Efficient Adaptively-Secure Byzantine Agreement for Long Messages

Amey Bhangale[1], Chen-Da Liu-Zhang[2*], Julian Loss[3], and Kartik Nayak[4]

[1] amey.bhangale@ucr.edu, UC Riverside
[2] cliuzhan@andrew.cmu.edu, Carnegie Mellon University
[3] lossjulian@gmail.com, CISPA Helmholtz Center
[4] kartik@cs.duke.edu, Duke University

Abstract. We investigate the communication complexity of Byzantine agreement protocols for long messages against an adaptive adversary. In this setting, prior results either achieved a communication complexity of $O(nl \cdot \mathsf{poly}(\kappa))$ or $O(nl + n^2 \cdot \mathsf{poly}(\kappa))$ for l-bit long messages. We improve the state of the art by presenting protocols with communication complexity $O(nl + n \cdot \mathsf{poly}(\kappa))$ in both the synchronous and asynchronous communication models. The synchronous protocol tolerates $t \leq (1-\varepsilon)\frac{n}{2}$ corruptions and assumes a VRF setup, while the asynchronous protocol tolerates $t \leq (1-\varepsilon)\frac{n}{3}$ corruptions under further cryptographic assumptions. Our protocols are very simple and combine subcommittee election with the recent approach of Nayak et al. (DISC '20). Surprisingly, the analysis of our protocols is *all but simple* and involves an interesting new application of Mc Diarmid's inequality to obtain *almost optimal* corruption thresholds.

Keywords: adaptive adversary, Byzantine agreement, long messages, communication complexity

1 Introduction

Byzantine agreement (BA) is a fundamental problem in distributed computing. In a Byzantine agreement protocol consisting of n parties, each party starts with an input value, and at the end of the protocol, all honest (non-faulty) parties output a value. Byzantine agreement protocols guarantee that if all honest parties input the same value v, then they must output v; otherwise, they output any agreed upon value. Moreover, this holds even if some threshold t out of n parties are Byzantine (arbitrarily malicious).

Byzantine agreement and other consensus primitives form a core abstraction for many blockchains where consensus is required on large values among a large number of parties [6, 21]. Moreover, due to the value of the transactions contained in these blockchains, they need to tolerate strong adaptive adversaries who are capable of corrupting any party based on the state of the protocols subject to the Byzantine threshold constraint. These requirements lead to the following natural question: *What is the lowest communication complexity possible for Byzantine agreement protocols on large values tolerating an adaptive adversary?*

This question has been partially answered in the literature. For instance, it has been shown that BA can be solved with $o(n^2)$ communication complexity against an adaptive adversary [11, 1, 6]. At a high level, these protocols take the approach of electing committees of size κ (where κ is a security parameter) and only the committee members send messages to all parties. This allows achieving a communication complexity of $O(n \cdot \mathsf{poly}(\kappa))$. However, this computation implicitly assumes inputs with a constant number of bits. If the inputs are of size l bits, the communication complexity is $O(nl \cdot \mathsf{poly}(\kappa))$.

A different line of work on extension protocols seeks to achieve the optimal communication complexity of $O(nl)$ for *long messages*. Currently, these works are only capable of considering messages that are very long, i.e., $l \gg n$ [19, 9, 16, 22, 10]. The best known protocols in this area achieve a communication complexity of $O(nl + \kappa n^2)$ [19] and the main goal of these works is to further reduce the latter term as much as possible. At a high level, these protocols take the approach of agreeing on the hash of an input value with $O(\kappa n^2)$ communication (κ is the size of a hash) assuming appropriate BA protocols for κ-sized

[*] Supported in part by the NSF award 1916939, DARPA SIEVE program, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award.

inputs and then use erasure coding techniques to distribute the l-bit long blocks with communication $O(nl + \kappa n^2)$. In this work, we ask whether we can achieve the best of both approaches. In particular,

Does there exist a Byzantine agreement protocol for l bit values tolerating an adaptive adversary with $O(nl + n \cdot \mathsf{poly}(\kappa))$ communication complexity?

We answer this question positively. Surprisingly, the techniques from the two lines of work do not compose in a straightforward manner to achieve the desired communication complexity. In fact, Nayak et al. [19] present a lower bound of $\Omega(nl + A(\kappa) + n^2)$ where $A(\kappa)$ is the communication complexity of Byzantine agreement on κ bit inputs. However, the bound holds only for deterministic protocols. For the first time, we use randomization in the extension part (as well as the underlying protocol) to circumvent the lower bound and achieve $O(nl + n \cdot \mathsf{poly}(\kappa))$ complexity. We present two protocols one assuming synchronous network and another assuming asynchronous network, that achieve these guarantees.

In the following table, we show our improvements on both communication complexity and input range of l to reach optimality, with respect to the previous most efficient l-bit Byzantine agreement protocol due work by Nayak et al. [19].

Threshold	**Model**	**Communication Complexity**	**Input range l to reach optimality**	**Reference**
$t < n/2$	sync.	$O(nl + n^2\kappa)$	$\Omega(\kappa n)$	[19]
		$O(nl + n \cdot \kappa^3)$	$\Omega(\kappa^3)$	**This work**
$t < n/3$	async.	$O(nl + n^2\kappa)$	$\Omega(\kappa n)$	[19]
		$O(nl + n \cdot \kappa^6)$	$\Omega(\kappa^6)$	**This work**

Table 1. Comparison with state-of-the-art Byzantine agreement for l-bit messages.

1.1 Simple Adaptively Secure BA Protocols for Long Messages

Our first result is a synchronous, adaptively secure BA protocol tolerating $t \le (1-\varepsilon) \cdot \frac{n}{2}$ Byzantine parties, for some arbitrary constant $\varepsilon > 0$. The second result is asynchronous and tolerates $t \le (1-\varepsilon) \cdot \frac{n}{3}$ corruptions.

Theorem 1. *For all constants $\varepsilon > 0$, assuming appropriate cryptographic setup assumptions, there exists an adaptively secure synchronous Byzantine agreement protocol achieving a communication complexity of $O(nl + n \cdot \mathsf{poly}(\kappa))$ for l-bit values for*

1. *$t \le (1-\varepsilon) \cdot \frac{n}{2}$ Byzantine parties under a synchronous network, and*
2. *$t \le (1-\varepsilon) \cdot \frac{n}{3}$ Byzantine parties under an asynchronous network.*

We describe a very high-level intuition behind the synchronous protocol. Using an adaptively-secure subquadratic 1-bit BA protocol from [1], all parties can agree on a κ-bit accumulator value corresponding to one of the inputs with a communication of $O(\kappa^3 n)$. Thus the key challenge is to distribute the l-bit value to all parties with linear communication while tolerating an adaptive adversary. Typically, distributing a large value to n parties using erasure codes is performed in two steps. First, create n encoded shares of the value, one for each party, of size $O(\frac{l}{n})$, and send the shares to the respective parties. Then, each party sends its own share to all other parties. If every party receives sufficiently many shares (Byzantine parties may not send shares), they can reconstruct the l-bit value. Observe that the latter step incurs $\Omega(n^2)$ communication, thus dominating the $n \cdot \mathsf{poly}(\kappa)$ term of the desired communication complexity. To make this approach efficient, we have to find the right amount of shares to create and the right parties to share them with. If we naïvely create one share per party, we will need all parties to speak so that we can reconstruct the long message. Clearly, this results in poor communication complexity. On the other hand, if we share the messages with only a small committee C, an adaptive adversary can corrupt all the parties in C and prevent reconstruction of the long message.

To address these concerns, our solution relies on a "public" partition of parties into one of κ buckets such that each bucket holds n/κ parties. We then elect κ-sized committees at random (using the standard

VRF approach for cryptographic sortition) to perform each of the two steps described earlier. In the first step, the value is encoded into κ shares of size $O(\frac{l}{\kappa})$ and the j-th share is sent to parties in bucket j. In the second step, the elected committee members from each of the κ buckets send their share to all parties. This incurs an $O(\kappa n \cdot \frac{l}{\kappa}) = O(ln)$ bits of communication. The crux of our argument lies in showing that when $t \leq (1-\varepsilon) \cdot \frac{n}{2}$, sufficiently many buckets contain an honest party who is also elected as a committee member. Thus, the shares that these honest parties send are sufficient to reconstruct the initial value. If we elect parties to the committee C using the common approach of verifiable random functions, it is not possible to argue via standard Chernoff-type bounds that sufficiently many of the buckets will be covered by members of C. This is because the number of committee members across buckets are correlated and a rushing adaptive adversary can observe the number of committee members for any subset of the buckets before corrupting others. Instead, our argument relies on a subtle application of McDiarmid's inequality, which, to the best of our knowledge, has not been explored in this type of protocol. Our analysis shows that choosing a committee of (expected) size $O(\kappa)$ is enough for our purposes.

Using our insights from the synchronous setting, we also obtain a protocol for the asynchronous setting by substituting the 1-bit agreement protocol with the recent (asynchronous) BA construction of Blum et al. [3].

1.2 Related Work

Work related to extension protocols. In the following, we denote as $\mathcal{A}(1), \mathcal{A}(\kappa)$ the communication complexity of a BA protocol with input domain of size 1 and κ bits, respectively. The problem of extending the domain of Byzantine agreement protocols is a well-studied one in the literature. To the best of our knowledge, the first work that considered this problem is that of Turpin and Coan [22] who showed how to reach agreement on messages from arbitrary domains given agreement on binary values in the corruption regime $t < n/3$ with synchrony. The problem has also been considered for other related primitives such as Byzantine broadcast [12, 7] or reliable broadcast [4, 19]. Previous works that focus on this problem are the works by Fitzi and Hirt [9], and that of Liang and Vaidya [16]. In the synchronous setting with $t < n/3$ and error-freeness, the protocol of Ganesh and Patra [10] previously provided the best known protocol which achieves $O(nl + n^2 \cdot \mathcal{A}(1))$ communication complexity. For the computational setting with $t < n/2$, the protocols of Ganesh and Patra [10] previously provided the best known solution achieving $O(nl + n\mathcal{A}(\kappa) + \kappa n^3)$. These complexities were recently further improved by the protocols of Nayak et al. [19] who gave protocols that achieve $O(nl + \mathcal{A}(\kappa) + n^2\kappa)$ communication complexity for the computational setting when $t < n/3$ or $t < n/2$. Nayak et al. also improved on error-free protocols in the $t < n/3$ setting, giving a protocol that achieves $O(nl + n\mathcal{A}(1) + n^3)$ communication complexity.

Our work improves over previous works on Byzantine agreement for long messages, both in the communication complexity as well as the input range of l to reach optimality for both the synchronous and asynchronous setting. However, our protocols require further tools. Compared to the protocol of Nayak et al. [19], our synchronous protocol requires an additional setup for verifiable random functions [17], and our asynchronous protocol also requires several cryptographic assumptions, including non-interactive zero-knowledge, threshold fully-homomorphic encryption and anonymous public-key encryption.

Work related to adaptively secure sub-quadratic communication protocols. Dolev and Reischuk [8] first showed that deterministic Byzantine agreement protocols incur $\Omega(t^2)$ communication complexity when tolerating $t < n$ Byzantine faults. King et al. [15, 13, 14] presented the first Byzantine agreement protocols that can be solved with subquadratic communication complexity under inverse polynomial in n error probability. More recently, Algorand [11, 6] showed constructions with $O(n \cdot \mathsf{poly}(\kappa))$ communication complexity for adaptively secure Byzantine agreement tolerating $t < (1-\varepsilon)n/3$ Byzantine parties in the synchronous setting assuming memory erasures. This was further improved by Abraham et al. [1] in the synchronous and partially synchronous network setting tolerating $t < (1-\varepsilon)n/2$ and $t < (1-\varepsilon)n/3$ respectively without assuming memory erasures. Finally, Blum et al. [3] presented a subquadratic communication protocol in the asynchronous setting tolerating $t < (1-\varepsilon)n/3$ faults. As discussed above, these protocols achieve subquadratic communication complexity, but fail to provide the asymptotically optimal complexity $O(nl)$ when l grows beyond n. Nonetheless, these protocols do serve as important building blocks in extension protocols such as the ones presented here (i.e., to agree efficiently on the short message shares).

2 Model and Preliminaries

We consider a setting with n parties $P_1, \ldots, P_n$ that have access to a complete network of pairwise authenticated channels. The adversary is adaptive, and can corrupt up to t parties at any point of the protocol execution in an arbitrary manner. However, we make two standard assumptions on the capability of the adversary (see, e.g., [5, 3]). First, parties can perform an *atomic send operation*, i.e., they can send a message to any number of parties *simultaneously* and without the adversary corrupting them in between (different) sends. Second, the adversary cannot perform *after-the-fact* removal, i.e., cannot take back messages sent by parties while they were still honest.[5] We consider protocols in the synchronous and asynchronous network settings. In a synchronous network, we assume communication in lock-step rounds where messages sent by a party at the start of a round arrives at its destination by the end of that round. On the other hand, in an asynchronous network, messages are assumed to arrive at their destination eventually.

2.1 Definitions

Let us recap the definition of Byzantine agreement.

Definition 1 (Byzantine Agreement). *Let Π be a protocol executed by parties $P_1, \ldots, P_n$, where each party P_i starts with an input x_i and parties terminate upon generating output. We say that Π is an t-secure Byzantine agreement protocol if the following properties hold when up to t parties are corrupted:*

- *Validity: If all honest parties start with the same input x, then every honest party outputs x.*
- *Consistency: All honest parties output the same value.*

2.2 Primitives

Our protocols will make use of standard linear error correcting codes and cryptographic accumulators.

Linear error correcting code. We use standard Reed-Solomon (RS) codes with parameters (κ, b). The codewords are elements in a Galois Field $GF(2^a)$ with $\kappa \leq 2^a - 1$. There are two algorithms:

- Encoding. Given inputs $m_1, \ldots, m_b$, the encoding function outputs κ codewords (a.k.a. shares) $(s_1, \ldots, s_\kappa)$ of length κ, such that any b codewords uniquely determine the input message and the other codewords.
- Decoding. Given κ codewords $(s_1, \ldots, s_\kappa)$, one can reconstruct the original message $(m_1, \ldots, m_b)$ even when $\kappa - b$ values are erased.

Looking ahead in our protocols, we will choose random committee subsets of κ parties out of the n parties, and we will set the parameter to b, to a lower bound on the number of honest parties in a committee.

Cryptographic accumulators. We recall the definition of cryptographic accumulators [2]. Given a set of values, the primitive can produce an accumulated value and a witness for each element in the set. Then, given the accumulated value and a witness, one can verify that a particular element is in the set.

Definition 2. *A cryptographic accumulator consists of a tuple of algorithms* $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{CreateWit}, \mathsf{Verify})$*, where:*

- $\mathsf{Gen}(1^\kappa, T)$*: It takes a parameter κ and an accumulation threshold T and returns an accumulator key* $\mathtt{ak}$*.*
- $\mathsf{Eval}(\mathtt{ak}, \mathcal{D})$*: It takes an accumulator key* $\mathtt{ak}$ *and a set of values to accumulate $\mathcal{D}$ and returns an accumulated value z for $\mathcal{D}$.*
- $\mathsf{CreateWit}(\mathtt{ak}, z, d_i)$*: It takes an accumulator key* $\mathtt{ak}$*, an accumulated value z for $\mathcal{D}$ and a value d_i, and returns $\perp$ if $d_i \notin \mathcal{D}$ or a witness w_i otherwise.*
- $\mathsf{Verify}(\mathtt{ak}, z, w_i, d_i)$*: It takes an accumulator key, accumulated value z for $\mathcal{D}$, witness w_i, value d_i, and returns 1 if w_i is a witness for $d_i \in \mathcal{D}$ and 0 otherwise.*

We require our accumulator to satisfy standard collision-free properties [20].

[5] In the absence of this assumption, no protocol (deterministic or randomized) can achieve $o(t^2)$ communication complexity as shown in Abraham et al. [1].

2.3 Concentration Bounds I

We recall the Chernoff concentration bound.

Lemma 1 (Homogenous Chernoff Bound). *Let $X_1, ..., X_n$ be i.i.d. Bernoulli random variables with parameter p. Let $X := \sum_i X_i$, so $\mu := E[X] = p \cdot n$. Then, for $\delta \in [0, 1]$,*

$$\Pr[X \geq (1+\delta) \cdot \mu] \leq e^{-\delta^2 \mu/(2+\delta)} \qquad \text{and} \qquad \Pr[X \leq (1-\delta) \cdot \mu] \leq e^{-\delta^2 \mu/2}.$$

Let $\chi_{s,n}$ denote the distribution that samples a subset of the n parties, where each party is included independently with probability s/n. The following lemma will be useful in our analysis.

Corollary 1. *Fix $\kappa \leq s \leq n$ and $\varepsilon > 0$, and let $t = (1-\varepsilon)n/2$ be the number of corrupted parties. If $C \leftarrow \chi_{s,n}$, then C contains less than $(1 - \frac{2}{3}\varepsilon)s/2$ corrupted parties except with negligible probability.*

Proof. Let $H \subseteq [n]$ be the indices of the honest parties. Let X_j be the Bernoulli random variable indicating if $P_j \in C$, so $\Pr[X_j = 1] = s/n$. Define $Z := \sum_{j \notin H} X_j$. Then, since $E[Z] = t \cdot s/n = (1-\varepsilon)s/2$, setting $\delta = \frac{\varepsilon}{3(1-\varepsilon)}$ in Lemma 1 yields

$$\Pr\left[Z \geq (1 - \frac{2}{3}\varepsilon)s/2\right] \leq \mathtt{neg}(\kappa).$$

(Almost) the same proof yields:

Corollary 2. *Fix $\kappa \leq s \leq n$ and $\varepsilon > 0$, and let $t = (1-\varepsilon)n/3$ be the number of corrupted parties. If $C \leftarrow \chi_{s,n}$, then C contains less than $(1 - \frac{2}{3}\varepsilon)s/3$ corrupted parties except with negligible probability.*

Corollary 3. *Fix $s \leq n$ and $0 < \varepsilon < 1$. If $C \leftarrow \chi_{s,n}$, then C contains more than $(1-\varepsilon) \cdot s$ many parties except with probability at most $O(e^{-\varepsilon^2 s})$.*

Proof. Let $H \subseteq [n]$ be the indices of the honest parties. Let X_j be the Bernoulli random variable indicating if $P_j \in C$, so $\Pr[X_j = 1] = s/n$. Define $Z := \sum_{j \notin H} X_j$. Then, since $E[Z] = s$, setting $\delta = \varepsilon$ in Lemma 1 yields

$$\Pr[Z \leq (1-\varepsilon) \cdot s] \leq e^{\varepsilon^2 \cdot s/2}.$$

3 Balls and Buckets Analysis for Throwing *ck* Balls in *k* Buckets

Our protocols in subsequent sections rely on publicly partitioning n parties in κ distinct buckets and then electing $c\kappa$ out of the n parties uniformly at random. Some of the elected parties can be Byzantine; and our protocols require some properties on the number of distinct buckets containing honest elected parties. In this section, we present the technical inequality that will be used in our protocols in subsequent sections. We use the notation $[k]$ to denote the set $\{1, 2, \ldots, k\}$. We will start with the following concentration bound:

Theorem 2. *(McDiarmid's Inequality) Let $X_1, X_2, \ldots, X_n$ be independent random variables such that $X_j \in \mathcal{K}_j$, for some measurable set $\mathcal{K}_j$. Suppose $f : \prod_{j=1}^n \mathcal{K}_j \to R$ is 'Lipschitz' in the following sense: there exist $\sigma_1, \sigma_2 \ldots, \sigma_n \geq 0$ such that for each $1 \leq k \leq n$ and any two input sequences $x, x' \in \prod_j \mathcal{K}_j$, that differ only in the k^{th} coordinate,*

$$|f(x) - f(x')| \leq \sigma_k.$$

Let $Y = f(X_1, X_2, \ldots, X_n)$. Then for any $\alpha > 0$,

$$\mathbf{Pr}[|Y - \mathbf{E}[Y]| \geq \alpha] \leq 2 \cdot \exp\left(-\frac{2\alpha^2}{\sum_{j=1}^n \sigma_j^2}\right).$$

The binomial distribution with parameters n and p is the discrete probability distribution of the number of successes in a sequence of n independent experiments, each asking a yes-no question, and each with its own Boolean-valued outcome: success (with probability p) or failure (with probability $1-p$).

Let $c \geq 1$ and $k \geq 5$ be the parameters where k is the number of buckets and ck is the number of balls (committee members). Consider the following random experiment: We throw ck balls in k buckets independently and uniformly at random. Let b_i be the expected number of buckets with exactly i balls.

Let X_j^i be the indicator random variable that the j^{th} bucket has exactly i balls. Thus, using linearity of expectation, we can write b_i as:

$$b_i = \sum_{j=1}^{k} \mathbf{E}[X_j^i].$$

We also have,

$$\mathbf{E}[X_j^i] = \binom{ck}{i} \cdot \left(\frac{1}{k}\right)^i \cdot \left(1 - \frac{1}{k}\right)^{ck-i}.$$

By linearity of expectation,

$$\begin{aligned} b_i &= \sum_{j=1}^{k} \binom{ck}{i} \cdot \left(\frac{1}{k}\right)^i \cdot \left(1 - \frac{1}{k}\right)^{ck-i} \\ &= k \cdot \binom{ck}{i} \cdot \left(\frac{1}{k}\right)^i \cdot \left(1 - \frac{1}{k}\right)^{ck-i}. \end{aligned}$$

The following lemma shows that the number of buckets with exactly i balls is concentrated around b_i.

Lemma 2. *For $k \geq 2$ and $0 \leq i \leq c$, $\mathbf{Pr}\left[\left|\begin{matrix}\textit{number of buckets} \\ \textit{with exactly } i \textit{ balls}\end{matrix} - b_i\right| \geq \varepsilon \cdot b_i\right] \leq 2\exp(-\frac{\varepsilon^2}{e^{5c}} \cdot k)$.*

Proof. Suppose the k buckets are labeled with (distinct) numbers from $[k]$. Let $m = ck$ and define a function $f : [k]^{ck} \to R$ as follows.

$$f(a_1, a_2, \ldots, a_m) = \left|\{\ell \in [k] \mid \ell \text{ appears exactly } i \text{ times in } (a_1, a_2, \ldots, a_m)\}\right|.$$

We are interested in the random variable $Y = f(x_1, x_2, \ldots, x_m)$ where each x_j is distributed independently and uniformly in $[k]$. This is because we can think of x_j as the bucket number in which the jth ball lands. Therefore, $f(a_1, a_2, \ldots, a_m)$ is precisely the number of buckets that contain exactly i balls, when the jth ball goes into the bucket a_j for all $j \in [m]$.

It is clear that f is Lipschitz with a Lipschitz constant of 1, i.e, if you change only one input coordinate, then the function value changes by at most 1. Towards applying Theorem 2, we have $\sigma_j = 1$ for all $j \in [m]$ and hence $\sum_j \sigma_j^2 = m$. Note that

$$\begin{aligned} b_i &= k \cdot \binom{ck}{i} \cdot \left(\frac{1}{k}\right)^i \cdot \left(1 - \frac{1}{k}\right)^{ck-i} \\ &\geq k \cdot \left(\frac{ck}{i}\right)^i \cdot \left(\frac{1}{k}\right)^i \left(1 - \frac{1}{k}\right)^{ck} \\ &\geq k \cdot \frac{c^i}{i^i} \left(1 - \frac{1}{k}\right)^{ck} \\ &\geq k \cdot \frac{c^i}{i^i} \left(e^{-2\frac{1}{k}}\right)^{ck} && \text{(Using } 1 - x \geq e^{-2x} \text{ for } x \in [0, \tfrac{1}{2}]) \\ &\geq \frac{c^i}{i^i e^{2c}} k. \end{aligned}$$

Using McDiarmid's inequality 2,

$$\begin{aligned}
\mathbf{Pr}[|Y - b_i| \geq \varepsilon \cdot b_i] &\leq 2\exp\left(-\frac{2\varepsilon^2 b_i^2}{m}\right) \\
&\leq 2\exp\left(-\frac{2\varepsilon^2 b_i^2}{ck}\right) \\
&\leq 2\exp\left(-\frac{2\varepsilon^2 c^{2i}}{i^{2i} e^{4c} \cdot c} \cdot k\right) && \text{(Using } b_i \geq \tfrac{c^i}{i^i e^{2c}} k\text{)} \\
&\leq 2\exp\left(-\frac{\varepsilon^2}{e^{5c}} \cdot k\right). && \text{(Using } i \leq c \text{ and } c \leq e^c\text{)}
\end{aligned}$$

We only need concentration for $i = 0, 1, \ldots, c-1$ for the overall argument that follows next. Since each holds with probability $1 - \exp(-\varepsilon^2 k/e^{O(c)})$, by union bound, we have that the number of buckets with i balls is concentrated around its expectation for $i = 0, 1, \ldots, c-1$ happens with probability at least $1 - c \cdot \exp(-\varepsilon^2 k/e^{O(c)})$.

Claim. Let $\tau \in (0, 1/2]$ be any constant. There exists a constant $0 \leq c_\tau \leq c$ such that the following two inequalities hold simultaneously. We have,

$$\sum_{i=0}^{c_\tau} b_i \leq \tau k \tag{1}$$

and

$$\sum_{i=1}^{c_\tau} i \cdot b_i \geq (\tau - o_k(1)) \cdot ck. \tag{2}$$

Proof. Let c_τ be the largest constant such that (1) holds. The sum $\sum_{i=0}^{c_\tau} b_i/k$ is the cumulative density of the binomial distribution with parameters ck and $\frac{1}{k}$ at c_τ. As the median of the binomial distribution with parameters ck and $\frac{1}{k}$ is c, we have $c_\tau \leq c$ for $\tau \in (0, 1/2]$. We will show that, for this constant c_τ, the inequality (2) holds.

$$\begin{aligned}
\sum_{i=1}^{c_\tau} i \cdot b_i &= \sum_{i=0}^{c_\tau} i \cdot k\binom{ck}{i} \cdot \left(\frac{1}{k}\right)^i \cdot \left(1 - \frac{1}{k}\right)^{ck-i} \\
&= k \cdot \sum_{i=0}^{c_\tau} i \cdot \binom{ck}{i} \cdot \left(\frac{1}{k}\right)^i \cdot \left(1 - \frac{1}{k}\right)^{ck-i} \\
&= k \cdot \sum_{i=1}^{c_\tau} ck \cdot \binom{ck-1}{i-1} \cdot \left(\frac{1}{k}\right)^i \cdot \left(1 - \frac{1}{k}\right)^{ck-i} && \left(k\tbinom{n}{k} = n\tbinom{n-1}{k-1}\right) \\
&= k \cdot ck \cdot \frac{1}{k} \cdot \sum_{i=1}^{c_\tau} \binom{ck-1}{i-1} \cdot \left(\frac{1}{k}\right)^{i-1} \cdot \left(1 - \frac{1}{k}\right)^{(ck-1)-(i-1)} \\
&= ck \cdot \sum_{i=0}^{c_\tau - 1} \binom{ck-1}{i} \cdot \left(\frac{1}{k}\right)^i \cdot \left(1 - \frac{1}{k}\right)^{(ck-1)-i}. && \text{(Setting } i \leftarrow i-1\text{)}
\end{aligned}$$

Now, the summation is precisely the cumulative density of the binomial distribution with parameters $ck - 1$ and $\frac{1}{k}$ at $c_\tau - 1$. We now rearrange the terms to get the cumulative density of the binomial distribution with parameters ck and $\frac{1}{k}$ at $c_\tau + 1$ in the summation. This way we can relate it to the constant τ.

$$
\begin{aligned}
\sum_{i=1}^{c_\tau} i \cdot b_i &= ck \cdot \sum_{i=0}^{c_\tau - 1} \binom{ck-1}{i} \cdot \left(\frac{1}{k}\right)^i \cdot \left(1 - \frac{1}{k}\right)^{(ck-1)-i} \\
&= ck \cdot \sum_{i=0}^{c_\tau - 1} \frac{\frac{ck-i}{ck}}{(1-1/k)} \binom{ck}{i} \cdot \left(\frac{1}{k}\right)^i \cdot \left(1 - \frac{1}{k}\right)^{ck-i} \\
&\geq ck \cdot \sum_{i=0}^{c_\tau - 1} \binom{ck}{i} \cdot \left(\frac{1}{k}\right)^i \cdot \left(1 - \frac{1}{k}\right)^{ck-i} \\
&= ck \cdot \sum_{i=0}^{c_\tau - 1} \frac{b_i}{k} \\
&= ck \cdot \left(\left(\sum_{i=0}^{c_\tau + 1} \frac{b_i}{k} \right) - \frac{b_{c_\tau}}{k} - \frac{b_{c_\tau+1}}{k} \right) \\
&\geq ck \cdot \left(\tau - \frac{b_{c_\tau}}{k} - \frac{b_{c_\tau+1}}{k} \right) \\
&= ck \left(\tau - o_k(1)\right).
\end{aligned}
$$

Here, in the first inequality, we used the fact that c_τ is at most c. The second inequality uses the fact that the constant c_τ is the largest constant that satisfies inequality (1) from the claim. Therefore, $\sum_{i=0}^{c_\tau+1} \frac{b_i}{k} \geq \tau$.

Now, $c_\tau \leq c$ for every $\tau \in (0, 1/2]$. Using this, we combine Lemma 2 and Claim 3 along with a simple application of union bound and the fact that the sums are natural numbers, to get the following Corollary.

Corollary 4. *For all $\varepsilon > 0, c \geq 1, k \geq 2$ and $\tau \in (0, 1/2]$ there exists a constant $c_\tau \leq c$ such that the following holds. Suppose we throw ck balls in k buckets, each uniformly and independently at random. Let b'_i be the number of buckets with exactly i balls. Then the following two inequalities hold with probability at least $1 - 2 \cdot c \cdot \exp(-\frac{\varepsilon^2}{e^{5c}} \cdot k)$.*

1. $\sum_{i=0}^{c_\tau} b'_i \leq \lfloor (1+\varepsilon)\tau k \rfloor$.
2. $\sum_{i=1}^{c_\tau} i \cdot b'_i \geq (1-\varepsilon)\left(\tau - o_k(1)\right) \cdot ck$.

4 Adaptively Secure Synchronous Communication-Efficient Protocol for Long Messages

In this section, we describe an adaptively-secure communication-efficient protocols for long messages of size $l = O(\kappa)$. In particular, we will achieve a communication complexity of $ln + n \cdot \mathsf{poly}(\kappa)$ under the synchrony assumption while tolerating $t \leq (1-\varepsilon)\frac{n}{2}$ faults.

4.1 Intuition

The $O(ln + \kappa n^2)$ approach [19]. Let us start by recalling the extension protocol proposed by Nayak et al. [19] which achieves a communication complexity of $ln + \kappa n^2$ when $l \gg n$. The protocol splits the l-bit agreement task into two sub-goals. The first sub-goal is to identify whether all honest parties can agree on one of the honest inputs. The second sub-goal ensures parties share the l-bit value efficiently if they have decided to agree upon an honest input in the first sub-goal.

To achieve the first sub-goal, the protocol requires every party to create a cryptographic accumulator corresponding to their input values and run a κ-bit Byzantine agreement (BA) protocol to agree on the accumulated value. If the κ-bit BA protocol outputs the same value as their input, they engage in another 1-bit BA protocol with input 1. Otherwise, they input 0 to the 1-bit BA protocol. Finally, if the 1-bit BA protocol outputs 1, the parties proceed with the second sub-goal related to sharing the long inputs (described in the next paragraph). If the 1-bit BA protocol outputs 0, parties output $\perp$ and end the protocol. Observe that the κ-bit BA protocol ensures that parties have the same accumulated value as their output; however, the 1-bit BA protocol is the one which puts all parties in agreement on whether to engage in the "sharing" phase or not. If yes, it ensures that some honest party does have the input corresponding to the agreed-upon value (since otherwise, the 1-bit BA would output 0).

The second sub-goal relates to sharing the l-bit long input (for $l \gg n$) with every other party with communication complexity $O(ln)$. Observe that if all honest parties share this value with every party, trivially, we have a communication complexity of $O(ln^2)$. Moreover, we cannot rely on a single honest party (say a chosen leader) to share this value directly with other parties either. For one, we do not know which honest party has the input to be shared. Even if we did, a Byzantine party can always claim to not have received this value from the honest leader. This cannot be distinguished from an honest party legitimately claiming to not receive the value from a Byzantine leader. To address this concern, Nayak et al. [19] rely on using erasure coding techniques instead. Each party that inputs 1 to the 1-bit BA protocol must have the same l-bit value (corresponding to the agreed-upon accumulated value). Thus, each such party can create appropriate (deterministic) n shares (using RS codes) with appropriate witnesses (cf. Definition 2) such that each share is of size $O(\frac{l}{n})$. We call this the *distribute* step of the protocol. The l-bit value can be reconstructed if a party receives a majority of distinct shares. Thus, when a party receives a share in the distribute step, using the witness, it verifies whether the share matches the agreed upon accumulated value. If yes, it shares this value with all other parties. This step is called the *reshare* step of the protocol. On receiving a majority of the shares, every party can reconstruct the l-bit value. The honest majority assumption ensures that all parties will be able to reconstruct the value; thus, no (Byzantine) party can claim to not have received it.

Observe that the BA protocol used in the first sub-goal requires a communication complexity of κn^2, which can be achieved using [18]. The communication complexity to achieve the second sub-goal is $O(\frac{l}{n} \cdot n^2) = O(ln)$ (sharing the witnesses along with the share to verify the correctness of shares incurs and additional $O(\kappa n^2)$ term, not described here in this intuition).

Towards $O(ln + n \cdot \mathsf{poly}(\kappa))$ communication complexity with adaptive security. We now describe our approach. At a high level, we maintain a similar structure and have similar sub-goals as that of Nayak et al. [19]. However, we need to achieve a better communication complexity while being adaptively secure which brings in several subtleties.

To achieve the first sub-goal, we rely on an underlying adaptively-secure BA protocol that has a communication complexity of $O(n \cdot \mathsf{poly}(\kappa))$ for κ and 1-bit inputs [1]. At a high-level, this protocol achieves a 1-bit sub-quadratic BA by selecting uniformly random and verifiable committees of size κ for each round of the execution. The parties in the committees send protocol messages to all other parties, who update their state based on the messages received. The committees are elected using verifiable random functions [17] which depends on the party's secret key; thus, adversary cannot predict whether a given party would be elected until the party sends a message to all other parties. Since this is a 1-bit BA protocol, for κ-bit BA we use κ independent instances of this protocol.

To achieve the second sub-goal of sharing the l-bit inputs, observe that the solution by [19] does not work in our scenario. In particular, in the reshare step, even if each party sends a 1-bit value to every other party, this trivially incurs a communication complexity of $O(n^2)$. When l is large, this is bounded by $O(ln)$; however, when l is small, e.g., $l < n$, this term is still quadratic. Thus, our goal is to achieve a communication complexity of $O(ln + n \cdot \mathsf{poly}(\kappa))$ even when $\kappa \leq l < n$ while achieving adaptive security. We take inspiration from the sharing technique used in [19] but attempt to constrain the number of parties that can send messages in the distribute and reshare steps of the protocol using committee-based election techniques.

However, achieving this goal is riddled with challenges. First, in the distribute step, due to the adaptive security requirement, a party does not know which party will be in the committee in the reshare step. Second, in the distribute step, to respect the communication complexity requirement, parties can only share $\frac{l}{\kappa}$-sized shares with other parties, and not the entire l-bit message. Finally, the sharing must happen in such a way that sufficiently many honest parties in the reshare committee must have distinct shares so that the value can eventually be reconstructed by all parties.

Our solution relies on publicly partitioning the parties into κ different buckets $B_1, \dots, B_\kappa$, e.g., based on their IDs. A visual sketch is depicted in Figure 1. The input would be split into κ different shares

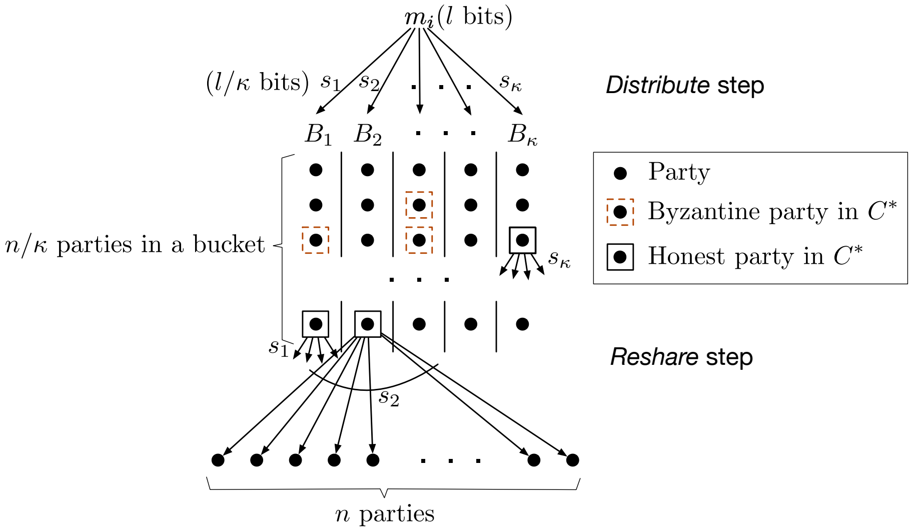

Fig. 1. Graphical representation of the distribute and reshare steps in our approach. The figure assumes party P_i is the only party engaging in the distribute step. In the distribute step, share s_j is shared with each of the parties in bucket B_j. In the reshare phase, a party in committee C^* and in bucket B_j shares share s_j with all the n parties. The figure does not show the process involved in the agreement of accumulated value and additional information such as witnesses shared by parties in the distribute and reshare step.

$s_1, \ldots, s_\kappa$ of size $\frac{l}{\kappa}$ such that $(1-\varepsilon)\frac{\kappa}{2}$ distinct shares are sufficient to reconstruct the block ($\varepsilon > 0$, ε slack relates to the committee election in the reshare step).

During the distribute step, parties in bucket B_j would receive share s_j. In the reshare step, we elect an independent reshare committee C^* of size $O(\kappa)$. Thus, if a party belongs to C^* and is from bucket B_j, then this party shares s_j with all the parties. Note that due to the use of an accumulator, parties can verify the correctness of the shares relative to the (agreed-upon) accumulated value, and thus would ignore incorrect shares. If we can ensure that all parties receive $(1-\varepsilon)\frac{\kappa}{2}$ distinct shares to reconstruct the l-bit input, then we would achieve the goal with the desirable communication complexity.

The crux of the concern is that, there are only $O(\kappa)$ parties in C^*. Thus, in expectation, each bucket B_j has $O(1)$ parties who can potentially share s_j. Due to the stochasticity, some buckets may not have any parties elected. Moreover, in a round, we cannot expect parties in the committee C^* to send reshare messages at exactly the same instant. This difference in times can potentially be used by an adaptive adversary to adaptively corrupt parties such that sufficiently many distinct shares are not shared. In particular, based on the number parties elected in the first few buckets, the adversary can decide its corruption strategy for the remaining buckets. This disallows the use of standard Chernoff-type bounds which requires independence across buckets.

To address this, we instead rely on the balls and buckets analysis described in Section 3. In particular, we can present this abstraction as throwing $c\kappa$ balls (elected committee members in C^*) into κ buckets uniformly at random. The adversary has a corruption quota of slightly less than $\frac{c\kappa}{2}$ among the committee members. The goal is then to ensure that there exists a constant c such that $(1-\varepsilon)\frac{\kappa}{2}$ distinct buckets have at least one honest committee member in C^*, so that these members are guaranteed to send the corresponding messages during the reshare step of the protocol. This is ensured in Corollary 4, which requires a careful analysis on the balls and buckets process via McDiarmid's inequality.

A final subtlety relates to the committee that should perform the distribute step. It turns out that due to the adaptive security and communication complexity requirement, the committee for the distribute step is the same as the parties that input 1 and participate in the first step of the 1-bit BA protocol (to achieve the first sub-goal). Intuitively, these are the parties which have access to the l-bit value corresponding to the agreed-upon accumulated value and are thus championing for all parties to agree upon the l-bit value. Thus, these parties *need* to send both these messages (the one in distribute step and the other in the first message for the 1-bit BA protocol) as a part of the same message; otherwise, the adaptive adversary can corrupt the parties after sending one of the two messages.

This completes the key intuition behind our protocol.

4.2 Protocol Description

As sketched above, our protocol makes use of several building blocks and setup.

Accumulator Setup. We assume a setup that chooses and distributes the accumulator keys.

Protocol Setup for $\Pi_{\texttt{sprBA}}$

Accumulators

Generate the accumulator key $\texttt{ak} = \mathsf{Gen}(1^\kappa, \kappa)$ and give it to all parties.

Verifiable Random Functions. When describing our protocol, we assume that parties have available via a trusted setup efficient algorithms ComProve and ComVer that allow them to prove and verify membership of a committee, and we do not make this explicit in our protocol (this can typically be achieved via a VRF setup).

Short BA Protocols. We make use of the adaptively-secure sub-quadratic BA protocol of Abraham et al. [1] as a building block. for both the κ-valued BA, denoted as $\mathsf{BA}(\kappa)$, and the binary-valued BA, denoted $\mathsf{BA}(1)$. (Note that the protocol in [1] is binary, but one can simply run it κ many times in parallel to agree on a κ bit message.)

In the protocol [1], all parties participate in the protocol, and different subsets of parties speak at different rounds. More concretely, at each round i, a committee is chosen uniformly at random for each $b \in \{0, 1\}$. Here, the committee is tied to the round number as well as the value b. Then, the committee members reveal themselves only when it is their turn to speak in the protocol. Intuitively, once members of the committee send their messages for round i, it is too late for the adversary to corrupt them, as they can not take back messages that were previously sent by honest parties.[6]

Finally, we would like to remark that the protocol in [1] has the property that if all honest parties in the first chosen committee have the same input value b, all honest parties output b. This property is in contrast to the standard notion of validity, which requires *all* honest parties to have the same input. Our protocol will make use of this in an essential way.

Reed-Solomon Codes. We will use two sub-protocols, Encode and Reconstruct, which are based on RS codes. We specify them relative to t_κ which in our protocol is set as $t_\kappa = (1 + \varepsilon)\frac{\kappa}{2}$.

- $\mathsf{Encode}(m)$. Given a message of size l, it divides the message into b blocks, and computes κ codewords $(s_1, \dots, s_\kappa)$ using RS codes, such that even when t_κ values are erased, one can recover the original message.
- $\mathsf{Reconstruct}(\mathcal{S}_i, \texttt{ak}, z, t_\kappa)$ removes incorrect values s_j for each pair $(s_j, w_j) \in \mathcal{S}_i$ that cannot be verified by the witness w_j and accumulation value z. And then reconstructs the message using RS code, where at most t_κ values are removed.

Our protocol starts from a fixed (arbitrary) partition of the n parties into κ buckets $B_1, \dots, B_\kappa$ of n/κ parties each. When describing our protocol, we will refer to C_i^b as the committee for the i-th round of an execution of $\mathsf{BA}(1)$ for the bit b. In our protocol specification, we make explicit the input value of *first* committee in $\mathsf{BA}(1)$, whereas the rest of the committees are implicit inside the protocol $\mathsf{BA}(1)$ and do not show up in the specification. Finally, we denote as C^* a special committee (also selected at random using ComProve) whose members are designated to perform the re-sharing step in our protocol.

Protocol $\Pi_{\texttt{sprBA}}$

Let $t_\kappa = \lfloor(1 + \varepsilon)\frac{\kappa}{2}\rfloor$. The protocol is described from the point of view of party P_i who holds an l-bit input message m_i.

```
1: Compute $\mathcal{D}_i := (s_1, \dots, s_\kappa) = \mathsf{Encode}(m_i)$, the accumulation value $z_i = \mathsf{Eval}(\texttt{ak}, \mathcal{D}_i)$. Input $z_i$ to $\mathsf{BA}(\kappa)$.
2: When the above BA outputs $z$, if $z = z_i$ and $P_i \in C_1^1$, input 1 to $\mathsf{BA}(1)$. Moreover, distribute the long
   block as follows. Compute a witness $w_j = \mathsf{CreateWit}(\texttt{ak}, z, s_j)$ for each share $s_j$ in Step 1 and send the
   tuple $(s_j, w_j)$ to each party $P_k \in B_j$. Otherwise, if $z \neq z_i$ and $P_i \in C_1^0$, input 0 to $\mathsf{BA}(1)$.
```

[6] Observe that, since committee members are tied to the bit b, the newly corrupted members cannot equivocate unless they are in both committees. The likelihood of that event is negligible.

3: If the output of the above BA is 0, output $\bot$ and abort. Otherwise, if $P_i \in C^* \cap B_j$: For the set of tuples $\{(s_j, w_j)\}$ received in the previous step from parties in C_1^1, if there exists an (s_j, w_j) such that $\mathsf{Verify}(\mathtt{ak}, z, w_j, s_j) = 1$, then send (s_j, w_j) to all parties.

4: Let $\mathcal{S}_i \coloneqq \{(s_j, w_j)\}$ be the set of messages received from the previous step from parties in C^*. If there are messages from parties belonging to at least $(1-\varepsilon)\frac{\kappa}{2}$ different buckets, output the reconstructed value $\mathsf{Reconstruct}(\mathcal{S}_i, \mathtt{ak}, z, t_\kappa)$. Otherwise, output $\bot$.

The following theorem is proven via a sequence of lemmas.

Theorem 3. *Let $0 < \varepsilon < 1/6$. Assuming a setup for VRFs and accumulators, $\Pi_{\mathtt{sprBA}}$ is a synchronous Byzantine agreement protocol secure up to $t < (1-6\varepsilon)n/2$ adaptive corruptions. The communication complexity is $O(nl + \kappa^3 n)$ for l-bit input values.*

In the proofs, we will need that the sub-protocol $\mathsf{BA}(1)$ satisfies the following somewhat stronger *committee-based* notion of validity described in the lemma below.

Lemma 3. *If all honest parties in C_1^b input b to $\mathsf{BA}(1)$, and no honest party in C_1^{1-b} inputs $1-b$ to $\mathsf{BA}(1)$, then the output of $\mathsf{BA}(1)$ is b.*

Proof. This follows from the fact that in protocol $\mathsf{BA}(1)$ only parties in the committee for the first round, which is C_1^b or C_1^{1-b}, speak and send their input to all other parties. Hence, if only honest parties in C_1^b input to $\mathsf{BA}(1)$ and no honest party from C_1^{1-b} inputs to $\mathsf{BA}(1)$, then it follows immediately from the validity proof given in [1] that the protocol should output b.

Lemma 4. *$\Pi_{\mathtt{sprBA}}$ satisfies validity for $t < (1-6\varepsilon)n/2$.*

Proof. If all honest parties have the same input message $m_i = m$, then all honest parties input the same accumulated value $z = z_i$ to $\mathsf{BA}(\kappa)$ in Step 1. By validity of $\mathsf{BA}(\kappa)$, all honest parties receive z as output. Hence, all honest parties in C_1^1 input 1 to $\mathsf{BA}(1)$ in Step 2 and distribute the shares of m. By Lemma 3, they receive 1 as output from $\mathsf{BA}(1)$.

Each honest party $P_j \in C^* \cap B_j$ receives a valid share s_j^i from each honest party $P_i \in C_1^1$, and forwards one of these shares to all parties. Parties are added to C^* uniformly at random, each with probability $c\kappa/n$, for constant c. Denote E_0 the event that fewer than $(1-\varepsilon)c\kappa$ parties are in C^* for $\varepsilon > 0$. By Corollary 3, we have that $\mathbf{Pr}[E_0]$ is negligible.

Whenever E_0 does not occur, we can map the process of distributing parties from C^* into buckets to the process of throwing $c\kappa$ or more balls at κ buckets. Moreover, the optimal strategy for the adversary to minimize the number of buckets in which an honest party sends a share is clearly to corrupt the buckets that contain smaller amount of parties from C^*.

Let us denote E_1 the event that $\frac{c\kappa}{2}(1-4\varepsilon)$ or more parties in C^* are corrupted. When $t < (1-6\varepsilon)n/2$, by Corollary 1, $\mathbf{Pr}[E_1]$ is negligible. Therefore, by a union bound, $\mathbf{Pr}[E_0 \cup E_1]$ is also negligible.

In the following, we condition on the event $\neg E_0 \wedge \neg E_1$ (which by the above occurs with overwhelming probability).

Let $c' = (1-\varepsilon)c$. By Corollary 4, and choosing $\tau = 1/2$, there is a constant $c'_{1/2}$ such that $\sum_{i=1}^{c'_{1/2}} i \cdot b_i \geq (1-\varepsilon)\left(1/2 - o_\kappa(1)\right) \cdot c'\kappa \geq \frac{c'\kappa}{2}(1-2\varepsilon)$, where the last inequality holds as long as $o_\kappa(1) \leq \frac{\varepsilon}{2(1-\varepsilon)}$. Substituting, $c' = (1-\varepsilon)c$, we have $\sum_{i=1}^{c'_{1/2}} i \cdot b_i \geq \frac{c\kappa}{2}(1-3\varepsilon)$. Therefore, the adversary can not corrupt all committee members in the buckets that contain up to $c'_{1/2}$ or less committee members. These amounts of buckets, t_κ, correspond to at most $\lfloor (1+\varepsilon)\kappa/2 \rfloor$ buckets, by Corollary 4.

Putting things together, at Step 4, at least honest parties from $\kappa - t_\kappa \geq (1-\varepsilon)\frac{\kappa}{2}$ buckets send a share, and thus every honest party receives at least that many valid shares. This way, all honest parties can reconstruct and output the long message m.

Lemma 5. *$\Pi_{\mathtt{sprBA}}$ satisfies consistency for $t < (1-6\varepsilon)n/2$.*

Proof. If $\mathsf{BA}(1)$ outputs 0, all honest parties output $\bot$. If $\mathsf{BA}(1)$ outputs 1, then by Lemma 3, there must exist an honest party $P_i \in C_1^1$ that input 1 to $\mathsf{BA}(1)$. First, this party P_i distributes its long messages m_i. Second, by Step 2 of the protocol, it must be the case that this honest party has received $z = z_i$. Using the consistency property of $\mathsf{BA}(\kappa)$, all honest parties must have delivered $z = z_i$. Thus, every honest party $P_j \in C^*$ obtains a valid tuple (s_j, w_j) from P_i and can verify its correctness using the accumulator value z and forward it. Hence, in Step 4, we can use the same argumentation as in the previous lemma to establish that at least $\kappa - t_\kappa$ honest parties in C^* send a share and every honest party can subsequently reconstruct m_i. Note that no other value can be reconstructed, because security of the accumulator and consistency of $\mathsf{BA}(\kappa)$ ensures that all honest parties share the same long message, and dishonest parties cannot compute valid pairs of share-witness different from those received by honest parties.

Communication complexity. The most expensive steps in the protocol are the run of $\mathsf{BA}(\kappa)$ in Step 1 (which itself consists of κ parallel runs of $\mathsf{BA}(1)$) and the distribution of the long blocks in Step 2. The costs for Step 1 are bounded as $O(\kappa^3 \cdot n)$ since every run of $\mathsf{BA}(1)$ costs $O(\kappa^2 \cdot n)$. The costs for Step 2 are bounded by $O(l \cdot n + \kappa^2 n)$, given that each of $O(\kappa)$ parties send a message of length $\ell/\kappa + \kappa$ to all parties. Overall, we obtain a complexity of $O(n \cdot l + \kappa^3 \cdot n)$.

5 Adaptively Secure Asynchronous Communication-Efficient Protocol for Long Messages

We briefly recall the asynchronous adaptively-secure BA protocol of Blum et al. [3]. As for the previous protocol, the step of each round i is performed by a randomly chosen committee C_i, who reveals itself only when it is their turn to speak in the protocol. Again, we assume that parties are endowed (via some trusted setup) with efficient routines $\mathsf{ComProve}$ and ComVer that allow to prove and verify committee membership. The remaining accumulator setup is as for Π_{sprABA} and we also reuse the routines Encode and $\mathsf{Reconstruct}$ introduced in the previous section.

Again, we run two versions of the protocol, the first is for κ-valued messages and denoted as $\mathsf{ABA}(\kappa)$, the other for binary-valued messages, and denoted as $\mathsf{ABA}(1)$. Since the protocol in [3] is binary, we simply run it κ many times in parallel to agree on a κ bit message. As before, we choose the committees with expected size $c\kappa$. Note that in protocol [3], contrary to the synchronous case, the committees are not tied to a specific value.

Protocol Π_{sprABA}

Let $t_\kappa = \lfloor (1+\varepsilon) \cdot \frac{\kappa}{3} \rfloor$. The protocol is described from the point of view of party P_i who holds an l-bit input message m_i.

1: Compute $\mathcal{D}_i := (s_1, \ldots, s_\kappa) = \mathsf{Encode}(m_i)$, the accumulation value $z_i = \mathsf{Eval}(\mathsf{ak}, \mathcal{D}_i)$. Input z_i to $\mathsf{ABA}(\kappa)$.

2: When the above BA outputs z, if $z = z_i$ and $P_i \in C_1$, input 1 to $\mathsf{ABA}(1)$. Moreover, distribute the long block as follows. Compute a witness $w_j = \mathsf{CreateWit}(\mathsf{ak}, z, s_j)$ for each share s_j in Step 1 and send the tuple (s_j, w_j) to each party $P_k \in B_j$. Otherwise, if $z \neq z_i$ and $P_i \in C_1$, input 0 to $\mathsf{ABA}(1)$.

3: If the output of the above BA is 0, output $\perp$ and abort. Otherwise, if $P_i \in C^* \cap B_j$: For the set of tuples $\{(s_j, w_j)\}$ received in the previous step from parties in C_1, if there exists an (s_j, w_j) such that $\mathsf{Verify}(\mathsf{ak}, z, w_j, s_j) = 1$, then send (s_j, w_j) to all parties.

4: Let $\mathcal{S}_i := \{(s_j, w_j)\}$ be the set of messages received from the previous step from parties in C^*. If there are messages from parties belonging to at least $\frac{2\kappa}{3} \cdot (1-\varepsilon)$ different buckets, output the reconstructed value $\mathsf{Reconstruct}(\mathcal{S}_i, \mathsf{ak}, z, t_\kappa)$. Otherwise, output $\perp$.

We follow a very similar strategy as in the previous section. In our main theorem statement, we include the cryptographic setup required to run the protocol of Blum et al. [3] without going in to much details as to how they work. Roughly speaking, their protocol starts from an initial setup provided by a trusted dealer. This initial setup allows parties to run a fixed number of multi-party computations (MPCs) and BAs with subquadratic communication complexity. The parties use these cheap (in terms of communication) MPCs to emulate the trusted dealer and refresh the setup for future cheap MPCs and BAs for any number of times. To run MPC with these complexities, their protocol requires strong setup assumptions including threshold fully homomorphic encryption, non-interactive zero knowledge, and anonymous public key encryption (where a ciphertext can not be linked to a public key without knowing the secret key).

Theorem 4. *Let $0 < \varepsilon < 1/4$. Assuming a setup for non-interactive zero-knowledge, threshold fully homomorphic encryption, and anonymous public key encryptions, Π_{sprABA} is an asynchronous Byzantine* *agreement protocol secure up to $t \leq (1-6\varepsilon)n/3$ adaptive corruptions. The communication complexity is $O(nl + \kappa^6 n)$ for l-bit values.*

The proof of the following lemma is almost identical to that of Lemma 3.

Lemma 6. *If all honest parties in C_1 input b to $\mathsf{ABA}(1)$ then the output of $\mathsf{ABA}(1)$ is b.*

Lemma 7. *Π_{sprABA} satisfies validity if $t \leq (1-6\varepsilon)n/3$ parties are corrupted.*

Proof. If all honest parties have the same input message $m_i = m$, then all honest parties input the same accumulated value $z = z_i$ to $\mathsf{ABA}(\kappa)$ in Step 1. By validity of $\mathsf{ABA}(\kappa)$, all honest parties receive z as output. Hence, all honest parties in C_1 input 1 to $\mathsf{ABA}(1)$ in Step 2 and distribute the shares of m. By Lemma 6, they receive 1 as output from $\mathsf{ABA}(1)$.

Each honest party $P_j \in C^* \cap B_j$ receives a valid share s^i_j from each honest party $P_i \in C_1$, and forwards one of these shares to all parties. Parties are added to C^* uniformly at random via $\mathsf{ComProve}$ with probability $c\kappa/n$. Denote E_0 the event that fewer than $(1-\varepsilon)c\kappa$ parties are in C^*, for $\varepsilon > 0$.

Whenever E_0 does not occur, we can map the process of adding parties to C^* (via $\mathsf{ComProve}$) to the process of throwing $c\kappa$ or more balls at κ buckets. By Corollary 3, we have that $\mathbf{Pr}[E_0]$ is negligible. Moreover, the optimal strategy for the adversary to minimize the number of buckets in which an honest party sends a share is clearly to corrupt the buckets that contain smaller amount of parties from C^*.

Let us denote E_1 the event that $\frac{c\kappa}{3}(1-4\varepsilon)$ or more parties in C^* are corrupted. By Corollary 2, $\mathbf{Pr}[E_1]$ is negligible. Therefore, by a union bound, $\mathbf{Pr}[E_0 \cup E_1]$ is also negligible.

In the following, we condition on the event $\neg E_0 \wedge \neg E_1$ (which by the above occurs with overwhelming probability).

Let $c' = (1-\varepsilon)c$. By Corollary 4, and choosing $\tau = 1/3$, there is a constant $c'_{1/3}$ such that $\sum_{i=1}^{c'_{1/3}} i \cdot b_i \geq (1-\varepsilon)\cdot(1/3 - o_\kappa(1))\cdot c'\kappa \geq \frac{c'\kappa}{3}(1-2\varepsilon)$, where the last inequality holds as long as $o_\kappa(1) \leq \frac{\varepsilon}{3(1-\varepsilon)}$. Therefore, the adversary can not corrupt all committee members in the buckets that contain up to $c'_{1/3}$ many committee members. These amounts of buckets correspond to at most $\lfloor(1+\varepsilon)\kappa/3\rfloor$ buckets, by Corollary 4.

Putting things together, at Step 4, at least $\kappa - t_\kappa \geq (1-\varepsilon)\frac{2\kappa}{3}$ honest parties in C^* send a share, and thus every honest party receives at least that many valid shares. This way, all honest parties can reconstruct and output the long message m.

The proof of the following lemma is identical as for the synchronous case.

Lemma 8. *Π_{sprABA} satisfies consistency if $t \leq (1-6\varepsilon)n/3$ parties are corrupted.*

Communication complexity. The most expensive steps in the protocol are the run of $\mathsf{BA}(\kappa)$ in Step 1 (which itself consists of κ parallel runs of $\mathsf{BA}(1)$) and the distribution of the long blocks in Step 2. The costs for Step 1 are bounded as $O(\kappa^6 \cdot n)$ since every run of $\mathsf{BA}(1)$ costs $O(\kappa^5 \cdot n)$. The costs for Step 2 are bounded by $O(l \cdot n)$. Overall, we obtain a complexity of $O(n \cdot l + \kappa^6 \cdot n)$.

References

[1] Ittai Abraham, T.-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. In *38th ACM PODC*, pages 317–326. ACM, 2019.

[2] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *International conference on the theory and applications of cryptographic techniques*, pages 480–494. Springer, 1997.

[3] Erica Blum, Jonathan Katz, Chen-Da Liu-Zhang, and Julian Loss. Asynchronous byzantine agreement with subquadratic communication. In *TCC 2020, Part I*, LNCS, pages 353–380. Springer, Heidelberg, March 2020.

[4] Christian Cachin and Stefano Tessaro. Asynchronous verifiable information dispersal. In *IEEE Symposium on Reliable Distributed Systems (SRDS'05)*, pages 191–201. IEEE, 2005.

[5] T.-H. Hubert Chan, Rafael Pass, and Elaine Shi. Sublinear-round byzantine agreement under corrupt majority. LNCS, pages 246–265. Springer, Heidelberg, 2020.

[6] Jing Chen, Sergey Gorbunov, Silvio Micali, and Georgios Vlachos. ALGORAND AGREEMENT: Super fast and partition resilient byzantine agreement. Cryptology ePrint Archive, Report 2018/377, 2018. `https://eprint.iacr.org/2018/377`.

[7] Wutichai Chongchitmate and Rafail Ostrovsky. Information-theoretic broadcast with dishonest majority for long messages. In *TCC 2018, Part I*, LNCS, pages 370–388. Springer, Heidelberg, March 2018.

[8] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. In Robert L. Probert, Michael J. Fischer, and Nicola Santoro, editors, *1st ACM PODC*, pages 132–140. ACM, August 1982.

[9] Matthias Fitzi and Martin Hirt. Optimally efficient multi-valued Byzantine agreement. In Eric Ruppert and Dahlia Malkhi, editors, *25th ACM PODC*, pages 163–168. ACM, July 2006.

[10] Chaya Ganesh and Arpita Patra. Broadcast extensions with optimal communication and round complexity. In George Giakkoupis, editor, *35th ACM PODC*, pages 371–380. ACM, July 2016.

[11] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. Cryptology ePrint Archive, Report 2017/454, 2017. `http://eprint.iacr.org/2017/454`.

[12] Martin Hirt and Pavel Raykov. Multi-valued byzantine broadcast: The $t < n$ case. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 448–465. Springer, Heidelberg, December 2014.

[13] Valerie King and Jared Saia. From almost everywhere to everywhere. In *DISC*, 2009.

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

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

[16] Guanfeng Liang and Nitin Vaidya. Error-free multi-valued consensus with byzantine failures. In *In Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing.*

[17] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, October 1999.

[18] Atsuki Momose and Ling Ren. Optimal communication complexity of authenticated byzantine agreement. In *35th International Symposium on Distributed Computing*, 2021.

[19] Kartik Nayak, Ling Ren, Elaine Shi, Nitin H. Vaidya, and Zhuolun Xiang. Improved extension protocols for byzantine broadcast and agreement. In *DISC*, 2020.

[20] Lan Nguyen. Accumulators from bilinear pairings and applications. In *Cryptographers' track at the RSA conference*, pages 275–292. Springer, 2005.

[21] The DFINITY Team. The internet computer for geeks. Cryptology ePrint Archive, Report 2022/087, 2022. `https://ia.cr/2022/087`.

[22] Russell Turpin and Brian A. Coan. Extending binary byzantine agreement to multivalued byzantine agreement. *Inf. Process. Lett.*, 18(2):73–76, 1984.