Communication-Optimal Convex Agreement

Diana Ghinea*
Lucerne University of Applied Sciences and Arts
Zug, Switzerland
diana.ghinea@hslu.ch

Chen-Da Liu-Zhang
Lucerne University of Applied Sciences and Arts & Web3 Foundation
Zug, Switzerland
chen-da.liuzhang@hslu.ch

Roger Wattenhofer
ETH Zürich
Zurich, Switzerland
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 synchronous CA protocol with 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.

CCS Concepts

• **Theory of computation → Cryptographic protocols**.

Keywords

convex agreement, optimal communication, long messages

ACM Reference Format:
Diana Ghinea, Chen-Da Liu-Zhang, and Roger Wattenhofer. 2025. Communication-Optimal Convex Agreement. In *ACM Symposium on Principles of Distributed Computing (PODC '25), June 16–20, 2025, Huatulco, Mexico.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3732772.3733551

Related Version: A full version of this paper is available at [27].

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 [14, 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 [4], transaction ordering in blockchain [13], and distributed machine learning [9, 17, 18, 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 $\Omega(\ell n)$ bits [22, 41], and this approach incurs a sub-optimal cost of at least $O(\ell n^2)$ bits. In fact, the communication complexity of existing CA protocols [14, 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 [6, 22, 23, 34, 41] via so-called *extension protocols*: these are protocols that achieve optimal communication cost of $O(\ell n)$ bits for sufficiently large values ℓ, relying on protocols for *short messages* as building blocks. Concretely, these protocols achieve a communication cost of $O(\ell n + \mathrm{poly}(n, \kappa))$ bits, where κ is a security parameter (they make use of cryptographic primitives that compress the input values to $O(\kappa)$ bits). 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

*This work was partially carried out while the author was at ETH Zürich.

PODC '25, Huatulco, Mexico

ACM ISBN 979-8-4007-1885-4/25/06
https://doi.org/10.1145/3732772.3733551

synchronous model for CA. Concretely, we address the following question:

Can we achieve CA with communication complexity $O(\ell n + \mathrm{poly}(n, \kappa))$?

We answer this question in the affirmative. More concretely, we introduce a 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 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). Our protocol makes use of collision-resistant hash functions.

1.1 Related Work

Convex Agreement. The requirement of obtaining outputs within the honest inputs' range has been first introduced in [15] for Approximate Agreement (AA). AA relaxes the agreement requirement, allowing the parties' outputs to deviate by a predefined error $\varepsilon > 0$. This relaxation allows for deterministic asynchronous protocols, circumventing the FLP result [21]. AA has been a subject of an extensive line of works, focusing on optimal convergence rates [5, 19, 20], higher resilience [1, 25, 33], and different input spaces, such as multidimensional inputs [26, 37, 50], or graphs and abstract convexity spaces [3, 14, 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 [14, 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 [13, 47], or 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 [22] later achieve BA in the honest majority setting with the asymptotically optimal communication complexity of $O(\ell n + \mathrm{poly}(n, \kappa))$ bits, assuming a universal hash function. Further works have provided error-free solutions focusing on reducing the additional $\mathrm{poly}(n, \kappa)$ factor in the communication complexity both in the $t < n/3$ setting [23, 35, 41] and in the honest-majority setting [6, 23, 41]. Extension protocols have also been a topic of interest for problems related to BA, such as BC in the $t < n$ setting [10, 28], or asynchronous Reliable Broadcast [8, 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 [16] 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 [11, 40]. However, randomized protocols have offered a path to subquadratic communication [2, 7, 12, 24, 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 for shorter messages as an interesting open problem.

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. The network is synchronous: the parties' clocks are synchronized, all messages get delivered within Δ time, and Δ is publicly known.

We assume 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. Throughout the paper, we use the terms *byzantine* and *corrupted* interchangeably.

In addition, we consider a security parameter κ and we make use of a collision-resistant hash function $H_\kappa : \{0,1\}^\star \to \{0,1\}^\kappa$. Informally, a hash function $H_\kappa : \{0,1\}^\star \to \{0,1\}^\kappa$ 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] As a result, our protocols have a negligible failure probability.

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

2.1 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 up to 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 up to 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 $v_{OUT} = v$;*
- *(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 Π's worst-case round complexity.

2.2 Binary Representations

We establish a few notations that will be used throughout the paper.

For a value $v \in \mathbb{N}$, we may define its binary representation as $\textsc{bits}(v) := \textsc{b}_1\textsc{b}_2 \ldots \textsc{b}_k$ such that the following hold: $2^{k-1} \le v < 2^k$, $\textsc{b}_i \in \{0,1\}$ for every $1 \le i \le k$, and $\sum_{i=1}^{k} \textsc{b}_i \cdot 2^{k-i} = v$.

For $\ell \ge k$, 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 $1 \le i \le \ell$, $\textsc{b}^i_\ell(v)$ denotes the i-th leftmost bit in v's ℓ-bit representation $\textsc{bits}_\ell(v)$.

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 $\textsc{b}_i \in \{0,1\}$), $\textsc{val}(\textsc{bits}) := \sum_{i=1}^{k} \textsc{b}_i \cdot 2^{k-i}$.

We denote the length of a bitstring $\textsc{bits}$ by $|\textsc{bits}|$, and $\|$ is the concatenation operator.

We 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 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 $\textsc{prefix}^\star$ as a prefix: such values are either lower than any value with prefix $\textsc{prefix}^\star$, meaning that $\textsc{min}_\ell(\textsc{prefix}^\star)$ is valid, or higher than any value with prefix $\textsc{prefix}^\star$, which means that $\textsc{max}_\ell(\textsc{prefix}^\star)$ is valid. These parties will announce which of these two options they believe to be 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 $\textsc{prefix}^\star$, and each party obtains two ℓ-bit valid values v and $v_\perp$ such that: (i) the values v have $\textsc{prefix}^\star$ as a prefix, and (ii) for any bitstring $\textsc{bits}$ of $|\textsc{prefix}^\star| + 1$ bits, there are $t+1$ honest parties whose values $v_\perp$ do not have $\textsc{bits}$ as a prefix. If $|\textsc{prefix}^\star| = \ell$, 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 $\textsc{prefix}^\star$ using their values v, ensuring that the extended $\textsc{prefix}^\star$ is still some valid values' prefix. Now there are $t+1$ honest parties holding values $v_\perp$ that do not have prefix $\textsc{prefix}^\star$, 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

```
PREFIX*, v, v⊥ := FindPrefix(ℓ, v).
If |PREFIX*| = ℓ, return v.
PREFIX* := AddLastBit(ℓ, v, PREFIX*).
Return v := GetOutput(ℓ, v⊥, PREFIX*).
```

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

3.1 Finding a Valid Value's Prefix

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 agreed upon 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 $\bot$.*

Definition 4 (Bounded Pre-Agreement). *If the honest parties agree on $\bot$, fewer than $n-2t$ honest parties hold the same input value.*

In Section 7, we present a protocol $\Pi_{\ell\text{BA+}}$ achieving these guarantees with communication complexity $O(\ell n + \text{poly}(n, \kappa))$, as described in Theorem 1.

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

We proceed in $O(\log \ell)$ iterations. In the first iteration, the parties check whether $\Pi_{\ell\text{BA+}}$ returns $\bot$ on the first half of their values' ℓ-bit representations $\text{B}_1 \| \ldots \| \text{B}_{\text{MID}}$. If $\Pi_{\ell\text{BA+}}$ returns $\bot$, $\text{MID} - 1$ 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 $v_\bot := v$ and then continue the search for $i^\star$ within bits $\text{B}_1, \ldots, \text{B}_{\text{MID}-1}$ in the next iteration, using an identical approach.

Otherwise, if $\Pi_{\ell\text{BA+}}$ returns a bitstring of MID bits $\text{PREFIX}^\star_1 \| \ldots \| \text{PREFIX}^\star_{\text{MID}}$, 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 $\text{VAL}(\text{B}_1 \| \ldots \| \text{B}_{\text{MID}}) < \text{VAL}(\text{PREFIX}^\star_1 \| \ldots \| \text{PREFIX}^\star_{\text{MID}})$, meaning that $v < v^\star$, then $\text{MIN}_\ell(\text{PREFIX}^\star_1 \| \ldots \| \text{PREFIX}^\star_{\text{MID}})$ is in $[v, v^\star]$ and therefore is valid. Otherwise, if $\text{VAL}(\text{B}_1 \| \ldots \| \text{B}_{\text{MID}}) > \text{VAL}(\text{PREFIX}^\star_1 \| \ldots \| \text{PREFIX}^\star_{\text{MID}})$, meaning that $v > v^\star$, then $\text{MAX}_\ell(\text{PREFIX}^\star_1 \| \ldots \| \text{PREFIX}^\star_{\text{MID}})$ 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 $\bot$ and the parties hold identical values v, or $i^\star$ is found. We present the code below.

FindPrefix(ℓ, v)

Code for party P

$\text{LEFT} := 1$; $\text{RIGHT} := \ell + 1$; $v := v_{\text{IN}}$.
$v_\bot := v_{\text{IN}}$; $\text{PREFIX}^\star :=$ empty string.
loop
  If $\text{LEFT} = \text{RIGHT}$, exit the loop.
  $(\text{B}_1, \text{B}_2, \ldots, \text{B}_\ell) := \text{BITS}_\ell(v)$, $\text{MID} := \lfloor(\text{LEFT} + \text{RIGHT})/2\rfloor$.
  Join $\Pi_{\ell\text{BA+}}$ with input $\text{B}_{\text{LEFT}} \| \ldots \| \text{B}_{\text{MID}}$.
  If $\Pi_{\ell\text{BA+}}$ has returned $\bot$, set $v_\bot := v$ and $\text{RIGHT} := \text{MID}$.
  If $\Pi_{\ell\text{BA+}}$ has returned bits $\text{PREFIX}^\star_{\text{LEFT}} \| \ldots \| \text{PREFIX}^\star_{\text{MID}}$:
    $\text{PREFIX}^\star := \text{PREFIX}^\star \| \text{PREFIX}^\star_{\text{LEFT}} \| \ldots \| \text{PREFIX}^\star_{\text{MID}}$.
    If $\text{VAL}(\text{B}_1 \| \ldots \| \text{B}_{\text{MID}}) < \text{VAL}(\text{PREFIX}^\star)$:
      $v := \text{MIN}_\ell(\text{PREFIX}^\star)$.
    If $\text{VAL}(\text{B}_1 \| \ldots \| \text{B}_{\text{MID}}) > \text{VAL}(\text{PREFIX}^\star)$:
      $v := \text{MAX}_\ell(\text{PREFIX}^\star)$.
    Set $\text{LEFT} := \text{MID} + 1$.
end loop
Return $\text{PREFIX}^\star$, v, $v_\bot$.

The lemma below describes the guarantees of FindPrefix. The formal proof can be found in the full version of our paper.

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, $v_\bot$ such that:

(i) $\text{PREFIX}^\star$ is a prefix of $\text{BITS}_\ell(v)$;
(ii) for any bitstring BITS consisting of $\left|\text{PREFIX}^\star\right| + 1$ bits, there are at least $t+1$ honest parties that hold values $v_\bot$ such that $\text{BITS}_\ell(v_\bot)$ does not have prefix BITS.

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

Proof Sketch. We consider the properties below. If these properties are satisfied at the beginning of iteration $i \geq 1$ of the loop, then either the stopping condition $\text{LEFT} = \text{RIGHT}$ is met 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 the first iteration due to the variables' initialization.

(A) All honest parties hold the same indices LEFT and RIGHT such that $1 \leq \text{LEFT} \leq \text{RIGHT} \leq \ell + 1$, and the same bitstring $\text{PREFIX}^\star$ consisting of $\text{LEFT} - 1$ bits.
(B) Honest parties' indices LEFT and RIGHT satisfy the following condition: $0 \leq \text{RIGHT} - \text{LEFT} \leq 2^{\lceil \log_2 \ell \rceil - (i-1)}$.
(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 $v_\bot$. In addition, for any bitstring BITS of RIGHT bits, there are $t+1$ honest parties holding values $v_\bot$ such that $\text{BITS}_\ell(v_\bot)$ does not have prefix BITS.

Property (B) implies that the condition $\text{LEFT} = \text{RIGHT}$ 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 honest parties hold valid ℓ-bit values v with prefix $\text{PREFIX}^\star$. Finally, Property (D) ensures that, honest parties hold valid ℓ-bit values $v_\bot$ 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 $v_\bot$ do not have $\text{PREFIX}^\star$ as a prefix.

As each of the $O(\log \ell)$ iterations invokes $\Pi_{\ell\text{BA+}}$ once, we may write the round complexity as $\text{ROUNDS}_\ell(\text{FindPrefix}) = O(\log \ell) \cdot \text{ROUNDS}_\ell(\Pi_{\ell\text{BA+}})$. Afterwards, Theorem 1 enables us to obtain the round complexity claimed in the lemma's statement.

For the communication complexity, we note that, in each iteration $i < \lceil \log_2 n \rceil + 2$, Property (B) ensures that FindPrefix runs $\Pi_{\ell\text{BA+}}$ on bitstrings consisting of at most $2^{\lceil \log_2 \ell \rceil - i} \leq \ell / 2^{i-1}$ bits. Then, we may write the communication complexity of FindPrefix as follows:

$$\text{BITS}_\ell(\text{FindPrefix}) = \sum_{i=1}^{\lceil \log_2 \ell \rceil + 1} \text{BITS}_{\ell/2^{i-1}}(\Pi_{\ell\text{BA+}}).$$

Using Theorem 1 and that $\sum_{i=0}^{\infty} 1/2^i \leq 2$, we obtain our claimed communication cost. □

3.2 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 value's 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.

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 lemma below describes the properties of AddLastBit and is proven in the paper's full version.

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 $\text{BITS}_\ell(\text{AddLastBit}) = \text{BITS}_1(\Pi_{BA})$ and round complexity $\text{ROUNDS}_\ell(\text{AddLastBit}) = \text{ROUNDS}_1(\Pi_{BA})$.*

3.3 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 $\text{MAX}_\ell(\text{PREFIX}^\star)$, hence 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

```
If PREFIX* is not a prefix of BITS_ℓ(v⊥):
      Set B := 0 if v⊥ < MIN_ℓ(PREFIX*) and B := 1 otherwise.
      Send B to all parties.
Set m := the number of bits B received.
Set CHOICE := a bit B received from ⌈m/2⌉ parties.
Join Π_BA with input CHOICE. If the bit agreed upon is 0, return
   MIN_ℓ(PREFIX*). Otherwise, return MAX_ℓ(PREFIX)*.
```

The next lemma presents the properties of GetOutput. The proof is included our paper's full version.

Lemma 3. *Assume a BA protocol Π_{BA}, and that honest parties join GetOutput with the same value ℓ, and with the same bitstring $\text{PREFIX}^\star$ 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 $\text{PREFIX}^\star$ as a prefix. Then, the honest parties obtain the same valid value v_{OUT}. GetOutput has communication complexity $\text{BITS}_\ell(\text{GetOutput}) = O(n^2) + \text{BITS}_1(\Pi_{BA})$ and round complexity $\text{ROUNDS}_\ell(\text{GetOutput}) = O(1) + \text{ROUNDS}_1(BA)$.*

3.4 Protocol Analysis

We have now reached the final theorem of this section, which presents the guarantees of FixedLengthCA. We highlight that, when considering inputs of ℓ bits such that $\ell \in \text{poly}(n)$ and therefore $\log \ell \in O(\log n)$, the communication complexity of our protocol FixedLengthCA becomes $\text{BITS}_\ell(\text{FixedLengthCA}) = O(\ell n + \kappa \cdot n^2 \log^2 n) + O(\log n) \cdot \text{BITS}_\kappa(\Pi_{\text{BA}})$. Similarly, the round complexity becomes $\text{ROUNDS}_\ell(\text{FixedLengthCA}) = O(\log n) \cdot \text{ROUNDS}_\kappa(\Pi_{\text{BA}})$.

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

Proof. Lemma 1 ensures that FindPrefix enables the parties to agree on a bitstring $\text{PREFIX}^\star$, and provides them with valid ℓ-bit values $v, v_\perp$ such that the values v have prefix $\text{PREFIX}^\star$.

If $\left|\text{PREFIX}^\star\right| = \ell$, the honest parties hold the same valid value v, and therefore CA is achieved.

Otherwise, Lemma 2 ensures that parties obtain the same bitstring $\text{PREFIX}^\star$ such that there are $t+1$ honest parties whose values $v_\perp$ do not have $\text{PREFIX}^\star$ 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 *Longer* Inputs in $\mathbb{N}$ of Fixed length

The protocol FixedLengthCA presented in Section 3 achieves our communication complexity goal when $\ell \in \text{poly}(n)$. 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 $O(\ell n + \text{poly}(n, \kappa))$ 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. Then, each party will split its ℓ-bit value into n^2 blocks $\text{BLOCK}_1, \text{BLOCK}_2, \ldots, \text{BLOCK}_{n^2}$ of ℓ/n^2 bits each. For an ℓ-bit value $v \in \mathbb{N}$, we define $\text{BLOCKS}(v) := (\text{BLOCK}_1, \text{BLOCK}_2, \ldots, \text{BLOCK}_{n^2})$ such that $\text{BITS}_\ell(v) = \text{BLOCK}_1 \parallel \text{BLOCK}_2 \parallel \ldots \parallel \text{BLOCK}_{n^2}$, and, for every $1 \leq i \leq n^2$, $\left|\text{BLOCK}_i\right| = \ell/n^2$. We denote BLOCK_i by $\text{BLOCK}_i(v)$, 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 $\text{B}_{\text{LEFT}} \parallel \ldots \parallel \text{B}_{\text{MID}}$ of honest parties' values v, we compare sequences of blocks $\text{BLOCK}_{\text{LEFT}} \| \ldots \| \text{BLOCK}_{\text{MID}}$. 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 $\text{PREFIX}^\star$ of $i^\star$ *blocks*, and each party obtains two ℓ-bit valid values $v, v_\perp$. The values v have prefix $\text{PREFIX}^\star$, 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

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

The lemma below is the *block version* of Lemma 1. The proof is included in the full version of our paper.

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.

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

AddLastBit becomes AddLastBlock: in order to decide on a final output using the values $v_\perp$, we need to append one block to $\text{PREFIX}^\star$, so that the extended $\text{PREFIX}^\star$ is a valid values' prefix. As the honest parties' values v have $\text{PREFIX}^\star$ 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. The full version of our paper provides a detailed description of this protocol.

Theorem 3 (Theorem 4 of [47]). *There is a protocol HighCostCA achieving CA for $\mathbb{N}$ up to $t < n/3$ 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. Lemma 5 describes the guarantees of AddLastBlock, and the proof is deferred to the full version of our paper.

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

Code for party P

Set $i^\star := \left|\text{PREFIX}^\star\right| / (\ell/n^2)$, i.e., the number of blocks in $\text{PREFIX}^\star$.
Set $\text{BLOCK}'_{i^\star+1} := \text{HighCostCA}(\text{BLOCK}_{i^\star+1}(v))$.
Return $\text{PREFIX}^\star \parallel \text{BLOCK}'_{i^\star+1}$.

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 $\text{PREFIX}^\star$ of $i^\star < n^2$ blocks, and with valid ℓ-bit values v that have $\text{PREFIX}^\star$ 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 $\text{BITS}_\ell(\text{AddLastBlock}) = O(\ell \cdot n)$ and round complexity $\text{ROUNDS}_\ell(\text{AddLastBlock}) = O(n)$.*

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

FixedLengthCABlocks(ℓ, v)

Code for party P

$\text{PREFIX}^\star, v, v_\perp := \text{FindPrefixBlocks}(\ell, v)$.
If $\left|\text{PREFIX}^\star\right| := \ell$, return v.
$\text{PREFIX}^\star := \text{AddLastBlock}(\ell, v, \text{PREFIX}^\star)$.
Return $v := \text{GetOutput}(\ell, v_\perp, \text{PREFIX}^\star)$.

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 the full version of our paper.

Theorem 4. *Assume a BA protocol Π_{BA} resilient against $t < n/3$ corruptions. If the honest parties hold ℓ-bit input values $v_{IN} \in \mathbb{N}$, where $\ell > 0$ is a publicly known multiple of n^2, FixedLengthCABlocks is a CA protocol resilient against $t < n/3$ corruptions. In addition, the communication complexity is $\text{BITS}_\ell(\text{FixedLengthCABlocks}) = O(\ell n + \kappa \cdot n^2 \log^2 n) + O(\log n) \cdot \text{BITS}_\kappa(\Pi_{BA})$ and the round complexity is $\text{ROUNDS}_\ell(\text{FixedLengthCABlocks}) = O(n) + O(\log n) \cdot \text{ROUNDS}_\kappa(\Pi_{BA})$.*

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. FixedLengthCA matches our communication complexity goal when $\ell \in \text{poly}(n)$, while 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 $\left|\text{BITS}(v_{\text{IN}})\right| \leq n^2$, and input 1 otherwise.

If Π_{BA} returns 0, the parties obtain an estimation for ℓ within $O(\log n) \cdot \text{ROUNDS}_1(\Pi_{\text{BA}})$ rounds by comparing their inputs' length with powers of two. Afterwards, they run FixedLengthCA.

Otherwise, if Π_{BA} returns 1, the parties obtain an estimation for ℓ by agreeing on a block size using the high-communication-cost protocol HighCostCA. Once the estimation is obtained, they run FixedLengthCABlocks.

We present the code of our protocol $\Pi_{\mathbb{N}}$ below.

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

Code for party P with input v_{IN}

1: Join Π_{BA} with input 0 if $|\text{BITS}(v_{\text{IN}})| \leq n^2$ and 1 otherwise.
2: If Π_{BA} has returned 0:
3:     If $|\text{BITS}(v_{\text{IN}})| > n^2$, set $v_{\text{IN}} := 2^{n^2} - 1$.
4:     For $i = 0 \ldots \lceil \log_2 n^2 \rceil$:
5:         Let $\text{B} := 0$ if $|\text{BITS}(v_{\text{IN}})| \leq 2^i$ and 1 otherwise.
6:         Join Π_{BA} with input B. If Π_{BA} returns 0:
7:             Let $\ell_{\text{EST}} := 2^i$.
8:             If $|\text{BITS}(v_{\text{IN}})| > \ell_{\text{EST}}$, set $v_{\text{IN}} := 2^{\ell_{\text{EST}}} - 1$.
9:             Set $v_{\text{OUT}} := \text{FixedLengthCA}(\ell_{\text{EST}}, v)$.
10:            Output v_{OUT} and terminate.
11: Otherwise, if Π_{BA} has returned 1:
12:    Set $\text{blockSize} := \lceil |\text{BITS}(v_{\text{IN}})| / n^2 \rceil$.
13:    Set $\text{blockSize}' := \text{HighCostCA}(\text{blockSize})$.
14:    Set $\ell_{\text{EST}} := \text{blockSize}' \cdot n^2$.
15:    If $|\text{BITS}(v_{\text{IN}})| \geq \ell_{\text{EST}}$, set $v_{\text{IN}} := 2^{\ell_{\text{EST}}} - 1$.
16:    Set $v_{\text{OUT}} := \text{FixedLengthCABlocks}(\ell_{\text{EST}}, v)$.
17:    Output v_{OUT} and terminate.

The following theorem presents the properties of $\Pi_{\mathbb{N}}$.

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

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. Therefore, 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 of the loop, then at least one honest party has joined this Π_{BA} invocation with input 0, meaning that $\ell_{\min} \leq 2^i$. 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 $\ell_{\text{EST}} \leq 2 \cdot \min(\ell_{\max}, n^2) \leq 2 \cdot n^2$ with $O(\log n)$ iterations. Finally, parties join FixedLengthCA with the same value $\ell_{\text{EST}} \leq 2 \cdot \min(\ell, n^2)$ 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 $O(\log n) \cdot \text{ROUNDS}_1(\Pi_{\text{BA}}) + \text{ROUNDS}_{2 \cdot \min(\ell, n^2)}(\text{FixedLengthCA})$, hence $O(\log n) \cdot \text{ROUNDS}_\kappa(\Pi_{\text{BA}})$.

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 the invocation of HighCostCA with inputs $\text{blockSize} := \lceil |\text{BITS}(v_{\text{IN}})| / n^2 \rceil$ and obtain a value $\text{blockSize}'$ with the property that $\lceil \ell_{\min}/n^2 \rceil \leq \text{blockSize}' \leq \lceil \ell_{\max}/n^2 \rceil$. Note that the values blockSize can be represented using $O(\log(\ell/n^2))$ bits, hence $O(\ell/n^2)$ bits, and therefore this step has communication cost $O(\ell n)$. We have also obtained that $\ell_{\text{EST}} := \text{blockSize}' \cdot n^2$ satisfies the following: $\ell_{\min} \leq \ell_{\text{EST}} \leq \ell_{\max} + n^2 \leq 2 \cdot \ell$.

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 communication complexity achieved by $\Pi_{\mathbb{N}}$ can be computed as $\text{BITS}_\ell(\Pi_{\mathbb{N}}) = \text{BITS}_1(\Pi_{\text{BA}}) + \text{BITS}_{O(\ell/n^2)}(\text{HighCostCA}) + \text{BITS}_{2\ell}(\text{FixedLengthCABlocks})$, hence we obtain $\text{BITS}_\ell(\Pi_{\mathbb{N}}) = O(\ell n + \kappa n^2 \log^2 n)$. Regarding the round complexity, $\text{ROUNDS}_\ell(\Pi_{\mathbb{N}})$ can be written as $\text{ROUNDS}_1(\Pi_{\text{BA}}) + \text{ROUNDS}_{O(\ell/n^2)}(\text{HighCostCA}) + \text{ROUNDS}(\text{FixedLengthCABlocks})$, and therefore $\text{ROUNDS}_\ell(\Pi_{\mathbb{N}}) = O(n) + O(\log n) \cdot \text{ROUNDS}_\kappa(\Pi_{\text{BA}})$. □

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 the full version of our paper.

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

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

Join Π_{BA} with input SIGN_{IN} and obtain output SIGN_{OUT}.
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 $v_{\text{OUT}}^{\mathbb{N}}$.
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 $(-1)^{\text{SIGN}_{IN}} \cdot v_{IN}^{\mathbb{N}} \in \mathbb{Z}$, such that $v_{IN}^{\mathbb{N}} \in \mathbb{N}$ with $|\text{BITS}(v_{IN}^{\mathbb{N}})| \leq \ell$, $\Pi_{\mathbb{Z}}$ is a CA protocol resilient against $t < n/3$ corruptions, with communication complexity $\text{BITS}_\ell(\Pi_{\mathbb{Z}}) = O(\ell n + \kappa \cdot n^2 \log^2 n) + O(\log n) \cdot \text{BITS}_\kappa(\Pi_{BA})$, and round complexity $\text{ROUNDS}_\ell(\Pi_{\mathbb{Z}}) = O(n) + O(\log n) \cdot \text{ROUNDS}_\kappa(\Pi_{BA})$.*

We state the final corollary, where we instantiate the assumed BA protocol Π_{BA} with a deterministic BA protocol with quadratic communication (e.g. [11]).

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

7 BA for Long Messages with Additional Properties

We recall that our CA protocol relies on a BA protocol $\Pi_{\ell\mathrm{BA+}}$ with communication complexity $O(\ell n + \mathrm{poly}(n, \kappa))$ 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 $\Pi_{\ell BA+}$ resilient against $t < n/3$ corruptions that achieves Intrusion Tolerance and Bounded Pre-Agreement, with communication complexity $\textsc{bits}_\ell(\Pi_{\ell BA+}) = O(\ell n + \kappa \cdot n^2 \log n) + \textsc{bits}_\kappa(\Pi_{BA})$, and with round complexity $\textsc{rounds}_\ell(\Pi_{\ell BA+}) = O(1) + \textsc{rounds}_\kappa(\Pi_{BA})$.*

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 $\Pi_{\mathrm{BA+}}$. Afterwards, $\Pi_{\ell\mathrm{BA+}}$ is constructed using the outline of prior works [6, 41].

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

In our protocol $\Pi_{\mathrm{BA+}}$, the parties first distribute their input values. Each party P receives $n - t$ values 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 $\textsc{vote}(\cdot)$, $\textsc{vote}(v_1)$, or $\textsc{vote}(v_1, v_2)$, 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 $a := \bot$ and $b := \bot$.

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 of $\Pi_{\mathrm{BA+}}$ below.

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

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 $\textsc{vote}(\cdot)$ to all parties. If there is only one, let this value be v_1 and send $\textsc{vote}(v_1)$ to all parties. If there are two, let these values be v_1 and v_2 and send $\textsc{vote}(v_1, v_2)$ 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.

The next theorem discusses the guarantees of $\Pi_{\mathrm{BA+}}$.

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

Proof. We first show that $\Pi_{\mathrm{BA+}}$ 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 $(\textsc{vote}, v)$. 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 second Π_{BA} invocation of line 4 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: if a party has received at least three input values sent by $n - 2t$ parties each, we obtain $3 \cdot (n - 2t) \leq n$, contradicting $t < n/3$. Hence, each party sees at most two values from $n - 2t$ parties each, and one of these is v. Then, each honest party sends a vote for v, and possibly for a second value.

Afterwards, each party receives the $n - t$ votes for v from the honest parties. In addition, each party receives $n - t$ votes for two values at most: 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 $3 \cdot (n-t) \leq 2n$, which contradicts that $t < n/3$.

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 a value v'' such that $v'' \notin \{v, v'\}$. 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.

It follows that 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 honest parties hold different values a: $a = v$ for some honest parties, and $a = v'$ for the others. As 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. □

7.2 From $\Pi_{\mathrm{BA+}}$ to $\Pi_{\ell\mathrm{BA+}}$

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

Then, $\Pi_{\ell\mathrm{BA+}}$ consists of three steps. In the first step, every party computes $s_1, \ldots, s_n :=$ RS.ENCODE(v_{IN}) and $z, w_1, \ldots, w_n :=$ MT.BUILD($\{s_1, \ldots, s_n\}$). Second, the parties agree on an encoding $z^\star$ with the help of $\Pi_{\mathrm{BA+}}$. In the third step, the parties obtain the final output. If $\Pi_{\mathrm{BA+}}$ returns $\perp$, the parties output $\perp$. Otherwise, if $\Pi_{\mathrm{BA+}}$ returns $z^\star$, every party $P^\star$ holding $z = z^\star$ distributes $v^\star := v_{\mathrm{IN}}$ to all the parties. To achieve this using only $O(\ell n + \mathrm{poly}(n, \kappa))$ bits, $P^\star$ sends s_i and its MT witness w_i to each party P_i. The MT witnesses allow the parties to detect and discard any corrupted codewords. In addition, RS codes are deterministic, so each party P_i obtains a unique codeword s_i from RS.ENCODE($v^\star$). Every party P_i then sends (s_i, w_i) to all parties, which allows the parties to reconstruct $v^\star$.

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

Code for party P_i with input v_{IN}

```
Let s_1, s_2, ..., s_n := RS.ENCODE(v_IN).
Let z, w_1, w_2, ..., w_n := MT.BUILD({s_1, s_2, ..., s_n}).
Join Π_BA+ with input z.
If Π_BA+ has returned ⊥, output ⊥ and terminate.
If Π_BA+ has returned z* ≠ ⊥, run the distributing step:
    If z* = z:
        For every 1 ≤ j ≤ n, send (j, s_j, w_j) to P_j.
    If you received a tuple (i, s_i, w_i):
        If MT.VERIFY(i, z*, s_i, w_i) = true:
            Send (i, s_i, w_i) to all parties.
    Let S := the set of correct tuples received.
    Output v* := RS.DECODE(S).
```

We may now sketch the proof of Theorem 1. The formal proof is included in the full version of our paper.

PROOF SKETCH OF THEOREM 1. As $\Pi_{\mathrm{BA+}}$ achieves Termination, $\Pi_{\ell\mathrm{BA+}}$ achieves Termination as well.

Then, note that $\Pi_{\ell\mathrm{BA+}}$ returns $\perp$ whenever $\Pi_{\mathrm{BA+}}$ returns $\perp$, and the parties obtain non-$\perp$ in $\Pi_{\ell\mathrm{BA+}}$ whenever $\Pi_{\mathrm{BA+}}$ returns a non-bot value. Moreover, the Intrusion Tolerance property of $\Pi_{\mathrm{BA+}}$ ensures that, whenever $\Pi_{\mathrm{BA+}}$ returns non-$\perp$, the parties agree on the encoding $z^\star$ of an honest party's input, therefore 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 $\Pi_{\mathrm{BA+}}$ only returns $\perp$ when fewer $n-2t$ parties hold the same input value, $\Pi_{\ell\mathrm{BA+}}$ also only returns $\perp$ 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. $\square$

8 Conclusions

Achieving solutions with optimal communication has been the subject of an extensive line of works [6, 22, 23, 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 $O(\ell n + \text{poly}(n, \kappa))$, 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 extending our question to input spaces beyond $\mathbb{Z}$.

It is also important to examine our protocol from a practical lens, particularly in light of the $\text{poly}(n, \kappa)$ factor in the communication complexity. Given that modern hash functions are typically 256 bits long, our protocol offers optimal communication guarantees for inputs of the order of a few kilobytes if $n = 10$ or megabytes if $n = 1000$. Whether the $\text{poly}(n, \kappa)$ overhead can be reduced remains an open question.

References

[1] Ittai Abraham, Yonatan Amit, and Danny Dolev. 2004. Optimal resilience asynchronous approximate agreement. In *Proceedings of the 8th International Conference on Principles of Distributed Systems* (Grenoble, France) *(OPODIS'04)*. Springer-Verlag, Berlin, Heidelberg, 229–239. https://doi.org/10.1007/11516798_17

[2] Ittai Abraham, T.-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. 2022. Communication complexity of byzantine agreement, revisited. *Distributed Computing* 36, 1 (July 2022), 3–28. https://doi.org/10.1007/s00446-022-00428-8

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

[4] A. Bandarupalli, A. Bhat, S. Bagchi, A. Kate, C.-D. Liu-Zhang, and M. K. Reiter. 2024. Delphi: Efficient Asynchronous Approximate Agreement for Distributed Oracles. In *Proceedings of the 2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, Brisbane, Australia, 456–469. https://doi.org/10.1109/DSN58291.2024.00051

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

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

[7] Erica Blum, Jonathan Katz, Chen-Da Liu-Zhang, and Julian Loss. 2020. Asynchronous Byzantine Agreement with Subquadratic Communication. In *TCC 2020, Part I (LNCS)*. Springer, Heidelberg, Cham, 353–380. https://doi.org/10.1007/978-3-030-64375-1_13

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

[9] Mélanie Cambus, Darya Melnyk, Tijana Milentijević, and Stefan Schmid. 2025. Approximate Agreement Algorithms for Byzantine Collaborative Learning. arXiv:2504.01504 [cs.LG] https://arxiv.org/abs/2504.01504

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

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

[12] Shir Cohen, Idit Keidar, and Alexander Spiegelman. 2020. Not a COINcidence: Sub-Quadratic Asynchronous Byzantine Agreement WHP. In *34th International Symposium on Distributed Computing (DISC 2020) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 179)*, Hagit Attiya (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 25:1–25:17. https://doi.org/10.4230/LIPIcs.DISC.2020.25

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

[14] Andrei Constantinescu, Diana Ghinea, Roger Wattenhofer, and Floris Westermann. 2024. Convex Consensus with Asynchronous Fallback. In *38th International Symposium on Distributed Computing (DISC 2024) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 319)*, Dan Alistarh (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 15:1–15:23. https://doi.org/10.4230/LIPIcs.DISC.2024.15

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

[16] Danny Dolev and Rüdiger Reischuk. 1982. Bounds on Information Exchange for Byzantine Agreement. In *1st ACM PODC*, Robert L. Probert, Michael J. Fischer, and Nicola Santoro (Eds.). ACM, New York, NY, USA, 132–140. https://doi.org/10.1145/800220.806690

[17] El-Mahdi El-Mhamdi, Sadegh Farhadkhani, Rachid Guerraoui, Arsany Guirguis, Lê-Nguyên Hoang, and Sébastien Rouault. 2021. Collaborative learning in the jungle (decentralized, byzantine, heterogeneous, asynchronous and nonconvex learning). In *Proceedings of the 35th International Conference on Neural Information Processing Systems (NIPS '21)*. Curran Associates Inc., Red Hook, NY, USA, Article 1918, 14 pages.

[18] El-Mahdi El-Mhamdi, Rachid Guerraoui, Arsany Guirguis, Lê Nguyên Hoang, and Sébastien Rouault. 2020. Genuinely Distributed Byzantine Machine Learning. In *Proceedings of the 39th Symposium on Principles of Distributed Computing* (Virtual Event, Italy) *(PODC '20)*. Association for Computing Machinery, New York, NY, USA, 355–364. https://doi.org/10.1145/3382734.3405695

[19] A. D. Fekete. 1987. Asynchronous approximate agreement. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing* (Vancouver, British Columbia, Canada) *(PODC '87)*. Association for Computing Machinery, New York, NY, USA, 64–76. https://doi.org/10.1145/41840.41846

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

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

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

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

[24] Yuval Gelles and Ilan Komargodski. 2024. Optimal Load-Balanced Scalable Distributed Agreement. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing* (Vancouver, BC, Canada) *(STOC 2024)*. Association for Computing Machinery, New York, NY, USA, 411–422. https://doi.org/10.1145/3618260.3649736

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

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

[27] Diana Ghinea, Chen-Da Liu-Zhang, and Roger Wattenhofer. 2024. Communication-Optimal Convex Agreement. Cryptology ePrint Archive, Paper 2024/251. https://eprint.iacr.org/2024/251

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

[29] Valerie King and Jared Saia. 2009. From Almost Everywhere to Everywhere: Byzantine Agreement with $\tilde{O}(n^{3/2})$ Bits. In *Distributed Computing*, Idit Keidar (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 464–478.

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

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

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

[33] Christoph Lenzen and Julian Loss. 2022. Optimal Clock Synchronization with Signatures. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing* (Salerno, Italy) *(PODC'22)*. Association for Computing Machinery, New York, NY, USA, 440–449. https://doi.org/10.1145/3519270.3538444

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

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

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

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

[38] Hammurabi Mendes, Maurice Herlihy, Nitin Vaidya, and Vijay K. Garg. 2015. Multidimensional agreement in Byzantine systems. *Distributed Computing* 28, 6 (Dec. 2015), 423–441. https://doi.org/10.1007/s00446-014-0240-5

[39] Ralph C. Merkle. 1987. A Digital Signature Based on a Conventional Encryption Function. In *A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology (CRYPTO '87)*. Springer-Verlag, Berlin, Heidelberg, 369–378.

[40] Atsuki Momose and Ling Ren. 2021. Optimal Communication Complexity of Authenticated Byzantine Agreement. In *35th International Symposium on Distributed Computing (DISC 2021) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 209)*, Seth Gilbert (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 32:1–32:16. https://doi.org/10.4230/LIPIcs.DISC.2021.32

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

[42] Lan Nguyen. 2005. Accumulators from bilinear pairings and applications. In *Proceedings of the 2005 International Conference on Topics in Cryptology* (San Francisco, CA) *(CT-RSA'05)*. Springer-Verlag, Berlin, Heidelberg, 275–292. https://doi.org/10.1007/978-3-540-30574-3_19

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

[44] Maria Potop-Butucaru, Michel Raynal, and Sebastien Tixeuil. 2011. Distributed Computing with Mobile Robots: An Introductory Survey. In *Proceedings of the 2011 14th International Conference on Network-Based Information Systems (NBIS '11)*. IEEE Computer Society, USA, 318–324. https://doi.org/10.1109/NBiS.2011.55

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

[46] Phillip Rogaway. 2006. Formalizing Human Ignorance. In *Progress in Cryptology - VIETCRYPT 2006*, Phong Q. Nguyen (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 211–228.

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

[48] Lili Su and Nitin H. Vaidya. 2016. Fault-Tolerant Multi-Agent Optimization: Optimal Iterative Distributed Algorithms. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing* (Chicago, Illinois, USA) *(PODC '16)*. Association for Computing Machinery, New York, NY, USA, 425–434. https://doi.org/10.1145/2933057.2933105

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

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