Communication-Optimal Convex Agreement

Diana Ghinea
ETH Zürich
ghinead@ethz.ch

Chen-Da Liu-Zhang
Lucerne University
of Applied Sciences and Arts
& Web3 Foundation
chendaliu@gmail.com

Roger Wattenhofer
ETH Zürich
wattenhofer@ethz.ch

Abstract

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 $O(\ell n^2)$. 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 $\ell = \Omega(\kappa \cdot n \log^2 n)$, where κ is the security parameter.

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), enabling n parties to agree on a value even if 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 reporting the room's temperature. One can expect minor errors in the measurements, such as correct sensors obtaining temperatures between $-10.05°C$ and $-10.03°C$. Standard BA would then allow the 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). CA-related problems have gained significant attention in recent years. The interest is driven by both theoretical curiosity [15, 43, 50] and practical concerns. Real-world applications of CA-related problems span diverse fields, including aviation control systems [36, 47], robotic coordination [44], blockchain oracles [5], transaction ordering in blockchain [14], and distributed machine learning [4, 18, 19, 48].

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, also known as *Byzantine Broadcast*). 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: if the honest parties hold inputs of at most ℓ bits, a lower bound on the communication complexity is $\Omega(\ell n + n^2)$ bits [41], and this approach incurs a sub-optimal cost of at least $O(\ell n^2)$ bits ($O(\ell n^2 + n^3)$ for deterministic protocols). In fact, the communication complexity of existing CA protocols [15, 41, 50] is adversarially chosen, as they involve steps where honest parties forward messages sent by corrupted parties. In real-world distributed systems, excessive communication can be detrimental: it may lead to network congestion and hence cause messages to be delayed or lost, compromising the system's reliability.

For regular BA and BC, this issue regarding communication complexity was solved in a line of works [8, 23, 24, 34, 41] via so-called *extension protocols*, that achieve a communication cost of $O(\ell n + \mathsf{poly}(n, \kappa))$ bits, where κ is a security parameter (these protocols make use of cryptographic primitives that compress the input values to $O(\kappa)$ bits). Note that this communication cost is optimal for large enough ℓ. While these results are adequate for real-world scenarios such as fault-tolerant distributed storage systems handling large files, they fall short in scenarios where CA is more suitable than BA. Our work will then focus on closing the communication complexity gap in the synchronous model for CA. Concretely, we ask the following question:

Can we achieve CA *with communication complexity* $O(\ell n + \mathsf{poly}(n, \kappa))$*,*
which is optimal for large enough ℓ*?*

We answer this question in the affirmative. More concretely, we introduce a deterministic protocol in the plain model (i.e. unauthenticated setting) that achieves the optimal resilience $t < n/3$, communication complexity $O(\ell n + n^2 \kappa \log^2 n)$, and round complexity $O(n \log n)$. The protocol makes use of collision-resistant hash functions and takes as inputs bitstrings 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 Agreement. The requirement of obtaining outputs within the honest inputs' range has been first introduced in [16] for Approximate Agreement (AA). AA relaxes the agreement requirement, where parties' outputs may deviate by a predefined error $\varepsilon > 0$. This relaxation allows for deterministic asynchronous protocols, circumventing the FLP result [22]. AA has been a subject of an extensive line of works, focusing on optimal convergence rates [6, 20, 21], higher resilience [1, 26, 33], and different input spaces, such as multidimensional inputs [27, 37, 50], or graphs and abstract convexity spaces [3, 15, 32, 43]. CA was formally defined by Vaidya and Garg in [38, 50] for multidimensional input values. Feasibility with optimal resilience has been considered for abstract convexity spaces as well [15, 43]. Another line of works has investigated the feasibility of an even stronger requirement for inputs in $\mathbb{R}$, i.e. that the output is *close* to the median of the honest inputs [14, 47], or, more generally, to the k-th lowest honest input [36].

Extension protocols. The problem of reducing the communication complexity of BA on multi-valued inputs was first addressed by Turpin and Coan [49], where the authors assume $t < n/3$ and give a reduction from long-messages BA to short-messages BA with a communication cost of $O(\ell n^2)$ bits. Fitzi and Hirt [23] 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 $t < n/3$ [24, 35, 41] setting and in the honest-majority setting [8, 24, 41]. Extension protocols have also been a topic of interest for problems related to BA, such as BC in the $t < n$ setting [11, 28], or asynchronous Reliable Broadcast [10, 41].

Protocols for short messages. Reducing the communication complexity is not only a topic of interest for long inputs, but also for short inputs (i.e., one bit or a constant number of bits). Dolev and Reischuk [17] showed that deterministic BA protocols (and hence also deterministic CA protocols) incur communication complexity $\Omega(t^2)$ if t of the parties involved are byzantine, therefore $\Omega(n^2)$ if $t = \Theta(n)$. This lower bound is tight [12, 40]. However, randomized protocols have offered a path to subquadratic communication [2,9,13,25,29–31] under different assumptions.

To the best of our knowledge, our work introduces the first CA protocol with asymptotically optimal-communication for sufficiently long messages, at least $\ell = \Omega(\kappa \cdot n \log^2 n)$. Our protocol is also deterministic. We leave the question of achieving communication-optimal CA protocols for shorter messages as an interesting open question.

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 $z^\star$ non-trivially distribute $v^\star$ to all the parties. The main issue when trying to adapt this approach to CA is that the short κ-bit encodings lost information about the ordering of 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 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 the byzantine parties prevent us from finding the exact longest common prefix of the honest inputs, the longest common prefix of any values in the honest inputs' range will suffice to obtain an output.

2 Preliminaries

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, all messages get delivered within Δ time, and 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. We consider a security parameter κ, a collision-resistant hash function

$H_\kappa : \{0,1\}^\star \to \{0,1\}^\kappa$, and we assume that the adversary is computationally bounded. Informally, $H_\kappa : \{0,1\}^\star \to \{0,1\}^\kappa$ is collision-resistant if, for any computationally-bounded adversary $\mathcal{A}$, the probability that $\mathcal{A}(1^\kappa)$ outputs two values $x \neq y$ such that $H_\kappa(x) = H_\kappa(y)$ is negligible in κ.[1] For simplicity of presentation, our proofs will assume that H_κ is *collision-free*; our protocols are secure conditioned on the event that no collision occurs.

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 Validity, 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 t of the n parties are corrupted: (Termination) All honest parties terminate; (Convex Validity) Honest parties' outputs lie in the honest inputs' convex hull; (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 t of the n parties are corrupted: (Termination) All honest parties terminate; (Validity) If all honest parties hold the same input value v, they output $v_{\text{OUT}} = v$; (Agreement) All honest parties output the same value.*

We denote the communication complexity for ℓ-bit inputs of a protocol Π by $\textsc{bits}_\ell(\Pi)$: this is the worst-case total number of bits sent by honest parties if they all hold inputs of at most ℓ bits. In addition, $\textsc{rounds}_\ell(\Pi)$ denotes the worst-case round complexity of Π.

Binary representations. We establish a few notations that will be used throughout the paper. 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 also define v's ℓ-bit representation $\textsc{bits}_\ell(v)$ as the ℓ-bit string obtained by prepending $\ell - k$ zeroes to $\textsc{bits}(v)$. In addition, for $1 \leq i \leq \ell$, $\textsc{b}_\ell^i(v)$ denotes the i-th leftmost bit in the ℓ-bit representation of v $\textsc{bits}_\ell(v)$.

The reverse operation of $\textsc{bits}(\cdot)$ will be $\textsc{val}(\textsc{bits})$: given a bitstring $\textsc{bits} := \textsc{b}_1\textsc{b}_2 \ldots \textsc{b}_k$ (where every $\textsc{b}_i \in \{0,1\}$), $\textsc{val}(\textsc{bits}) := \sum_{i=1}^{k} \textsc{b}_i \cdot 2^{k-i} = v$. We denote the length of a bitstring $\textsc{bits}$ by $|\textsc{bits}|$, and $\|$ is the concatenation operator.

We will also need to define $\textsc{max}_\ell(\textsc{bits})$ and $\textsc{min}_\ell(\textsc{bits})$. $\textsc{max}_\ell(\textsc{bits})$ is the highest ℓ-bit value having prefix $\textsc{bits}$, obtained by appending $\ell - |\textsc{bits}|$ ones to $\textsc{bits}$. Similarly, $\textsc{min}_\ell(\textsc{bits})$ is the lowest ℓ-bit value having prefix $\textsc{bits}$, obtained by appending $\ell - |\textsc{bits}|$ zeroes to $\textsc{bits}$.

3 CA for Long Inputs in $\mathbb{N}$ of Fixed Length

Building towards our CA protocol for $\mathbb{Z}$, we first focus on $\mathbb{N}$. For now, we assume that the inputs' length ℓ is fixed: there is a publicly known ℓ such that every honest input v_{IN} satisfies $v_{\text{IN}} < 2^\ell$. In this section, we present a protocol FixedLengthCA achieving CA under these assumptions. When $\ell \in \mathsf{poly}(n)$, FixedLengthCA has communication complexity $O(\ell \cdot n + \mathsf{poly}(n, \kappa))$ and round complexity $O(n \log n)$.

FixedLengthCA searches for a valid value by only working with values' prefixes. We first use the honest parties' inputs to identify a bitstring $\textsc{prefix}^\star$ that is the prefix of an ℓ-bit valid value. If $\textsc{prefix}^\star$ consists of ℓ bits, the parties may output $\textsc{val}(\textsc{prefix}^\star)$. Otherwise, we ensure that $\textsc{prefix}^\star$ satisfies a few special properties enabling the parties to efficiently find an output. Concretely, we ensure that sufficiently many honest parties *know* valid values that do not have

[1] See [46] for a formal definition.

$\text{PREFIX}^\star$ as a prefix: such values are either lower than any value with prefix $\text{PREFIX}^\star$, meaning that $\text{MIN}_\ell(\text{PREFIX}^\star)$ is valid, or higher than any value with prefix $\text{PREFIX}^\star$, meaning that $\text{MAX}_\ell(\text{PREFIX}^\star)$ is valid. These parties will announce which of these two options they believe to valid, enabling all parties to decide on the final output.

We split the implementation of FixedLengthCA into three subprotocols. The first one is FindPrefix, where the parties agree on a bitstring $\text{PREFIX}^\star$, and each party obtains two ℓ-bit valid values v and $v_\perp$ such that: (i) the values v have $\text{PREFIX}^\star$ as a prefix, and (ii) for any bitstring BITS of $|\text{PREFIX}^\star| + 1$ bits, there are $t+1$ honest parties whose values $v_\perp$ do not have BITS as a prefix. If $|\text{PREFIX}^\star| = \ell$, then the parties hold the same valid value v, and may simply output v. Otherwise, in our second subprotocol, AddLastBit, the parties append one bit to $\text{PREFIX}^\star$ using their values v, ensuring that the extended $\text{PREFIX}^\star$ is still some valid values' prefix. Now there are $t+1$ honest parties holding values $v_\perp$ that do not have prefix $\text{PREFIX}^\star$, and these differences are announced in the third subprotocol, GetOutput, where the parties agree on the final output.

FixedLengthCA(ℓ, v)

Code for party P

$\text{PREFIX}^\star$, v, $v_\perp$:= FindPrefix(v, ℓ). If $|\text{PREFIX}^\star| = \ell$, output v and terminate.
$\text{PREFIX}^\star$:= AddLastBit$(\text{PREFIX}^\star, v, \ell)$.
Return v := GetOutput$(\text{PREFIX}^\star, v_\perp, \ell)$.

In the following, we present each of these subprotocols in detail.

Subprotocol FindPrefix. Roughly, FindPrefix aims to identify the longest common prefix of the *honest parties' inputs* using binary search. Identifying the precise longest common prefix is impossible, as the byzantine parties can act as honest parties with inputs of their own choice. However, we can identify a valid value's prefix that is at least as long as the honest inputs' longest common prefix. Roughly, the parties will be looking for some index $i^\star$ such that running BA on their values' $i^\star$-bit prefixes would return an honest prefix, but running BA on their $(i^\star+1)$-bit prefixes would not offer the same guarantee. We need a BA protocol *for long messages* that satisfies two additional properties, defined below. The first one is *Intrusion Tolerance*: recall that, unless the honest parties hold identical inputs, BA's Validity condition does not impose any restrictions. Intrusion Tolerance requires that the output is either an honest party's input or a special symbol $\perp$. The second property, *Bounded Pre-Agreement*, prevents agreement on $\perp$ when an honest input can be easily identified.

Definition 3. *Intrusion Tolerance: Honest parties output an honest party's input or $\perp$.*

Definition 4. *Bounded Pre-Agreement: If the parties agree on $\perp$, then there are fewer than $n-2t$ honest parties holding the same input value.*

In Section 7, we describe a protocol $\Pi_{\ell\text{BA}+}$ achieving these guarantees with communication complexity $O(\ell n + \mathsf{poly}(n, \kappa))$, as described in Theorem 1. The main technical challenge behind $\Pi_{\ell\text{BA}+}$ is to design a protocol for κ-bit messages with communication complexity $O(\kappa \cdot n^2)$.

Theorem 1. *Given a* BA *protocol* Π_{BA} *resilient against* $t < n/3$ *corruptions, there is a* BA *protocol* $\Pi_{\ell\text{BA}+}$ *resilient against* $t < n/3$ *corruptions that achieves Intrusion Tolerance and Bounded Pre-Agreement, with communication complexity* $\text{BITS}_\ell(\Pi_{\ell\text{BA}+}) = O(\ell n + \kappa \cdot n^2 \log n) + \text{BITS}_\kappa(\Pi_{\text{BA}})$*, and round complexity* $\text{ROUNDS}_\ell(\Pi_{\ell\text{BA}+}) = O(1) + \text{ROUNDS}_\kappa(\Pi_{\text{BA}})$.

We proceed in $O(\log \ell)$ iterations. In the first iteration, the parties check whether $\Pi_{\ell\text{BA}+}$ returns $\perp$ on the first half of their values' bitstrings $\text{B}_1 \| \ldots \| \text{B}_{\text{MID}}$. If $\Pi_{\ell\text{BA}+}$ returns $\perp$, $\text{MID} - 1$ is an upper bound for index $i^\star$. The Bounded Pre-agreement property ensures that at most $n-2t$ honest parties hold values v with the same prefix of MID bits. This means that, for any bitstring BITS of MID bits, there are at least $(n-t)-(n-2t) \geq t+1$ honest parties holding values v that do not have BITS as a prefix. The parties set $v_\perp := v$ and then continue the search for $i^\star$ within bits $\text{B}_1, \dots, \text{B}_{\text{MID}-1}$ in the next iteration, using an identical approach.

Otherwise, if $\Pi_{\ell\text{BA}+}$ returns a bitstring of MID bits $\text{PREFIX}^\star_1 \| \dots \| \text{PREFIX}^\star_{\text{MID}}$, Intrusion Tolerance ensures that this is the prefix of an honest party's (valid) value $v^\star$. The parties holding values v with a different prefix update v to match this prefix: if $\text{VAL}(\text{B}_1 \| \dots \| \text{B}_{\text{MID}}) < \text{VAL}(\text{PREFIX}^\star_1 \| \dots \| \text{PREFIX}^\star_{\text{MID}})$, meaning that $v < v^\star$, then $\text{MIN}_\ell(\text{PREFIX}^\star_1 \| \dots \| \text{PREFIX}^\star_{\text{MID}})$ is in $[v, v^\star]$ and therefore is valid. Otherwise, if $\text{VAL}(\text{B}_1 \| \dots \| \text{B}_{\text{MID}}) > \text{VAL}(\text{PREFIX}^\star_1 \| \dots \| \text{PREFIX}^\star_{\text{MID}})$, meaning that $v > v^\star$, then $\text{MAX}_\ell(\text{PREFIX}^\star_1 \| \dots \| \text{PREFIX}^\star_{\text{MID}})$ is in $[v^\star, v]$ and therefore is valid. The parties then proceed to the next iteration, where they continue the search for $i^\star$ on the second half of their (updated) values' ℓ-bit representation. After $O(\log \ell)$ iterations, either $\Pi_{\ell\text{BA}+}$ never returned $\perp$ and the parties hold identical values v, or $i^\star$ is found. We present the code below.

FINDPREFIX(ℓ, v)

Code for party P

$\text{LEFT} := 1, \text{RIGHT} := \ell + 1$; $v := v_{\text{IN}}$; $v_\perp := v_{\text{IN}}$; $\text{PREFIX}^\star :=$ empty string.
loop
  If $\text{LEFT} = \text{RIGHT}$, exit the loop.
  $(\text{B}_1, \text{B}_2, \dots, \text{B}_\ell) := \text{BITS}_\ell(v)$.
  Join $\Pi_{\ell\text{BA}+}$ with input $\text{B}_{\text{LEFT}} \| \dots \| \text{B}_{\text{MID}}$, 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 $\text{MID} - \text{LEFT} + 1$ bits $\text{PREFIX}^\star_{\text{LEFT}} \| \dots \| \text{PREFIX}^\star_{\text{MID}}$:
    $\text{PREFIX}^\star := \text{PREFIX}^\star \| \text{PREFIX}^\star_{\text{LEFT}} \| \dots \| \text{PREFIX}^\star_{\text{MID}}$.
    If $\text{VAL}(\text{B}_1 \| \dots \| \text{B}_{\text{MID}}) < \text{VAL}(\text{PREFIX}^\star)$: $v := \text{MIN}_\ell(\text{PREFIX}^\star)$.
    If $\text{VAL}(\text{B}_1 \| \dots \| \text{B}_{\text{MID}}) > \text{VAL}(\text{PREFIX}^\star)$: $v := \text{MAX}_\ell(\text{PREFIX}^\star)$.
    Set $\text{LEFT} := \text{MID} + 1$.
end loop
Return $\text{PREFIX}^\star$, v, $v_\perp$.

The formal proof of the lemma below is included in Appendix A.2.

Lemma 1. *Assume a* BA *protocol* Π_{BA}*, and that honest parties join* FINDPREFIX *with the same* ℓ*, and with valid* ℓ*-bit values* v*. Then, the honest parties obtain the same bitstring* $\text{PREFIX}^\star$*, and each honest party obtains two valid* ℓ*-bit values* v*,* $v_\perp$ *such that: (i)* $\text{PREFIX}^\star$ *is a prefix of* $\text{BITS}_\ell(v)$*; (ii) for any bitstring* BITS *of* $|\text{PREFIX}^\star| + 1$ *bits, at least* $t+1$ *honest parties hold values* $v_\perp$ *such that* $\text{BITS}_\ell(v_\perp)$ *does not have prefix* BITS*.*

FINDPREFIX *achieves communication complexity* $\text{BITS}_\ell(\text{FINDPREFIX}) = O(\ell \cdot n + \kappa \cdot n^2 \log n \log \ell) + O(\log \ell) \cdot \text{BITS}_\kappa(\Pi_{\text{BA}})$*, and round complexity* $\text{ROUNDS}_\ell(\text{FINDPREFIX}) = O(\log \ell) \cdot \text{ROUNDS}_\kappa(\Pi_{\text{BA}})$*.*

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 in iteration i, 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 $1 \leq \text{LEFT} \leq \text{RIGHT} \leq \ell + 1$, and the same bitstring $\text{PREFIX}^\star$ consisting of $\text{LEFT} - 1$ bits.
(B) $0 \leq \text{RIGHT} - \text{LEFT} \leq 2^{\lceil \log_2 \ell \rceil - (i-1)}$.
(C) Honest parties hold valid ℓ-bit values v such that $\text{BITS}_\ell(v)$ has $\text{PREFIX}^\star$ as a prefix.

(D) Honest parties hold valid ℓ-bit values $v_\perp$, and, for any bitstring BITS of RIGHT bits, there are $t+1$ honest parties holding values $v_\perp$ such that $\text{BITS}_\ell(v_\perp)$ does not have prefix BITS.

Property (B) implies that the condition LEFT = RIGHT is met by iteration $i = \lceil \log_2 \ell \rceil + 2$. Once this condition is met, Property (A) ensures that parties hold the same bitstring $\text{PREFIX}^\star$ of $\text{LEFT} - 1$ bits. Property (C) ensures that parties hold valid ℓ-bit values v with prefix $\text{PREFIX}^\star$. Finally, property (D) ensures that, honest parties hold valid ℓ-bit values $v_\perp$ such that for any bitstring BITS of $\text{RIGHT} = \text{LEFT} = \left|\text{PREFIX}^\star\right| + 1$ bits, there are $t+1$ honest parties whose values $v_\perp$ do not have $\text{PREFIX}^\star$ as a prefix.

Since each iteration invokes $\Pi_{\ell\text{BA}+}$ once, $\text{ROUNDS}_\ell(\text{FindPrefix}) = O(\log \ell) \cdot \text{ROUNDS}_\ell(\Pi_{\ell\text{BA}+})$, and applying Theorem 1 gives our claimed round complexity. In each iteration $i < \lceil \log_2 n \rceil + 2$, Property (B) ensures that FindPrefix runs $\Pi_{\ell\text{BA}+}$ on bitstrings of at most $2^{\lceil \log_2 \ell \rceil - i} \leq \ell / 2^{i-1}$ bits. Therefore, $\text{BITS}_\ell(\text{FindPrefix}) = \sum_{i=1}^{\lceil \log_2 \ell \rceil + 1} \text{BITS}_{\ell/2^{i-1}}(\Pi_{\ell\text{BA}+})$. Using Theorem 1 and that $\sum_{i=0}^{\infty} 1/2^i \leq 2$, we obtain our claimed communication cost. □

Extending the prefix agreed upon. We now describe the subprotocol AddLastBit, where parties extend the bitstring $\text{PREFIX}^\star$ agreed upon in FindPrefix with one bit (assuming $\left|\text{PREFIX}^\star\right| < \ell$). The resulting bitstring should still be a valid values' prefix. As each party holds a valid value v with prefix $\text{PREFIX}^\star || 0$ or $\text{PREFIX}^\star || 1$, we extend $\text{PREFIX}^\star$ by using a bit BA protocol Π_{BA}.

AddLastBit($\ell, v, \text{PREFIX}^\star$)

Code for party P.

1: Join Π_{BA} with input $\text{B}_\ell^{i^\star+1}(v)$, where $i^\star = \left|\text{PREFIX}^\star\right|$, and obtain output $\text{B}^\star$. Return $\text{PREFIX}^\star \parallel \text{B}^\star$.

The proof of the lemma below is included in Appendix A.2.

Lemma 2. *Assume a* BA *protocol* Π_{BA}*, and that honest parties join* AddLastBit *with the same value* ℓ*, the same bitstring* $\text{PREFIX}^\star$ *of* $i^\star < \ell$ *bits, and with valid* ℓ*-bit values v such that* $\text{BITS}_\ell(v)$ *has prefix* $\text{PREFIX}^\star$*. Then, the honest parties agree on a bitstring of* $i^\star + 1$ *bits that is the prefix of a valid value's* ℓ*-bit representation.* AddLastBit *has communication complexity* $\text{BITS}_\ell(\text{AddLastBit}) = \text{BITS}_1(\Pi_{\text{BA}})$ *and round complexity* $\text{ROUNDS}_\ell(\text{AddLastBit}) = \text{ROUNDS}_1(\Pi_{\text{BA}})$.

Obtaining the final output. After running AddLastBit, the parties hold a bitstring $\text{PREFIX}^\star$ of $i^\star + 1$ bits that is a valid value's prefix. Moreover, $t+1$ honest parties hold valid values $v_\perp$ that do not have prefix $\text{PREFIX}^\star$. These parties' values $v_\perp$ are either lower than $\text{MIN}_\ell(\text{PREFIX}^\star)$ or higher than $\text{MAX}_\ell(\text{PREFIX}^\star)$, meaning that at least one of these two options is valid. Each of these parties may announce (by sending a bit) which of the two options it believes to be valid. Then, every party becomes aware of a valid option by looking at the bit received the most (and therefore sent by at least one honest party). Afterwards, the parties use Π_{BA} to agree on a valid option.

GetOutput($\ell, v_\perp, \text{PREFIX}^\star$)

Code for party P

1: If $\text{PREFIX}^\star$ is not a prefix of $\text{BITS}_\ell(v_\perp)$:
2: Set $\text{B} := 0$ if $v_\perp < \text{MIN}_\ell(\text{PREFIX}^\star)$ and $\text{B} := 1$ otherwise.
3: Send B to all parties.
4: Set $m :=$ the number of bits B received, $\text{CHOICE} :=$ a bit B received from $\lceil m/2 \rceil$ parties.
5: Join Π_{BA} with input CHOICE. If the bit agreed upon is 0, return $\text{MIN}_\ell(\text{PREFIX}^\star)$. Otherwise, return $\text{MAX}_\ell(\text{PREFIX})^\star$.

The proof of the lemma below is included in Appendix A.2.

Lemma 3. *Assume a* BA *protocol* Π_{BA}*, and that honest parties join* GetOutput *with the same value* ℓ*, and with the same bitstring* $\textsc{prefix}^\star$ *representing the prefix of some valid value's* ℓ*-bit representation. In addition, assume that each party joins with some valid* ℓ*-bit input* $v_\perp$ *such that the* ℓ*-bit representations of* $t+1$ *honest parties' values* $v_\perp$ *do not have* $\textsc{prefix}^\star$ *as a prefix. Then, the honest parties obtain the same valid value* v_{OUT}*.*

GetOutput *has communication complexity* $\textsc{bits}_\ell(\textsc{GetOutput}) = O(n^2) + \textsc{bits}_1(\Pi_{\text{BA}})$ *and round complexity* $\textsc{rounds}_\ell(\textsc{GetOutput}) = O(1) + \textsc{rounds}_1(\text{BA})$*.*

Protocol Analysis. We have now reached the final theorem of this section, which presents the guarantees of FixedLengthCA. We highlight that, when $\ell \in \mathsf{poly}(n)$ and therefore $\log \ell \in O(\log n)$, we achieve communication complexity $\textsc{bits}_\ell(\textsc{FixedLengthCA}) = O(\ell n + \kappa \cdot n^2 \log^2 n) + O(\log n) \cdot \textsc{bits}_\kappa(\Pi_{\text{BA}})$, and round complexity $\textsc{rounds}_\ell(\textsc{FixedLengthCA}) = O(\log n) \cdot \textsc{rounds}_\kappa(\Pi_{\text{BA}})$.

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

Proof. Lemma 1 ensures that FindPrefix enables the parties to agree on a bitstring $\textsc{prefix}^\star$, and provides them with valid ℓ-bit values $v, v_\perp$ such that the values v have prefix $\textsc{prefix}^\star$. If $|\textsc{prefix}^\star| = \ell$, the honest parties hold the same valid value v, and therefore CA is achieved. Otherwise, Lemma 2 ensures that parties obtain the same bitstring $\textsc{prefix}^\star$ such that there are $t+1$ honest parties whose values $v_\perp$ do not have $\textsc{prefix}^\star$ as a prefix. Then, GetOutput's preconditions are met, and Lemma 3 ensures that CA is achieved. The communication complexity and the round complexity follow from summing up the complexities of each subprotocol. □

4 CA for Very Long Inputs in $\mathbb{N}$ of Fixed length

The protocol FixedLengthCA presented in Section 3 achieves our communication complexity goal when $\ell \in \mathsf{poly}(n)$. We now consider values $\ell \geq n^2$ that may not satisfy this condition, and we build a round-efficient CA protocol with communication complexity $O(\ell n + \mathsf{poly}(n, \kappa))$ by making small adjustments to the protocol of Section 3. We maintain the same assumptions: parties hold ℓ-bit inputs in $\mathbb{N}$, and ℓ is publicly known.

We first describe the adjustments for FindPrefix. Instead of comparing substrings *of bits* in each iteration, we compare substrings *of blocks*. For simplicity, we assume that ℓ is a multiple of n^2. Each party will split its ℓ-bit value into n^2 blocks $\textsc{block}_1, \textsc{block}_2, \ldots, \textsc{block}_{n^2}$ of ℓ/n^2 bits each. For an ℓ-bit value $v \in \mathbb{N}$, we define $\textsc{blocks}(v) := (\textsc{block}_1, \textsc{block}_2, \ldots, \textsc{block}_{n^2})$ such that $\textsc{bits}_\ell(v) = \textsc{block}_1 \,\|\, \textsc{block}_2 \,\|\, \ldots \,\|\, \textsc{block}_{n^2}$, and, for any $1 \leq i \leq n^2$, $|\textsc{block}_i| = \ell/n^2$. We denote $\textsc{block}_i$ by $\textsc{block}_i(v)$, and we use the term *block* to refer to such sequences of ℓ/n^2 bits.

Then, in each iteration, instead of comparing via $\Pi_{\ell\text{BA}+}$ the sequences of bits $\textsc{b}_{\text{LEFT}} \,\|\, \ldots \,\|\, \textsc{b}_{\text{MID}}$ of honest parties' values v, we compare sequences of blocks $\textsc{block}_{\text{LEFT}} \,\|\, \ldots \,\|\, \textsc{block}_{\text{MID}}$. This change reduces the number of iterations from $O(\log \ell)$ to $O(\log n)$: after $O(\log n)$ iterations, parties agree on a bitstring $\textsc{prefix}^\star$ of $i^\star$ *blocks*, and each party obtains two ℓ-bit valid values $v, v_\perp$. The values v have prefix $\textsc{prefix}^\star$, and, for any bitstring of $i^\star + 1$ *blocks*, there are $t+1$ honest parties whose values $v_\perp$ do not have that bitstring as prefix. We present the modified subprotocol below.

FindPrefixBlocks(ℓ_{EST}, v)

Code for party P

```
LEFT := 1, RIGHT := n + 1; v := v_IN, v_⊥ := v_IN, PREFIX* := empty string.
loop
    If LEFT = RIGHT, set i* := LEFT and exit the loop.
    (BLOCK_1, BLOCK_2, ..., BLOCK_n) := BLOCKS(v).
    Join Π_ℓBA+ with input BLOCK_LEFT ‖ ... ‖ BLOCK_MID, where MID := ⌊(LEFT + RIGHT)/2⌋.
    If Π_ℓBA+ has returned ⊥, set v_⊥ := v and RIGHT := MID.
    Otherwise, if Π_ℓBA+ has returned MID − LEFT + 1 blocks PREFIX*_LEFT ‖ ... ‖ PREFIX*_MID:
        PREFIX* := PREFIX* ‖ PREFIX*_LEFT ‖ ... ‖ PREFIX*_MID.
        If VAL(BLOCK_1 ‖ ... ‖ BLOCK_MID) < VAL(PREFIX*): v := MIN_ℓEST(PREFIX*).
        If VAL(BLOCK_1 ‖ ... ‖ BLOCK_MID) > VAL(PREFIX*): v := MAX_ℓEST(PREFIX*).
        Set LEFT := MID + 1.
end loop
Return PREFIX*, v, v_⊥.
```

The lemma below is the *block version* of Lemma 1. The proof is included in Appendix A.3.

Lemma 4. *Assume a* BA *protocol* Π_{BA}*, and that honest parties join* FindPrefixBlocks *with the same (multiple of* n^2*)* ℓ*, and with valid* ℓ*-bit values* v*. Then, the honest parties obtain the same bitstring* PREFIX* *of* $i^\star$ *blocks, and each honest party obtains two valid* ℓ*-bit values* v, $v_\perp$ *such that: (i) the* ℓ*-bit representations of the values* v *have prefix* PREFIX**; (ii) for any bitstring* BITS *of* $i^\star + 1$ *blocks, at least* $t + 1$ *honest parties hold values* $v_\perp$ *such that* $\text{BITS}_\ell(v_\perp)$ *does not have prefix* BITS*.*

FindPrefixBlocks *achieves communication complexity* $\text{BITS}_\ell(\text{FindPrefixBlocks}) = O(\ell \cdot n + \kappa \cdot n^2 \log^2 n) + O(\log n) \cdot \text{BITS}_\kappa(\Pi_{\text{BA}})$ *and round complexity* $\text{ROUNDS}_\ell(\text{FindPrefixBlocks}) = O(\log n) \cdot \text{ROUNDS}_\kappa(\Pi_{\text{BA}})$*.*

AddLastBit becomes AddLastBlock: in order to decide on a final output using the values $v_\perp$, we need to append one block to PREFIX*, so that the extended PREFIX* is a valid values' prefix. As the honest parties values' v have PREFIX* as a common prefix of $i^\star$ blocks, any block within the range of values $\text{VAL}(\text{BLOCK}_{i^\star+1}(v))$ of the honest parties' values v suffices. Finding such a block comes down to solving CA on inputs of ℓ/n^2 bits. Since we only run this step once, and on inputs of ℓ/n^2 bits, we may use a high communication complexity approach. For instance, we may use the protocol of [47], with minor adjustments. We describe this protocol in Appendix A.4.

Theorem 3 (Theorem 4 of [47])**.** *There is a* CA *protocol* HighCostCA *for* $\mathbb{N}$ *resilient against* $t < n/3$ *corruptions, with communication complexity* $\text{BITS}_\ell(\text{HighCostCA}) = O(\ell \cdot n^3)$*, and round complexity* $\text{ROUNDS}_\ell(\text{HighCostCA}) = O(n)$*.*

We present AddLastBlock below. The proof of Lemma 5 is deferred to Appendix A.3.

AddLastBlock(ℓ, v, PREFIX*)

Code for party P

```
Set i* := |PREFIX*|/(ℓ/n^2), i.e., the number of blocks in PREFIX*.
Set BLOCK'_{i*+1} := HighCostCA(BLOCK_{i*+1}(v)). Return PREFIX* ‖ BLOCK'_{i*+1}.
```

Lemma 5. *Assume that the honest parties join* AddLastBlock *with the same value* ℓ *(that is a multiple of* n^2*), with the same bitstring prefix* PREFIX* *of* $i^\star < n^2$ *blocks, and with valid* ℓ*-bit values*

v that have $\textsc{prefix}^\star$ *as a prefix. Then, the honest parties agree on a bitstring of* $i^\star + 1$ *blocks that is the prefix of a valid value's* ℓ*-bit representation.* $\textsc{AddLastBlock}$ *has communication complexity* $\textsc{bits}_\ell(\textsc{AddLastBlock}) = O(\ell \cdot n)$ *and round complexity* $\textsc{rounds}_\ell(\textsc{AddLastBlock}) = O(n)$*.*

Afterwards, as in $\textsc{FixedLengthCA}$, the parties obtain their output using the subprotocol $\textsc{GetOutput}$ presented in Section 3. We present the code of $\textsc{FixedLengthCABlocks}$ below.

$\textsc{FixedLengthCABlocks}(\ell, v)$

Code for party P

```
PREFIX*, v, v⊥ := FindPrefixBlocks(v, ℓ). If |PREFIX*| := ℓ, output v and terminate.
PREFIX* := AddLastBlock(PREFIX*, v, ℓ).
Return v := GetOutput(PREFIX*, v⊥, ℓ).
```

Similarly to the proof of Theorem 2, we can prove the following theorem by showing that the preconditions of each subprotocol are met. We have included the proof in Appendix A.3.

Theorem 4. *Assume a* BA *protocol* Π_{BA} *resilient against* $t < n/3$ *corruptions. If the honest parties hold* ℓ*-bit inputs* $v_{\text{IN}} \in \mathbb{N}$*, where* ℓ *is a publicly known multiple of* n^2*,* $\textsc{FixedLengthCABlocks}$ *is a* CA *protocol resilient against* $t < n/3$ *corruptions.*

The protocol achieves communication complexity $\textsc{bits}_\ell(\textsc{FixedLengthCABlocks}) = O(\ell n + \kappa \cdot n^2 \log^2 n) + O(\log n) \cdot \textsc{bits}_\kappa(\Pi_{\text{BA}})$*, and round complexity* $\textsc{rounds}_\ell(\textsc{FixedLengthCABlocks}) = O(n) + O(\log n) \cdot \textsc{rounds}_\kappa(\Pi_{\text{BA}})$*.*

5 Final CA Protocol for $\mathbb{N}$

In the previous sections, we have presented two protocols achieving CA given that the honest parties hold ℓ-bit input values in $\mathbb{N}$ and ℓ is publicly known. $\textsc{FixedLengthCA}$ matches our communication complexity goal when $\ell \in \mathsf{poly}(n)$, while $\textsc{FixedLengthCABlocks}$ does not impose an upper bound on ℓ, but instead implicitly requires that $\ell \geq n^2$. Our final protocol combines these two, and removes the assumption that ℓ is publicly known.

The parties decide which protocol to run using a bit BA protocol Π_{BA}: each party joins Π_{BA} with input 0 if $|\textsc{bits}(v_{\text{IN}})| \leq n^2$, and input 1 otherwise. If Π_{BA} returns 0, the parties obtain the estimation ℓ_{EST} in $O(\log n) \cdot \textsc{rounds}_1(\Pi_{\text{BA}})$ rounds by comparing their inputs' length with powers of two, and afterwards they run $\textsc{FixedLengthCA}$. Otherwise, if Π_{BA} returns 1, the parties obtain the estimation ℓ_{EST} by agreeing on a block size using the high-communication-cost protocol $\textsc{HighCostCA}$, and afterwards they run $\textsc{FixedLengthCABlocks}$.

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

Code for party P with input v_{in}

```
Join Π_BA with input 0 if |BITS(v_IN)| ≤ n^2 and otherwise with input 1.
If the bit agreed upon is 0:
    If |BITS(v_IN)| > n^2, set v_IN := 2^(n^2) − 1.
    For i = 0...⌈log_2 n^2⌉:
        Join Π_BA with input 0 if |BITS(v_IN)| ≤ 2^i and with input 1 otherwise. If Π_BA returns 0:
            Let ℓ_EST := 2^i. If |BITS(v_IN)| > ℓ_EST, set v_IN := 2^(ℓ_EST) − 1.
            Output v_OUT := FixedLengthCA(ℓ_EST, v) and terminate.
If the bit agreed upon is 1:
Set BLOCKSIZE := ⌈|BITS(v_IN)|/n^2⌉, and BLOCKSIZE' := HighCostCA(BLOCKSIZE).
```

10: Set $\ell_{\text{EST}} := \text{BLOCKSIZE}' \cdot n^2$. If $|\text{BITS}(v_{\text{IN}})| \geq \ell_{\text{EST}}$, $v_{\text{IN}} := 2^{\ell_{\text{EST}}} - 1$.
11: Output $v_{\text{OUT}} := \text{FIXEDLENGTHCABLOCKS}(\ell_{\text{EST}}, v)$ and terminate.

Theorem 5. *Assume a* BA *protocol* Π_{BA} *resilient against* $t < n/3$ *corruptions. Then, if the honest parties hold* ℓ*-bit inputs* $v_{\text{IN}} \in \mathbb{N}$*,* $\Pi_{\mathbb{N}}$ *is a* CA *protocol resilient against* $t < n/3$ *corruptions, with communication complexity* $\text{BITS}_\ell(\Pi_{\mathbb{N}}) = O(\ell n + \kappa \cdot n^2 \log^2 n) + O(\log n) \cdot \text{BITS}_\kappa(\Pi_{\text{BA}})$*, and round complexity* $\text{ROUNDS}_\ell(\Pi_{\mathbb{N}}) = O(n) + O(\log n) \cdot \text{ROUNDS}_\kappa(\Pi_{\text{BA}})$*.*

Proof. Let $\ell_{\min}$ and $\ell_{\max}$ denote the lengths of the lowest and the highest honest inputs respectively.

We first assume that the Π_{BA} invocation in line 1 returns 0. Then, at least one honest party has joined with input 0 due to Π_{BA}'s Validity condition and therefore $\ell_{\min} \leq n^2$. If any honest party holds an input value longer than n^2 bits, $2^{n^2} - 1$ is within the honest inputs' range. Hence, the parties join the loop in line 4 with valid values of at most n^2 bits. If Π_{BA} returns 0 in some iteration i, then at least one honest party has joined with input 0 and therefore $\ell_{\min} \leq 2^i$. Then, if an honest party holds an input value longer than i bits, $2^i - 1$ is in the honest inputs range. Note that Π_{BA} returns 0 by iteration $i = \lceil \log_2 \min(\ell_{\max}, n^2) \rceil$ the latest, since all honest parties hold values of at most $\min(\ell_{\max}, n^2)$ bits. This ensures that, the parties agree on an estimation $\ell_{\text{EST}} \leq 2 \cdot \min(\ell_{\max}, n^2) \leq 2n^2$ with $O(\log n)$ iterations. Finally, parties join FIXEDLENGTHCA with the same value $\ell_{\text{EST}} \leq n^2$ and valid ℓ_{EST}-bit values. The parties agree on a valid output v_{OUT}, which ensures that CA is achieved. The communication complexity in this case is $O(\log n) \cdot \text{BITS}_1(\Pi_{\text{BA}}) + \text{BITS}_{2 \cdot \min(\ell, n^2)}(\text{FIXEDLENGTHCA}) = O(\ell n + \kappa n^2 \log^2 n) + O(\log n) \cdot \text{BITS}_\kappa(\Pi_{\text{BA}})$, while the round complexity is $O(\log n) \cdot \text{ROUNDS}_1(\Pi_{\text{BA}}) + \text{ROUNDS}_{2 \cdot \min(\ell, n^2)}(\text{FIXEDLENGTHCA}) = O(\log n) \cdot \text{ROUNDS}_\kappa(\Pi_{\text{BA}})$.

Otherwise, if the Π_{BA} invocation in line 1 returns 1, then there is an honest party holding an input value longer than n^2 bits: $n^2 < \ell_{\max} \leq \ell$. Parties join HIGHCOSTCA with inputs $\text{BLOCKSIZE} := \lceil |\text{BITS}(v_{\text{IN}})| / n^2 \rceil$ and obtain a value $\text{BLOCKSIZE}'$ satisfying $\lceil \ell_{\min}/n^2 \rceil \leq \text{BLOCKSIZE}' \leq \lceil \ell_{\max}/n^2 \rceil$. Note that the values BLOCKSIZE can be represented via $O(\log(\ell/n^2))$ bits, hence $O(\ell/n^2)$ bits, and therefore this step has communication cost $O(\ell n)$. Therefore, $\ell_{\text{EST}} := \text{BLOCKSIZE}' \cdot n^2$ satisfies $\ell_{\min} \leq \ell_{\text{EST}} \leq \ell_{\max} + n^2 \leq 2 \cdot \ell$. Since $\ell_{\text{EST}} \geq \ell_{\min}$, if an honest party's input value is longer than ℓ_{EST} bits, $2^{\ell_{\text{EST}}} - 1$ is guaranteed to be in the honest inputs' range. It follows that the parties join FIXEDLENGTHCABLOCKS with the same value $\ell_{\text{EST}} \leq 2 \cdot \ell$ (that is a multiple of n^2) and valid ℓ_{EST}-bit values. The parties agree on a valid value v_{OUT} and therefore CA is achieved. The total bit complexity is $\text{BITS}_1(\Pi_{\text{BA}}) + \text{BITS}_{O(\ell/n^2)}(\text{HIGHCOSTCA}) + \text{BITS}_{2\ell}(\text{FIXEDLENGTHCABLOCKS}) = O(\ell n + \kappa n^2 \log^2 n)$, and the round complexity is $\text{ROUNDS}_1(\Pi_{\text{BA}}) + \text{ROUNDS}_{O(\ell/n^2)}(\text{HIGHCOSTCA}) + \text{ROUNDS}(\text{FIXEDLENGTHCABLOCKS}) = O(n) + O(\log n) \cdot \text{ROUNDS}_\kappa(\Pi_{\text{BA}})$. □

6 CA Protocol for $\mathbb{Z}$

To extend the input space to $\mathbb{Z}$, we assume that the parties' inputs v_{IN} are represented as $(-1)^{\text{SIGN}_{\text{IN}}} \cdot v_{\text{IN}}^{\mathbb{N}}$, where $\text{SIGN}_{\text{IN}} \in \{0, 1\}$ and $v_{\text{IN}}^{\mathbb{N}} \in \mathbb{N}$. 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 SIGN_{OUT}, differs from a party P's SIGN_{IN}, P updates $v_{\text{IN}}^{\mathbb{N}}$ to 0, since it is guaranteed to be valid. Afterwards, the parties join $\Pi_{\mathbb{N}}$ with their possibly updated inputs $v_{\text{IN}}^{\mathbb{N}}$ and agree on $v_{\text{OUT}}^{\mathbb{N}}$ such that $v_{\text{OUT}} := (-1)^{\text{SIGN}_{\text{OUT}}} \cdot v_{\text{IN}}^{\mathbb{N}}$ is valid. We present the code and the guarantees of $\Pi_{\mathbb{Z}}$ below. The formal proof is included in Appendix A.5.

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

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

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

Corollary 1. *Assume a* BA *protocol* Π_{BA} *resilient against* $t < n/3$ *corruptions. Then, if the honest parties hold inputs* $(-1)^{\text{SIGN}_{\text{IN}}} \cdot v_{\text{IN}}^{\mathbb{N}} \in \mathbb{Z}$*, such that* $v_{\text{IN}}^{\mathbb{N}} \in \mathbb{N}$ *with* $\left|\text{BITS}(v_{\text{IN}}^{\mathbb{N}})\right| \leq \ell$*,* $\Pi_{\mathbb{Z}}$ *is a* CA *protocol resilient against* $t < n/3$ *corruptions, with communication complexity* $\text{BITS}_{\ell}(\Pi_{\mathbb{Z}}) = O(\ell n + \kappa \cdot n^2 \log^2 n) + O(\log n) \cdot \text{BITS}_{\kappa}(\Pi_{\text{BA}})$*, and round complexity* $\text{ROUNDS}_{\ell}(\Pi_{\mathbb{Z}}) = O(n) + O(\log n) \cdot \text{ROUNDS}_{\kappa}(\Pi_{\text{BA}})$.

We state the final corollary, where we instantiate the assumed BA protocol Π_{BA} with a deterministic BA protocol with quadratic communication (e.g. [12]).

Corollary 2. *If the honest parties hold inputs* $(-1)^{\text{SIGN}_{\text{IN}}} \cdot v_{\text{IN}}^{\mathbb{N}} \in \mathbb{Z}$*, such that* $v_{\text{IN}}^{\mathbb{N}} \in \mathbb{N}$ *with* $\left|\text{BITS}(v_{\text{IN}}^{\mathbb{N}})\right| \leq \ell$*,* $\Pi_{\mathbb{Z}}$ *is a* CA *protocol resilient against* $t < n/3$ *corruptions, with communication complexity* $\text{BITS}_{\ell}(\Pi_{\mathbb{Z}}) = O(\ell n + \kappa \cdot n^2 \log^2 n)$*, and round complexity* $\text{ROUNDS}_{\ell}(\Pi_{\mathbb{Z}}) = O(n \log n)$.

7 BA for Long Messages with Additional Properties

We recall that our CA protocol relies on a BA protocol $\Pi_{\ell\text{BA}+}$ with communication complexity $O(\ell n + \mathsf{poly}(n, \kappa))$ that satisfies two additional properties: *Intrusion Tolerance* and *Bounded Pre-Agreement*, introduced in Section 3. We restate the theorem describing this protocol below.

Theorem 1. *Given a* BA *protocol* Π_{BA} *resilient against* $t < n/3$ *corruptions, there is a* BA *protocol* $\Pi_{\ell\text{BA}+}$ *resilient against* $t < n/3$ *corruptions that achieves Intrusion Tolerance and Bounded Pre-Agreement, with communication complexity* $\text{BITS}_{\ell}(\Pi_{\ell\text{BA}+}) = O(\ell n + \kappa \cdot n^2 \log n) + \text{BITS}_{\kappa}(\Pi_{\text{BA}})$*, and round complexity* $\text{ROUNDS}_{\ell}(\Pi_{\ell\text{BA}+}) = O(1) + \text{ROUNDS}_{\kappa}(\Pi_{\text{BA}})$.

In this section, we describe the construction behind this theorem. The main technical challenge lies in building a communication-efficient BA protocol *for short messages* (κ bits) that achieves the two additional properties, denoted by $\Pi_{\text{BA}+}$. Afterwards, $\Pi_{\ell\text{BA}+}$ is constructed using the outline of prior works [8, 41].

Protocol $\Pi_{\text{BA}+}$. In our protocol $\Pi_{\text{BA}+}$, the parties first distribute their input values. Each party P receives $n - t$ values v from honest parties, plus at most t values from corrupted parties, and checks whether there is any value received from $n - 2t$ parties. Note that P sees at most two values v_1, v_2 with this property. Moreover, if there is some value v held as input by $n - 2t$ honest parties, all honest parties observe this value.

The parties then *vote* for the values they have seen $n - 2t$ times by sending either $\text{VOTE}(\cdot)$, $\text{VOTE}(v_1)$, or $\text{VOTE}(v_1, v_2)$, depending on how many such values they have seen. Each party P looks at the values that have received $n - t$ votes: there will be at most two such values. If there are two, P denotes them by a and b such that $a \leq b$. If there is only one such value v, P sets $a := v$ and $b := v$. If there are none, P simply sets $a := \perp$ and $b := \perp$.

The key observation will be that, if there is a value v held as input by $n - 2t$ honest parties, then the honest parties hold the same $a = v$ or the same $b = v$. The parties then first try to agree on a: they run a BA protocol Π_{BA} on their values a and obtain an output a'. Afterwards they check whether they are happy with a' by joining Π_{BA} once again: with input 1 if $a = a'$ and 0 otherwise. If Π_{BA} returns 1, the parties output a'. If Π_{BA} returns 0, the parties check whether they hold the same value b with the same strategy: they join Π_{BA} with input b, and obtain output b'. Afterwards, they join Π_{BA} again with input 1 if $b = b'$ and 0 otherwise. If the output is 1, they output b', and otherwise they output $\bot$.

This way, if $n - 2t$ honest parties hold the same input, the output is guaranteed to be non-$\bot$, which ensures that Bounded Pre-Agreement holds. In addition, if the parties output a non-$\bot$ value, we are able to show that some honest party has received it from $n - 2t > t$ parties in the first step, and therefore it is an honest input, which ensures that Intrusion Tolerance holds. We present the code below.

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

Code for party P with input v_{in}

1: Send v_{IN} to all parties.
2: Check if there is any value received from $n-2t$ parties. If there is none, send $\text{VOTE}(\cdot)$ to all parties. If there is only one, let this value be v_1 and send $\text{VOTE}(v_1)$ to all parties. If there are two, let these values be v_1 and v_2 and send $\text{VOTE}(v_1, v_2)$ to all parties.
3: Let $a := \bot, b := \bot$. If there is a single value v voted by $n-t$ parties, set $a := v$ and $b := v$. If there are two values $v \leq v'$ voted by $n-t$ parties, set $a := v$ and $b := v'$.
4: Join Π_{BA} with input a and obtain output a'. If $a' = a \neq \bot$, join Π_{BA} with input 1, and otherwise with input 0. If Π_{BA} returns 1, output a and terminate.
5: Join Π_{BA} with input b and obtain output b'. If $b' = b \neq \bot$, join Π_{BA} with input 1, and otherwise with input 0. If Π_{BA} returns 1, output b and terminate. Otherwise, output $\bot$ and terminate.

Theorem 6. *Assume a* BA *protocol* Π_{BA} *resilient against* $t < n/3$ *corruptions. Then, there is a* BA *protocol* $\Pi_{\text{BA+}}$ *resilient against* $t < n/3$ *that additionally achieves Intrusion Tolerance and Bounded Pre-Agreement.* $\Pi_{\text{BA+}}$ *has communication complexity* $\text{BITS}_\kappa(\Pi_{\text{BA+}}) = O(\kappa n^2) + \text{BITS}_\kappa(\Pi_{\text{BA}})$ *and round complexity* $\text{ROUNDS}(\text{BA+}) = O(1) \cdot \text{ROUNDS}_\kappa(\Pi_{\text{BA}})$.

Proof. We first show that $\Pi_{\text{BA+}}$ is indeed a BA protocol. Agreement and Termination follow from the fact that Π_{BA} satisfies these properties. If all honest parties hold the same input value v, then no honest party receives $v' \neq v$ from $n - 2t > t$ parties, and therefore all honest parties send (VOTE, v). Then, the honest parties see at most t votes for any value $v' \neq v$, and therefore all honest parties set $a = b = v$. They join the Π_{BA} invocation in line 4 with input $a = v$, and since Π_{BA} achieves Validity, they agree on $a' = v$. All honest parties join the next Π_{BA} invocation with input 1, and therefore they agree on 1 and output v, hence Validity holds.

We now focus on Intrusion Tolerance. If the honest parties output some value $v \neq \bot$, then this is the value a or b obtained by an honest party P (due to Π_{BA}'s Validity). P has received $n-t$ votes for v, hence at least one vote from some honest party P'. Then, P' has received v from $n - 2t > t$ parties, hence from at least one honest party, and therefore Intrusion Tolerance holds.

For Bounded Pre-Agreement, we show that, if there is a value v held as input by $n-2t$ honest parties, then the value agreed upon is not $\bot$.

We establish that, in line 2, every party sees at most two input values from $n-2t$ parties each. Assuming that a party has received at least three input values sent by $n - 2t$ parties each, we obtain that $3 \cdot (n - 2t) \leq n$, which contradicts $n > 3t$. Then, each party sees at most two values from $n - 2t$ parties each, and one of these is v. Hence, each honest party sends a vote for v, and possibly for a second value.

Then, each party receives the $n - t$ votes for v from the honest parties. We add that each party receives $n - t$ votes for two values at most. Otherwise, since each party votes for at most two values, there are at most $2n$ votes in total. Assuming that a party receives $n-t$ votes for at least three values implies $3 \cdot (n-t) \leq 2n$, contradicting $n > 3t$.

If every honest party sees v as the only value with $n-t$ votes, then every honest party sets $a = b = v$, and therefore all honest parties output v in line 4. Otherwise, let P and P' be two honest parties. We assume that P sees two values with $n-t$ votes each: one of these we know to be v, and the other is $v' \neq v$. We have showed that P' also sees v with $n-t$ (honest) votes, and we assume that P' receives $n-t$ votes for $v'' \notin \{v, v'\}$. This leads to a contradiction as P' has received $n-t$ honest votes for v, and at least $n-2t$ honest votes for v'. Since every party may vote for at most two different values, there are at most t honest votes and t votes from byzantine parties left for v'': these are $2t < n-t$ in total.

Hence, every honest party obtains a, b such that $v \in \{a, b\} \subseteq \{v, v'\}$. The parties then try to agree on a in line 4. If the second Π_{BA} invocation of line 4 returns 0, then the parties hold different values a: $a = v$ for some honest parties, and $a = v'$ for the others. Since every honest party has set a and b such that $a \leq b$, this means that the honest parties hold the same value $b = v$ and output v in line 5.

For the round complexity, note that $\Pi_{\mathrm{BA+}}$ incurs two rounds of communication and afterwards runs Π_{BA} at most four times. For the communication complexity, note that each party sends at most three values to all parties, and each of these values is an honest party's input, and therefore consists of κ bits. Afterwards, they run Π_{BA} on κ-bit inputs at most twice and on bits at most twice. □

From $\Pi_{\mathbf{BA+}}$ to $\Pi_{\boldsymbol{\ell}\mathbf{BA+}}$. We may now describe our protocol $\Pi_{\ell\mathrm{BA+}}$ for long messages, relying on $\Pi_{\mathrm{BA+}}$ and on the outline of prior works [8, 41].

$\Pi_{\ell\mathrm{BA+}}$ makes use of Reed-Solomon (RS) codes [45], 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 values, prior works make use of collision-free cryptographic accumulators [42]. 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) [39], which do not require a trusted dealer. We briefly describe RS codes and MT below.

$\Pi_{\ell\mathrm{BA+}}$ assumes 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 $(\mathrm{s}_1, \ldots, \mathrm{s}_n)$ of $O(|\mathrm{BITS}(v)|/n)$ 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.

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 (with high probability) that some value belongs to the compressed multiset. Given a multiset $S = \{\mathrm{s}_1, \ldots, \mathrm{s}_n\}$, the MT is built bottom-up, using the collision-resistant hash function H_κ: starting with n leaves, where the i-th leaf stores $H_\kappa(\mathrm{s}_i)$. Each non-leaf node stores $H_\kappa(h_{\mathrm{LEFT}} \,\|\, h_{\mathrm{RIGHT}})$, 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, $\text{MT.}\textsc{Verify}(z, i, s_i, w_i)$ 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.

Then, $\Pi_{\ell\text{BA}+}$ consists of three steps. In the first step, every party computes $s_1, \ldots, s_n := \text{RS.}\textsc{encode}(v_{\text{IN}})$ and $z, w_1, \ldots, w_n := \text{MT.}\textsc{Build}(\{s_1, \ldots, s_n\})$. In the second step, the parties agree on an encoding $z^\star$ with the help of $\Pi_{\text{BA}+}$. In the third step, the parties obtain the final output. If $\Pi_{\text{BA}+}$ returns $\bot$, the parties output $\bot$. Otherwise, if $\Pi_{\text{BA}+}$ returns $z^\star$, every party $P^\star$ holding $z = z^\star$ distributes $v^\star := v_{\text{IN}}$ to all the parties. To achieve this using only $O(\ell n + \mathsf{poly}(n, \kappa))$ 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 $\text{RS.}\textsc{encode}(v^\star)$. Every party P_i then sends (s_i, w_i) to all parties, which allows the parties to reconstruct $v^\star$.

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

Code for party P_i with input v_{in}

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

We may now sketch the proof of Theorem 1. The formal proof is included in Section A.1.

Proof sketch of Theorem 1. As $\Pi_{\text{BA}+}$ achieves Termination, $\Pi_{\ell\text{BA}+}$ achieves Termination as well. Then, note that $\Pi_{\ell\text{BA}+}$ returns $\bot$ whenever $\Pi_{\text{BA}+}$ returns $\bot$, and the parties obtain non-$\bot$ in $\Pi_{\ell\text{BA}+}$ whenever $\Pi_{\text{BA}+}$ returns a non-bot value. Moreover, the Intrusion Tolerance property of $\Pi_{\text{BA}+}$ ensures that, whenever $\Pi_{\text{BA}+}$ returns non-$\bot$, the parties agree on the encoding $z^\star$ of an honest party's input, which means that the parties successfully decode a value that is an honest party's input. Hence, both Agreement and Intrusion Tolerance hold. For Bounded Pre-Agreement, since $\Pi_{\text{BA}+}$ only returns $\bot$ when fewer $n - 2t$ parties hold the same input value, $\Pi_{\ell\text{BA}+}$ also only returns $\bot$ when fewer $n - 2t$ parties hold the same input value. Finally, for Validity, if all honest parties hold the same input value $v^\star$, $\Pi_{\text{BA}+}$'s Validity ensures that the parties agree on the encoding $z^\star$ of $v^\star$, and therefore the parties successfully decode $z^\star$.

The round complexity follows immediately from the round complexity of $\Pi_{\text{BA}+}$. For the communication complexity, note that, in Step 3, each party sends at most two shares and two MT witnesses to each party. This leads $O(\ell n + \kappa \cdot n^2 \log n) + \textsc{bits}_\kappa(\Pi_{\text{BA}+})$ bits of communication. □

8 Conclusions

Achieving solutions with optimal-communication has been the subject of an extensive line of works [8, 23, 24, 34, 41]. These works have primarily focused on the fundamental primitives BA and BC, where $\Omega(\ell n)$ bits of communication are necessary, and have presented protocols with communication complexity $O(\ell n + \mathsf{poly}(n, \kappa))$, proving the lower bound tight for large enough ℓ. Our work shows that the lower bound $\Omega(\ell n)$ is also tight for synchronous CA on integers given that ℓ is large enough, namely $\ell = \Omega(\kappa \cdot n \log^2 n)$. We have presented a protocol that relies on finding some valid values' longest common prefix, achieving CA with optimal resilience, asymptotically optimal communication complexity, and efficient round complexity. Our protocol is also deterministic and operates without trusted setup.

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 $\mathsf{poly}(n, \kappa)$ factor, or extending our question to input spaces beyond $\mathbb{Z}$.

References

[1] Ittai Abraham, Yonatan Amit, and Danny Dolev. Optimal resilience asynchronous approximate agreement. In Teruo Higashino, editor, *Principles of Distributed Systems*, pages 229–239, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[2] Ittai Abraham, TH Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 317–326, 2019.

[3] Dan Alistarh, Faith Ellen, and Joel Rybicki. Wait-free approximate agreement on graphs. In Tomasz Jurdziński and Stefan Schmid, editors, *Structural Information and Communication Complexity*, pages 87–105, Cham, 2021. Springer International Publishing. doi:10.1007/978-3-030-79527-6_6.

[4] Dan Alistarh, Jerry Ho, Sashank Kannan, Aditya Sarwate, and Leonidas Zhang. Collaborative learning in the jungle (decentralized, byzantine, heterogeneous, asynchronous and nonconvex learning). In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 11990–12001, 2021.

[5] Akhil Bandarupalli, Adithya Bhat, Saurabh Bagchi, Aniket Kate, Chen-Da Liu-Zhang, and Michael K. Reiter. Delphi: Efficient asynchronous approximate agreement for distributed oracles. In *54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2024.

[6] Michael Ben-Or, Danny Dolev, and Ezra N. Hoch. Brief announcement: Simple gradecast based algorithms. In Nancy A. Lynch and Alexander A. Shvartsman, editors, *Distributed Computing*, pages 194–197, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[7] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Towards optimal distributed consensus (extended abstract). In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 410–415. IEEE Computer Society, 1989.

[8] Amey Bhangale, Chen-Da Liu-Zhang, Julian Loss, and Kartik Nayak. 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*, pages 504–525, Berlin, Heidelberg, 2023. Springer, Springer-Verlag.

[9] Erica Blum, Jonathan Katz, Chen-Da Liu-Zhang, and Julian Loss. Asynchronous byzantine agreement with subquadratic communication. In *TCC 2020, Part I*, LNCS, pages 353–380. Springer, Heidelberg, March 2020. doi:10.1007/978-3-030-64375-1_13.

[10] Christian Cachin and Stefano Tessaro. Asynchronous verifiable information dispersal. In *24th IEEE Symposium on Reliable Distributed Systems (SRDS'05)*, pages 191–201, Orlando, FL, USA, 2005. IEEE, IEEE Computer Society. doi:10.1109/RELDIS.2005.9.

[11] Wutichai Chongchitmate and Rafail Ostrovsky. Information-theoretic broadcast with dishonest majority for long messages. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 370–388, Cham, November 2018. Springer, Heidelberg. doi:10.1007/978-3-030-03807-6_14.

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

[13] Shir Cohen, Idit Keidar, and Alexander Spiegelman. Not a coincidence: Sub-quadratic asynchronous byzantine agreement whp. In *34th International Symposium on Distributed Computing*, 2020.

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

[15] Andrei Constantinescu, Diana Ghinea, Roger Wattenhofer, and Floris Westermann. Convex Consensus with Asynchronous Fallback. In *38th International Symposium on Distributed Computing (DISC), Madrid, Spain*, October 2024.

[16] Danny Dolev, Nancy A. Lynch, Shlomit S. Pinter, Eugene W. Stark, and William E. Weihl. Reaching approximate agreement in the presence of faults. *J. ACM*, 33(3):499–516, May 1986. `doi:10.1145/5925.5931`.

[17] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. In Robert L. Probert, Michael J. Fischer, and Nicola Santoro, editors, *1st ACM PODC*, pages 132–140. ACM, August 1982. `doi:10.1145/800220.806690`.

[18] El-Mahdi El-Mhamdi, Rachid Guerraoui, Arsany Guirguis, Lê Nguyên Hoang, and Sébastien Rouault. Genuinely distributed byzantine machine learning. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pages 355–364, 2020.

[19] El-Mahdi El-Mhamdi, Rachid Guerraoui, Arsany Guirguis, Lê Nguyên Hoang, and Sébastien Rouault. Genuinely distributed byzantine machine learning. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, PODC '20, page 355–364, New York, NY, USA, 2020. Association for Computing Machinery. `doi:10.1145/3382734.3405695`.

[20] A. D. Fekete. Asynchronous approximate agreement. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, PODC '87, page 64–76, New York, NY, USA, 1987. Association for Computing Machinery. `doi:10.1145/41840.41846`.

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

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

[23] Matthias Fitzi and Martin Hirt. Optimally efficient multi-valued Byzantine agreement. In Eric Ruppert and Dahlia Malkhi, editors, *25th ACM PODC*, pages 163–168, New York, NY, USA, July 2006. ACM. `doi:10.1145/1146381.1146407`.

[24] Chaya Ganesh and Arpita Patra. Broadcast extensions with optimal communication and round complexity. In George Giakkoupis, editor, *35th ACM PODC*, pages 371–380, New York, NY, USA, July 2016. ACM. doi:10.1145/2933057.2933082.

[25] Yuval Gelles and Ilan Komargodski. Optimal load-balanced scalable distributed agreement. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, STOC 2024, page 411–422, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3618260.3649736.

[26] Diana Ghinea, Chen-Da Liu-Zhang, and Roger Wattenhofer. Optimal synchronous approximate agreement with asynchronous fallback. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, PODC'22, page 70–80, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519270.3538442.

[27] Diana Ghinea, Chen-Da Liu-Zhang, and Roger Wattenhofer. Multidimensional approximate agreement with asynchronous fallback. In *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '23, page 141–151, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3558481.3591105.

[28] Martin Hirt and Pavel Raykov. Multi-valued byzantine broadcast: The $t < n$ case. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 448–465, Berlin, Heidelberg, December 2014. Springer, Heidelberg. doi:10.1007/978-3-662-45608-8_24.

[29] Valerie King and Jared Saia. From almost everywhere to everywhere: Byzantine agreement with \tilde{o}($n^{\{}$3/2}) bits. In Idit Keidar, editor, *Distributed Computing*, pages 464–478, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[30] Valerie King and Jared Saia. Breaking the $O(n^2)$ bit barrier: scalable byzantine agreement with an adaptive adversary. In Andréa W. Richa and Rachid Guerraoui, editors, *29th ACM PODC*, pages 420–429. ACM, July 2010. doi:10.1145/1835698.1835798.

[31] Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Scalable leader election. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, page 990–999, USA, 2006. Society for Industrial and Applied Mathematics.

[32] Jérémy Ledent. Brief announcement: Variants of approximate agreement on graphs and simplicial complexes. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, PODC'21, page 427–430, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3465084.3467946.

[33] Christoph Lenzen and Julian Loss. Optimal clock synchronization with signatures. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, PODC'22, page 440–449, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519270.3538444.

[34] Guanfeng Liang and Nitin Vaidya. Error-free multi-valued consensus with byzantine failures. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 11–20, New York, NY, USA, 2011. Association for Computing Machinery.

[35] Guanfeng Liang and Nitin H. Vaidya. Error-free multi-valued consensus with byzantine failures. In Cyril Gavoille and Pierre Fraigniaud, editors, *30th ACM PODC*, pages 11–20. ACM, June 2011. doi:10.1145/1993806.1993809.

[36] Darya Melnyk and Roger Wattenhofer. Byzantine agreement with interval validity. In *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*, pages 251–260, Salvador, Brazil, 2018. IEEE Computer Society. doi:10.1109/SRDS.2018.00036.

[37] Hammurabi Mendes and Maurice Herlihy. Multidimensional approximate agreement in byzantine asynchronous systems. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 391–400, Palo Alto, CA, USA, June 2013. ACM Press. doi:10.1145/2488608.2488657.

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

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

[40] Atsuki Momose and Ling Ren. Optimal communication complexity of authenticated byzantine agreement. In *35th International Symposium on Distributed Computing (DISC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[41] Kartik Nayak, Ling Ren, Elaine Shi, Nitin H. Vaidya, and Zhuolun Xiang. Improved Extension Protocols for Byzantine Broadcast and Agreement. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing (DISC 2020)*, volume 179 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2020/13106, doi:10.4230/LIPIcs.DISC.2020.28.

[42] Lan Nguyen. 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*, pages 275–292. Springer, 2005.

[43] Thomas Nowak and Joel Rybicki. Byzantine Approximate Agreement on Graphs. In Jukka Suomela, editor, *33rd International Symposium on Distributed Computing (DISC 2019)*, volume 146 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 29:1–29:17, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: http://drops.dagstuhl.de/opus/volltexte/2019/11336, doi:10.4230/LIPIcs.DISC.2019.29.

[44] Maria Potop-Butucaru, Michel Raynal, and Sebastien Tixeuil. Distributed computing with mobile robots: An introductory survey. pages 318 – 324, 10 2011. doi:10.1109/NBiS.2011.55.

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

[46] Phillip Rogaway. Formalizing human ignorance. In Phong Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 2006*, pages 211–228, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[47] David Stolz and Roger Wattenhofer. Byzantine Agreement with Median Validity. In Emmanuelle Anceaume, Christian Cachin, and Maria Potop-Butucaru, editors, *19th International Conference on Principles of Distributed Systems (OPODIS 2015)*, volume 46 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1–14, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2016/6591`, `doi:10.4230/LIPIcs.OPODIS.2015.22`.

[48] Lili Su and Nitin H Vaidya. Fault-tolerant multi-agent optimization: optimal iterative distributed algorithms. In *Proceedings of the 2016 ACM symposium on principles of distributed computing*, pages 425–434, 2016.

[49] Russell Turpin and Brian A Coan. Extending binary byzantine agreement to multivalued byzantine agreement. *Information Processing Letters*, 18(2):73–76, 1984.

[50] Nitin H. Vaidya and Vijay K. Garg. Byzantine vector consensus in complete graphs. In Panagiota Fatourou and Gadi Taubenfeld, editors, *32nd ACM PODC*, pages 65–73, Montreal, QC, July 2013. ACM. `doi:10.1145/2484239.2484256`.

A Appendix

A.1 BA for long messages with additional properties

This section provides the analysis of our protocol $\Pi_{\ell\text{BA}+}$. We formally prove Theorem 1, restated below.

Theorem 1. *Given a* BA *protocol* Π_{BA} *resilient against* $t < n/3$ *corruptions, there is a* BA *protocol* $\Pi_{\ell\text{BA}+}$ *resilient against* $t < n/3$ *corruptions that achieves Intrusion Tolerance and Bounded Pre-Agreement, with communication complexity* $\text{BITS}_\ell(\Pi_{\ell\text{BA}+}) = O(\ell n + \kappa \cdot n^2 \log n) + \text{BITS}_\kappa(\Pi_{\text{BA}})$*, and round complexity* $\text{ROUNDS}_\ell(\Pi_{\ell\text{BA}+}) = O(1) + \text{ROUNDS}_\kappa(\Pi_{\text{BA}})$.

The lemma below ensures that, if the $\Pi_{\text{BA}+}$ invocation in line 2 of $\Pi_{\ell\text{BA}+}$ returns a non-$\bot$ value, the honest parties agree on an honest input. The argument is identical to [41, Lemma 6].

Lemma 6. *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 $\text{s}_1, \ldots, \text{s}_n$ 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 $\text{MT.Verify}(z^\star, j, \text{s}_j, w_j) = \texttt{true}$.

Note that party P_j ignores any tuples (j, s'_j, w'_j) with $\text{s}'_j \neq \text{s}_j$: a different RS encoding $(\text{s}'_1, \ldots, \text{s}'_n) \neq (\text{s}_1, \ldots, \text{s}_n)$ 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 $\text{MT.Verify}(z^\star, j, \text{s}'_j, w'_j) = \texttt{true}$ requires the adversary to find collisions for H_κ, which we assumed to be impossible. Therefore, $\text{MT.Verify}(z^\star, j, \text{s}'_j, w'_j) = \texttt{false}$, 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: $\text{MT.Verify}(z^\star, j, \text{s}'_j, w'_j) = \texttt{false}$. 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. ☐

Below we provide the analysis of $\Pi_{\ell\text{BA}+}$. This is also similar to the analysis of [41], with the exception that we also verify the additional properties Intrusion Tolerance and Bounded Pre-Agreement. Theorem 6 and the lemma presented below directly imply Theorem 1.

Lemma 7. *Assume a* BA *protocol* $\Pi_{\text{BA}+}$ *secure against* $t < n/3$ *corruptions that additionally achieves Intrusion Tolerance and Bounded Pre-Agreement. Then,* $\Pi_{\ell\text{BA}+}$ *achieves the same guarantees, with communication complexity* $\text{BITS}_\ell(\Pi_{\ell\text{BA}+}) = O(\ell n + \kappa \cdot n^2 \log n) + \text{BITS}_\kappa(\Pi_{\text{BA}+})$*, and round complexity* $\text{ROUNDS}_\ell(\Pi_{\ell\text{BA}+}) = O(1) + \text{ROUNDS}_\kappa(\Pi_{\text{BA}+})$.

Proof. In the following, we first show that $\Pi_{\ell\text{BA}+}$ achieves the standard BA properties.

The parties obtain the same output in $\Pi_{\text{BA}+}$: either $z^\star$ or $\bot$. If the output returned by $\Pi_{\text{BA}+}$ is $\bot$, the parties output $\bot$, hence Agreement holds in this case. Otherwise, there is an honest party who proposed $z = z^\star$ since $\Pi_{\mathrm{BA}+}$ achieves Intrusion Tolerance, and Lemma 6 ensures that the parties output 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. This implies that, if all honest parties hold the same input v, then all honest parties obtain the same value z, and $\Pi_{\mathrm{BA}+}$ returns $z^\star = z$. Lemma 6 ensures that the parties output an honest party's input, 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 this step is only executed if there is an honest party holding $z = z^\star$, Lemma 6 ensures that the Intrusion Tolerance property holds.

If there are $n - 2t$ honest parties holding the same input value v, then these parties join $\Pi_{\mathrm{BA}+}$ with the same encoding z. The Bounded Pre-Agreement property of $\Pi_{\mathrm{BA}+}$ ensures that the parties agree on $z^\star \neq z$. Afterwards, Lemma 6 ensures that the honest parties agree on a non-$\bot$ value in $\Pi_{\ell\mathrm{BA}+}$, and therefore Bounded Pre-Agreement holds.

We have obtained that $\Pi_{\ell\mathrm{BA}+}$ indeed maintains the properties of $\Pi_{\mathrm{BA}+}$. Running $\Pi_{\mathrm{BA}+}$ with inputs z requires $\textsc{bits}_\kappa(\Pi_{\mathrm{BA}+})$ bits and $\textsc{rounds}_\kappa(\Pi_{\mathrm{BA}+})$ rounds. If the output is $\bot$, there is no further communication. Otherwise, the parties run the distributing step, and Lemma 6 shows that this step has an additional cost of $O(\ell n + \kappa \cdot n^2 \log n)$ bits and $O(1)$ rounds. Then, the total bit complexity of $\Pi_{\ell\mathrm{BA}+}$ is $O(\ell n + \kappa \cdot n^2 \log n) + \textsc{bits}_\kappa(\Pi_{\mathrm{BA}+})$, and the round complexity is $O(1) + \textsc{rounds}_\kappa(\Pi_{\mathrm{BA}+})$. □

A.2 Missing proofs for FixedLengthCA

In this section, we focus on analyzing each of the subprotocols of FixedLengthCA. We first include two remarks which will enable us to show that values computed by comparing prefixes of valid values are valid.

Remark 1. *Consider two values $v, v' \in \mathbb{N}$ satisfying $v \leq v' < 2^\ell$, and let* COMMON_PREFIX *denote the* **longest** *common prefix of* $\textsc{bits}_\ell(v)$ *and* $\textsc{bits}_\ell(v')$*. If* $|\textsc{common\_prefix}| < \ell$*, then* $\textsc{max}_\ell(\textsc{common\_prefix} \parallel 0), \textsc{min}_\ell(\textsc{common\_prefix} \parallel 1) \in [v, v']$.

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

We first note that, since $v \leq v'$, $\textsc{bits}_\ell(v)$ has prefix $\textsc{common\_prefix} \parallel 0$, while $\textsc{bits}_\ell(v')$ has prefix $\textsc{common\_prefix} \parallel 1$. Secondly, since $\textsc{max}_\ell(\textsc{common\_prefix} \parallel 0)$ is the highest ℓ-bit value having prefix $\textsc{common\_prefix} \parallel 0$, and v is an ℓ-bit value with the same prefix, $v \leq \textsc{max}_\ell(\textsc{common\_prefix} \parallel 0)$.

In addition, note that $\textsc{max}_\ell(\textsc{common\_prefix} \parallel 0) + 1 = \textsc{min}_\ell(\textsc{common\_prefix} \parallel 1)$.

We use a similar argument to show that $v' \geq \textsc{min}_\ell(\textsc{common\_prefix} \parallel 1)$: v' is an ℓ-bit value with prefix $\textsc{common\_prefix} \parallel 1$, while $\textsc{min}_\ell(\textsc{common\_prefix} \parallel 1)$ is the lowest ℓ-bit value having prefix $\textsc{common\_prefix} \parallel 1$. □

Remark 2. *Consider two values $v, v' \in \mathbb{N}$ such that $v, v' < 2^\ell$, and let* COMMON_PREFIX *denote the* **longest** *common prefix of* $\textsc{bits}_\ell(v)$ *and* $\textsc{bits}_\ell(v')$*. Let* NEXT_BITS *and* NEXT_BITS′ *denote two non-empty bitstrings of equal length such that* $\textsc{common\_prefix} \parallel \textsc{next\_bits}$ *is a prefix of* $\textsc{bits}_\ell(v)$*, and* $\textsc{common\_prefix} \parallel \textsc{next\_bits}'$ *is a prefix of* $\textsc{bits}_\ell(v')$*.*

If $\textsc{val}(\textsc{next\_bits}) < \textsc{val}(\textsc{next\_bits}')$*, then:*

$$\textsc{min}_\ell(\textsc{common\_prefix} \parallel \textsc{next\_bits}'), \textsc{max}_\ell(\textsc{common\_prefix} \parallel \textsc{next\_bits}) \in [v, v'].$$

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

Missing proofs for FindPrefix. We first prove the invariants of each iteration, as described in the proof sketch of Lemma 1.

Lemma 8. *Assume that the following properties hold at the beginning of iteration i.*

(A) All honest parties hold the same indices $1 \leq \textsc{left} \leq \textsc{right} \leq \ell + 1$, and the same bitstring $\textsc{prefix}^\star$ consisting of $\textsc{left} - 1$ bits.

(B) $0 \leq \textsc{right} - \textsc{left} \leq 2^{\lceil \log_2 \ell \rceil - (i-1)}$.

(C) Honest parties hold valid ℓ-bit values v such that $\textsc{bits}_\ell(v)$ has $\textsc{prefix}^\star$ as a prefix.

(D) Honest parties hold valid ℓ-bit values $v_\perp$, and, for any bitstring $\textsc{bits}$ of $\textsc{right}$ bits, there are $t+1$ honest parties holding values $v_\perp$ such that $\textsc{bits}_\ell(v_\perp)$ does not have prefix $\textsc{bits}$.

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

Proof. We assume that the condition $\textsc{left} = \textsc{right}$ is not yet met in iteration i (otherwise, the statement trivially holds). Then, Property (B) ensures that $\textsc{left} < \textsc{right}$, 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 bits, 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. $\textsc{prefix}^\star(i)$ is the value held at the beginning of iteration i, and $\textsc{prefix}^\star(i+1)$ 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 $\textsc{right}(i+1)$ index identically, while all other values remain unchanged. Note that $\textsc{left}(i) \leq \textsc{mid} < \textsc{right}(i)$ and $\textsc{right}(i+1) := \textsc{mid}$ still satisfies $1 \leq \textsc{right}(i+1) \leq \ell + 1$. Therefore, Property (A) holds at the beginning of iteration $i+1$.

(B) All honest parties compute $\textsc{right}(i+1) := \textsc{mid} \geq \textsc{left}(i)$, while the $\textsc{left}$ index remains unchanged: $\textsc{left}(i+1) := \textsc{left}(i)$. We obtain the inequality below, which ensures that Property (B) holds at the beginning of iteration $i+1$.

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

(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-bit value according to Property (C). We also need to show that, given an arbitrary bitstring $\textsc{b}_1 \parallel \ldots \parallel \textsc{b}_{\textsc{mid}}$ of $\textsc{right}(i+1) = \textsc{mid}$ bits, there are $t+1$ honest parties holding values $v(i)$ such that $\textsc{bits}_\ell(v(i))$ does not have $\textsc{bits}$ as a prefix. This is ensured by the *Bounded Pre-Agreement* property of $\Pi_{\ell\text{BA}+}$: fewer than $n - 2t$ honest parties have proposed $\text{B}_{\text{LEFT}(i)} \| \ldots \| \text{B}_{\text{MID}}$. Therefore, at least $(n-t)-(n-2t-1) \geq t+1$ honest parties hold values $v(i)$ satisfying $\text{B}_{\text{LEFT}(i)}(v(i)) \| \ldots \| \text{B}_{\text{MID}}(v(i)) \neq \text{B}_{\text{LEFT}(i)} \| \ldots \| \text{B}_{\text{MID}}$, which implies that the bit representations $\text{BITS}_\ell(v(i))$ do not have prefix $\text{B}_1 \| \ldots \| \text{B}_{\text{MID}}$. Therefore, Property (D) holds at the beginning of iteration $i+1$.

We now assume that $\Pi_{\ell\text{BA}+}$ returns $\text{PREFIX}^\star_{\text{LEFT}(i)} \| \ldots \| \text{PREFIX}^\star_{\text{MID}}$:

(A) The parties compute their $\text{LEFT}(i+1)$ index and the sequence of bits $\text{PREFIX}^\star(i+1)$ identically, while the RIGHT index remains unchanged ($\text{RIGHT}(i+1) := \text{RIGHT}(i)$). Note that $\text{PREFIX}^\star(i+1)$ is obtained by adding $\text{LEFT}(i+1) - \text{LEFT}(i)$ bits to $\text{PREFIX}^\star(i)$, therefore $\text{PREFIX}^\star(i+1)$ consists of $\text{LEFT}(i+1) - 1$ bits. In addition, $\text{LEFT}(i) \leq \text{MID} < \text{RIGHT}(i)$ and $\text{LEFT}(i+1) := \text{MID} + 1$ satisfies $1 \leq \text{LEFT}(i+1) \leq \ell + 1$. Therefore, Property (A) holds at the beginning of iteration $i+1$.

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

$$\begin{aligned} 0 \leq \text{RIGHT}(i+1) - \text{LEFT}(i+1) &= \text{RIGHT}(i) - (\lfloor (\text{LEFT}(i) + \text{RIGHT}(i))/2 \rfloor + 1) \\ &\leq \text{RIGHT}(i) - (\text{LEFT}(i) + \text{RIGHT}(i))/2 \\ &\leq (\text{RIGHT}(i) - \text{LEFT}(i))/2 \leq 2^{\lceil \log_2 \ell \rceil - ((i+1)-1)}. \end{aligned}$$

(C) Honest parties either hold values $v(i)$ having $\text{PREFIX}^\star(i+1)$ as a prefix, or they set $v(i+1)$ to some ℓ-bit value having prefix $\text{PREFIX}^\star(i+1)$. This implies that, at the beginning of iteration $i+1$, all honest parties hold ℓ-bit values $v(i)$ with prefix $\text{PREFIX}^\star(i+1)$. 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 *Intrusion Tolerance* property of $\Pi_{\ell\text{BA}+}$ ensures that parties agree on a sequence of bits $\text{PREFIX}^\star_{\text{LEFT}(i)} \| \ldots \| \text{PREFIX}^\star_{\text{MID}}$ that was proposed by an honest party holding value $v^\star$. Then, Property (C) ensures that $v^\star$ is a valid ℓ-bit value such that $\text{BITS}_\ell(v^\star)$ has prefix $\text{PREFIX}^\star(i+1)$. In addition, $v(i)$ is a valid ℓ-bit value, such that $\text{BITS}_\ell(v(i))$ does not have prefix $\text{PREFIX}^\star(i+1)$. Then, Remark 2 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-bit value within the honest inputs' range.

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

□

We may now focus on the proof of Lemma 1.

Lemma 1. *Assume a* BA *protocol* Π_{BA}*, and that honest parties join* FINDPREFIX *with the same* ℓ*, and with valid* ℓ*-bit values* v*. Then, the honest parties obtain the same bitstring* $\text{PREFIX}^\star$*, and each honest party obtains two valid* ℓ*-bit values* v*,* $v_\perp$ *such that: (i)* $\text{PREFIX}^\star$ *is a prefix of* $\text{BITS}_\ell(v)$*; (ii) for any bitstring* BITS *of* $|\text{PREFIX}^\star| + 1$ *bits, at least* $t+1$ *honest parties hold values* $v_\perp$ *such that* $\text{BITS}_\ell(v_\perp)$ *does not have prefix* BITS*.*

FINDPREFIX *achieves communication complexity* $\text{BITS}_\ell(\text{FINDPREFIX}) = O(\ell \cdot n + \kappa \cdot n^2 \log n \log \ell) + O(\log \ell) \cdot \text{BITS}_\kappa(\Pi_{\text{BA}})$*, and round complexity* $\text{ROUNDS}_\ell(\text{FINDPREFIX}) = O(\log \ell) \cdot \text{ROUNDS}_\kappa(\Pi_{\text{BA}})$*.*

Proof. The properties listed in Lemma 8 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. Once the condition LEFT = RIGHT is met, Property (C) guarantees that honest parties hold valid ℓ-bit values v having the bitstring PREFIX* as a common prefix. According to Property (A), this common prefix consists of $i^\star :=$ LEFT $- 1$ bits. From Property (D), it follows that the honest parties hold valid ℓ-bit values $v_\perp$. The same property implies that, for any bitstring BITS of RIGHT $= i^\star + 1$ bits, the ℓ-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 LEFT = RIGHT is met (for all honest parties simultaneously, due to Property (A)) by iteration $i := \lceil \log_2 \ell \rceil + 2$. Property (B) ensures that, at the beginning of iteration i, $0 \leq \text{RIGHT} - \text{LEFT} \leq 2^{\lceil \log_2 \ell \rceil - (i-1)}$. Then, if this condition was not met by iteration $i := \lceil \log_2 \ell \rceil + 2$, the indices LEFT and RIGHT obtained by the honest parties in iteration i satisfy $0 \leq \text{RIGHT} - \text{LEFT} \leq 2^{\lceil \log_2 \ell \rceil - (\lceil \log_2 \ell \rceil + 1)} \leq 2^{-1}$. Since the indices LEFT and RIGHT are natural numbers, we may conclude that RIGHT $-$ LEFT $= 0$.

We may then discuss the round complexity of FindPrefix: since $O(\log \ell)$ iterations are sufficient and each iteration invokes $\Pi_{\ell\text{BA}+}$ once, we obtain that $\text{ROUNDS}_\ell(\text{FindPrefix}) = O(\log \ell) \cdot \text{ROUNDS}_\ell(\Pi_{\ell\text{BA}+})$. Then, Theorem 1 leads to the result claimed in the lemma's statement.

For the communication complexity, Property (B) of Lemma 8 ensures that, in each iteration $i < \lceil \log_2 \ell \rceil + 2$, FindPrefix runs $\Pi_{\ell\text{BA}+}$ on inputs of at most $2^{\lceil \log_2 \ell \rceil - i}$ bits, hence of at most $\ell / 2^{i-1}$ bits. Therefore, $\text{BITS}_\ell(\text{FindPrefix}) = \sum_{i=1}^{\lceil \log_2 \ell \rceil + 1} \text{BITS}_{\ell/2^{i-1}}(\Pi_{\ell\text{BA}+})$. Using Theorem 1, and the fact that $\sum_{i=0}^{\infty} 1/2^i \leq 2$, we obtain that $\text{BITS}_\ell(\text{FindPrefix}) = O(\ell \cdot n + \kappa \cdot n^2 \log n \log \ell) + O(\log \ell) \cdot \text{BITS}_\kappa(\Pi_{\text{BA}})$. ☐

Missing proofs for AddLastBit. We present the proof of Lemma 2. This provides the guarantees of AddLastBit, which enables the honest parties to extend the prefix obtained in FindPrefix with one bit.

Lemma 2. *Assume a* BA *protocol* Π_{BA}*, and that honest parties join* AddLastBit *with the same value* ℓ*, the same bitstring* PREFIX* *of* $i^\star < \ell$ *bits, and with valid* ℓ*-bit values* v *such that* $\text{BITS}_\ell(v)$ *has prefix* PREFIX**. Then, the honest parties agree on a bitstring of* $i^\star + 1$ *bits that is the prefix of a valid value's* ℓ*-bit representation.* AddLastBit *has communication complexity* $\text{BITS}_\ell(\text{AddLastBit}) = \text{BITS}_1(\Pi_{\text{BA}})$ *and round complexity* $\text{ROUNDS}_\ell(\text{AddLastBit}) = \text{ROUNDS}_1(\Pi_{\text{BA}})$.

Proof. Since the honest parties hold the same bitstring PREFIX* when joining the subprotocol, Π_{BA} ensures that they obtain the same bitstring of PREFIX$^\star \parallel$ B* of $|\text{PREFIX}^\star| + 1$ bits. Moreover, the Validity property of Π_{BA} ensures that the bit agreed upon B* was proposed by an honest party. Hence, there is an honest party holding a valid value v whose ℓ-bit representation has PREFIX$^\star \parallel$ B* as prefix. The communication complexity and the round complexity follow from the fact that AddLastBit only invokes Π_{BA} once on one-bit inputs. ☐

Missing proofs for GetOutput. We present the proof of Lemma 3. This describes the subprotocol GetOutput, which enables the honest parties to obtain the final output.

Lemma 3. *Assume a* BA *protocol* Π_{BA}*, and that honest parties join* GetOutput *with the same value* ℓ*, and with the same bitstring* PREFIX* *representing the prefix of some valid value's* ℓ*-bit representation. In addition, assume that each party joins with some valid* ℓ*-bit input* $v_\perp$ *such that* *the ℓ-bit representations of $t+1$ honest parties' values $v_\perp$ do not have* $\textsc{prefix}^\star$ *as a prefix. Then, the honest parties obtain the same valid value v_{OUT}.*

$\textsc{GetOutput}$ *has communication complexity* $\textsc{bits}_\ell(\textsc{GetOutput}) = O(n^2) + \textsc{bits}_1(\Pi_{\text{BA}})$ *and round complexity* $\textsc{rounds}_\ell(\textsc{GetOutput}) = O(1) + \textsc{rounds}_1(\text{BA})$.

Proof. There are are $t+1$ honest parties holding values $v_\perp$ such that $\textsc{bits}_\ell(v_\perp)$ does not have $\textsc{prefix}^\star$ as prefix. For each of these parties, $v_\perp$ is either lower than $\textsc{min}_\ell(\textsc{prefix}^\star)$ or higher than $\textsc{max}_\ell(\textsc{prefix}^\star)$. This implies that there are at least $t+1$ honest parties sending bits $\textsc{b}$. Hence, each party receives $m \geq t+1$ bits $\textsc{b}$.

We need to show that each honest party's $\textsc{choice}$ is a bit $\textsc{b}$ sent by an honest party. Let P denote an honest party that obtained some value $\textsc{choice}$, and assume that no honest party has sent $\textsc{b} = \textsc{choice}$. Hence, P has received at most t bits $\textsc{choice}$, and at least $t+1$ honest bits $1 - \textsc{choice}$. We obtain a contradiction: P has received $m \geq 2t+1$ bits $\textsc{b}$, and $\lceil m/2 \rceil > t$. This means that P did not receive $\textsc{choice}$ from $\lceil m/2 \rceil$ parties.

Then, each honest party joins Π_{BA} with an honest party's bit $\textsc{b}$ as input, and therefore they agree on an honest party's bit $\textsc{b}$ due to Π_{BA}'s Validity condition.

If the bit agreed upon is 0, some honest party holds $v_\perp < \textsc{min}_\ell(\textsc{prefix}^\star)$. Since $\textsc{prefix}^\star$ is some valid value's prefix, $\textsc{min}_\ell(\textsc{prefix}^\star)$ is valid. Similarly, if the bit agreed upon is 1, some honest party holds $v_\perp > \textsc{max}_\ell(\textsc{prefix}^\star)$. Since $\textsc{prefix}^\star$ is some valid value's prefix, $\textsc{max}_\ell(\textsc{prefix}^\star)$ is a valid value.

For the communication complexity and round complexity, note that $\textsc{GetOutput}$ makes use of one round of communication where the parties sent at most a bit to all parties, and afterwards the parties run Π_{BA} on one-bit inputs. □

A.3 Missing proofs for FixedLengthCABlocks

Missing proofs for FindPrefixBlocks. We recall that the main difference between $\textsc{FindPrefix}$ and $\textsc{FindPrefixBlocks}$ is that the first implements binary search on *bits*, while the latter implements binary search on *blocks*. This will be the main difference in the analysis as well.

We start by analyzing the invariants of each iteration. The lemma below is a variant of Lemma 8 on blocks, and the proof is also reflects this.

Lemma 9. *Assume that the following properties hold at the beginning of iteration i.*

(A) All honest parties hold the same indices $1 \leq \textsc{left} \leq \textsc{right} \leq \ell + 1$, and the same bitstring $\textsc{prefix}^\star$ consisting of $\textsc{left} - 1$ blocks.

(B) $0 \leq \textsc{right} - \textsc{left} \leq 2^{\lceil \log_2 n^2 \rceil - (i-1)}$.

(C) Honest parties hold valid ℓ-bit values v such that $\textsc{bits}_\ell(v)$ has $\textsc{prefix}^\star$ as a prefix.

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

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

Proof. We assume that the condition $\textsc{left} = \textsc{right}$ is not yet met in iteration i (otherwise, the statement trivially holds). Then, Property (B) ensures that $\textsc{left} < \textsc{right}$, 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. $\textsc{prefix}^\star(i)$ is the value held at the beginning of iteration i, and $\textsc{prefix}^\star(i+1)$ 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 $\bot$:

(A) Honest parties compute the $\textsc{right}(i+1)$ index identically, while all other values remain unchanged. Note that $\textsc{left}(i) \leq \textsc{mid} < \textsc{right}(i)$ and therefore $\textsc{right}(i+1) := \textsc{mid}$ still satisfies $1 \leq \textsc{right}(i+1) \leq n^2+1$. Therefore, Property (A) holds at the beginning of iteration $i+1$ as well.

(B) All honest parties compute $\textsc{right}(i+1) := \textsc{mid} \geq \textsc{left}(i)$, while the $\textsc{left}$ index remains unchanged: $\textsc{left}(i+1) := \textsc{left}(i)$. We obtain the inequality below, which ensures that Property (B) holds at the beginning of iteration $i+1$.

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

(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 $v_\bot(i+1) := v(i)$ is a valid ℓ-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 $\textsc{bits}_\ell(v(i))$ does not have $\textsc{bits}$ as a prefix. This is ensured by the *Bounded Pre-Agreement* 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 $\textsc{bits}_\ell(v(i))$ 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 $1 \leq \textsc{left}(i+1) \leq n^2+1$. 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.

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

(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-bit value having prefix $\textsc{prefix}^\star(i+1)$. This implies that, at the beginning of iteration $i+1$, all honest parties hold ℓ-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 *Intrusion Tolerance* property of $\Pi_{\ell\text{BA}+}$ ensures that parties agree on a sequence of blocks $\text{PREFIX}^\star_{\text{LEFT}(i)} \| \ldots \| \text{PREFIX}^\star_{\text{MID}}$ that was proposed by an honest party holding value $v^\star$. Then, Property (C) ensures that $v^\star$ is a valid ℓ-bit value such that $\text{BITS}_\ell(v^\star)$ has prefix $\text{PREFIX}^\star(i+1)$. On the other hand, $v(i)$ is a valid ℓ-bit value such that $\text{BITS}_\ell(v(i))$ has $\text{PREFIX}^\star(i)$ as a prefix, but not $\text{PREFIX}^\star(i+1)$. Remark 2 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-bit value within the honest inputs' range.

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

□

Then, the proof of Lemma 4 will be similar to that of Lemma 1: the main difference is, once again, that we consider blocks instead of bits.

Lemma 4. *Assume a* BA *protocol* Π_{BA}, *and that honest parties join* FindPrefixBlocks *with the same (multiple of* n^2*)* ℓ, *and with valid* ℓ*-bit values* v. *Then, the honest parties obtain the same bitstring* $\text{PREFIX}^\star$ *of* $i^\star$ *blocks, and each honest party obtains two valid* ℓ*-bit values* v, $v_\perp$ *such that: (i) the* ℓ*-bit representations of the values* v *have prefix* $\text{PREFIX}^\star$*; (ii) for any bitstring* BITS *of* $i^\star + 1$ *blocks, at least* $t+1$ *honest parties hold values* $v_\perp$ *such that* $\text{BITS}_\ell(v_\perp)$ *does not have prefix* BITS.

FindPrefixBlocks *achieves communication complexity* $\text{BITS}_\ell(\text{FindPrefixBlocks}) = O(\ell \cdot n + \kappa \cdot n^2 \log^2 n) + O(\log n) \cdot \text{BITS}_\kappa(\Pi_{\text{BA}})$ *and round complexity* $\text{ROUNDS}_\ell(\text{FindPrefixBlocks}) = O(\log n) \cdot \text{ROUNDS}_\kappa(\Pi_{\text{BA}})$.

Proof. The properties listed in Lemma 9 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. Once the condition $\text{LEFT} = \text{RIGHT}$ is met, Property (C) guarantees that honest parties hold valid ℓ-bit values v having the bitstring $\text{PREFIX}^\star$ as a common prefix. According to Property (A), this common prefix consists of $i^\star := \text{LEFT} - 1$ blocks. From Property (D), it follows that the honest parties hold valid ℓ-bit values $v_\perp$. The same property implies that, for any bitstring BITS of $\text{RIGHT} = i^\star + 1$ blocks, the ℓ-bit representations of $t+1$ honest parties' values $v_\perp$ do not have BITS as a prefix. Hence, once the stopping condition is met, 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 $\text{LEFT} = \text{RIGHT}$ is met (for all honest parties simultaneously, due to Property (A)) by iteration $i := \lceil \log_2 n^2 \rceil + 2$. Property (B) ensures that, at the beginning of iteration i, $0 \le \text{RIGHT} - \text{LEFT} \le 2^{\lceil \log_2 n^2 \rceil - (i-1)}$. Then, if this condition was not met by iteration $i := \lceil \log_2 n^2 \rceil + 2$, the indices LEFT and RIGHT obtained by the honest parties in iteration i satisfy $0 \le \text{RIGHT} - \text{LEFT} \le 2^{\lceil \log_2 n^2 \rceil - (\lceil \log_2 n^2 \rceil + 1)} \le 2^{-1}$. Since the indices LEFT and RIGHT are natural numbers, we may conclude that $\text{RIGHT} - \text{LEFT} = 0$.

We may then discuss the round complexity of FindPrefixBlocks: since $O(\log n)$ iterations are sufficient and each iteration invokes $\Pi_{\ell\text{BA}+}$ once, we obtain that $\text{ROUNDS}_\ell(\text{FindPrefixBlocks}) = O(\log n) \cdot \text{ROUNDS}_\ell(\Pi_{\ell\text{BA}+})$. Then, Theorem 1 leads to the result claimed in the lemma's statement.

For the communication complexity, Property (B) of Lemma 9 ensures that, in each iteration $i < \lceil \log_2 n^2 \rceil + 2$, FindPrefixBlocks runs $\Pi_{\ell\text{BA}+}$ on inputs of at most $2^{\lceil \log_2 n^2 \rceil - i}$ blocks, hence of at most $2^{\lceil \log_2 n^2 \rceil - i} \cdot \ell / n^2 \le \ell / 2^{i-1}$ bits. Therefore, $\text{BITS}_\ell(\text{FindPrefixBlocks}) = \sum_{i=1}^{\lceil \log_2 n^2 \rceil + 1} \text{BITS}_{\ell/2^{i-1}}(\Pi_{\ell\text{BA}+})$. Using Theorem 1, and the fact that $\sum_{i=0}^{\infty} 1/2^i \le 2$, we obtain that $\text{BITS}_\ell(\text{FindPrefixBlocks}) = O(\ell \cdot n + \kappa \cdot n^2 \log^2 n) + O(n \log n) \cdot \text{BITS}_\kappa(\Pi_{\text{BA}})$. □

Missing proofs for AddLastBlock. We include the proof of Lemma 5 below. This describes the subprotocol AddLastBlock, which enables the honest parties to extend the prefix obtained in FindPrefixBlocks with one block.

Lemma 5. *Assume that the honest parties join* AddLastBlock *with the same value ℓ (that is a multiple of n^2), with the same bitstring prefix* $\textsc{prefix}^\star$ *of $i^\star < n^2$ blocks, and with valid ℓ-bit values v that have* $\textsc{prefix}^\star$ *as a prefix. Then, the honest parties agree on a bitstring of $i^\star + 1$ blocks that is the prefix of a valid value's ℓ-bit representation.* AddLastBlock *has communication complexity* $\textsc{bits}_\ell(\textsc{AddLastBlock}) = O(\ell \cdot n)$ *and round complexity* $\textsc{rounds}_\ell(\textsc{AddLastBlock}) = O(n)$.

Proof. HighCostCA ensures that the honest parties obtain the same bitstring $\textsc{block}'_{i^\star+1}$, that is within the honest range of blocks $\textsc{block}_{i^\star+1}(v)$. That is, some honest parties P_1 and P_2 have joined with block $\textsc{block}^1, \textsc{block}^2$ such that $\textsc{val}(\textsc{block}^1) \le \textsc{val}(\textsc{block}'_{i^\star+1}) \le \textsc{val}(\textsc{block}^2)$.

Then, since all honest parties have joined HighCostCA with bitstrings of ℓ/n^2 bits, $\textsc{block}'_{i^\star+1}$ is also a block. Then, since the honest parties hold the same bitstring $\textsc{prefix}^\star$ of $i^\star$ blocks, they obtain the same bitstring $\textsc{prefix}^\star \parallel \textsc{block}'_{i^\star+1}$ of $i^\star + 1$ blocks.

It remains to show that $\textsc{prefix}^\star \parallel \textsc{block}'_{i^\star+1}$ is some valid values' prefix. The honest parties P_1 and P_2 hold valid values v^1 and v^2 such that $\textsc{bits}_\ell(v^1)$ has prefix $\textsc{prefix}^\star \parallel \textsc{block}^1$, and $\textsc{bits}_\ell(v^2)$ has prefix $\textsc{prefix}^\star \parallel \textsc{block}^2$. Then, since $\textsc{val}(\textsc{block}^1) \le \textsc{val}(\textsc{block}'_{i^\star+1}) \le \textsc{val}(\textsc{block}^2)$, there is a valid value $v^\star$ such that $\textsc{bits}_\ell(v^\star)$ has $\textsc{prefix}^\star \parallel \textsc{block}'_{i^\star+1}$ as prefix.

The communication and round complexities follow from those of HighCostCA presented in Theorem 3, as AddLastBlock invokes HighCostCA once on inputs of ℓ/n^2 bits. □

Missing proofs for FixedLengthCABlocks. We include the proof of Theorem 4, describing the guarantees of FixedLengthCABlocks, restated below.

Theorem 4. *Assume a* BA *protocol* Π_{BA} *resilient against $t < n/3$ corruptions. If the honest parties hold ℓ-bit inputs $v_{\text{IN}} \in \mathbb{N}$, where ℓ is a publicly known multiple of n^2,* FixedLengthCABlocks *is a* CA *protocol resilient against $t < n/3$ corruptions.*

The protocol achieves communication complexity $\textsc{bits}_\ell(\textsc{FixedLengthCABlocks}) = O(\ell n + \kappa \cdot n^2 \log^2 n) + O(\log n) \cdot \textsc{bits}_\kappa(\Pi_{\text{BA}})$*, and round complexity* $\textsc{rounds}_\ell(\textsc{FixedLengthCABlocks}) = O(n) + O(\log n) \cdot \textsc{rounds}_\kappa(\Pi_{\text{BA}})$.

Proof. Lemma 4 enables the parties to agree on a bitstring $\textsc{prefix}^\star$, and provides them with valid ℓ-bit values $v, v_\perp$ such that the values v have prefix $\textsc{prefix}^\star$. If $|\textsc{prefix}^\star| = \ell$, the honest parties hold the same valid value v, and therefore CA is achieved. Otherwise, Lemma 5 ensures that parties obtain the same bitstring $\textsc{prefix}^\star$ such that there are $t + 1$ honest parties whose values $v_\perp$ do not have $\textsc{prefix}^\star$ as a prefix. Then, GetOutput's preconditions are met, and Lemma 3 ensures that CA is achieved. The communication complexity and the round complexity follow by summing up the complexities of each subprotocol. □

A.4 High-Communication-Cost CA

In subprotocol AddLastBlock of FixedLengthCABlocks, we have used a high-communication-cost CA protocol, described in Theorem 3, restated below.

Theorem 3 (Theorem 4 of [47]). *There is a* CA *protocol* HighCostCA *for* $\mathbb{N}$ *resilient against $t < n/3$ corruptions, with communication complexity* $\textsc{bits}_\ell(\textsc{HighCostCA}) = O(\ell \cdot n^3)$*, and round complexity* $\textsc{rounds}_\ell(\textsc{HighCostCA}) = O(n)$.

In this section, we present the protocol HighCostCA, obtained by making minor adjustments to the Median Validity protocol of [47]. This is a variant of the well-known King BA protocol [7].

This protocol involves a *setup stage*, where the parties distribute their inputs and each party P estimates a *trusted* interval. In the protocol of [47], this is an interval containing values *close* to the honest median. For us, any interval included in the honest inputs' rage suffices, allowing us to slightly simplify the protocol. If a party receives $n - t + k$ values, then at most k out of these values were sent by the byzantine parties. Hence, the interval between the $(k+1)$-th lowest and the $(k+1)$-th highest values received is included in the honest inputs' range. Each party afterwards sends its trusted interval to all parties. We need to be careful about a small technical detail here – byzantine parties may send non-integer values that fit into the trusted intervals, and honest parties forward forwarding such values would increase the communication cost. To prevent this, we take into account that we only run HighCostCA on values in $\mathbb{N}$, and therefore the honest parties may ignore any values outside $\mathbb{N}$ in each step of the protocol.

At the end of the setup stage, each party chooses a suggestion value: this is a value that appears in $n - t$ of the intervals received, hence in at least $t + 1$ honest intervals, which is roughly *likely* to get support in the *search stage*.

The parties then start the *search stage*. The only adjustment we make here is that parties ignore all values outside $\mathbb{N}$. The parties run $t + 1$ sequential phases. In each phase i, P_i is *the king*. Roughly, the king distributes the value it believes the parties should agree on. If the king is honest (and $t + 1$ phases means at least one honest king), all honest parties accept the king's suggestion, and agreement is achieved. In addition, the agreement obtained is maintained in all further phases. We present the code below.

HighCostCA

Code for party P with input v_{in}

Setup stage
Send v_{IN} to all parties.
Out of the $n - t + k$ values in $\mathbb{N}$ received, set intervalMin := the $(k+1)$-th lowest value received, and intervalMax := the $(k+1)$-th highest value received.
Send intervalMin, intervalMax to all parties. (ℓn^2).
Let suggestion := some value in $\mathbb{N}$ that appears in $n - t$ of the intervals received.
Let current := suggestion.

Search stage
for $i = 1 \dots t + 1$ **do**
    Send current to all parties.
    If you have received the same value $v \in \mathbb{N}$ from $n - t$ parties, send (propose, v) to all parties.
    If you have received the same (propose, v) with $v \in \mathbb{N}$ from $t + 1$ parties, set current $= v$.
    King P_i only:
        If you have received the same (propose, v) with $v \in \mathbb{N}$ from $t+1$ parties: kingValue := v.
        Otherwise, set kingValue := suggestion.
        Send kingValue to all parties.
    If kingValue = current or kingValue $\in$ [intervalMin, intervalMax] $\cap\, \mathbb{N}$:
        Send (vote, kingValue) to all parties.
    If you have not received $n - t$ messages (propose, v') for any value $v' \in \mathbb{N}$:
        If you have received $t + 1$ messages (vote, kingValue) for some value kingValue $\in \mathbb{N}$:
            Set current := kingValue.
end for
Output current.

We now present the analysis of HighCostCA. Most of the lemmas below follow the analysis of [47].

The lemma below ensures that the interval obtained by each honest party is indeed in the honest inputs' range.

Lemma 10. *Let $v_1, v_2, \dots v_{n-t}$ be the $n-t$ honest inputs arranged in increasing order. Then, for every honest party P,* intervalMin *and* intervalMax *are well-defined and satisfy:* $v_1 \leq$ intervalMin $\leq v_{t+1} \leq$ intervalMax $\leq v_{n-t}$.

Proof. P receives $n-t+k$ values v_{IN}, where $0 \leq k \leq t$. Since the $n-t$ honest inputs are received, only k of these $n-t+k$ values are sent by byzantine parties, and hence may be outside the honest inputs' range. Hence, there are at most k values lower than v_1, and at most k values higher than v_{n-t}.

Note that both intervalMin and intervalMax are well defined, since $k+1 \leq (n-t+k)$: there is a $(k+1)$-th lowest value received, and a $(k+1)$-th highest value received.

Since there are at most k values lower than v_1, intervalMin := the $(k+1)$-th lowest value received is at least v_1. Moreover, since all honest inputs are received, intervalMin $\leq v_{k+1} \leq v_{t+1}$.

Similarly, since there are at most k values higher than v_{n-t}, intervalMax := the $(k+1)$-th highest value received is at most $v_{\max}$. Moreover, since all honest inputs are received, intervalMax $\geq v_{(n-t)-k} \geq v_{n-2t}$. Since $n > 3t$, we obtain that $n-2t \geq t+1$, and therefore intervalMax $\geq v_{t+1}$. □

The following two properties are immediate corollaries of Lemma 10 and ensure the success of the setup stage.

Corollary 3. *For any honest party P,* [intervalMin, intervalMax] *is non-empty and it is a subset of the honest inputs' range.*

Corollary 4. *The intervals* [intervalMin, intervalMax] *obtained by the honest parties have a non-empty intersection. Moreover, the intersection contains some natural number.*

The next lemma ensures that, at all times, every honest party holds a value current that is in some honest party's trusted interval. This also implies that honest parties hold valid values at all times.

Lemma 11. *Assume that at the beginning of iteration i, every honest party P holds a value* current *that is in some honest party's interval* [intervalMin, intervalMax].

Then, at the end of iteration i, the same property holds: every honest party P holds a value current *that is in some honest party's interval* [intervalMin, intervalMax].

Proof. P may update its value current to some value v if it receives $t+1$ messages (propose, v). If this is the case, at least one of these propose messages was sent by an honest party P', and v is the value current held by P' at the beginning of iteration i. This means that v is also a value in within some honest party's interval [intervalMin, intervalMax]. Hence, up to line 12, all honest parties hold values current satisfying this property.

Afterwards, if P did not receive $n-t$ messages (propose, v) for some value v, it may try to update its value current to the king's suggestion. P first checks if it has received $t+1$ messages (vote, kingValue). If this is the case, then at least one honest party P' has sent a (vote, kingValue) message. Then, there are two cases: kingValue satisfies kingValue $\in$ [intervalMin, intervalMax] $\cap \mathbb{N}$ for P', or kingValue is the value current held by P', which we have proved to be in some honest party's interval [intervalMin, intervalMax]. □

We now ensure that, if two honest parties update their value CURRENT at the beginning of a phase, then they update CURRENT to the same value.

Lemma 12. *If, in iteration i, an honest party P receives $t+1$ messages* (PROPOSE, v) *for some value v, no honest party receives $t+1$ messages* (PROPOSE, v') *for some value $v' \neq v$.*

Proof. Assume that an honest party P' receives $t+1$ messages (PROPOSE, v') for some value $v' \neq v$.

P has received $t+1$ PROPOSE messages for v, hence at least one honest party P'' has sent (PROPOSE, v). This implies that P'' has received v from $n-t$ parties in line 11, hence from at least $n-2t$ honest parties.

Similarly, P' has received $t+1$ PROPOSE messages for v', hence at one honest party P''' has sent (PROPOSE, v'). This implies that P''' has received v' from $n-t$ parties in line 10, hence from at least $n-2t$ honest parties. We obtain a contradiction: at least $n-2t$ out honest parties have sent $v \neq v'$ and there are only $(n-t)-(n-2t) = t < n-2t$ honest parties that could have sent v'. □

The lemma below ensures that, once agreement is reached in some phase, it is maintained in all further phases.

Lemma 13. *If all honest parties hold the same value* CURRENT *at the beginning of iteration i, no honest party changes its value* CURRENT *during iteration i.*

Proof. All honest parties send CURRENT, and therefore all honest parties receive $n-t$ PROPOSE messages for this value CURRENT. No honest party sends (PROPOSE, v) with $v \neq$ CURRENT, and therefore no honest party receives $t+1$ PROPOSE messages for $v \neq$ CURRENT. Regardless of whether the king of this iteration is honest or not, parties have received $n-t$ messages PROPOSE for CURRENT and therefore no honest party updates its value to the king's suggestion. Therefore, the honest parties maintain their value CURRENT. □

The lemma below ensures that, in the first phase with an honest king, agreement is reached.

Lemma 14. *If, in iteration i, the king P_i is honest, then the honest parties hold the same value* CURRENT *at the end of iteration i.*

Proof. If the honest parties have started iteration i with the same value CURRENT, Lemma 13 ensures that the honest parties complete iteration i with the same value CURRENT.

We may then assume that honest parties held different values CURRENT at the beginning of iteration i. The remainder of the proof will be split into two cases, depending on how the honest king P_i defines its KINGVALUE.

Case 1: P_i has received the same (propose, v) from $t+1$ parties. Then, P_i sets KINGVALUE := v.

We first show that at least $t+1$ honest parties send (VOTE, v) in line 18. Since P_i has received $t+1$ PROPOSE messages for v, at least one honest party has sent (PROPOSE, v) and therefore it has received v from $n-t$ parties. This implies that $t+1$ honest parties held CURRENT := v at the beginning of the iteration. Lemma 12 ensures that every honest party receives at most t PROPOSE messages for any value $v' \neq v$, and therefore these honest parties still hold CURRENT := v in line 17. Therefore, the condition KINGVALUE = CURRENT in line 17 holds for at least $t+1$ honest parties, and $t+1$ honest parties send (VOTE, v). Moreover, since P_i is honest, no honest party sends (VOTE, v') for $v' \neq v$.

We now show that every honest party P holds CURRENT = v by the end of the iteration. P receives $t+1$ messages (VOTE, v) and at most t votes for any other values. Then, if P has not received $n-t$ PROPOSE messages for some value v', it sets CURRENT := v in line 21. Otherwise, if P has received $n-t$ PROPOSE messages for some value v', these messages are for $v'=v$, as guaranteed by Lemma 12. This implies that P has set CURRENT $:= v$ in line 12.

Case 2: P_i has not received the same (propose, v) from $t+1$ parties. In this case, P_i sets KINGVALUE := SUGGESTION. P_i has set SUGGESTION in line 5 to some natural number that appears in $n-t$ of the intervals [INTERVALMIN, INTERVALMAX] it has received, therefore in at least $n-2t \geq t+1$ of the honest parties' intervals.

Then, the condition KINGVALUE $\in$ [INTERVALMIN, INTERVALMAX] $\cap \mathbb{N}$ holds for at least $t+1$ honest parties, and these honest parties send (VOTE, KINGVALUE). In addition, since P_i is honest, every honest party receives at most t messages (VOTE, v') for $v' \neq$ KINGVALUE.

Note that no honest party has received $n-t$ messages (PROPOSE, v) for some value v: otherwise, P_i would have received at least the $n-2t \geq t+1$ messages (PROPOSE, v) sent by honest parties, which contradicts our assumption for this case. Hence, all honest parties reach line 12, and all honest parties have received the $t+1$ messages (VOTE, KINGVALUE) sent by honest parties, and at most t messages (VOTE, v') for $v' \neq$ KINGVALUE. This means that every honest party sets CURRENT := KINGVALUE. □

We may now present the proof of Theorem 3.

Proof of Theorem 3. Termination follows from the protocol's construction.

For Agreement, note that at least one of the $t+1$ kings of the $t+1$ iterations is honest. Lemma 14 ensures that, in the first iteration i where the king P_i is honest, the honest parties obtain the same value CURRENT. Afterwards, Lemma 13 ensures that the honest parties do not change their CURRENT value in any further iteration, and therefore they output the same value CURRENT.

For Convex Validity, Lemma 6 guarantees that the honest parties enter the loop with valid values. Then, Lemma 11 implies that, in each of the iterations, each honest party holds a value CURRENT that is in some honest party's interval [INTERVALMIN, INTERVALMAX], which is a subset of the honest inputs' range according to Lemma 11. Therefore, at the end of the $t+1$ iterations, the honest parties output a valid value.

The round complexity follows from the protocol's construction: the setup stage consists of $O(1)$ communication rounds, and the search stage of $t+1$ iterations, and each iteration consists of $O(1)$ communication rounds.

It remains to discuss the communication complexity. In the setup phase, each honest party sends three ℓ-bit values to all parties, with a communication cost of $O(\ell n^2)$ bits. Afterwards, in each iteration in the search stage, each honest party sends at most four ℓ-bit values to all parties, implying a communication cost of $O(\ell n^2)$ bits per iteration. Since there are $t+1$ iterations, we obtain a communication cost of $O(\ell n^3)$ bits in total. □

A.5 Protocol for $\mathbb{Z}$: missing proofs

We include the proof of Corollary 1, describing protocol $\Pi_{\mathbb{Z}}$.

Corollary 1. *Assume a* BA *protocol* Π_{BA} *resilient against* $t < n/3$ *corruptions. Then, if the honest parties hold inputs* $(-1)^{\mathrm{SIGN_{IN}}} \cdot v_{\mathrm{IN}}^{\mathbb{N}} \in \mathbb{Z}$*, such that* $v_{\mathrm{IN}}^{\mathbb{N}} \in \mathbb{N}$ *with* $\left|\mathrm{BITS}(v_{\mathrm{IN}}^{\mathbb{N}})\right| \leq \ell$*,* $\Pi_{\mathbb{Z}}$ *is a* CA *protocol resilient against* $t < n/3$ *corruptions, with communication complexity* $\mathrm{BITS}_\ell(\Pi_{\mathbb{Z}}) = O(\ell n + \kappa \cdot n^2 \log^2 n) + O(\log n) \cdot \mathrm{BITS}_\kappa(\Pi_{\mathrm{BA}})$*, and round complexity* $\mathrm{ROUNDS}_\ell(\Pi_{\mathbb{Z}}) = O(n) + O(\log n) \cdot \mathrm{ROUNDS}_\kappa(\Pi_{\mathrm{BA}})$.

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 $\mathrm{SIGN_{OUT}} = 0$, 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 5. 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 5, 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 $\text{ROUNDS}_{\ell}(\Pi_{\mathbb{Z}}) = \text{ROUNDS}_1(\Pi_{\text{BA}}) + \text{ROUNDS}_{\ell}(\Pi_{\mathbb{N}})$. Theorem 5 leads to the results claimed in the corollary's statement. □