Communication-Optimal Convex Agreement

DIANA GHINEA, ETH Zurich
CHEN-DA LIU-ZHANG, Luzern University of Applied Sciences and Arts & Web3 Foundation
ROGER WATTENHOFER, ETH Zurich

Byzantine Agreement (BA) allows a set of n parties to agree on a value even when up to t of the parties involved are corrupted. While previous works have shown that, for ℓ-bit inputs, BA can be achieved with the optimal communication complexity $O(\ell n)$ for sufficiently large ℓ, BA only ensures that honest parties agree on a meaningful output when they hold the same input, rendering the primitive inadequate for many real-world applications.

This gave rise to the notion of Convex Agreement (CA), introduced by Vaidya and Garg [PODC'13], which requires the honest parties' outputs to be in the convex hull of the honest inputs. Unfortunately, all existing CA protocols incur a communication complexity of at least $\Omega(\ell n^2)$. In this work, we introduce the first CA protocol with the optimal communication of $O(\ell n)$ bits for inputs in $\mathbb{Z}$ of size $\ell = \Omega(\kappa \cdot n^2 \log n)$, where κ is the security parameter.

CCS Concepts: • **Theory of computation → Cryptographic protocols**.

Additional Key Words and Phrases: convex agreement, optimal communication, long messages

1 INTRODUCTION

Reaching collaborative decisions becomes tricky in decentralized systems, especially when participants might be unreliable or even malicious. This is where agreement protocols come in, acting as crucial tools for finding common ground. One such primitive is Byzantine Agreement (BA), where a group of n parties agree on a value, even if up to t of the parties are byzantine.

The standard BA definition comes with certain limitations when applied to real-world scenarios. Consider, for instance, a network of sensors deployed within a cooling room, responsible for measuring and reporting the room's temperature. One can expect minor discrepancies in the measurements, such as correct sensors obtaining temperatures between $-10.05°C$ and $-10.03°C$. In such a scenario, standard BA allows the honest parties to agree on a value proposed by the byzantine parties, such as $+100°C$, instead of requiring the output to reflect the correct sensors' measurements.

A stronger variant of BA, known as Convex Agreement (CA), addresses this issue, as it requires the honest parties to agree on a value within the convex hull of their inputs (or within the range of their inputs, if the input space is uni-dimensional). The synchronous model, where parties have synchronized clocks and messages get delivered within a publicly known amount of time, facilitates a straightforward approach for achieving CA through Synchronous Broadcast (BC). Essentially, each party sends its input value via BC, which provides the parties with an identical view of the inputs. Afterwards, the parties decide on a common output by applying a deterministic function to the values received. While this approach yields optimal solutions in terms of resilience and round complexity, there is still a gap in terms of communication. Specifically, if the honest parties hold inputs of at most ℓ bits, a lower bound on the communication complexity is $\Omega(\ell n)$ bits [28], and this approach incurs a sub-optimal communication cost of $\Omega(\ell n^2)$ bits. For BA and BC, this gap was long closed in a beautiful line of works [4, 15, 16, 22, 28] via so-called *extension protocols*, that achieve a communication complexity of $O(\ell n + \mathsf{poly}(n, \kappa))$ bits, where κ is a security parameter. In this work, we focus on closing this gap in the synchronous model for CA. In this setting, we ask the following question:

Can we achieve CA with the asymptotically optimal communication of $O(\ell n + \mathsf{poly}(n, \kappa))$ *bits?*

We answer this question in the affirmative. More concretely, we introduce a deterministic protocol in the plain model (no setup) that achieves the optimal resilience $t < n/3$, optimal asymptotic communication complexity of $O(\ell n + \mathsf{poly}(n, \kappa))$ and round complexity $O(n \log n)$.[1] The protocol makes use of collision-resistant hash functions and takes as inputs ℓ-bit strings interpreted as integer values. This is without loss of generality and only used to establish an ordering between the inputs (one could alternatively interpret the inputs being rational numbers with some arbitrary pre-defined precision).

1.1 Related work

Convex-Hull Validity. The requirement of obtaining outputs within the honest inputs' range has been first introduced in [10] for Approximate Agreement (AA). AA relaxes the agreement requirement, allowing the parties' outputs to deviate by a predefined error $\varepsilon > 0$. While this relaxation allows for deterministic asynchronous protocols, circumventing the FLP result [14], it also has advantages in the synchronous model if n is $\Omega(\ell)$. Namely, the runtime of deterministic AA algorithms may only depend on ℓ instead of n, bypassing the $O(n)$ rounds requirement [11]. Such algorithms proceed in iterations, where each iteration involves a step where parties send a value to all parties, hence incurring a communication cost of $\Omega(\ell n^2)$ bits. AA has been a subject of an extensive line of works, focusing on optimal convergence rates [3, 12, 13], higher resilience thresholds both in asynchronous and synchronous networks [1, 17, 21], and different input spaces, such as multidimensional inputs [18, 25, 34], or abstract convexity spaces [2, 9, 20, 30].

CA was formally defined by Vaidya and Garg in [26, 34], assuming that the input space consists of multidimensional values. Feasibility with optimal resilience has been considered for abstract convexity spaces as well [9, 30]. Another line of works has investigated the feasibility of an even stronger requirement for inputs in $\mathbb{R}$ or $\mathbb{Z}$, i.e. that the output is *close* to the median of the honest inputs [8, 32], or, more generally, to the k-th lowest honest input [24].

Extension Protocols. The problem of reducing the communication complexity of BA on multi-valued inputs was first addressed by Turpin and Coan [33], where the authors assume $t < n/3$ and give a reduction from long-messages BA to short-messages BA with a communication cost of $\Omega(\ell n^2)$ bits. Fitzi and Hirt [15] later achieve BA in the honest majority setting with the asymptotically optimal communication complexity $O(\ell n + \mathsf{poly}(n, \kappa))$ bits, assuming a universal hash function. Further works have provided error-free solutions focusing on reducing the additional $\mathsf{poly}(n, \kappa)$ factor in the communication complexity both in the $t < n/3$ [16, 23, 28] setting and in the honest-majority setting [4, 16, 28].

Extension protocols have also been a topic of interest for problems related to BA, such as BC in the $t < n$ setting [6, 19], or asynchronous Reliable Broadcast [5, 28].

1.2 Comparison to previous works

In terms of techniques, our solution differs significantly from both prior works on BA extension protocols and prior works on CA or AA. In comparison to BA, the honest-range requirement of CA adds a new level of challenges when it comes to reducing the communication. Roughly, in prior works on communication-optimal BA, each party first computes a short κ-bit encoding of its long ℓ-bit input value (using e.g. a hash function). Afterwards, the parties agree on an encoding $z^\star$ using a BA protocol for short messages. Finally, parties holding the (unique) input value $v^\star$ matching the encoding $z^\star$ distribute $v^\star$ to all the parties in a non-trivial manner. The main issue when trying to adapt this approach to CA is that the short κ-bit encodings lost information about the ordering of

[1] With randomization, our protocol can be made to achieve $O(\kappa \log n) = \tilde{O}(1)$ rounds.

the original values, and in particular cannot reflect the honest inputs' range. On the other hand, existing protocols satisfying this validity requirement, regardless of whether they achieve CA or its weaker variant AA, involve some step where all parties send their ℓ-bit values to all other parties. It might seem intuitive that the parties need a possibly consistent or identical view over their actual values to decide on a valid output. However, we show that this intuition is not true.

Our protocol relies on a byzantine variant of the *longest common prefix* problem, and makes use of a BA protocol for short messages as a building block. The central insight behind our approach is that the longest common prefix of the honest parties' inputs represented as bitstrings reveals a subset of the honest inputs' range. While finding the exact longest common prefix of the honest inputs is impossible due to the byzantine parties involved, the longest common prefix of any values in the honest inputs' range will suffice to obtain an output.

2 PRELIMINARIES

We denote by κ the security parameter. We consider a setting with n parties $P_1, P_2, \ldots, P_n$ in a fully connected network, where each pair of parties is connected by an authenticated channel. We assume that the network is synchronous: the parties' clocks are synchronized and all messages get delivered within Δ time, where Δ is publicly known. We consider an adaptive adversary that can corrupt up to $t < n/3$ parties at any point in the protocol's execution, causing them to become byzantine: corrupted parties may deviate arbitrarily from the protocol. Our protocols make use of a collision-resistant hash function $H_\kappa : \{0,1\}^\star \rightarrow \{0,1\}^\kappa$, and we assume that the adversary is computationally bounded. For simplicity of presentation, our proofs will assume that H_κ is *collision-free*; our protocols are secure conditioned on the event that a collision occurs.

2.1 Binary representations

We need to establish a few notations and implicit remarks. For a value $v \in \mathbb{N}$, we define its binary representation $\textsc{bits}(v) := \textsc{b}_1\textsc{b}_2 \ldots \textsc{b}_k$ such that $2^{k-1} \leq v < 2^k$, $\textsc{b}_i \in \{0,1\}$ for every $1 \leq i \leq k$, and $\sum_{i=1}^{k} \textsc{b}_i \cdot 2^{k-i} = v$. For $\ell \geq k$, we additionally define $\textsc{bits}_\ell(v)$ as the ℓ-bit string obtained by prepending $\ell - k$ zeroes to $\textsc{bits}(v)$. We denote the length of a bitstring $\textsc{bits}$ by $\left|\textsc{bits}\right|$. The reverse operation will be $\textsc{val}(\textsc{bits})$: given a bitstring $\textsc{bits} := \textsc{b}_1\textsc{b}_2 \ldots \textsc{b}_k$ (where every $\textsc{b}_i \in \{0,1\}$), $\textsc{val}(\textsc{bits}) := \sum_{i=1}^{k} \textsc{b}_i \cdot 2^{k-i} = v$. For any $\ell \geq \left|\textsc{bits}(v)\right|$, $\textsc{val}(\textsc{bits}_\ell(v)) = v$. We also include the remark below, which we use implicitly in our proofs. Note that $||$ is the concatenation operator.

Remark 1. *Consider some bitstrings* $\textsc{prefix}$, $\textsc{bits}^1$ *and* $\textsc{bits}^2$, $\textsc{suffix}^1$ *and* $\textsc{suffix}^2$ *such that* $\left|\textsc{bits}^1\right| = \left|\textsc{bits}^2\right| > 0$, *and* $\left|\textsc{suffix}^1\right| = \left|\textsc{suffix}^2\right|$. *If* $\textsc{val}(\textsc{bits}^1) > \textsc{val}(\textsc{bits}^2)$, *then* $\textsc{val}(\textsc{prefix} || \textsc{bits}^1 || \textsc{suffix}^1) > \textsc{val}(\textsc{prefix} || \textsc{bits}^2 || \textsc{suffix}^2)$.

2.2 Definitions

We recall the definitions of CA and BA. We mention that, throughout the paper, we will use *valid value* to refer to a value satisfying Convex-Hull Validity (as opposed to the Validity definition of BA), as defined below.

Definition 1 (Convex Agreement). *Let* Π *be an* n*-party protocol where each party holds a value* v_{IN} *as input, and parties terminate upon generating an output* v_{OUT}. Π *achieves Convex Agreement if the following properties hold even when up to* t *of the parties involved are corrupted:*

- *(Termination) All honest parties terminate;*
- *(Convex-Hull Validity) Honest parties' outputs lie in the honest inputs' convex hull (i.e. range);*
- *(Agreement) All honest parties output the same value.*

Definition 2 (Byzantine Agreement)**.** *Let* Π *be an* n*-party protocol where each party holds a value* v_{IN} *as input, and parties terminate upon generating an output* v_{OUT}*.* Π *achieves Byzantine Agreement if the following properties hold even when up to* t *of the parties involved are corrupted:*

- *(Termination) All honest parties terminate;*
- *(Validity) If all honest parties hold the same input value* v*, they output* $v_{OUT} = v$*;*
- *(Agreement) All honest parties output the same value.*

For a BA or CA protocol Π, we use $\textsc{bits}_\ell(\Pi)$ to refer to its communication complexity for ℓ-bit inputs. That is, $\textsc{bits}_\ell(\Pi)$ denotes the worst-case total number of bits sent by honest parties, assuming that they hold inputs of at most ℓ bits. In addition, we denote the worst-case round complexity of Π by $\textsc{rounds}(\Pi)$ (note that this is independent of the inputs' length ℓ).

We also need to recall the definition of BC.

Definition 3 (Synchronous Broadcast)**.** *Let* Π *be a protocol where a designated party* S *(called the sender) holds a value* v_S*, and every party* P *terminates upon generating an output* v_{OUT}*. We say that* Π *achieves BC if the following properties hold even when* t *of the parties involved are corrupted:*

- *(Termination) All honest parties terminate;*
- *(Validity) If* S *is honest, every party outputs* $v_{OUT} = v_S$*;*
- *(Consistency) All honest parties output the same value.*

Similarly to BA and CA protocols, for a BC protocol Π, we use $\textsc{rounds}(\Pi)$ to denote its round complexity. For the bit complexity, we will distinguish between an honest sender and a byzantine sender. Hence, $\textsc{bits}_\ell(\Pi)$ denotes the worst-case bit complexity assuming that the sender is honest and its input consists of at most ℓ bits. For the worst-case bit complexity in the case of a byzantine sender, we use the notation $\textsc{bits}_{\textsc{byz}}(\Pi)$.

3 OVERVIEW

We provide an overview of our main protocol, outlining the main challenges and techniques. First, note that $O(\ell n + \text{poly}(n, k))$ bits do not allow for a step where the parties distribute their ℓ-bit values. Instead, we aim to only work with the prefixes of the values' ℓ-bit representations. In the following, and in our main protocol, we solely concentrate on interpreting the input bitstrings as values in $\mathbb{N}$. The extension to $\mathbb{Z}$ is explained in Section 6.

For intuition, it will be useful to arrange the honest inputs' range in a so-called *prefix tree* (or *trie*). As shown in Figure 1, a prefix tree is a (rooted) tree where each node stores a string's prefix. The edges from nodes to their children are labelled with characters (0 or 1) indicating the prefixes stored on the children. Note that CA requires the parties to find a leaf in this prefix tree.

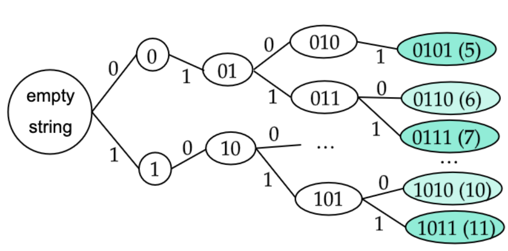

Fig. 1. Prefix tree storing the honest inputs' range, assuming that $\ell = 4$ and the honest inputs are 5, 7 and 11.

3.1 Inputs' length

The bit representations $\textsc{bits}(v_{\text{IN}})$ of the parties' inputs v_{IN} may be of different lengths, and, to compare prefixes effectively, the parties first agree on a common input length of ℓ_{EST} bits, such that every party can modify its input to a valid input of length ℓ_{EST}.

A straightforward approach for parties to estimate ℓ_{EST} is to simply run iterations, where at each iteration i, parties use a BA protocol to agree on whether their input is smaller than 2^i (which is within the convex hull of original inputs). If the output is 1, the parties that had a larger input modify their input to the value $2^i - 1$. This approach would require a number of rounds that is in the worst case proportional to the longest original input.

We provide instead an optimized mechanism that requires a number of rounds independent of the length of the original inputs and is based on BC. A challenge here is that the parties are not aware of what message length to expect, which enables the byzantine parties to blow up the communication complexity by sending very long messages via BC. We prevent this by providing the parties with (possibly distinct) limits on the inputs' lengths, and by requiring our BC protocol to achieve the property below. See Section 4.2 for details.

Definition 4. *(Limited Length) Let Π be a broadcast protocol. We say that Π achieves Limited Length if the following property holds: Assume that every party P holds a value $\textsc{length\_limit}$ such that, if S is honest, its input v_S satisfies $|\textsc{bits}(v_S)| \leq \textsc{length\_limit}$. Then, $\textsc{bits}_{\textsc{byz}}(\Pi) \leq \textsc{bits}_{\textsc{length\_limit}_{\max}}(\Pi)$, where $\textsc{length\_limit}_{\max}$ is the highest among the values $\textsc{length\_limit}$ held by honest parties.*

3.2 Warm-up

As a starting point towards our final solution, we describe a simple (yet inefficient) approach that finds a leaf in the prefix tree of the honest inputs' range using ℓ_{EST} iterations. In iteration i, the parties hold valid values v such that the bit representations $\textsc{bits}_{\ell_{\text{EST}}}(v)$ share a common prefix $\textsc{prefix}^\star$ of $i-1$ bits. The parties extend the common prefix with one bit with the help of a BA protocol Π_{BA}: they join Π_{BA} with input $\textsc{b}_i :=$ the i-th bit of $\textsc{bits}_{\ell_{\text{EST}}}(v)$ and agree on bit $\textsc{prefix}^\star_i$. Parties holding $\textsc{b}_i \neq \textsc{prefix}^\star_i$ need to update their value v to some *valid* value matching the prefix agreed upon. We know that $\textsc{prefix}^\star_i$ was proposed by an honest party, hence $\textsc{prefix} \,\|\, \textsc{prefix}^\star_i$ is the prefix of a valid ℓ_{EST}-bit value $v^\star$. This allows the parties to update their values as follows: if $\textsc{b}_i = 0$ and $\textsc{prefix}^\star_i = 1$, meaning that $v < v^\star$, then the lowest ℓ_{EST}-bit value having prefix $\textsc{prefix}^\star \,\|\, \textsc{prefix}^\star_i$ is in $[v, v^\star]$ and therefore is valid. Similarly, if $\textsc{b}_i = 1$ and $\textsc{prefix}^\star_i = 0$, meaning that $v > v^\star$, then the highest ℓ_{EST}-bit value having prefix $\textsc{prefix}^\star \,\|\, \textsc{prefix}^\star_i$ is in $[v^\star, v]$ and therefore is valid.

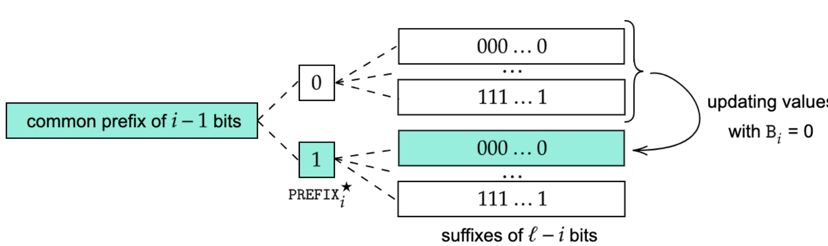

Fig. 2. In iteration i, the parties hold values with a common prefix of $i-1$ bits, and agree on the i-th bit $\textsc{prefix}^\star_i$. In this figure, $\textsc{prefix}^\star_i = 1$, and parties holding values with $\textsc{b}_i = 0$ update their values.

For a bitstring $\textsc{prefix}$ of at most ℓ bits, $\textsc{max}_\ell(\textsc{prefix})$ denotes the highest ℓ-bit value having $\textsc{prefix}$ as prefix (obtained by concatenating $\textsc{prefix}$ with $\ell - |\textsc{prefix}|$ ones). Similarly, $\textsc{min}_\ell(\textsc{prefix})$ denotes the lowest ℓ-bit value having $\textsc{prefix}$ as prefix (obtained by concatenating $\textsc{prefix}$ with

$\ell - |\text{PREFIX}|$ zeroes). The remark below then ensures that the update step indeed leads to valid values. The proof is included in Appendix A.

Remark 2. *Consider two values $v, v' \in \mathbb{N}$ satisfying $v \leq v' < 2^\ell$, and let COMMON_PREFIX denote the **longest** common prefix of $\text{BITS}_\ell(v)$ and $\text{BITS}_\ell(v')$.*
If $|\text{COMMON\_PREFIX}| < \ell$, then $\text{MAX}_\ell(\text{COMMON\_PREFIX} \,\|\, 0), \text{MIN}_\ell(\text{COMMON\_PREFIX} \,\|\, 1) \in [v, v']$.

This way, at the end of iteration ℓ_{EST}, CA is achieved: the parties hold valid ℓ_{EST}-bit values with a common prefix of ℓ_{EST} bits, and therefore they have agreed on a valid value.

3.3 From bits to blocks

Instead of building some valid values' prefix bit by bit, we may do so *block by block*. Assume without loss of generality that ℓ_{EST} is a multiple of n. Then, for $v \in \mathbb{N}$ satisfying $|\text{BITS}(v)| \leq \ell_{\text{EST}}$, we define $\text{BLOCKS}(v) := (\text{BLOCK}_1, \text{BLOCK}_2, \ldots, \text{BLOCK}_n)$ such that $\text{BITS}_{\ell_{\text{EST}}}(v) = \text{BLOCK}_1 \,\|\, \text{BLOCK}_2 \,\|\, \ldots \,\|\, \text{BLOCK}_n$, and, for any $1 \leq i \leq n$, $|\text{BLOCK}_i| = \ell_{\text{EST}}/n$. For $1 \leq i \leq n$, use $\text{BLOCK}_i(v)$ to refer to BLOCK_i. We will use the term *block* to refer to such sequences of ℓ_{EST}/n bits.

Following the outline of the warm-up approach, in iteration i, the parties hold valid ℓ_{EST}-bit values v having a common prefix $\text{PREFIX}^\star$ of $i-1$ blocks. In an attempt to extend $\text{PREFIX}^\star$ by one block $\text{PREFIX}_i^\star$, the parties join a BA protocol $\Pi_{\ell\text{BA}}$ (for long messages) with $\text{BLOCK}_i(v)$ as input.

When the parties agree on a block. If the parties agree on a block $\text{PREFIX}_i^\star$, the honest parties holding $\text{BLOCK}_i \neq \text{PREFIX}_i^\star$ should update their values v to match the prefix agreed upon. However, unless all honest parties hold $\text{BLOCK}_i = \text{PREFIX}_i^\star$, $\text{PREFIX}_i^\star$ may be a block proposed by a corrupted party, forcing the updated values outside the honest range. To prevent this, we make use of the special symbol $\bot$, and we require $\Pi_{\ell\text{BA}}$ to achieve an additional property, as defined below.

Definition 5. *No Corrupted Output: If honest parties output $v \neq \bot$, v is some honest party's input.*

If $\Pi_{\ell\text{BA}}$ satisfies No Corrupted Output and the parties agree on a block $\text{PREFIX}_i^\star$, then $\text{PREFIX}^\star \,\|\, \text{PREFIX}_i^\star$ is the prefix of an honest party's (valid) value. If a party P holds a value v with $\text{BLOCK}_i(v) \neq \text{PREFIX}_i^\star$, it updates its value to match the prefix agreed upon. If $\text{BLOCK}_i < \text{PREFIX}_i^\star$, then P updates its value as $v := \text{MIN}_{\ell_{\text{EST}}}(\text{PREFIX}^\star \,\|\, \text{PREFIX}_i^\star)$, and, if $\text{BLOCK}_i > \text{PREFIX}_i^\star$, P updates its value as $v := \text{MAX}_{\ell_{\text{EST}}}(\text{PREFIX}^\star \,\|\, \text{PREFIX}_i^\star)$. The result below is a more general version of Remark 2 and ensures that updated values indeed remain valid. The proof is included in Appendix A.

Remark 3. *Consider two values $v, v' \in \mathbb{N}$ such that $v, v' < 2^\ell$, and let COMMON_PREFIX denote the **longest** common prefix of $\text{BITS}_\ell(v)$ and $\text{BITS}_\ell(v')$. Let NEXT_BITS and $\text{NEXT\_BITS}'$ denote two non-empty bitstrings of equal length such that $\text{COMMON\_PREFIX} \,\|\, \text{NEXT\_BITS}$ is a prefix of $\text{BITS}_\ell(v)$, and $\text{COMMON\_PREFIX} \,\|\, \text{NEXT\_BITS}'$ is a prefix of $\text{BITS}_\ell(v')$. Then, if $\text{VAL}(\text{NEXT\_BITS}) < \text{VAL}(\text{NEXT\_BITS}')$, then $\text{MIN}_\ell(\text{COMMON\_PREFIX} \,\|\, \text{NEXT\_BITS}'), \text{MAX}_\ell(\text{COMMON\_PREFIX} \,\|\, \text{NEXT\_BITS}) \in [v, v']$.*

When the parties agree on $\bot$. If $\Pi_{\ell\text{BA}}$ returns $\bot$ in some iteration $i^\star \leq n$, honest parties hold different blocks $\text{BLOCK}_{i^\star}$. In fact, this means that we are very close to finding a valid output.

Looking at Figure 1, a crucial observation is that nodes that have two children, and hence that store valid values' longest common prefixes, reveal subsets of the honest inputs' range. For example, the node storing 01 indicates that the highest 4-bit value having prefix 010 (in this case, this is 5) and the lowest value having prefix 011 (namely, 6) are valid. This means that, once the parties identify some valid values' longest common prefix, they may immediately derive an output with the help of Remark 2. On the other hand, this property applies to valid values' longest common prefix in terms of bits, while, at this point, the parties are only aware of a longest common prefix in terms of blocks: some of the bits in block $i^\star$ may be common. We then enable honest parties to find two different valid values' prefixes, and hence a longest common prefix in terms of bits, by requiring $\Pi_{\ell\text{BA}}$ to achieve a second additional property, defined below. Note that, when $t < n/3$, this property is equivalent to requiring that at most t honest parties have the same input value.

Definition 6. $(t+1)$-Disagreement*: If the honest parties output* $\bot$*, then, for any value* v*, there are* $t+1$ *honest parties holding inputs* $v_{IN} \neq v$*.*

3.4 A round-efficient approach

Although the approach described so far already achieves our goal regarding communication complexity, the round complexity will be $O(n) \cdot \textsc{rounds}(\Pi_{\ell\text{BA}})$. We reduce the number of iterations from $O(n)$ to $O(\log n)$ (while maintaining the communication complexity) by employing *binary search*: the parties are looking for an index $i^\star$ such that, roughly, running $\Pi_{\ell\text{BA}}$ on valid values' prefixes of $i^\star$ blocks returns $\bot$, while $\Pi_{\ell\text{BA}}$ returns a non-$\bot$ output on valid values' prefixes of $i^\star - 1$ blocks. Then, we proceed as follows: in the first iteration, the parties check whether $\Pi_{\ell\text{BA}}$ returns $\bot$ on the first half of their blocks $\textsc{block}_1 \parallel \ldots \parallel \textsc{block}_{\textsc{mid}}$. If $\Pi_{\ell\text{BA}}$ returns $\bot$, $\textsc{mid}$ is an upper bound for $i^\star$, and we continue the search for $i^\star$ within the first half of the blocks $\textsc{block}_1, \ldots, \textsc{block}_{\textsc{mid}-1}$ in the next iteration, using an identical approach. Otherwise, if $\Pi_{\ell\text{BA}}$ returns a bitstring of $\textsc{mid}$ blocks $\textsc{prefix}^\star_1 \parallel \ldots \parallel \textsc{prefix}^\star_{\textsc{mid}}$, the parties update their values to match this prefix and use the same approach to find $i^\star$ within the second half of their updated values' blocks in the next iteration. After $O(\log n)$ iterations, either $\Pi_{\ell\text{BA}}$ never returned $\bot$ and the parties now hold identical values, or $i^\star$ is found.

While this approach is more efficient, it adds a couple of challenges for deciding on the final output once $i^\star$ is found. The values held by the honest parties at the end of the $O(\log n)$ iterations will indeed have a common prefix of $i^\star - 1$ blocks. However, as opposed to the $O(n)$-iterations approach, these values might have been updated, which prevents us from using the $(t+1)$-Disagreement property of $\Pi_{\ell\text{BA}}$ to immediately obtain an output. Instead, we need to make use of the values $v_\bot$ held in the last iteration where $\Pi_{\ell\text{BA}}$ has returned $\bot$: $(t+1)$-Disagreement holds for these values' prefixes of $i^\star$ blocks. Hence, we use the values v to derive some valid value's prefix of $i^\star$ blocks, denoted by $\textsc{prefix}^1$, and the $t+1$ honest parties that hold values $v_\bot$ not having $\textsc{prefix}^1$ as a prefix will *complain* by announcing the first mismatch between $\textsc{prefix}^1$ and the values $v_\bot$. Each of the honest complaints will lead to a valid value, enabling the parties to agree on a valid output. See Section 5 for details.

4 BUILDING BLOCKS

In Section 3, we have introduced a few additional properties which enable us to use BC and BA protocols as building blocks. In the following, we describe how these properties can be achieved.

4.1 Recap: BA for long messages

We describe an extension protocol for BA following the outline of prior works [4, 28]. We may remove the trusted setup assumption since we only focus on $t < n/3$ corruptions instead of an honest majority. We make use of Reed-Solomon (RS) codes [31], which allow each party to split its value into n codewords so that reconstructing the original value only requires $n-t$ of these n codewords. To enable the parties to detect corrupted codewords, and also to compress the parties' values, prior works [4, 28] make use of collision-free cryptographic accumulators [29]. Essentially, accumulators convert a set (in our case, the n codewords) into a κ-bit value and provide witnesses confirming the accumulated set's contents. For this task, we use Merkle Trees (MT) [27], which do not require a trusted dealer. We define RS codes and MT below.

Linear erasure-correcting codes. [31] We use standard RS codes with parameters $(n, n-t)$. This provides us with a deterministic algorithm RS.ENCODE(v), which takes a value v as input and converts it into n codewords $(s_1, \ldots, s_n)$ of $O(|\text{BITS}(v)|/n)$ bits each. The codewords s_i are elements of a Galois Field $\mathbb{F} = GF(2^a)$ with $n \leq 2^a - 1$. To reconstruct the original value, RS codes provide a decoding algorithm, RS.DECODE, which takes as input $n-t$ of the n codewords and returns the original value v. Any $n-t$ of the n codewords uniquely determine the original value v.

Merkle trees. [27] An MT is a balanced binary tree that enables us to compress a multiset of values into a κ-bit encoding, and to efficiently verify that some value belongs to the compressed multiset. Given a multiset $S = \{s_1, \ldots, s_n\}$, the MT is built bottom-up: starting with n leaves, where the i-th leaf stores $H_\kappa(s_i)$. Each non-leaf node stores $H_\kappa(h_{\text{LEFT}} \parallel h_{\text{RIGHT}})$, where h_{LEFT} and h_{RIGHT} are the hashes stored by the node's left and resp. right child. This way, the hash stored by the root represents the encoding of S. Given the root's hash z, one can prove that s_i belongs to the compressed multiset using a witness w_i of $O(\kappa \cdot \log n)$ bits. The witness w_i contains the hashes needed to verify the path from the i-th leaf to the root. Note that the collision-resistance assumption leads to different encodings for different multisets, and prevents the adversary from producing witnesses for values of its own choice. We will use MT.BUILD(S) to denote the (deterministic) algorithm that creates the MT for the given multiset S and returns the hash stored by the root z and the witnesses $w_1, w_2, \ldots w_n$. Afterwards, MT.VERIFY(z, i, s_i, w_i) returns `true` if w_i proves that $H_\kappa(s_i)$ is indeed stored on the i-th leaf of the MT with root hash z and `false` otherwise.

BA for long messages. [4, 28] We sketch the outline of existing BA protocols for long messages.

(1) Each party computes $s_1, \ldots, s_n :=$ RS.ENCODE(v_{IN}); $z, w_1, \ldots, w_n :=$ MT.BUILD$(\{s_1, \ldots s_n\})$.
(2) The parties agree on an encoding $z^\star$ with the help of a BA protocol for short messages. To ensure that $z^\star$ was proposed by an honest party (i.e., that the No Corrupted Output property holds) and therefore the value $v^\star$ behind $z^\star$ can be reconstructed, the parties join BA with input 1 if $z = z^\star$ and 0 otherwise.
(3) If the bit agreed upon is 0, the parties output $\perp$. Otherwise, every party $P^\star$ holding $z = z^\star$ distributes $v^\star := v_{\text{IN}}$ to all the parties. To achieve this using only $O(\ell n + \text{poly}(n, \kappa))$ bits, $P^\star$ sends s_i and its MT witness w_i to each party P_i. The MT witnesses allow the parties to detect and discard any corrupted codewords. In addition, RS codes are deterministic, so each party P_i obtains a unique codeword s_i from RS.ENCODE$(v^\star)$. Every party P_i then sends (s_i, w_i) to all parties, which allows the parties to reconstruct $v^\star$.

This leads to the result below. We include the formal presentation and analysis in Appendix B.1.

THEOREM 1 ([28]). *Given a BA protocol Π_{BA} secure against $t < n/3$ corruptions, there is a protocol $\Pi_{\ell BA}$ achieving BA secure against $t < n/3$ corruptions with communication complexity BITS$_\ell(\Pi_{\ell BA}) = O(\ell n + \kappa \cdot n^2 \log n) + O(1) \cdot$ BITS$_\kappa(\Pi_{BA})$, and round complexity ROUNDS$(\Pi_{\ell BA}) = O(1) \cdot$ ROUNDS(Π_{BA}).*

4.2 Limited Length Synchronous Broadcast

The lemma below explains how the Limited Length property can be achieved with the help of BA.

Lemma 1. *Given a BA protocol Π_{BA} resilient against $t < n/3$ corruptions, there is a BC protocol Π_{BC+} resilient against $t < n/3$ corruptions that achieves Limited Length, with communication complexity BITS$_\ell(\Pi_{BC+}) = O(\ell n) +$ BITS$_\ell(\Pi_{BA})$ and round complexity ROUNDS$(\Pi_{BC+}) = O(1) +$ ROUNDS(Π_{BA}).*

PROOF. In $\Pi_{\text{BC+}}$, the sender S sends its value v_S to all parties (hence, it sends $O(\ell n)$ bits). Afterwards, each party joins Π_{BA}: with input v if the value v received from S satisfies $|\text{BITS}(v)| \leq$ LENGTH_LIMIT, and with input $\perp$ otherwise. Then, the parties output the value returned by Π_{BA}. The round complexity is therefore ROUNDS$(\Pi_{\text{BC+}}) = O(1) +$ ROUNDS(Π_{BA}).

If S is honest, the precondition on the values $\textsc{length\_limit}$ (in Definition 4) ensures that v_S satisfies $\left|\textsc{bits}(v_S)\right| \leq \textsc{length\_limit}$ for every honest party. Then, all honest parties join Π_{BA} with v_S as input and agree on v_S, hence Validity holds. The total communication cost is $\textsc{bits}_\ell(\Pi_{\mathrm{BC+}}) = O(\ell n) + \textsc{bits}_\ell(\Pi_{\mathrm{BA}})$. Otherwise, if S is corrupted, every honest party joins Π_{BA} with an input of at most $\textsc{length\_limit}_{\max}$ bits. Then, the parties obtain the same output in Π_{BA}, which ensures Consistency, and a total communication cost $\textsc{bits}_{\mathrm{BYZ}}(\Pi_{\mathrm{BC+}}) \leq \textsc{bits}_{\textsc{length\_limit}_{\max}}(\Pi_{\mathrm{BA}})$. □

Theorem 1 and Lemma 1 imply the following result for long messages.

Corollary 1. *Given a BA protocol Π_{BA} secure against $t < n/3$ corruptions, there is a BC protocol $\Pi_{\ell BC+}$ secure against $t < n/3$ corruptions that achieves Limited Length, with communication complexity $\textsc{bits}_\ell(\Pi_{\ell BC+}) = O(\ell n + \kappa \cdot n^2 \log n) + O(1) \cdot \textsc{bits}_\kappa(\Pi_{BA})$ and round complexity $\textsc{rounds}(\Pi_{\ell BC+}) = O(1) \cdot \textsc{rounds}(\Pi_{BA})$.*

4.3 BA with additional properties

To obtain a BA protocol for long messages that achieves *No Corrupted Output* and *$(t+1)$-Disagreement*, we only need the underlying BA protocol for short messages to achieve these properties. We design such a protocol with the help of the BC protocol $\Pi_{\mathrm{BC+}}$ described in Lemma 1. Every party distributes its input z via $\Pi_{\mathrm{BC+}}$, where all parties set $\textsc{length\_limit} := \kappa$. Then, the parties receive the same (at least $n - t$) values z. If a value $z^\star$ was sent by $t + 1$ parties, this will be the value agreed upon. Otherwise, the parties output $\perp$.

Protocol $\Pi_{\mathrm{BA+}}$

Code for party P with input z

1: Send z to all the parties via $\Pi_{\mathrm{BC+}}$. (Set $\textsc{length\_limit} := \kappa$ in every $\Pi_{\mathrm{BC+}}$ invocation).
2: If there is no value z received from $t + 1$ parties, output $\perp$.
3: Otherwise, output $z^\star :=$ the lowest value z received from $t + 1$ parties.

$\Pi_{\mathrm{BA+}}$ achieves BA with the additional properties No Corrupted Output and $(t + 1)$-Disagreement when $t < n/3$. We may then replace Step (2) of the sketch included in Section 4.1 as follows: the parties join $\Pi_{\mathrm{BA+}}$. If $\Pi_{\mathrm{BA+}}$ returns $z^\star \neq \perp$, then $z^\star$ was proposed by an honest party, which is enough for $v^\star$ to be correctly distributed in Step 3. Otherwise, if $\Pi_{\mathrm{BA+}}$ returns $\perp$, the parties output $\perp$. This way, if the parties output $\perp$ in the long-messages protocol, then $\Pi_{\mathrm{BA+}}$ guarantees that there was no set of $t + 1$ honest parties holding the same encoding $z^\star$, and therefore no $t + 1$ honest parties held the same input value v_{IN}. Hence, the $(t + 1)$-Disagreement property is maintained. If the parties output a non-$\perp$ value, then $\Pi_{\mathrm{BA+}}$ guarantees that the encoding $z^\star$ that led to this output was proposed by an honest party, and therefore the No Corrupted Output property is also maintained. This leads us to the result below. For the formal presentation and proofs, see Appendix B.2.

Corollary 2. *Given a BA protocol Π_{BA} resilient against $t < n/3$ corruptions, there is a BA protocol $\Pi_{\ell BA+}$ resilient against $t < n/3$ corruptions that additionally achieves No Corrupted Output and $(t+1)$-Disagreement. The communication complexity of $\Pi_{\ell BA+}$ is $\textsc{bits}_\ell(\Pi_{\ell BA+}) = O(\ell n + \kappa \cdot n^2 \log n) + O(n) \cdot \textsc{bits}_\kappa(\Pi_{BA})$, and the round complexity is $\textsc{rounds}(\Pi_{\ell BA+}) = O(1) + \textsc{rounds}(\Pi_{BA})$.*

5 PROTOCOL FOR $\mathbb{N}$

We are now ready to present our protocol $\Pi_{\mathbb{N}}$ achieving CA on natural numbers with communication complexity $O(\ell n + \mathrm{poly}(n, \kappa))$ and round complexity $O(n \log n)$. In the following, we assume a BA protocol Π_{BA}, and we make use of the building blocks presented in Section 4: the BA protocol

$\Pi_{\ell\mathrm{BA}+}$ that also achieves *No Corrupted Output* and $(t+1)$*-Disagreement*, described in Corollary 2, and the BC protocol $\Pi_{\ell\mathrm{BC}+}$ that also achieves *Limited Length*, described in Corollary 1.

5.1 Estimating the inputs' length

The parties first estimate their inputs' length with the help of the subprotocol Estimate. In this subprotocol, the parties agree on a length ℓ_{EST} (that is a multiple of n) satisfying $\ell_{\min} \leq \ell_{\text{EST}} \leq \lceil \ell_{\max}/n \rceil \cdot n$. In addition, each party obtains a valid value v such that $|\text{BITS}(v)| \leq \ell_{\text{EST}}$.

As the parties are not yet aware of what message length to expect, we first allow each party to learn a LENGTH_LIMIT: every party sends $\lceil |\text{BITS}(v_{\text{IN}})|/n \rceil$ to all parties, which takes $O(n \cdot \log \ell)$ bits. Then, every party receives $n-t+k$ such values, out of which at most k are sent by corrupted parties. The lemma below (proven in Appendix C.1) ensures that the multiset SAFE_VALUES obtained by discarding the lowest k and the highest k values received is included in the range of values $\lceil |\text{BITS}(v_{\text{IN}})|/n \rceil$ sent by the honest parties.

Lemma 2. *Let RECEIVED_VALUES denote a multiset of $n-t+k$ values, where $0 \leq k \leq t$, and let HONEST_VALUES $\subseteq$ RECEIVED_VALUES denote a multiset of $n-t$ values. Then, if SAFE_VALUES is a multiset obtained by discarding the lowest k and highest k values in RECEIVED_VALUES, it holds that $|\text{SAFE\_VALUES}| \geq t+1$, and SAFE_VALUES $\subseteq$ [min HONEST_VALUES, max HONEST_VALUES].*

Note that both $n \cdot \min$ SAFE_VALUES and $n \cdot \max$ SAFE_VALUES are good candidates for ℓ_{EST}. To agree on ℓ_{EST}, each party distributes $l_{\min} := \min$ SAFE_VALUES through the BC protocol $\Pi_{\mathrm{BC}+}$ described in Corollary 1. Each party sets LENGTH_LIMIT $:= \lceil \log_2 \max \text{SAFE\_VALUES} \rceil + 1$ for these invocations: one can show that the multisets SAFE_VALUES obtained by honest parties pair-wise intersect, which implies that honest values $l_{\min}$ satisfy these limits. The parties then receive the same $n-t+k$ values $l_{\min}$, out of which k come from corrupted parties, and Lemma 2 ensures that the $(k+1)$-th lowest value $l_{\min}$ received, denoted by l_{EST}, is in the range of honest values $l_{\min}$. Parties then set $\ell_{\text{EST}} := n \cdot l_{\text{EST}}$. The communication cost of this step is at most $O(n) \cdot \text{BITS}_{\lceil \log_2 \lceil \ell_{\max}/n \rceil \rceil + 1}(\Pi_{\mathrm{BC}+})$.

Estimate(v_{IN})

Code for party P with input v_{IN}

1: Send $l := \lceil |\text{BITS}(v_{\text{IN}})|/n \rceil$ to all parties.
2: Out of the $n-t+k$ values received, discard the lowest k and the highest k values. Let SAFE_VALUES be the multiset containing the remaining values, and let $l_{\min} := \min$ SAFE_VALUES and $l_{\max} :=$ max SAFE_VALUES.
3: Send $l_{\min}$ to all parties via $\Pi_{\ell\mathrm{BC}+}$. Join each $\Pi_{\ell\mathrm{BC}+}$ invocation with LENGTH_LIMIT $:= \lceil \log_2 l_{\max} \rceil + 1$.
4: Out of the $n-t+k$ values $l_{\min}$ received, set $l_{\text{EST}} :=$ the $(k+1)$-th lowest value, and $\ell_{\text{EST}} := l_{\text{EST}} \cdot n$.
5: Set $v := v_{\text{IN}}$ if $|\text{BITS}(v_{\text{IN}})| \leq \ell_{\text{EST}}$ and $v := 2^{\ell_{\text{EST}}} - 1$ otherwise. Output ℓ_{EST}, v.

The lemma below states the guarantees of Estimate. The proof is included in Appendix C.1.

Lemma 3. *Assume a BA protocol Π_{BA} resilient against $t < n/3$ corruptions, and let $\ell_{\min}$ and $\ell_{\max}$ denote the lowest and resp. the highest lengths $|\text{BITS}(v_{IN})|$ of the honest inputs v_{IN}. Then, in Estimate, honest parties agree on a value ℓ_{EST} that is a multiple of n and satisfies $\ell_{\min} \leq \ell_{EST} \leq \lceil \ell_{\max}/n \rceil \cdot n$. In addition, every honest party obtains a valid ℓ_{EST}-bit value v. Estimate achieves communication complexity $\text{BITS}_\ell(\text{Estimate}) = O(\ell n + k \cdot n^3 \log n) + O(n) \cdot \text{BITS}_\kappa(\Pi_{BA})$, and round complexity $\text{ROUNDS}(\text{Estimate}) = O(1) \cdot \text{ROUNDS}(\Pi_{BA})$.*

5.2 Finding some valid values' longest common prefix

As described in Section 3, the parties' goal is to determine some valid values' longest common prefix. In our implementation, the honest parties try to find the longest common prefix of their ℓ_{EST}-bit values v. To achieve this, the parties first look for a longest common prefix *of blocks* with the help of a subprotocol BlocksLCP. In this subprotocol, the parties agree on an index $1 \leq i^\star \leq n+1$, and each party obtains valid values v and $v_\perp$ with the following properties:

- The ℓ_{EST}-bit representations of the honest parties' values v share a prefix of $i^\star - 1$ blocks.
- For any bitstring BITS of $i^\star$ blocks, there are $t+1$ honest parties holding values $v_\perp$ whose ℓ_{EST}-bit representations do not have BITS as a prefix.

If $i^\star > n$, CA is already achieved. Otherwise, the parties run a subprotocol AddLastBlock, where they append one block to the common prefix of their values v, with the guarantee that the prefix obtained is still a valid value's prefix. This way, there are $t+1$ honest parties holding valid values $v_\perp$ that do not match this prefix. The parties then run a third subprotocol Complain, where the parties announce where their values $v_\perp$ differ from the prefix of $i^\star$ blocks agreed upon, which enables the honest parties to obtain an output.

Valid values' longest common prefix of blocks. We zoom in on the subprotocol BlocksLCP, where the parties try to find the longest common prefix of their values v *in terms of blocks*, denoted by $\text{PREFIX}^\star$. The parties initially set $\text{PREFIX}^\star$ to an empty string. Then, they run $O(\log n)$ iterations where each party holds a pair of valid values v and $v_\perp$, and the parties compare pieces of their values v to decide whether and how $\text{PREFIX}^\star$ should be extended. In each iteration, the parties only focus on a substring of their values v, namely blocks $\text{BLOCK}_{\text{LEFT}}(v), \ldots, \text{BLOCK}_{\text{RIGHT}-1}(v)$, where the indices LEFT and RIGHT satisfy $1 \leq \text{LEFT} \leq \text{RIGHT} \leq n+1$ and have the following meaning:

- $\text{PREFIX}^\star$ consists of $\text{LEFT} - 1$ blocks and is a common prefix of the honest parties' values v.
- Roughly, the honest parties' initial values v did not have a common prefix of RIGHT blocks. More precisely, the honest parties hold valid values $v_\perp$ such that, for any bitstring BITS of RIGHT blocks, the values $v_\perp$ of $t+1$ honest parties do not have prefix BITS.

While $\text{LEFT} < \text{RIGHT}$ holds, the parties check if the first half of this substring, namely blocks $\text{BLOCK}_{\text{LEFT}}(v), \ldots, \text{BLOCK}_{\text{MID}}(v)$ where $\text{MID} := \lfloor(\text{LEFT} + \text{RIGHT})/2\rfloor$, should extend the current common prefix $\text{PREFIX}^\star$ of their values v (consisting of $\text{LEFT} - 1$ blocks). They do so by joining $\Pi_{\ell\text{BA}+}$ with inputs $\text{BLOCK}_{\text{LEFT}}(v) \parallel \ldots \parallel \text{BLOCK}_{\text{MID}}(v)$. If $\Pi_{\ell\text{BA}+}$ returns $\perp$, then, for any bitstring of MID blocks, there are $t+1$ honest parties whose values v do not have that bitstring as a prefix. Then, the parties set $v_\perp := v$ and continue the search within blocks $\text{BLOCK}_{\text{LEFT}}(v), \ldots, \text{BLOCK}_{\text{MID}-1}(v)$ in the next iteration. Otherwise, if $\Pi_{\ell\text{BA}+}$ returns $\text{MID} - \text{LEFT} + 1$ blocks $\text{PREFIX}^\star_{\text{LEFT}} \parallel \ldots \parallel \text{PREFIX}^\star_{\text{MID}}$, the parties update their values v to match the prefix agreed upon: No Corrupted Output ensures that there is some valid value having prefix $\text{PREFIX}^\star \parallel \text{PREFIX}^\star_{\text{LEFT}} \parallel \ldots \parallel \text{PREFIX}^\star_{\text{MID}}$. Then, if a party holds v with a lower prefix, it sets v to $\text{MIN}_{\ell_{\text{EST}}}(\text{PREFIX}^\star \parallel \text{PREFIX}^\star_{\text{LEFT}} \parallel \ldots \parallel \text{PREFIX}^\star_{\text{MID}})$. Otherwise, if v has a higher prefix, the party sets v to $\text{MAX}_{\ell_{\text{EST}}}(\text{PREFIX}^\star \parallel \text{PREFIX}^\star_{\text{LEFT}} \parallel \ldots \parallel \text{PREFIX}^\star_{\text{MID}})$. Remark 3 ensures that the updated values are indeed valid. Afterwards, the parties continue the search in the next iteration within blocks $\text{BLOCK}_{\text{MID}+1}(v), \ldots, \text{BLOCK}_{\text{RIGHT}-1}(v)$ of the updated values v.

The size of the sequence considered, namely $\text{RIGHT} - \text{LEFT}$, gets halved in each iteration. The stopping condition is $\text{LEFT} = \text{RIGHT}$, which enables us to set $i^\star := \text{LEFT}$: the parties hold valid values v with a common prefix of $i^\star - 1$ blocks, and valid values $v_\perp$ such that, for any bitstring of $i^\star$ blocks, the values $v_\perp$ of $t+1$ honest parties do not have this bitstring as a prefix.

$\text{BLOCKSLCP}(\ell_{\text{EST}}, v)$

Code for party P

$\text{LEFT} := 1, \text{RIGHT} := n+1$; $v := v_{\text{IN}}$, $v_\perp := v_{\text{IN}}$, $\text{PREFIX}^\star :=$ empty string.
loop
    If $\text{LEFT} = \text{RIGHT}$, set $i^\star := \text{LEFT}$ and exit the loop.
    $(\text{BLOCK}_1, \text{BLOCK}_2, \ldots, \text{BLOCK}_n) := \text{BLOCKS}(v)$.
    Join $\Pi_{\ell\text{BA+}}$ with input $\text{BLOCK}_{\text{LEFT}} \| \ldots \| \text{BLOCK}_{\text{MID}}$, where $\text{MID} := \lfloor(\text{LEFT} + \text{RIGHT})/2\rfloor$.
    If $\Pi_{\ell\text{BA+}}$ has returned $\perp$, set $v_\perp := v$ and $\text{RIGHT} := \text{MID}$.
    Otherwise, if $\Pi_{\ell\text{BA+}}$ has returned $\text{MID} - \text{LEFT} + 1$ blocks $\text{PREFIX}^\star_{\text{LEFT}} \| \ldots \| \text{PREFIX}^\star_{\text{MID}}$:
        $\text{PREFIX}^\star := \text{PREFIX}^\star \| \text{PREFIX}^\star_{\text{LEFT}} \| \ldots \| \text{PREFIX}^\star_{\text{MID}}$.
        If $\text{VAL}(\text{BLOCK}_1 \| \ldots \| \text{BLOCK}_{\text{MID}}) < \text{VAL}(\text{PREFIX}^\star)$: $v := \text{MIN}_{\ell_{\text{EST}}}(\text{PREFIX}^\star)$.
        If $\text{VAL}(\text{BLOCK}_1 \| \ldots \| \text{BLOCK}_{\text{MID}}) > \text{VAL}(\text{PREFIX}^\star)$: $v := \text{MAX}_{\ell_{\text{EST}}}(\text{PREFIX}^\star)$.
        Set $\text{LEFT} := \text{MID} + 1$.
end loop
Return $i^\star, v, v_\perp$.

We state the guarantees of BLOCKSLCP below. We highlight the main details of the proof, and we defer the formal analysis to Appendix C.2.

Lemma 4. *Assume a BA protocol Π_{BA}, and that the honest parties join BLOCKSLCP with the same value ℓ_{EST} (that is a multiple of n) and valid ℓ_{EST}-bit values v. Then, the honest parties obtain the same index $i^\star$, and each honest party obtains a pair of valid ℓ_{EST}-bit values $v, v_\perp$ such that:*

- *the ℓ_{EST}-bit representations of the values v have a common prefix of $i^\star - 1$ blocks;*
- *for any bitstring BITS of $i^\star$ blocks, there are $t+1$ honest parties holding values $v_\perp$ such that $\text{BITS}_{\ell_{EST}}(v_\perp)$ does not have prefix BITS.*

BLOCKSLCP has communication complexity $\text{BITS}_{\ell_{EST}}(\text{BLOCKSLCP}) = O(\ell_{EST} \cdot n + \kappa \cdot n^2 \log^2 n) + O(n \log n) \cdot \text{BITS}_\kappa(\Pi_{BA})$ and round complexity $\text{ROUNDS}(\text{BLOCKSLCP}) = O(\log n) \cdot \text{ROUNDS}(\Pi_{BA})$.

PROOF SKETCH. We consider the properties below. If these properties are satisfied at the beginning of iteration $i \geq 1$ of the loop, then either the stopping condition $\text{LEFT} = \text{RIGHT}$ is met, or the properties hold at the beginning of iteration $i+1$ as well. We also note that these properties hold at the beginning of iteration 1 due to the variables' initialization.

(A) All honest parties hold the same indices $1 \leq \text{LEFT} \leq \text{RIGHT} \leq n+1$, and the same bitstring $\text{PREFIX}^\star$ consisting of $\text{LEFT} - 1$ blocks.
(B) $0 \leq \text{RIGHT} - \text{LEFT} \leq 2^{\lceil \log_2 n \rceil - (i-1)}$.
(C) Honest parties hold valid ℓ_{EST}-bit values v such that $\text{BITS}_{\ell_{\text{EST}}}(v)$ has $\text{PREFIX}^\star$ as a prefix.
(D) Honest parties hold valid ℓ_{EST}-bit values $v_\perp$, and, for any bitstring BITS of RIGHT blocks, the ℓ_{EST}-bit representations of the values $v_\perp$ of $t+1$ honest parties do not have prefix BITS.

Property (B) implies that the condition $\text{LEFT} = \text{RIGHT}$ is met by iteration $i = \lceil \log_2 n \rceil + 2$. Then, due to Properties (A), (C) and (D), setting $i^\star := \text{LEFT}$ ensures the guarantees on $i^\star$, v, and $v_\perp$ given in the lemma's statement. We still need to discuss the round complexity and the communication complexity. Since each of the at most $O(\log n)$ iterations invokes $\Pi_{\ell\text{BA+}}$ once, $\text{ROUNDS}(\text{BLOCKSLCP}) = O(\log n) \cdot \text{ROUNDS}(\Pi_{\ell\text{BA+}})$, and applying Corollary 2 gives our claimed round complexity. In each iteration $i < \lceil \log_2 n \rceil + 2$, Property (B) ensures that BLOCKSLCP runs $\Pi_{\ell\text{BA+}}$ on a bitstring of at most $2^{\lceil \log_2 n \rceil - i}$ blocks, hence $2^{\lceil \log_2 n \rceil - i} \cdot \ell_{\text{EST}}/n \leq \ell_{\text{EST}}/2^{i-1}$ bits. Therefore, $\text{BITS}_{\ell_{\text{EST}}}(\text{BLOCKSLCP}) = \sum_{i=1}^{\lceil \log_2 n \rceil + 1} \text{BITS}_{\ell_{\text{EST}}/2^{i-1}}(\Pi_{\ell\text{BA+}})$. Using Corollary 2 and the fact that $\sum_{i=0}^{\infty} 1/2^i \leq 2$, we obtain that $\text{BITS}_{\ell_{\text{EST}}}(\text{BLOCKSLCP}) = O(\ell_{\text{EST}} \cdot n + \kappa \cdot n^2 \log^2 n) + O(n \log n) \cdot \text{BITS}_\kappa(\Pi_{\text{BA}})$. □

Valid values' longest common prefix of bits. If BlocksLCP has returned $i^\star > n$, the honest parties hold the same valid value v; therefore, CA is already achieved. Otherwise, we continue our search for some valid values' longest common prefix. As mentioned in Section 3, the parties use the values v returned by BlocksLCP to obtain a valid value's prefix of $i^\star$ blocks, denoted by PREFIX^1. This way, there will be $t+1$ honest parties whose values $v_\perp$ do not have PREFIX^1 as a prefix and that announce these differences, which afterwards enables the parties to agree on a valid output.

Hence, if $i^\star \leq n$, the parties run the subprotocol AddLastBlock, where they compute PREFIX^1: as the first $i^\star - 1$ blocks of their v (which are identical), plus one block $\text{PREFIX}^1_{i^\star}$. As ensured by the remark below, it will be sufficient if the parties agree on some block $\text{PREFIX}^1_{i^\star}$ such that $\text{VAL}(\text{PREFIX}^1_{i^\star})$ is within the range of values $\text{VAL}(\text{BLOCK}^\star_i(v))$ of the honest values v.

Remark 4. *Let $v, v' \in \mathbb{N}$ denote two values satisfying $v \leq v' < 2^{\ell_{EST}}$, and assume that v and v' have a common prefix $\text{PREFIX}^\star$ of $i^\star - 1$ blocks, but not of $i^\star$ blocks: $\forall i < i^\star : \text{BLOCK}_i(v) = \text{BLOCK}_i(v')$, but $\text{BLOCK}_{i^\star}(v) \neq \text{BLOCK}_{i^\star}(v')$. Then, for any block BLOCK satisfying $\text{VAL}(\text{BLOCK}_{i^\star}(v)) \leq \text{VAL}(\text{BLOCK}) \leq \text{VAL}(\text{BLOCK}_{i^\star}(v'))$, there is a value $v^\star \in [v, v']$ such that $\text{BITS}_{\ell_{EST}}(v^\star)$ has prefix $\text{PREFIX}^\star \parallel \text{BLOCK}$.*

To agree on $\text{PREFIX}^1_{i^\star}$, every party sends $\text{BLOCK}_{i^\star}(v)$ to all the parties using the protocol $\Pi_{\ell\text{BC+}}$ of Corollary 1 (where parties join with $\text{LENGTH\_LIMIT} = \ell_{\text{EST}}/n$). The parties receive the same $n - t + k$ blocks' values, out of which k are sent by byzantine parties. Then, Lemma 2 implies that the $(k+1)$-th lowest value received, denoted by SAFE_BLOCK_VAL, is within the range of honest values $\text{VAL}(\text{BLOCK}_{i^\star}(v))$, and therefore parties may set $\text{PREFIX}^1_{i^\star} := \text{BITS}_{\ell_{\text{EST}}/n}(\text{SAFE\_BLOCK\_VAL})$.

AddLastBlock($\ell_{\text{EST}}, i^\star, v$)

Code for party P

1: Send $\text{VAL}(\text{BLOCK}_{i^\star}(v))$ to all parties via $\Pi_{\ell\text{BC+}}$. (Join all invocations with $\text{LENGTH\_LIMIT} = \ell_{\text{EST}}/n$).
2: Let SAFE_BLOCK_VAL := the $(k+1)$-th lowest out of the $n - t + k$ values received.
3: Return $\text{PREFIX}^1 := \text{BLOCK}_1(v) \parallel \ldots \parallel \text{BLOCK}_{i^\star - 1}(v) \parallel \text{BITS}_{\ell_{\text{EST}}/n}(\text{SAFE\_BLOCK\_VAL})$.

The proof of the result below is included in Appendix C.3.

Lemma 5. *Assume a BA protocol Π_{BA}, and that honest parties join AddLastBlock with the same value ℓ_{EST} (that is a multiple of n), with the same index $1 \leq i^\star \leq n$, and with valid ℓ_{EST}-bit values v that have a common prefix of $i^\star - 1$ blocks. Then, honest parties agree on a bitstring PREFIX^1 of $i^\star$ blocks such that there is a valid value v^1 whose ℓ_{EST}-bit representation has prefix PREFIX^1.*

AddLastBlock has communication complexity $\text{BITS}_{\ell_{EST}}(\text{AddLastBlock}) = O(\ell_{EST} \cdot n + k \cdot n^3 \log n) + O(n) \cdot \text{BITS}_\kappa(\Pi_{BA})$ and round complexity $\text{ROUNDS}(\text{AddLastBlock}) = O(1) \cdot \text{ROUNDS}(\Pi_{BA})$.

Once PREFIX^1 is obtained, the parties run the subprotocol Complain. In this subprotocol, the parties' goal is to find a bitstring PREFIX^2 that is the prefix of a valid ℓ_{EST}-bit value v^2, such that PREFIX^1 and PREFIX^2 are not prefixes of one another. This way, Remark 2 enables us to obtain a valid output from the longest common prefix of PREFIX^1 and PREFIX^2, which is the longest common prefix of $\text{BITS}_{\ell_{\text{EST}}}(v^1)$ and $\text{BITS}_{\ell_{\text{EST}}}(v^2)$. Honest parties' values $v_\perp$ that do not have prefix PREFIX^1 are candidates for v^2, and their prefixes are candidates for PREFIX^2. If, for party P, the leftmost bit where $\text{BITS}_{\ell_{\text{EST}}}(v_\perp)$ differs from PREFIX^1 is in block $\text{BLOCK}_i(v_\perp)$, P sends $(i, \text{BLOCK}_i(v_\perp))$ to all parties via $\Pi_{\ell\text{BC+}}$. Then, for complaint (i, BLOCK_i) from a party P holding value $v_\perp$, the parties define PREFIX^2 as the first i blocks of $v_\perp$: the first $i - 1$ blocks of PREFIX^1, followed by BLOCK_i. Since PREFIX^1 and PREFIX^2 differ in block i, the parties obtain a potential output $v_{\text{OUT?}} \in [\min(v_\perp, v^1), \max(v_\perp, v^1)]$ as discussed in Remark 2. This ensures that, if P is honest, $v_{\text{OUT?}}$ is a valid output.

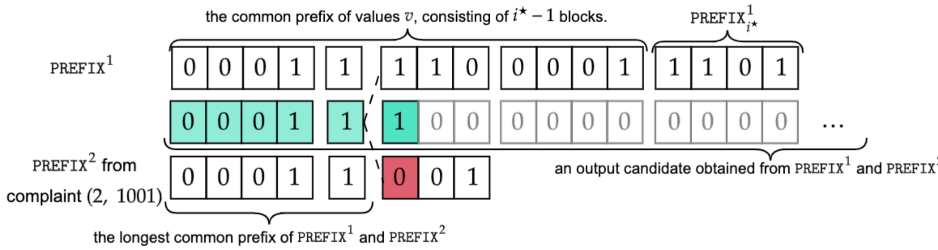

Fig. 3. This figure shows how an output candidate is obtained from a party's complaint. The first row shows $\textsc{prefix}^1$, the prefix of the ℓ_{EST}-bit representation of some valid value v^1. Some party holds a value $v_\perp$ with prefix 0001 1001 as opposed to 0001 1110, and therefore the party sends a complaint $(2, 1001)$. The prefix 0001 1001 becomes a candidate for $\textsc{prefix}^2$. Then, the parties obtain an output candidate $\textsc{min}_{\ell_{\text{EST}}}(0001\ 11) \in [v_\perp, v^1]$ from the longest common prefix of $\textsc{prefix}^1$ and $\textsc{prefix}^2$. If $v_\perp$ is valid, then this output candidate is also valid.

The parties receive a total of $(t+1)+k$ complaints, where $0 \le k \le n-(t+1)$. Since $t+1$ of these complaints are honest and all honest complaints lead to valid values $v_{\text{OUT?}}$, at most $\min(k,t)+1$ complaints lead to values $v_{\text{OUT?}}$ outside the honest range. Parties may then choose $v_{\text{OUT}} :=$ the $\min(k,t)+1$-th lowest value $v_{\text{OUT?}}$ as their final output.

$\textsc{Complain}(\ell_{\text{EST}}, v_\perp, \textsc{prefix}^1 = \textsc{prefix}^1_1 \,\|\, \ldots \,\|\, \textsc{prefix}^1_{i^\star})$

Code for party P

Find candidates for $\textsc{prefix}^2$:
If $\textsc{prefix}^1$ is not a prefix of $\textsc{bits}_{\ell_{\text{EST}}}(v_\perp)$: send $(i, \textsc{block}_i(v_\perp))$ to all parties via $\Pi_{\ell\text{BC+}}$, where i satisfies $\forall i' < i : \textsc{block}_{i'}(v_\perp) = \textsc{prefix}^1_{i'}$, and $\textsc{block}_i(v_\perp) \neq \textsc{prefix}^1_i$. (Join each $\Pi_{\ell\text{BC+}}$ invocation with $\textsc{length\_limit} := \ell_{\text{EST}}/n + \lceil \log_2 n \rceil + 1$).
$\textsc{outputs\_from\_complaints} :=$ empty multiset.
For every complaint $(i, \textsc{block}_i)$ received (with $i < i^\star$, $|\textsc{block}_i| = \ell_{\text{EST}}/n$, and $\textsc{block}_i \neq \textsc{prefix}^1_i$):
  $\textsc{prefix}^2 := \textsc{prefix}^1_1 \,\|\, \ldots \,\|\, \textsc{prefix}^1_{i-1} \,\|\, \textsc{block}_i$.
  $\textsc{common\_prefix} :=$ the longest common prefix of $\textsc{prefix}^1$ and $\textsc{prefix}^2$.
  Add $v_{\text{OUT?}} := \textsc{min}_{\ell_{\text{EST}}}(\textsc{common\_prefix} \,\|\, 1)$ to $\textsc{outputs\_from\_complaints}$.
Compute the output value v_{OUT}:
$k := |\textsc{outputs\_from\_complaints}| - (t+1)$.
Return $v_{\text{OUT}} :=$ the $\min(k,t)+1$-th lowest value in $\textsc{outputs\_from\_complaints}$.

The proof of the following lemma is included in Appendix C.4.

Lemma 6. *Assume a BA protocol Π_{BA}, and that honest parties join* Complain *with the same value ℓ_{EST} (that is a multiple of n) and with the same bitstring of $1 \le i^\star \le n$ blocks $\textsc{prefix}^1$ representing the prefix of some valid value's ℓ_{EST}-bit representation. In addition, assume that each party joins with some valid ℓ_{EST}-bit input $v_\perp$ such that the ℓ_{EST}-bit representations of $t+1$ honest parties' values $v_\perp$ do not have $\textsc{prefix}^1$ as a prefix. Then, the honest parties obtain the same valid value v_{OUT}.*

Complain has communication complexity $\textsc{bits}_{\ell_{EST}}(\textsc{Complain}) = O(\ell_{EST} \cdot n + \kappa \cdot n^3 \log n) + O(n) \cdot \textsc{bits}_\kappa(\Pi_{BA})$ and round complexity $\textsc{rounds}(\textsc{Complain}) = O(1) \cdot \textsc{rounds}(BA)$.

5.3 Putting it all together

We present the code of our protocol $\Pi_{\mathbb{N}}$ and its analysis.

Protocol $\Pi_{\mathbb{N}}$

Code for party P with input $v_{\mathrm{IN}} \in \mathbb{N}$

Run $\textsc{Estimate}(v_{\mathrm{IN}})$ obtain ℓ_{EST}, v.
Run $\textsc{BlocksLCP}(\ell_{\mathrm{EST}}, v)$ and obtain $i^\star, v, v_\perp$. If $i^\star > n$, output $v_{\mathrm{OUT}} := v$. Otherwise:
  Run $\textsc{AddLastBlock}(\ell_{\mathrm{EST}}, i^\star, v)$ and obtain $\textsc{prefix}^1$.
  Run $\textsc{Complain}(\ell_{\mathrm{EST}}, v_\perp, \textsc{prefix}^1)$ and obtain v_{OUT}. Output v_{OUT}.

Theorem 2. *Assume a BA protocol Π_{BA} resilient against $t < n/3$ corruptions. Then, if the honest parties hold ℓ-bit inputs $v_{IN} \in \mathbb{N}$, $\Pi_{\mathbb{N}}$ is a CA protocol resilient against $t < n/3$ corruptions, with communication complexity $\textsc{bits}_\ell(\Pi_{\mathbb{N}}) = O(\ell n + \kappa \cdot n^3 \log n) + O(n \log n) \cdot \textsc{bits}_\kappa(\Pi_{BA})$, and round complexity $\textsc{rounds}(\Pi_{\mathbb{N}}) = O(\log n) \cdot \textsc{rounds}(\Pi_{BA})$.*

Proof. According to Lemma 3, $\textsc{Estimate}$ enables the parties to agree on (a multiple of n) ℓ_{EST} and obtain valid values v such that: $\ell_{\mathrm{EST}} \leq \lceil \ell/n \rceil \cdot n$ and $|\textsc{bits}(v)| \leq \ell_{\mathrm{EST}}$. Hence, parties' values ℓ_{EST} and v meet the preconditions of $\textsc{BlocksLCP}$. We may then apply Lemma 4 and conclude that parties obtain the same index $i^\star$ satisfying $1 \leq i^\star \leq n+1$, and valid ℓ_{EST}-bit values $v, v_\perp$ such that the values v share a common prefix of $i^\star - 1$ blocks. If $i^\star > n$, the honest parties hold the same valid value v, and therefore CA is achieved. Otherwise, Lemma 5 ensures that parties obtain the same bitstring $\textsc{prefix}^1$ of $i^\star$ blocks that is the prefix of a valid value's ℓ_{EST}-bit representation. Since Lemma 4 additionally implies that there are $t+1$ honest parties whose values $v_\perp$ do not have $\textsc{prefix}^1$ as a prefix, $\textsc{Complain}$'s preconditions are met, and Lemma 6 ensures that CA is achieved.

The communication complexity and the round complexity follow by summing up the complexities of each subprotocol, taking into account that $\ell_{\mathrm{EST}} \leq \lceil \ell/n \rceil \cdot n$.

□

6 PROTOCOL FOR $\mathbb{Z}$

To extend the input space to $\mathbb{Z}$, we assume that the parties' inputs v_{IN} are represented as $(-1)^{\textsc{sign}_{\mathrm{IN}}} \cdot v_{\mathrm{IN}}^{\mathbb{N}}$, where $\textsc{sign}_{\mathrm{IN}} \in \{0,1\}$ and $v_{\mathrm{IN}}^{\mathbb{N}} \in \mathbb{N}$. Then, in order to cover negative numbers using $\Pi_{\mathbb{N}}$, the parties make use of the assumed BA protocol Π_{BA} to agree on their values' sign. If the sign agreed upon, denoted by $\textsc{sign}_{\mathrm{OUT}}$, differs from a party P's $\textsc{sign}_{\mathrm{IN}}$, then P will update its input value $v_{\mathrm{IN}}^{\mathbb{N}}$ to 0, since it is guaranteed to be valid. Afterwards, the parties join $\Pi_{\mathbb{N}}$ with their possibly updated inputs $v_{\mathrm{IN}}^{\mathbb{N}}$ and agree on $v_{\mathrm{OUT}}^{\mathbb{N}}$ such that $v_{\mathrm{OUT}} := (-1)^{\textsc{sign}_{\mathrm{OUT}}} \cdot v_{\mathrm{IN}}^{\mathbb{N}}$ is valid. We present the code and the guarantees of $\Pi_{\mathbb{Z}}$ below. The formal proof is included in Appendix D.

Protocol $\Pi_{\mathbb{Z}}$

Code for party P with input $v_{\mathrm{IN}} = (-1)^{\textsc{sign}_{\mathrm{IN}}} \cdot v_{\mathrm{IN}}^{\mathbb{N}}$

Join Π_{BA} with input $\textsc{sign}_{\mathrm{IN}}$ and obtain output $\textsc{sign}_{\mathrm{OUT}}$.
If $\textsc{sign}_{\mathrm{OUT}} \neq \textsc{sign}_{\mathrm{IN}}$, set $v_{\mathrm{IN}}^{\mathbb{N}} := 0$. Join $\Pi_{\mathbb{N}}$ with input $v_{\mathrm{IN}}^{\mathbb{N}}$ and obtain output $v_{\mathrm{OUT}}^{\mathbb{N}}$.
Output $v_{\mathrm{OUT}} := (-1)^{\textsc{sign}_{\mathrm{OUT}}} \cdot v_{\mathrm{OUT}}^{\mathbb{N}}$.

Corollary 3. *Assume a BA protocol Π_{BA} resilient against $t < n/3$ corruptions. Then, if the honest parties hold inputs $v_{IN} \in \mathbb{Z}$ with $\textsc{bits}(|v_{IN}|)$ consisting of at most ℓ bits, $\Pi_{\mathbb{Z}}$ is a CA protocol resilient against $t < n/3$ corruptions, with communication complexity $\textsc{bits}_\ell(\Pi_{\mathbb{Z}}) = O(\ell n + \kappa \cdot n^3 \log n) + O(n \log n) \cdot \textsc{bits}_\kappa(\Pi_{BA})$, and round complexity $\textsc{rounds}(\Pi_{\mathbb{Z}}) = O(\log n) \cdot \textsc{rounds}(\Pi_{BA})$.*

We state the final corollary, where we instantiate the BA protocol with a deterministic BA protocol with quadratic communication (e.g. [7]).

Corollary 4. *If the honest parties hold inputs $v_{IN} \in \mathbb{Z}$ with $\textsc{bits}(|v_{IN}|)$ consisting of at most ℓ bits, $\Pi_{\mathbb{Z}}$ is a CA protocol resilient against $t < n/3$ corruptions, with communication complexity $\textsc{bits}_\ell(\Pi_{\mathbb{Z}}) = O(\ell n + \kappa \cdot n^3 \log n)$, and round complexity $\textsc{rounds}(\Pi_{\mathbb{Z}}) = O(n \log n)$.*

7 CONCLUSIONS

Our work investigates whether $O(\ell n)$ bits of communication are sufficient for achieving CA on $\mathbb{N}$, and shows that this lower bound is tight when $\ell = \Omega(\kappa \cdot n^2 \log n)$. We have presented a synchronous protocol $\Pi_{\mathbb{N}}$ that relies on finding some valid values' longest common prefix, achieving CA with optimal resilience (without cryptographic setup), asymptotically optimal communication complexity, and efficient round complexity. In addition, we have extended this result to $\mathbb{Z}$.

We leave a number of exciting open problems. While we expect that our techniques can be easily extended to the asynchronous setting for a lower number of corruptions $t < n/5$, it would be interesting to see whether achieving asymptotically optimal communication complexity for $t < n/3$ corruptions in the asynchronous model is possible. The same question applies to the synchronous model with $t < n/2$ corruptions assuming cryptographic setup. A different direction could investigate whether the round complexity can be reduced from $O(n \log n)$ to the optimal $O(n)$ while maintaining the communication complexity. Further works could also consider reducing the $\mathrm{poly}(n, \kappa)$ factor, or extending our question to input spaces beyond $\mathbb{Z}$.

REFERENCES

[1] Ittai Abraham, Yonatan Amit, and Danny Dolev. 2005. Optimal Resilience Asynchronous Approximate Agreement. In *Principles of Distributed Systems*, Teruo Higashino (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 229–239.

[2] Dan Alistarh, Faith Ellen, and Joel Rybicki. 2021. Wait-Free Approximate Agreement on Graphs. In *Structural Information and Communication Complexity*, Tomasz Jurdziński and Stefan Schmid (Eds.). Springer International Publishing, Cham, 87–105. https://doi.org/10.1007/978-3-030-79527-6_6

[3] Michael Ben-Or, Danny Dolev, and Ezra N. Hoch. 2010. Brief Announcement: Simple Gradecast Based Algorithms. In *Distributed Computing*, Nancy A. Lynch and Alexander A. Shvartsman (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 194–197.

[4] Amey Bhangale, Chen-Da Liu-Zhang, Julian Loss, and Kartik Nayak. 2023. Efficient adaptively-secure byzantine agreement for long messages. In *Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part I*. Springer, 504–525.

[5] Christian Cachin and Stefano Tessaro. 2005. Asynchronous verifiable information dispersal. In *24th IEEE Symposium on Reliable Distributed Systems (SRDS'05)*. IEEE, 191–201.

[6] Wutichai Chongchitmate and Rafail Ostrovsky. 2018. Information-Theoretic Broadcast with Dishonest Majority for Long Messages. In *TCC 2018, Part I (LNCS, Vol. 11239)*, Amos Beimel and Stefan Dziembowski (Eds.). Springer, Heidelberg, 370–388. https://doi.org/10.1007/978-3-030-03807-6_14

[7] Brian A Coan and Jennifer L Welch. 1992. Modular construction of a Byzantine agreement protocol with optimal message bit complexity. *Information and Computation* 97, 1 (1992), 61–85.

[8] Andrei Constantinescu, Diana Ghinea, Lioba Heimbach, Zilin Wang, and Roger Wattenhofer. 2024. A Fair and Resilient Decentralized Clock Network for Transaction Ordering. In *27th International Conference on Principles of Distributed Systems (OPODIS 2023) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 286)*, Alysson Bessani, Xavier Défago, Junya Nakamura, Koichi Wada, and Yukiko Yamauchi (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 8:1–8:20. https://doi.org/10.4230/LIPIcs.OPODIS.2023.8

[9] Andrei Constantinescu, Diana Ghinea, Roger Wattenhofer, and Floris Westermann. 2023. Meeting in a Convex World: Convex Consensus with Asynchronous Fallback. Cryptology ePrint Archive, Paper 2023/1364. https://eprint.iacr.org/2023/1364 https://eprint.iacr.org/2023/1364.

[10] Danny Dolev, Nancy A. Lynch, Shlomit S. Pinter, Eugene W. Stark, and William E. Weihl. 1986. Reaching Approximate Agreement in the Presence of Faults. *J. ACM* 33, 3 (May 1986), 499–516. https://doi.org/10.1145/5925.5931

[11] Danny Dolev and H. Raymond Strong. 1983. Authenticated algorithms for Byzantine agreement. *SIAM J. Comput.* 12, 4 (1983), 656–666.

[12] Alan David Fekete. 1987. Asynchronous Approximate Agreement. In *6th ACM PODC*, Fred B. Schneider (Ed.). ACM, 64–76. https://doi.org/10.1145/41840.41846

[13] Alan David Fekete. 1990. Asymptotically optimal algorithms for approximate agreement. *Distributed Computing* 4, 1 (1990), 9–29.

[14] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. 1985. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)* 32, 2 (1985), 374–382.

[15] Matthias Fitzi and Martin Hirt. 2006. Optimally efficient multi-valued Byzantine agreement. In *25th ACM PODC*, Eric Ruppert and Dahlia Malkhi (Eds.). ACM, 163–168. https://doi.org/10.1145/1146381.1146407

[16] Chaya Ganesh and Arpita Patra. 2016. Broadcast Extensions with Optimal Communication and Round Complexity. In *35th ACM PODC*, George Giakkoupis (Ed.). ACM, 371–380. https://doi.org/10.1145/2933057.2933082

[17] Diana Ghinea, Chen-Da Liu-Zhang, and Roger Wattenhofer. 2022. Optimal Synchronous Approximate Agreement with Asynchronous Fallback. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing* (Salerno, Italy) *(PODC'22)*. Association for Computing Machinery, New York, NY, USA, 70–80. https://doi.org/10.1145/3519270.3538442

[18] Diana Ghinea, Chen-Da Liu-Zhang, and Roger Wattenhofer. 2023. Multidimensional Approximate Agreement with Asynchronous Fallback. In *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures* (Orlando, FL, USA) *(SPAA '23)*. Association for Computing Machinery, New York, NY, USA, 141–151. https://doi.org/10.1145/3558481.3591105

[19] Martin Hirt and Pavel Raykov. 2014. Multi-valued Byzantine Broadcast: The $t < n$ Case. In *ASIACRYPT 2014, Part II (LNCS, Vol. 8874)*, Palash Sarkar and Tetsu Iwata (Eds.). Springer, Heidelberg, 448–465. https://doi.org/10.1007/978-3-662-45608-8_24

[20] Jérémy Ledent. 2021. Brief Announcement: Variants of Approximate Agreement on Graphs and Simplicial Complexes. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing* (Virtual Event, Italy) *(PODC'21)*. Association for Computing Machinery, New York, NY, USA, 427–430. https://doi.org/10.1145/3465084.3467946

[21] Christoph Lenzen and Julian Loss. 2022. Optimal Clock Synchronization with Signatures. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing* (Salerno, Italy) *(PODC'22)*. Association for Computing

Machinery, New York, NY, USA, 440–449. https://doi.org/10.1145/3519270.3538444

[22] Guanfeng Liang and Nitin Vaidya. 2011. Error-free multi-valued consensus with Byzantine failures. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*. 11–20.

[23] Guanfeng Liang and Nitin H. Vaidya. 2011. Error-free multi-valued consensus with byzantine failures. In *30th ACM PODC*, Cyril Gavoille and Pierre Fraigniaud (Eds.). ACM, 11–20. https://doi.org/10.1145/1993806.1993809

[24] Darya Melnyk and Roger Wattenhofer. 2018. Byzantine Agreement with Interval Validity. In *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*. 251–260. https://doi.org/10.1109/SRDS.2018.00036

[25] Hammurabi Mendes and Maurice Herlihy. 2013. Multidimensional approximate agreement in Byzantine asynchronous systems. In *45th ACM STOC*, Dan Boneh, Tim Roughgarden, and Joan Feigenbaum (Eds.). ACM Press, 391–400. https://doi.org/10.1145/2488608.2488657

[26] Hammurabi Mendes, Maurice Herlihy, Nitin Vaidya, and Vijay K Garg. 2015. Multidimensional agreement in Byzantine systems. *Distributed Computing* 28, 6 (2015), 423–441.

[27] Ralph C Merkle. 1987. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*. Springer, 369–378.

[28] Kartik Nayak, Ling Ren, Elaine Shi, Nitin H. Vaidya, and Zhuolun Xiang. 2020. Improved Extension Protocols for Byzantine Broadcast and Agreement. In *34th International Symposium on Distributed Computing (DISC 2020) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 179)*, Hagit Attiya (Ed.). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 28:1–28:17. https://doi.org/10.4230/LIPIcs.DISC.2020.28

[29] Lan Nguyen. 2005. Accumulators from bilinear pairings and applications. In *Topics in Cryptology–CT-RSA 2005: The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005. Proceedings*. Springer, 275–292.

[30] Thomas Nowak and Joel Rybicki. 2019. Byzantine Approximate Agreement on Graphs. In *33rd International Symposium on Distributed Computing (DISC 2019) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 146)*, Jukka Suomela (Ed.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 29:1–29:17. https://doi.org/10.4230/LIPIcs.DISC.2019.29

[31] Irving S Reed and Gustave Solomon. 1960. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics* 8, 2 (1960), 300–304.

[32] David Stolz and Roger Wattenhofer. 2016. Byzantine Agreement with Median Validity. In *19th International Conference on Principles of Distributed Systems (OPODIS 2015) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 46)*, Emmanuelle Anceaume, Christian Cachin, and Maria Potop-Butucaru (Eds.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 1–14. https://doi.org/10.4230/LIPIcs.OPODIS.2015.22

[33] Russell Turpin and Brian A Coan. 1984. Extending binary Byzantine agreement to multivalued Byzantine agreement. *Inform. Process. Lett.* 18, 2 (1984), 73–76.

[34] Nitin H. Vaidya and Vijay K. Garg. 2013. Byzantine vector consensus in complete graphs. In *32nd ACM PODC*, Panagiota Fatourou and Gadi Taubenfeld (Eds.). ACM, 65–73. https://doi.org/10.1145/2484239.2484256

APPENDIX

A OVERVIEW: MISSING PROOFS

Remark 2. *Consider two values $v, v' \in \mathbb{N}$ satisfying $v \le v' < 2^\ell$, and let $\textsc{common\_prefix}$ denote the **longest** common prefix of $\textsc{bits}_\ell(v)$ and $\textsc{bits}_\ell(v')$.*

If $\left|\textsc{common\_prefix}\right| < \ell$, then $\textsc{max}_\ell(\textsc{common\_prefix} \,\|\, 0), \textsc{min}_\ell(\textsc{common\_prefix} \,\|\, 1) \in [v, v']$.

Proof. We show that $v \le \textsc{max}_\ell(\textsc{common\_prefix} \,\|\, 0) \le \textsc{min}_\ell(\textsc{common\_prefix} \,\|\, 1) \le v'$.

We first note that, since $v \le v'$, $\textsc{bits}_\ell(v)$ has prefix $\textsc{common\_prefix} \,\|\, 0$, while $\textsc{bits}_\ell(v')$ has prefix $\textsc{common\_prefix} \,\|\, 1$. Secondly, since $\textsc{max}_\ell(\textsc{common\_prefix} \,\|\, 0)$ is the highest ℓ-bit value having prefix $\textsc{common\_prefix} \,\|\, 0$, and v is an ℓ-bit value with the same prefix, $v \le \textsc{max}_\ell(\textsc{common\_prefix} \,\|\, 0)$. In addition, note that $\textsc{max}_\ell(\textsc{common\_prefix} \,\|\, 0) + 1 = \textsc{min}_\ell(\textsc{common\_prefix} \,\|\, 1)$. We use a similar argument to show that $v' \ge \textsc{min}_\ell(\textsc{common\_prefix} \,\|\, 1)$: v' is an ℓ-bit value with prefix $\textsc{common\_prefix} \,\|\, 1$, while $\textsc{min}_\ell(\textsc{common\_prefix} \,\|\, 1)$ is the lowest ℓ-bit value having prefix $\textsc{common\_prefix} \,\|\, 1$. □

Remark 3. *Consider two values $v, v' \in \mathbb{N}$ such that $v, v' < 2^\ell$, and let $\textsc{common\_prefix}$ denote the **longest** common prefix of $\textsc{bits}_\ell(v)$ and $\textsc{bits}_\ell(v')$. Let $\textsc{next\_bits}$ and $\textsc{next\_bits}'$ denote two non-empty bitstrings of equal length such that $\textsc{common\_prefix} \,\|\, \textsc{next\_bits}$ is a prefix of $\textsc{bits}_\ell(v)$, and $\textsc{common\_prefix} \,\|\, \textsc{next\_bits}'$ is a prefix of $\textsc{bits}_\ell(v')$. Then, if $\textsc{val}(\textsc{next\_bits}) < \textsc{val}(\textsc{next\_bits}')$, then $\textsc{min}_\ell(\textsc{common\_prefix} \,\|\, \textsc{next\_bits}'), \textsc{max}_\ell(\textsc{common\_prefix} \,\|\, \textsc{next\_bits}) \in [v, v']$.*

Proof. Since $\textsc{bits}_\ell(v)$ has prefix $\textsc{common\_prefix} \| \textsc{next\_bits}$, v is at most the highest ℓ-bit value having prefix $\textsc{common\_prefix} \,\|\, \textsc{next\_bits}$. Similarly, since $\textsc{bits}_\ell(v')$ has prefix $\textsc{common\_prefix} \,\|\, \textsc{next\_bits}'$, v is at least the lowest ℓ-bit value having this prefix $\textsc{common\_prefix} \,\|\, \textsc{next\_bits}'$. In addition, since $\textsc{val}(\textsc{next\_bits}) < \textsc{val}(\textsc{next\_bits}')$, we have that $\textsc{max}(\textsc{common\_prefix} \,\|\, \textsc{next\_bits}) \le \textsc{min}_\ell(\textsc{common\_prefix} \,\|\, \textsc{next\_bits}')$. Therefore, we have obtained the following inequality: $v \le \textsc{max}_\ell(\textsc{common\_prefix} \,\|\, \textsc{next\_bits}) \le \textsc{min}_\ell(\textsc{common\_prefix} \,\|\, \textsc{next\_bits}') \le v'$. □

B BA FOR LONG MESSAGES

B.1 BA for $t < n/3$, without setup

We restate the protocol and prove of the BA protocol for long messages described in Section 4.1, which was introduced in [28].

Protocol $\Pi_{\ell\mathrm{BA}}$

Code for party P_i with input v_{IN}

Let $s_1, s_2, \ldots, s_n := \mathrm{RS}.\textsc{encode}(v_{\mathrm{IN}})$; $z, w_1, w_2, \ldots, w_n := \mathrm{MT}.\textsc{Build}(\{s_1, s_2, \ldots s_n\})$.
Join Π_{BA} with input z and obtain output $z^\star$.
Join Π_{BA} with input $b = 1$ if $z = z^\star$ and $b = 0$ otherwise. Obtain output b'.
If $b' = 0$, output $\perp$.
Otherwise, if $b' = 1$, run the **distributing step**:
    If $z^\star = z$: for every $1 \le j \le n$, send (j, s_j, w_j) to P_j.
    If you have received a tuple (i, s_i, w_i) such that $\mathrm{MT}.\textsc{Verify}(i, z^\star, s_i, w_i) = \texttt{true}$:
        Send (i, s_i, w_i) to all parties.
    Discard any tuples (j, s_j, w_j) where $\mathrm{MT}.\textsc{Verify}(i, z^\star, s_i, w_i) = \texttt{false}$.
    Let $S :=$ the set of correct tuples received. Output $v^\star := \mathrm{RS}.\textsc{decode}(S)$.

Lemma 7. *Assume that the parties join the distributing step and that at least one honest party has proposed $z = z^\star$. Then, the honest party agree on a value $v^\star$ that is an honest party's input. In addition, this step has communication complexity $O(\ell n + \kappa \cdot n^2 \log n)$ and round complexity $O(1)$.*

Proof. Since at least one honest party P_i holds $z := z^\star$, P_i holds an input value $v^\star$ whose RS encoding $s_1, \ldots, s_n$ leads to an MT tree with root $z^\star$. P_i sends to each party P_j a tuple (j, s_j, w_j) such that MT.Verify$(z^\star, j, s_j, w_j) =$ `true`.

Note that party P_j ignores any tuples (j, s'_j, w'_j) with $s'_j \neq s_j$: a different RS encoding $(s'_1, \ldots, s'_n) \neq (s_1, \ldots, s_n)$ leads to an MT with root $z \neq z^\star$. Hence, such a tuple is sent by a corrupted party. We note that finding a witness w'_j with MT.Verify$(z^\star, j, s'_j, w'_j) =$ `true` requires the adversary to find collisions for H_κ, which we assumed to be impossible. Therefore, MT.Verify$(z^\star, j, s'_j, w'_j) =$ `false`, and P_j discards this tuple.

Then, every party P_i holds a unique correct tuple (i, s_i, w_i) (possibly received from multiple parties), and forwards this tuple to all parties. Each party P_i receives $n - t$ correct tuples from honest parties, plus at most t tuples from corrupted parties. Once again, if an honest party P_j receives (j, s'_j, w'_j) with an incorrect codeword s'_j, P_j discards this tuple: MT.Verify$(z^\star, j, s'_j, w'_j) =$ `false`. Hence, all (at least $n - t$) tuples remaining are correct, which allows the parties to reconstruct $v^\star$ correctly. Therefore, the parties agree on an honest party's input value.

It remains to discuss the communication complexity and the round complexity. There are two communication rounds, where every party sends to all parties at most two tuples. Each such tuple contains an index of $O(\log n)$ bits, a RS codeword of $O(\ell/n)$ bits, and a MT witness of $O(\kappa \cdot \log n)$ bits. Therefore, this step has a total communication cost of $O(\ell n + \kappa \cdot n^2 \log n)$ bits. □

Theorem 1 ([28]). *Given a BA protocol Π_{BA} secure against $t < n/3$ corruptions, there is a protocol $\Pi_{\ell BA}$ achieving BA secure against $t < n/3$ corruptions with communication complexity $\textsc{bits}_\ell(\Pi_{\ell BA}) = O(\ell n + \kappa \cdot n^2 \log n) + O(1) \cdot \textsc{bits}_\kappa(\Pi_{BA})$, and round complexity $\textsc{rounds}(\Pi_{\ell BA}) = O(1) \cdot \textsc{rounds}(\Pi_{BA})$.*

Proof. We first focus on Agreement: honest parties obtain the same bit b' in Π_{BA}. If $b' = 0$, all honest parties output $\perp$. Otherwise, at least one honest party has proposed $b = 1$ and therefore holds $z = z^\star$ and Lemma 7 ensures that the honest parties output the same value.

For Validity, if all honest parties hold the same input v, then they obtain the same encoding z, since the algorithm computing the RS encoding and the algorithm building the MT are deterministic. Then, Π_{BA} returns $z^\star = z$. Afterwards, all honest parties join Π_{BA} with input $b = 1$ and obtain $b' = 1$. Lemma 7 ensures that the honest parties output $v^\star = v$. Therefore, BA is achieved.

In terms of communication complexity and round complexity, the parties run Π_{BA} on κ-bit values, and afterwards on a single bit, leading to a cost of $\textsc{bits}_\kappa(\Pi_{\mathrm{BA}}) + \textsc{bits}_1(\Pi_{\mathrm{BA}})$ bits and $2 \cdot \textsc{rounds}(\Pi_{\mathrm{BA}})$ rounds. Afterwards, the parties join the distributing step only if some honest party holds $z = z^\star$, which incurs an additional cost of $O(\ell n + \kappa \cdot n^2 \log n)$ bits and $O(1)$ rounds according to Lemma 7. Therefore, the total communication complexity of Π_{BA} is $O(\ell n + \kappa \cdot n^2 \log n) + O(1) \cdot \textsc{bits}_\kappa(\Pi_{\mathrm{BA}})$, and the round complexity is $O(1) \cdot \textsc{rounds}(\Pi_{\mathrm{BA}})$. □

B.2 Additional properties

This section formally proves the Corollary 2, restated below.

Corollary 2. *Given a BA protocol Π_{BA} resilient against $t < n/3$ corruptions, there is a BA protocol $\Pi_{\ell BA+}$ resilient against $t < n/3$ corruptions that additionally achieves No Corrupted Output and $(t+1)$-Disagreement. The communication complexity of $\Pi_{\ell BA+}$ is $\textsc{bits}_\ell(\Pi_{\ell BA+}) = O(\ell n + \kappa \cdot n^2 \log n) + O(n) \cdot \textsc{bits}_\kappa(\Pi_{BA})$, and the round complexity is $\textsc{rounds}(\Pi_{\ell BA+}) = O(1) + \textsc{rounds}(\Pi_{BA})$.*

We first include the analysis of $\Pi_{\mathrm{BA+}}$.

Lemma 8. *Given a BA protocol Π_{BA} resilient against $t < n/3$ corruptions, Π_{BA+} is a BA protocol resilient against $t < n/3$ corruptions that additionally achieves No Corrupted Output and $(t+1)$-Disagreement. Π_{BA+} has communication complexity $\textsc{bits}_\kappa(\Pi_{BA+}) = O(\kappa n^2) + O(n) \cdot \textsc{bits}_\kappa(\Pi_{BA})$, and round complexity $\textsc{rounds}(\Pi_{BA+}) = O(1) + \textsc{rounds}(\Pi_{BA})$.*

Proof. The parties distribute their κ-bit input values through the BC protocol $\Pi_{\mathrm{BC+}}$ described in Lemma 1. Therefore, all parties receive the same values z, and afterwards obtain the same output, which ensures Agreement. In addition, if all honest parties hold the same input value z, then the parties receive $n - t > t + 1$ values z, and at most t different values. Hence, honest parties output $z^\star := z$, which ensures that Validity holds. Consequently, $\Pi_{\mathrm{BA+}}$ achieves BA.

If the output is $z^\star \neq \bot$, then at least $t+1$ parties, and hence at least one honest party, have proposed $z^\star$, which ensures that the No Corrupted Output property holds.

Finally, we show that the $(t+1)$-Disagreement property holds: if the output agreed upon is $\bot$, there was no value $z^\star$ proposed by $t+1$ parties. It follows that, for any value z, at most t honest parties hold input z, and at least $(n-t) - t \geq t+1$ honest parties do not have z as input.

For the communication complexity and round complexity, note that each party distributes a κ-bit value via $\Pi_{\mathrm{BC+}}$. This leads to $O(n) \cdot \textsc{bits}_\kappa(\Pi_{\mathrm{BC+}})$ bits, and, since the $\Pi_{\mathrm{BC+}}$ invocations are in parallel, $\textsc{rounds}(\Pi_{\mathrm{BC+}})$ rounds. Then, by applying Lemma 1 we obtain that $\textsc{bits}_\kappa(\Pi_{\mathrm{BA+}}) = O(\kappa n^2) + O(n) \cdot \textsc{bits}_\kappa(\Pi_{\mathrm{BA}})$ and $\textsc{rounds}(\Pi_{\mathrm{BA+}}) = O(1) + \textsc{rounds}(\Pi_{\mathrm{BA}})$. □

We now focus on the BA protocol that achieves No Corrupted Output and $(t+1)$-Disagreement and that is designed for long messages. As described in Section 4.3, to achieve these additional properties, we only need to replace the Π_{BA} invocations in Line 2 and Line 3 of $\Pi_{\ell\mathrm{BA}}$ with one invocation of $\Pi_{\mathrm{BA+}}$. We include the updated code below.

Protocol $\Pi_{\ell\mathrm{BA+}}$

Code for party P_i with input v_{IN}

1: Let $s_1, s_2, \ldots, s_n := \mathrm{RS.}\textsc{encode}(v_{\mathrm{IN}})$; $z, w_1, w_2, \ldots, w_n := \mathrm{MT.}\textsc{Build}(\{s_1, s_2, \ldots s_n\})$.
2: Join $\Pi_{\mathrm{BA+}}$ with input z.
3: If $\Pi_{\mathrm{BA+}}$ has returned $\bot$, output $\bot$.
4: Otherwise, if $\Pi_{\mathrm{BA+}}$ has returned $z^\star \neq \bot$, run the **distributing step**:
5:     If $z^\star = z$: for every $1 \leq j \leq n$, send (j, s_j, w_j) to P_j.
6:     If you have received a tuple (i, s_i, w_i) such that $\mathrm{MT.}\textsc{Verify}(i, z^\star, s_i, w_i) = \texttt{true}$:
7:         Send (i, s_i, w_i) to all parties.
8:     Discard any tuples (j, s_j, w_j) where $\mathrm{MT.}\textsc{Verify}(i, z^\star, s_i, w_i) = \texttt{false}$.
9:     Let $S :=$ the set of correct tuples received. Output $v^\star := \mathrm{RS.}\textsc{decode}(S)$.

Below we provide the analysis of $\Pi_{\ell\mathrm{BA+}}$. Lemma 8 and Lemma 9 directly imply Corollary 2.

Lemma 9. *Assume a BA protocol Π_{BA+} secure against $t < n/3$ corruptions that additionally achieves No Corrupted Output and $(t+1)$-Disagreement. Then, $\Pi_{\ell BA+}$ achieves the same guarantees, with communication complexity $\textsc{bits}_\ell(\Pi_{\ell BA+}) = O(\ell n + \kappa \cdot n^2 \log n) + \textsc{bits}_\kappa(\Pi_{BA+})$, and round complexity $\textsc{rounds}(\Pi_{\ell BA+}) = O(1) + \textsc{rounds}(\Pi_{BA+})$.*

Proof. We show that $\Pi_{\ell\mathrm{BA}}$ achieves BA using a similar argument to that of Theorem 1. The parties obtain the same output in $\Pi_{\mathrm{BA+}}$: either $z^\star$ or $\bot$. If the output returned by $\Pi_{\mathrm{BA+}}$ is $\bot$, the parties output $\bot$, hence Agreement holds in this case. Otherwise, then there is an honest party who proposed $z = z^\star$ since $\Pi_{\mathrm{BA+}}$ achieves No Corrupted Output and Lemma 7 ensures that the parties agree on the same value.

Honest parties holding the same value v_{IN} obtain the same encoding z since the algorithms for computing the RS encoding and the MT are deterministic. Then, if all honest parties hold the same input v, then all honest parties obtain the same value z, and $\Pi_{\text{BA+}}$ returns $z^{\star} = z$. Lemma 7 ensures that the parties output an honest party's output, therefore they output v. Therefore, Validity also holds and BA is achieved.

If the honest parties obtain a non-$\bot$ output, they have obtained this value via the distributing step. Since the distributing step is only run if there is an honest party holding $z = z^{\star}$, Lemma 7 ensures that the No Corrupted Output property holds.

If the parties output $\bot$, the $(t+1)$-Disagreement property of $\Pi_{\text{BA+}}$ ensures that there was no value z proposed by $t+1$ honest parties, and hence there is no value v held as input by at least $t+1$ honest parties. That is, for any value v, there are at least $(n-t)-t \geq t+1$ honest parties having $v_{\text{IN}} \neq v$, and therefore $(t+1)$-Disagreement is maintained.

We have obtained that $\Pi_{\ell\text{BA+}}$ indeed maintains the properties of $\Pi_{\text{BA+}}$. Running $\Pi_{\text{BA+}}$ with inputs z requires $\textsc{bits}_{\kappa}(\Pi_{\text{BA+}})$ bits and $\textsc{rounds}(\Pi_{\text{BA+}})$ rounds. If the output is $\bot$, there is no further communication. Otherwise, the parties run the distributing step, and Lemma 7 shows that this step has an additional cost of $(\ell n + \kappa \cdot n^2 \log n)$ bits and $O(1)$ rounds. Then, the total bit complexity of $\Pi_{\ell\text{BA+}}$ is $O(\ell n + \kappa \cdot n^2 \log n) + \textsc{bits}_{\kappa}(\Pi_{\text{BA+}})$, and the round complexity is $O(1) + \textsc{rounds}(\Pi_{\text{BA+}})$. □

C PROTOCOL FOR $\mathbb{N}$: MISSING PROOFS

C.1 Subprotocol Estimate

We include the missing proofs and the full analysis of the Estimate subprotocol.

Lemma 2. *Let* $\textsc{received\_values}$ *denote a multiset of* $n-t+k$ *values, where* $0 \leq k \leq t$*, and let* $\textsc{honest\_values} \subseteq \textsc{received\_values}$ *denote a multiset of* $n-t$ *values. Then, if* $\textsc{safe\_values}$ *is a multiset obtained by discarding the lowest* k *and highest* k *values in* $\textsc{received\_values}$*, it holds that* $|\textsc{safe\_values}| \geq t+1$*, and* $\textsc{safe\_values} \subseteq [\min \textsc{honest\_values}, \max \textsc{honest\_values}]$.

Proof. We first note that $|\textsc{safe\_values}| = (n-t+k) - 2k \geq n-2t \geq t+1$.

We now focus on safe_values being within the range of values honest_values. Only the k values that are in received_values but not in honest_values may be lower than min honest_values or higher than max honest_values. Then, since the multiset safe_values is obtained by discarding the lowest k and the highest k values from received_values, it follows that $\min \textsc{safe\_values} \geq \min \textsc{honest\_values}$ and $\max \textsc{safe\_values} \leq \max \textsc{honest\_values}$. □

Lemma 3. *Assume a BA protocol* Π_{BA} *resilient against* $t < n/3$ *corruptions, and let* $\ell_{\min}$ *and* $\ell_{\max}$ *denote the lowest and resp. the highest lengths* $|\textsc{bits}(v_{IN})|$ *of the honest inputs* v_{IN}*. Then, in* Estimate*, honest parties agree on a value* ℓ_{EST} *that is a multiple of* n *and satisfies* $\ell_{\min} \leq \ell_{EST} \leq \lceil \ell_{\max}/n \rceil \cdot n$*. In addition, every honest party obtains a valid* ℓ_{EST}*-bit value* v*.* Estimate *achieves communication complexity* $\textsc{bits}_{\ell}(\text{Estimate}) = O(\ell n + k \cdot n^3 \log n) + O(n) \cdot \textsc{bits}_{\kappa}(\Pi_{BA})$*, and round complexity* $\textsc{rounds}(\text{Estimate}) = O(1) \cdot \textsc{rounds}(\Pi_{BA})$.

Proof. In the first step, every party receives $n-t$ values l from the honest parties, plus $k \leq t$ values from corrupted parties. Lemma 2 ensures that each party obtains a multiset safe_values containing at least $t+1$ values that are within the range of values l sent by honest parties. The number of bits sent in this step is at most $O(\log(\ell/n) \cdot n^2)$, hence $O(\ell n)$.

Afterwards, the parties send $l_{\min}$ via $\Pi_{\ell\text{BC+}}$, while every party joins the $\Pi_{\ell\text{BC+}}$ invocations with $\textsc{length\_limit} := \lceil \log_2 l_{\max} \rceil + 1$. To ensure that the precondition of the Limited Length property holds and therefore honest parties' messages get delivered, it is sufficient to prove that, if P and P' are honest and obtain multisets safe_values and safe_values$'$ respectively, then $l_{\min} =$

$\min \text{SAFE\_VALUES} \le \max \text{SAFE\_VALUES}' = l'_{\max}$. Note that both P and P' have received the $n-t$ honest values, and obtained SAFE_VALUES and SAFE_VALUES′ by discarding the at most the lowest t and the highest t values received. Then, they discarded at most the lowest t and the highest t out of the $n-t \ge 2t+1$ honest values. There is at least one honest value left, which will be included in both SAFE_VALUES and SAFE_VALUES′. It follows that $\text{SAFE\_VALUES} \cap \text{SAFE\_VALUES}' \neq \emptyset$ and $l_{\min} \le l'_{\max}$. Note that this step has communication cost at most $O(n) \cdot \text{BITS}_{\lceil \log_2 \lceil \ell_{\max}/n \rceil \rceil + 1}(\Pi_{\ell\text{BC+}})$.

Then, honest parties receive the same values via $\Pi_{\ell\text{BC+}}$. These will be $n-t$ values $l_{\min}$ from honest parties, plus k values from corrupted parties. Applying Lemma 2 once again ensures that $l :=$ the $(k+1)$-th lowest value $l_{\min}$ received is within the range of values honest values $l_{\min}$, and therefore l satisfies $\lceil \ell_{\min}/n \rceil \le l \le \lceil \ell_{\max}/n \rceil$ and $\ell_{\text{EST}} := l \cdot n$ satisfies $\ell_{\min} \le \ell_{\text{EST}} \le \lceil \ell_{\max}/n \rceil \cdot n$.

We still need to show that the parties return valid ℓ_{EST}-bit values v. If a party P holds $v_{\text{IN}} < 2^{\ell_{\text{EST}}}$, it sets $v := v_{\text{IN}}$, and the claim follows immediately. Otherwise, if P holds $v_{\text{IN}} \ge 2^{\ell_{\text{EST}}}$, we use the guarantee that the lowest honest input $v_{\min}$ satisfies $v_{\min} < 2^{\ell_{\text{EST}}}$. This implies that $v := 2^{\ell_{\text{EST}}} - 1$ is in the interval $[v_{\min}, v_{\max}]$, hence it is a valid ℓ_{EST}-bit value.

For the communication complexity, we have obtained that $\text{BITS}_\ell(\text{ESTIMATE}) = O(\ell n) + O(n) \cdot \text{BITS}_{\lceil \log_2 \lceil \ell_{\max}/n \rceil \rceil + 1}(\Pi_{\ell\text{BC+}})$. The round complexity is $\text{ROUNDS}(\text{ESTIMATE}) = O(1) + \text{ROUNDS}(\Pi_{\ell\text{BC+}})$ since ESTIMATE runs n invocations of $\Pi_{\ell\text{BC+}}$ in parallel after one communication round. Afterwards, applying Corollary 1 leads to the results claimed in the lemma's statement. □

C.2 Subprotocol BLOCKSLCP

We first prove the invariants of each iteration, as described in the proof sketch of Lemma 4.

Lemma 10. *Assume that the following properties hold at the beginning of iteration i.*

(A) All honest parties hold the same indices $1 \le \text{LEFT} \le \text{RIGHT} \le n+1$, and the same bitstring $\text{PREFIX}^\star$ consisting of $\text{LEFT}-1$ blocks.

(B) $0 \le \text{RIGHT} - \text{LEFT} \le 2^{\lceil \log_2 n \rceil - (i-1)}$.

(C) Honest parties hold valid ℓ_{EST}-bit values v such that $\text{BITS}_{\ell_{EST}}(v)$ has $\text{PREFIX}^\star$ as a prefix.

(D) Honest parties hold valid ℓ_{EST}-bit values $v_\perp$, and, for any bitstring BITS of RIGHT blocks, the ℓ_{EST}-bit representations of the values $v_\perp$ of $t+1$ honest parties do not have prefix BITS.

Then, either the condition $\text{LEFT} = \text{RIGHT}$ is met in iteration i, or the properties hold at the beginning of iteration $i+1$.

PROOF. We assume that the condition $\text{LEFT} = \text{RIGHT}$ is not yet met in iteration i (otherwise, the statement trivially holds). Then, Property (B) ensures that $\text{LEFT} < \text{RIGHT}$, and we may prove that the properties hold at the beginning of iteration $i+1$ as well. The honest parties obtain the same output in the $\Pi_{\ell\text{BA+}}$ invocation of iteration i: either $\perp$ or a sequence of blocks, and we split the analysis into these two cases. In the following, we make the iteration number explicit to differentiate between variables' values at the beginning of iteration i and at the beginning of iteration $i+1$ (i.e. $\text{PREFIX}^\star(i)$ is the value held at the beginning of iteration i, and $\text{PREFIX}^\star(i+1)$ is the value computed during iteration i and held at the beginning of iteration $i+1$).

We first assume that $\Pi_{\ell\text{BA+}}$ returns $\perp$:

(A) Honest parties compute the $\text{RIGHT}(i+1)$ index identically, while all other values remain unchanged. Note that $\text{LEFT}(i) \le \text{MID} < \text{RIGHT}(i)$ and therefore $\text{RIGHT}(i+1) := \text{MID}$ still satisfies $1 \le \text{RIGHT}(i+1) \le n+1$. Therefore, iteration Property (A) holds at the beginning of iteration $i+1$ as well.

(B) All honest parties compute $\text{RIGHT}(i+1) := \text{MID} \ge \text{LEFT}(i)$, while the LEFT index remains unchanged: $\text{LEFT}(i+1) := \text{LEFT}(i)$. We obtain the inequality below, which ensures that

Property (B) holds at the beginning of iteration $i+1$.

$$0 \leq \textsc{right}(i+1) - \textsc{left}(i+1) = \lfloor (\textsc{left}(i) + \textsc{right}(i))/2 \rfloor - \textsc{left}(i)$$
$$\leq (\textsc{left}(i) + \textsc{right}(i))/2 - \textsc{left}(i)$$
$$= (\textsc{right}(i) - \textsc{left}(i))/2 \leq 2^{\lceil \log_2 n \rceil - ((i+1)-1)}.$$

(C) Since $v(i+1) := v(i)$, $\textsc{left}(i+1) := \textsc{left}(i)$ and $\textsc{prefix}^\star(i+1) := \textsc{prefix}^\star(i)$, Property (C) holds at the beginning of iteration $i+1$.

(D) Note that $v_\perp(i+1) := v(i)$ is a valid ℓ_{EST}-bit value according to Property (C). We also need to show that, given an arbitrary bitstring $\textsc{block}_1 \parallel \ldots \parallel \textsc{block}_{\textsc{mid}}$ of $\textsc{right}(i+1) = \textsc{mid}$ blocks, there are $t+1$ honest parties holding values $v(i)$ such that $\textsc{bits}_{\ell_{\text{EST}}}(v(i))$ does not have $\textsc{bits}$ as a prefix. This is ensured by the $(t+1)$-*Disagreement* property of $\Pi_{\ell\text{BA+}}$: $t+1$ honest parties hold values $v(i)$ satisfying $\textsc{block}_{\textsc{left}(i)}(v(i)) \parallel \ldots \parallel \textsc{block}_{\textsc{mid}}(v(i)) \neq \textsc{block}_{\textsc{left}(i)} \parallel \ldots \parallel \textsc{block}_{\textsc{mid}}$, which implies that the bit representations $\textsc{bits}_{\ell_{\text{EST}}}(v(i))$ do not have prefix $\textsc{block}_1 \parallel \ldots \parallel \textsc{block}_{\textsc{mid}}$. Therefore, Property (D) holds at the beginning of iteration $i+1$.

We now assume that $\Pi_{\ell\text{BA+}}$ returns $\textsc{prefix}^\star_{\textsc{left}(i)} \parallel \ldots \parallel \textsc{prefix}^\star_{\textsc{mid}}$:

(A) The parties compute their $\textsc{left}(i+1)$ index and the sequence of blocks $\textsc{prefix}^\star(i+1)$ identically, while the $\textsc{right}$ index remains unchanged ($\textsc{right}(i+1) := \textsc{right}(i)$). Note that $\textsc{prefix}^\star(i+1)$ is obtained by adding $\textsc{left}(i+1) - \textsc{left}(i)$ blocks to $\textsc{prefix}^\star(i)$, therefore $\textsc{prefix}^\star(i+1)$ consists of $\textsc{left}(i+1) - 1$ blocks. In addition, $\textsc{left}(i) \leq \textsc{mid} < \textsc{right}(i)$ and therefore $\textsc{left}(i+1) := \textsc{mid} + 1$ still satisfies $1 \leq \textsc{left}(i+1) \leq n+1$. Therefore, Property (A) holds at the beginning of iteration $i+1$.

(B) Since $\textsc{left}(i+1) := \textsc{mid} + 1 \leq \textsc{right}(i)$, while $\textsc{right}(i+1) := \textsc{right}(i)$, we obtain the inequality below, which ensures that Property (B) holds at the beginning of iteration $i+1$ as well.

$$0 \leq \textsc{right}(i+1) - \textsc{left}(i+1) = \textsc{right}(i) - (\lfloor (\textsc{left}(i) + \textsc{right}(i))/2 \rfloor + 1)$$
$$\leq \textsc{right}(i) - (\textsc{left}(i) + \textsc{right}(i))/2$$
$$\leq (\textsc{right}(i) - \textsc{left}(i))/2 \leq 2^{\lceil \log_2 n \rceil - ((i+1)-1)}.$$

(C) Honest parties either hold values $v(i)$ having $\textsc{prefix}^\star(i+1)$ as a prefix, or they set $v(i+1)$ to some ℓ_{EST}-bit value having prefix $\textsc{prefix}^\star(i+1)$. This implies that, at the beginning of iteration $i+1$, all honest parties hold ℓ_{EST}-bit values $v(i)$ with prefix $\textsc{prefix}^\star(i+1)$. We still need to prove that values $v(i+1)$ are valid. If $v(i+1) = v(i)$, this follows from Property (C) holding for values $v(i)$. Otherwise, let P denote an honest party holding $v(i+1) \neq v(i)$.
The *No Corrupted Output* property of $\Pi_{\ell\text{BA+}}$ ensures that parties agree on a sequence of blocks $\textsc{prefix}^\star_{\textsc{left}(i)} \parallel \ldots \parallel \textsc{prefix}^\star_{\textsc{mid}}$ that was proposed by an honest party holding value $v^\star$. Then, Property (C) ensures that $v^\star$ is a valid ℓ_{EST}-bit value such that $\textsc{bits}_{\ell_{\text{EST}}}(v^\star)$ has prefix $\textsc{prefix}^\star(i+1)$. On the other hand, $v(i)$ is a valid ℓ_{EST}-bit value such that $\textsc{bits}_{\ell_{\text{EST}}}(v(i))$ has $\textsc{prefix}^\star(i)$ as a prefix, but not $\textsc{prefix}^\star(i+1)$. Remark 3 guarantees that P's updated value $v(i+1)$ is in $[\min(v(i), v^\star), \max(v(i), v^\star)]$, and therefore it is an ℓ_{EST}-bit value within the honest inputs' range.

(D) Since $\textsc{right}(i+1) := \textsc{right}(i)$ and $v_\perp(i+1) := v_\perp(i)$, Property (D) is maintained.

□

We may now focus on the proof of Lemma 4.

Lemma 4. *Assume a BA protocol Π_{BA}, and that the honest parties join* BlocksLCP *with the same value ℓ_{EST} (that is a multiple of n) and valid ℓ_{EST}-bit values v. Then, the honest parties obtain the same index $i^\star$, and each honest party obtains a pair of valid ℓ_{EST}-bit values $v, v_\perp$ such that:*

- *the ℓ_{EST}-bit representations of the values v have a common prefix of $i^\star - 1$ blocks;*
- *for any bitstring* bits *of $i^\star$ blocks, there are $t + 1$ honest parties holding values $v_\perp$ such that $\textsc{bits}_{\ell_{EST}}(v_\perp)$ does not have prefix* bits.

BlocksLCP has communication complexity $\textsc{bits}_{\ell_{EST}}(\textsc{BlocksLCP}) = O(\ell_{EST} \cdot n + \kappa \cdot n^2 \log^2 n) + O(n \log n) \cdot \textsc{bits}_\kappa(\Pi_{BA})$ and round complexity $\textsc{rounds}(\textsc{BlocksLCP}) = O(\log n) \cdot \textsc{rounds}(\Pi_{BA})$.

Proof. The properties listed in Lemma 10 hold in iteration 1 due to the variables' initialization. Hence, these properties hold for every iteration of the loop.

Property (A) ensures that honest parties hold the same indices left and right in every iteration of the loop. Then, once the condition $\textsc{left} = \textsc{right}$ is met, the honest parties set $i^\star$ identically as $i^\star = \textsc{left}$, satisfying $1 \leq i^\star \leq n + 1$. Then, Property (C) guarantees that honest parties hold valid ℓ_{EST}-bit values v having the bitstring $\textsc{prefix}^\star$ as a common prefix. According to Property (A), this common prefix consists of $\textsc{left} - 1 = i^\star - 1$ blocks. From Property (D), it follows that the honest parties hold valid ℓ_{EST}-bit values $v_\perp$. The same property implies that, for any bitstring bits of $\textsc{right} = i^\star$ blocks, the ℓ_{EST}-bit representations of $t+1$ honest parties do not have bits as a prefix. Hence, once the stopping condition holds, honest parties hold values $i^\star$, v, and $v_\perp$ satisfying the guarantees in the lemma's statement. It remains to show that the stopping condition indeed holds eventually.

Note that the condition $\textsc{left} = \textsc{right}$ is met (for all honest parties simultaneously, due to Property (A)) by iteration $i := \lceil \log_2 n \rceil + 2$. Property (B) ensures that, at the beginning of iteration i, $0 \leq \textsc{right} - \textsc{left} \leq 2^{\lceil \log_2 n \rceil - (i-1)}$. Then, if this condition was not met by iteration $i := \lceil \log_2 n \rceil + 2$, the indices left and right obtained by the honest parties in iteration i satisfy $0 \leq \textsc{right} - \textsc{left} \leq 2^{\lceil \log_2 n \rceil - (\lceil \log_2 n \rceil + 1)} \leq 2^{-1}$. Since the indices left and right are natural numbers, we may conclude that $\textsc{right} - \textsc{left} = 0$.

We may then discuss the round complexity of BlocksLCP: since $O(\log n)$ iterations are sufficient and each iteration invokes $\Pi_{\ell\text{BA+}}$ once, we obtain that $\textsc{rounds}(\textsc{BlocksLCP}) = O(\log n) \cdot \textsc{rounds}(\Pi_{\ell\text{BA+}})$. Then, Corollary 2 leads to the result claimed in the lemma's statement.

For the communication complexity, Property (B) of Lemma 10 ensures that, in each iteration $i < \lceil \log_2 n \rceil + 2$, BlocksLCP runs $\Pi_{\ell\text{BA+}}$ on a bitstring of at most $2^{\lceil \log_2 n \rceil - i}$ blocks, hence $2^{\lceil \log_2 n \rceil - i} \cdot \ell_{\text{EST}}/n \leq \ell_{\text{EST}}/2^{i-1}$ bits. Therefore, $\textsc{bits}_{\ell_{\text{EST}}}(\textsc{BlocksLCP}) = \sum_{i=1}^{\lceil \log_2 n \rceil + 1} \textsc{bits}_{\ell_{\text{EST}}/2^{i-1}}(\Pi_{\ell\text{BA+}})$. Using Corollary 2, and the fact that $\sum_{i=0}^{\infty} 1/2^i \leq 2$, we obtain that $\textsc{bits}_{\ell_{\text{EST}}}(\textsc{BlocksLCP}) = O(\ell_{\text{EST}} \cdot n + \kappa \cdot n^2 \log^2 n) + O(n \log n) \cdot \textsc{bits}_\kappa(\Pi_{\text{BA}})$. ◻

C.3 Subprotocol AddLastBlock

Lemma 5. *Assume a BA protocol Π_{BA}, and that honest parties join* AddLastBlock *with the same value ℓ_{EST} (that is a multiple of n), with the same index $1 \leq i^\star \leq n$, and with valid ℓ_{EST}-bit values v that have a common prefix of $i^\star - 1$ blocks. Then, honest parties agree on a bitstring $\textsc{prefix}^1$ of $i^\star$ blocks such that there is a valid value v^1 whose ℓ_{EST}-bit representation has prefix $\textsc{prefix}^1$.*

AddLastBlock has communication complexity $\textsc{bits}_{\ell_{EST}}(\textsc{AddLastBlock}) = O(\ell_{EST} \cdot n + k \cdot n^3 \log n) + O(n) \cdot \textsc{bits}_\kappa(\Pi_{BA})$ and round complexity $\textsc{rounds}(\textsc{AddLastBlock}) = O(1) \cdot \textsc{rounds}(\Pi_{BA})$.

Proof. Since parties send their blocks' values $\textsc{val}(\textsc{block}_{i^\star}(v))$ (satisfying $\textsc{length\_limit} = \ell_{\text{EST}}/n$) via $\Pi_{\text{BC+}}$, the honest parties receive the same $n - t$ blocks from honest parties, and the same $k \leq t$ blocks from corrupted parties. Then, since honest values v have a common prefix of $i^\star - 1$ blocks, the honest parties obtain the same bitstring of $i^\star$ blocks $\textsc{prefix}^1 := \textsc{block}_1(v) \parallel \ldots \parallel \textsc{block}_{i^\star - 1}(v) \parallel \textsc{bits}_{\ell_{\text{EST}}/n}(\textsc{safe\_block\_val})$. In addition, Lemma 2 ensures that safe_block_val is within the range of values $\textsc{val}(\textsc{block}_{i^\star}(v))$ of honest values v, and therefore $\textsc{prefix}^1$ is the prefix of a valid value's ℓ_{EST}-bit representation according to Remark 4.

For the bit complexity, note that parties run $O(n)$ invocations of $\Pi_{\ell\text{BC+}}$ in parallel on ℓ_{EST}/n-bits inputs. Then, $\textsc{bits}_{\ell_{\text{EST}}}(\textsc{AddLastBlock}) = O(n) \cdot \textsc{bits}_{\ell_{\text{EST}}/n}(\Pi_{\ell\text{BC+}})$ and $\textsc{rounds}(\textsc{AddLastBlock}) = \textsc{rounds}(\Pi_{\ell\text{BC+}})$. Corollary 1 enables us to conclude that $\textsc{bits}_{\ell_{\text{EST}}}(\textsc{AddLastBlock}) = O(\ell_{\text{EST}} \cdot n + \kappa \cdot n^3 \log n) + O(n) \cdot \textsc{bits}_\kappa(\Pi_{\text{BA}})$ and $\textsc{rounds}(\textsc{AddLastBlock}) = O(1) \cdot \textsc{rounds}(\Pi_{\text{BA}})$. □

C.4 Subprotocol Complain

Lemma 6. *Assume a BA protocol Π_{BA}, and that honest parties join* Complain *with the same value ℓ_{EST} (that is a multiple of n) and with the same bitstring of $1 \leq i^\star \leq n$ blocks $\textsc{prefix}^1$ representing the prefix of some valid value's ℓ_{EST}-bit representation. In addition, assume that each party joins with some valid ℓ_{EST}-bit input $v_\perp$ such that the ℓ_{EST}-bit representations of $t+1$ honest parties' values $v_\perp$ do not have $\textsc{prefix}^1$ as a prefix. Then, the honest parties obtain the same valid value v_{OUT}.*

Complain has communication complexity $\textsc{bits}_{\ell_{EST}}(\textsc{Complain}) = O(\ell_{EST} \cdot n + \kappa \cdot n^3 \log n) + O(n) \cdot \textsc{bits}_\kappa(\Pi_{BA})$ and round complexity $\textsc{rounds}(\textsc{Complain}) = O(1) \cdot \textsc{rounds}(BA)$.

Proof. We first note that honest parties obtain the same multiset $\textsc{outputs\_from\_complaints}$ since all complaints are sent via $\Pi_{\ell\text{BC+}}$. This implies that the parties compute v_{OUT} identically. In addition, every honest complaint satisfies the $\textsc{length\_limit}$, so the honest complaints are delivered correctly.

We show that, if a complaint $(i, \textsc{block}_i)$ is sent by an honest party holding value $v_\perp$, then it leads to a valid value $v_{\text{OUT?}}$. Note that $i \leq i^\star$, and the binary representation of $v_\perp$, namely $\textsc{bits}_{\ell_{\text{EST}}}(v_\perp)$, has the bitstring $\textsc{prefix}^\star_1 \parallel \ldots \parallel \textsc{prefix}^\star_{i-1} \parallel \textsc{block}_i$ as prefix. Then, Remark 2 ensures that $v_{\text{OUT?}} := \textsc{min}_{\ell_{\text{EST}}}(\textsc{common\_prefix} \parallel 1) \in [\min(v_\perp, v^1), \max(v_\perp, v^1)]$, and therefore $v_{\text{OUT?}}$ is valid.

At least $t+1$ honest parties hold values $v_\perp$ that do not have $\textsc{prefix}^1$ as a prefix and send complaints. Therefore, all parties receive $t+1+k$ complaints, with $0 \leq k \leq n-(t+1)$. Out of these, $\min(k,t)$ are sent by corrupted parties and may lead to values outside the honest inputs' range. Note that v_{OUT} is well-defined, since $t+1+k \geq \min(t,k)+1$. To show that v_{OUT} is valid, we need to show that v_{OUT} is at least the $\min(k,t)+1$-th lowest value in $\textsc{outputs\_from\_complaints}$ (which is ensured by the way v_{OUT} is initialized), and at most the $\min(k,t)+1$-th highest value in $\textsc{outputs\_from\_complaints}$. This follows from the fact that $\textsc{outputs\_from\_complaints}$ contains $\min(k,t)$ values that are at least v_{OUT}, i.e. $|\textsc{outputs\_from\_complaints}| \geq 2 \cdot \min(k,t)+1$. This holds since, if $k \leq t$, $|\textsc{outputs\_from\_complaints}| = (t+1)+k \geq 2k+1$. Otherwise, if $k > t$, $|\textsc{outputs\_from\_complaints}| = (t+1)+k \geq 2t+1$, and therefore v_{OUT} is valid.

For the communication complexity and the round complexity, note that at most n parties send a block and an index of $\lceil \log_2 n \rceil + 1$ bits via $\Pi_{\ell\text{BC+}}$ (in parallel). Therefore, $\textsc{bits}_{\ell_{\text{EST}}}(\textsc{Complain}) = O(n) \cdot \textsc{bits}_{\ell_{\text{EST}}/n + \lceil \log_2 n \rceil + 1}\textsc{bits}(\Pi_{\ell\text{BC+}})$ and $\textsc{rounds}(\textsc{Complain}) = \textsc{rounds}(\Pi_{\ell\text{BC+}})$. Applying Corollary 1 leads to the results claimed in the lemma's statement. □

D PROTOCOL FOR $\mathbb{Z}$: MISSING PROOFS

We include the proof of Corollary 3.

Corollary 3. *Assume a BA protocol Π_{BA} resilient against $t < n/3$ corruptions. Then, if the honest parties hold inputs $v_{IN} \in \mathbb{Z}$ with $\textsc{bits}(|v_{IN}|)$ consisting of at most ℓ bits, $\Pi_{\mathbb{Z}}$ is a CA protocol resilient against $t < n/3$ corruptions, with communication complexity $\textsc{bits}_\ell(\Pi_{\mathbb{Z}}) = O(\ell n + \kappa \cdot n^3 \log n) + O(n \log n) \cdot \textsc{bits}_\kappa(\Pi_{BA})$, and round complexity $\textsc{rounds}(\Pi_{\mathbb{Z}}) = O(\log n) \cdot \textsc{rounds}(\Pi_{BA})$.*

Proof. We first show that $\Pi_{\mathbb{Z}}$ achieves CA. In $\Pi_{\mathbb{Z}}$, parties first agree on their values' sign with the help of Π_{BA}. If Π_{BA} returns $\textsc{sign}_{\text{OUT}} = 0$, then there is an honest party holding a non-negative input. If a party holds $v_{\text{IN}} < 0$, then $v_{\text{IN}}^{\mathbb{N}} := 0$ is a valid value. Parties then join $\Pi_{\mathbb{N}}$ with valid values $v_{\text{IN}}^{\mathbb{N}}$ and therefore agree on a valid output according to Theorem 2. Otherwise, if Π_{BA} returns $\text{SIGN}_{\text{OUT}} = 1$, there is an honest party holding a non-positive input. If a party holds $v_{\text{IN}} > 0$, then $v_{\text{IN}}^{\mathbb{N}} := 0$ is a valid value. Therefore, all honest parties hold valid values $(-1) \cdot v_{\text{IN}}^{\mathbb{N}}$. Parties then join $\Pi_{\mathbb{N}}$ with inputs $v_{\text{IN}}^{\mathbb{N}}$ and, according to Theorem 2, they agree on a value $v_{\text{OUT}}^{\mathbb{N}}$ such that $v_{\text{OUT}} := (-1) \cdot v_{\text{OUT}}^{\mathbb{N}}$ is valid.

$\Pi_{\mathbb{Z}}$ first runs Π_{BA} once with bits as inputs, and afterwards it runs $\Pi_{\mathbb{N}}$ on inputs of at most ℓ bits. Then, we obtain that $\text{BITS}_{\ell}(\Pi_{\mathbb{N}}) = \text{BITS}_{1}(\Pi_{\text{BA}}) + \text{BITS}_{\ell}(\Pi_{\mathbb{N}})$, and $\text{ROUNDS}(\Pi_{\mathbb{Z}}) = \text{ROUNDS}(\Pi_{\text{BA}}) + \text{ROUNDS}(\Pi_{\mathbb{N}})$. Theorem 2 leads to the results claimed in the corollary's statement. □