Byzantine Resilient Distributed Computing on External Data

John Augustine
Indian Institute of Technology Madras, Chennai, India

Jeffin Biju
Indian Institute of Technology Madras, Chennai, India

Shachar Meir
Weizmann Institute of Science, Rehovot, Israel

David Peleg
Weizmann Institute of Science, Rehovot, Israel

Srikkanth Ramachandran
Indian Institute of Technology Madras, Chennai, India

Aishwarya Thiruvengadam
Indian Institute of Technology Madras, Chennai, India

Abstract

We study a class of problems we call *retrieval* problems in which a distributed network has read-only access to a trusted external data source through queries, and each peer is required to output some computable function of the data. To formalize this, we propose the *Data Retrieval Model* comprising two parts: (1) a congested clique network with k peers, up to βk of which can be Byzantine in every execution (for suitable values of $\beta \in [0, 1)$); (2) a trusted source of data with no computational abilities, called the *External Data Source* (or just source for short). This source stores an array $\mathcal{X}$ of n bits ($n \gg k$), providing every peer in the congested clique read-only access to $\mathcal{X}$ through queries. It is assumed that a query to the source is significantly more expensive than a message between two peers in the network. Hence, we prioritize minimizing the number of queries a peer performs over the number of messages it sends. Retrieval problems are easily solved by having each peer query all of $\mathcal{X}$, so we focus on designing non-trivial query-efficient protocols for retrieval problems in the DR network that achieve low query performance per peer. Specifically, to initiate this study, we present deterministic and randomized upper and lower bounds for two fundamental problems. The first is the Download problem that requires every peer to output an array of n bits identical to $\mathcal{X}$. The second problem of focus, Disjunction, requires nodes to learn if some bit in $\mathcal{X}$ is set to 1.

2012 ACM Subject Classification Computing methodologies → Distributed algorithms

Keywords and phrases Byzantine Fault Tolerance, Blockchain Oracle, Congested Clique, Data Retrieval Model

Digital Object Identifier 10.4230/LIPIcs.DISC.2024.3

Related Version *Full Version*: `https://arxiv.org/abs/2309.16359`

Funding *John Augustine*: Supported by the Cybersecurity Centre, IIT Madras.
David Peleg: Venky Harinarayanan and Anand Rajaraman Visiting Chair Professor. The funds from this professorship enabled exchange visits between IIT Madras, India, and the Weizmann Institute of Science, Israel.

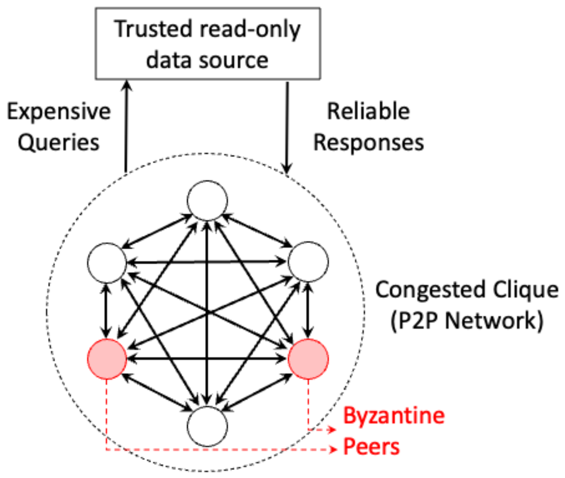
Acknowledgements We would like to thank Atharva Chougule for useful ideas.

1 Introduction

Background and Motivation

We study distributed systems in which a peer-to-peer (P2P) network retrieves data (or some boolean function of it) from a trusted source of data that is external to the network. To formalize this study, we propose a new model called the *Data Retrieval (DR) Model* comprising a congested clique network and an *External Data Source* (or source for short) with no computational capabilities. The DR model consists of a congested clique network with k peers, up to βk of which can be Byzantine in every execution (for suitable values of $\beta \in [0, 1)$). The source comprises an array $\mathcal{X}$ of n bits ($n \gg k$), providing every peer in the congested clique read-only access to $\mathcal{X}$ through queries. We prioritize minimizing the number of queries a peer performs over the number of messages it sends as we assume that a query to the source is significantly more expensive than a message between two peers in the network.

Our DR model is inspired by distributed Blockchain oracles [7, 12]. In such oracle systems, a decentralized P2P network with some Byzantine corruptions (modeled by our congested clique network) is tasked with retrieving information from trusted external data sources (e.g., stock prices, inflation indices, IoT sensors, etc.) through well defined *Application Programming Interface* (API) calls. Currently, nodes in state-of-the-art blockchain oracles do not cooperate, resulting in each node having to read all the information directly from the data source. These API calls can be expensive with cost scaling directly with their usage. The DR model provides a framework for designing Byzantine resilient mechanisms for nodes in such P2P networks to share the workload of queries, thus reducing the cost for each node.



Difference in approach from traditional BFT problems. The theory of Byzantine fault tolerance has been a fundamental part of distributed computing ever since its introduction by Pease, Shostak, and Lamport [33, 36] in the early 80's, and has had a profound influence on cryptocurrencies, blockchains, distributed ledgers, and other decentralized peer-to-peer systems. It largely focused on a canonical set of problems like Broadcast [18], Agreement [11, 33, 36, 37], k-set Agreement [14], Common Coin [34], and State Machine Replication [13]. Some studies have injected Byzantine fault tolerance into other related areas (cf. [5, 6, 9, 16, 17]). In most of these studies, the main parameter of interest is the maximum fraction β of the peers that can be corrupted by the adversary in an execution.

Consider the Byzantine Agreement problem that requires n peers, each with an input bit, to agree on a common output bit that is *valid*, in the sense that at least one honest (non-Byzantine) peer held it as input. In the synchronous setting, even without cryptographic assumptions, there are agreement algorithms that can tolerate any fraction $\beta < 1/3$ of Byzantine peers [33] (and this extends to asynchronous settings as well [11]). When $\beta \geq 1/3$, agreement becomes impossible in these settings [33]. However, the bound improves to $\beta < 1/2$ with message authentication by cryptographic digital signatures [38]. By the well-known network partitioning argument (discussed shortly), $\beta < 1/2$ is required for any form of Byzantine agreement. For most of the Byzantine fault tolerance literature, β hovers around either $1/3$ or $1/2$, with some notable exceptions like authenticated broadcast [18] that can tolerate any $\beta < 1$.

The main reason for this limitation stems from the inherent coupling of data and computing. Consider, for instance, any Byzantine Agreement variation with $\beta \geq 1/2$. When all honest peers have the same input bit (say, 1), the Byzantine peers hold at least half the input bits and can unanimously claim 0 as their input bits. This ability of Byzantine peers to spoof input bits makes it fundamentally impossible for honest peers to reach a correct agreement with the validity requirement intact. At the heart of this impossibility is the adversary's power to control information crucial to solving the problem. In fact, this issue leads to many impossibilities and inability to solve problems exactly (see e.g, [4]).

In contrast, having a reliable source that provides the data in read-only fashion yields a distributed computing context where access to data cannot be controlled by Byzantine peers. Taken to the extreme, any honest peer can individually solve all problems by directly querying the source for all required data. However, queries are charged for, and can be quite expensive. So the challenge is to design effective and secure collaborative techniques to solve the problem at hand while minimizing the number of queries made by each honest peer[1]. Hence, despite the source being passive (read-only with no computational power), its reliability makes the model stronger than the common Byzantine model.

The Model

A Data Retrieval model consists of (i) k peers that form a *congested clique* and (ii) a source of data that is external to the congested clique called the *source* that stores the input array comprising n bits and provides read-only access to its content through queries.

Congested Clique. The k peers are identified by unique ID's assumed to be from the range $[1, k]$. The peers are connected via a complete network. In each round, every peer can send at most one $O(\log n)$ bit message to each of the other peers. This communication mechanism is referred to as *peer-peer* communication.

The source. The n-bit input array $\mathcal{X} = \{x_1, \ldots, x_n\}$ (with $n \gg k$) is stored in the source. It allows peers to retrieve that data through queries of the form `Query`(i), for $1 \leq i \leq n$. The answer returned by the source would then be x_i, the i^{th} element in the array. This type of communication is referred to as *source-peer* communication.

[1] Note that appointing some individual peers to query each input bit and applying a *Byzantine Reliable Broadcast (BRB)* protocol [2, 11, 18] for disseminating the bits to all peers will not do, since the appointed peers might be Byzantine, in which case the BRB protocol can only guarantee agreement on *some* value, but not necessarily the true one. Moreover, Byzantine Reliable Broadcast (BRB) cannot be solved when $\beta \geq 1/3$ with no authenticated messages.

Synchrony and rounds. We consider a synchronous round setting where peers share a global clock, and the network delay is bounded by Δ. Each round has a total length of 3Δ and consists of two sub-rounds:

1. The *query sub-round* of length 2Δ of source-peer communication, comprising sending queries of the form `Query`$(\cdot)$ from a peer to the source and receiving the responses from the source. Every peer can send up to n queries per round to the source. (This is merely an upper limit; our protocols typically send significantly fewer queries).
2. The *message-passing sub-round* of length Δ of peer-peer communication, consisting of messages exchanged between peers. Every message is of size $O(\log n)$

We assume local computation takes 0 time and is performed at the beginning of a round. We assume that a peer M can choose to ignore (not process) messages received from another peer during the execution. Such messages incur no communication cost[2] for M.

The adversarial settings. The behavior of the environment in which our protocols operate is modeled via an adversary `Adv` that is in charge of selecting the input data and fixing the peers' failure pattern. In executing a protocol, a peer is considered *honest* if it obeys the protocol throughout the execution. A *Byzantine* peer can deviate from the protocol arbitrarily (controlled by `Adv`). The adversary `Adv` can corrupt at most βk peers for some given[3] $\beta \in [0, 1)$. This implies that `Adv` cannot corrupt all of the peers; our results are stated under this assumption. Letting $\gamma = 1 - \beta$, there is (at least) a γ fraction of honest peers. We denote the set of Byzantine (respectively, honest) peers in the execution by $\mathcal{B}$. (resp., $\mathcal{H}$).

We design both deterministic and randomized protocols. When the protocol is deterministic, the adversary can be thought of as all-knowing. Thus, `Adv` knows exactly how the complete execution will proceed and can select Byzantine nodes from the beginning based on this knowledge. When the protocol is randomized, the peers may generate random bits locally. At the beginning of each round i, `Adv` has knowledge of $\mathcal{X}$, all the local random bits generated up to round $i - 1$, and all peer-peer and source-peer communications up to round $i - 1$. At the start of round i, it can corrupt as many peers as it desires, provided the total number of peers corrupted since the beginning of the execution does not exceed βk. Such an adversary is said to be *adaptive*.

Complexity measures. The following complexity measures are used to analyze our protocols: (i) *Query* Complexity ($\mathcal{Q}$): the maximum number of queries made by an honest peer during the execution of the protocol, (ii) *Message* Complexity ($\mathcal{M}$): the total number of messages sent by honest peers during the execution of the protocol, and (iii) *Round* Complexity ($\mathcal{T}$): the number of rounds (or *time*) it takes for the protocol to terminate.

As queries to the source are expected to be the more expensive component in the foreseeable future, we primarily focus on optimizing the query complexity $\mathcal{Q}$, only trying to optimize $\mathcal{T}$ and $\mathcal{M}$ when $\mathcal{Q}$ is optimal (within $\log(n)$ factors). Our definition of $\mathcal{Q}$ (measuring the maximum cost per peer rather than the total cost) favors a fair and balanced load of queries across honest peers.

[2] Specifically, an honest peer M can ignore the messages of a known Byzantine peer M' and thus thwart any "denial of service" attack that M' attempts on M. Such messages sent by the Byzantine peer M' to M will not be counted towards the message complexity.

[3] We do not assume β to be a fixed constant (unless mentioned otherwise).

Problems Studied and Their Complexity in the Failure Free Model

We introduce the two main problems we focus on in this paper. To establish a baseline for our various results, we first outline the best possible complexity measures when there are no Byzantine failures. For $\mathcal{Q}$, the best bound is the total number of queries required divided by k, since this work of querying can be distributed evenly.

Download. We begin with the fundamental Download problem, where each of the k peers needs to obtain a copy of all n input bits from the cloud. This problem is the most fundamental retrieval problem since every computable function f of the input can be computed by the peers by first running a download protocol and then computing $f(\mathcal{X})$ locally at no additional costs. Hence, its query cost serves as a baseline against which to compare the costs of other specialized algorithms for specific problems. Observe that a $\mathcal{Q}$ lower bound for computing any Boolean function on $\mathcal{X}$ serves as a lower bound for Download as well.

To solve this problem in the absence of failures, all n bits need to be queried, and this workload can be shared evenly among k peers, giving $\mathcal{Q} = \Theta(n/k)$. The message complexity is $\mathcal{M} = \tilde{O}(nk)$ and round complexity is $\mathcal{T} = \tilde{O}(n/k)$ since $\Omega(n/k)$ bits need to be sent along each communication link when the workload is shared.

Disjunction. In the Disjunction problem, the honest peers must learn whether at least one of the input bits in $\mathcal{X}$ is a 1. We also consider an *Explicit* Disjunction version where each peer must learn an index i such that $\mathcal{X}[i] = 1$ (or output 0 if there are no 1's).

The Disjunction problem is a retrieval problem that illustrates the possibility of achieving better results than trivially using Download as a subroutine. The complexity of the problem is closely tied to the *density* δ (i.e., the fraction of ones) in the input. In fact, the relevant parameter is often $1/\boldsymbol{\delta}$ where $\boldsymbol{\delta} = \max(1/n, \delta)$ to handle the exceptional case when $\delta = 0$.

Let us consider the Explicit Disjunction problem. In the deterministic setting, at least $n - \boldsymbol{\delta} n + 1$ queries are required in total. Consequently, the best deterministic query complexity is $\mathcal{Q} = O(n(1-\boldsymbol{\delta})/k)$. The round complexity is $\mathcal{T} = O(1)$, and message complexity is $\mathcal{M} = O(k)$. Peers that find a 1-bit can send the index to a "leader" peer to broadcast the answer.

Randomization helps when δ is large. Querying $\left(\boldsymbol{\delta}^{-1} \cdot \ln n\right)$ bits uniformly at random in search for a 1 bit has failure probability of $(1-\boldsymbol{\delta})^{\ln n/\boldsymbol{\delta}} \leq 1/n$. Thus $O\left(\boldsymbol{\delta}^{-1} \cdot \ln n\right)$ queries are sufficient to find a 1 w.h.p. Even without knowledge of δ, one can simply try density values in decreasing powers of 2, starting with 1/2 and eventually land at a 1 having made at most $O\left(\boldsymbol{\delta}^{-1} \cdot \ln n\right)$ queries. We can distribute the work equally amongst k peers, and thus $\mathcal{Q} = O(1 + \boldsymbol{\delta}^{-1} \cdot \frac{1}{k} \cdot \ln n)$. The time and message analysis is similar to the deterministic case, i.e, $\mathcal{T} = O(1)$, $\mathcal{M} = O(k)$. Note that $\mathcal{Q} = \Omega(\frac{1}{k} \cdot \boldsymbol{\delta}^{-1})$, for any algorithm that solves the Disjunction problem with constant probability.

Our Contributions

We initiate the study of the Data Retrieval Model and retrieval problems. We present several deterministic and randomized protocols and some lower bounds for Download and Disjunction. Here, we state only simplified bounds, in which the $\tilde{O}(\cdot)$ notation hides factors dependent on β and poly log factors in n. The main results are summarized in Table 1 for convenience.

Download. For the deterministic model, the Download problem turns out to be expensive, requiring $\Omega(\beta n)$ queries in the worst case. Every peer essentially has to query the entire input array for itself. In the randomized model, we give an algorithm that solves the Download problem (and consequently *any* function of the input) for an *arbitrary* fraction $\beta < 1$ of Byzantine faults while requiring at most $\tilde{O}(n/k + \sqrt{n})$ queries per peer. The result is nearly as efficient as the *failure-free model* whenever $k < \sqrt{n}$. The time and message costs are $\mathcal{T} = O(n)$ and $\mathcal{M} = \tilde{O}(kn + k^2\sqrt{n})$. A natural question then, is whether the additive $\sqrt{n}$ term is necessary for $k > \sqrt{n}$. While we are not able to fully address this question, we show that for restricted β ($< 1/3$), we can be fully efficient for all $k \in [1, n]$, getting $\mathcal{Q} = \tilde{O}\left(\frac{n}{k}\right)$, $\mathcal{T} = \tilde{O}(n)$, and $\mathcal{M} = \tilde{O}(nk^2)$.

Disjunction. To show that for specific problems one can be more efficient, we consider Disjunction when the input bits have density δ. Naturally, the problem becomes easier as δ gets larger. We first show that any deterministic algorithm requires $\Omega(n/k + \boldsymbol{\delta}^{-1})$ queries in the worst case. Next, we show that for any $\beta < 1$, there exists a deterministic algorithm that makes $\tilde{O}(n/k + \boldsymbol{\delta}^{-1} + k)$ queries. This algorithm is nearly optimal whenever $k < \sqrt{n}$. Our second deterministic algorithm achieves near optimal complexity provided $\beta < 1/2$. Both algorithms require $\mathcal{T} = \tilde{O}(1)$ and $\mathcal{M} = \tilde{O}(k^2)$.

We then consider the randomized model. It is easy to see that any algorithm requires $\Omega(1/k \cdot \boldsymbol{\delta}^{-1})$ queries per peer. We show that this is nearly tight by presenting an algorithm that w.h.p. solves the Disjunction problem with $\mathcal{Q} = \tilde{O}\left(\frac{1}{k} \cdot \boldsymbol{\delta}^{-1}\right)$, $\mathcal{T} = \tilde{O}(1)$, $\mathcal{M} = \tilde{O}(k^2)$.

Table 1 Our Main Results (with β treated as constant).

Problem & Model	Query	Lower Bound	Round	Message	Theorem
Download					
Randomized $\beta < 1$	$\tilde{O}(n/k + \sqrt{n})$	$\Omega(n/k)$	$O(n)$	$\tilde{O}(nk + k^2\sqrt{n})$	Thm 4
Randomized $\beta < 1/3$	$\tilde{O}(n/k)$	$\Omega(n/k)$	$O(n)$	$\tilde{O}(nk^2)$	Thm 12
Disjunction					
Deterministic $\beta < 1$	$\tilde{O}(n/k + \boldsymbol{\delta}^{-1} + k)$	$\Omega\left(\boldsymbol{\delta}^{-1} + \frac{(1-\delta)n}{\gamma k}\right)$	$\tilde{O}(1)$	$\tilde{O}(k^2)$	Thm 17
Deterministic $\beta < 1/2$	$\tilde{O}(n/k + \boldsymbol{\delta}^{-1})$	$\Omega\left(\boldsymbol{\delta}^{-1} + \frac{(1-\delta)n}{\gamma k}\right)$	$\tilde{O}(1)$	$\tilde{O}(k^2)$	Thm 18
Randomized $\beta < 1$	$\tilde{O}(1/k \cdot \boldsymbol{\delta}^{-1})$	$\Omega(\frac{1}{\gamma k} \cdot \boldsymbol{\delta}^{-1})$	$\tilde{O}(1)$	$\tilde{O}(k^2)$	Thm 19

2 Methods

Private ρ-Representative Committees. Several of our protocols organize the peers in *committees*, assigned to perform a common task. In a *private ρ-representative committee*, every peer knows only whether it belongs to the committee and the committee is guaranteed to have at least ρ honest members, where ρ is known.

We present a probabilistic construction for a ρ-representative committee, where the guarantee of at least ρ honest members holds w.h.p. To construct such a committee, each peer adds itself to the committee with probability p. See Algorithm 1. By choosing an appropriate value of p, we can obtain high probability guarantees on the number of (honest) peers in a committee using standard Chernoff tail bounds. This yields the following result

▶ **Lemma 1.** *Consider k i.i.d Bernoulli random variables with bias $p = \min(1, \frac{9 \ln n + 4\rho}{\gamma k})$, $\beta \in [0, 1)$, $n > 1$ and $\rho \leq \gamma k$, we have with probability at least $1 - 2n^{-3}$,*

- *for any subset of γk variables, at least ρ of them are* 1.
- *At most $(18 \ln n + 8\rho)/\gamma$ variables are* 1.

Lemma 1 implies that w.h.p a committee $\mathcal{C}$ constructed by Algorithm 1 is indeed a private ρ-representative committee and it will have at most $(18 \ln n + 8\rho)/\gamma$ honest members.

Algorithm 1 Procedure `Elect_Private`.

```
1: Every peer tosses a biased coin with a probability of heads $p = \min\left\{\frac{6\ln n + 4\rho}{\gamma k}, 1\right\}$
2: return $\mathcal{C}$ = set of peers that tossed heads.
```

Commit Verification. Before a peer M commits b as x_i, it verifies that $x_i = b$ by one of several ways:

- *Direct-verification:* M directly queries the source and receives a reply that $x_i = b$.
- *Comm-verification:* M collects votes from a private ρ-representative committee $\mathcal{C}_i$. M learns that $x_i = b$ if it receives a message saying that $x_i = b$ from at least ρ members of $\mathcal{C}_i$, and a message saying that $x_i = 1 - b$ from fewer than ρ members of $\mathcal{C}_i$.
- *Gossip-verification:* M receives messages from $\beta k + 1$ or more peers, each testifying that it verified $x_i = b$. This suffices since necessarily at least one of these senders must have been an honest peer.

Blacklisting. During an execution, honest peers can *blacklist* Byzantine ones, after identifying a deviation from the behavior expected of an honest peer, and subsequently ignore their messages. A Byzantine peer M' can be blacklisted for several reasons. The most common reason to blacklist is when M' is directly "caught" in a lie about the value of some bit. The two other reasons for blacklisting are as follows.

- **Blacklisting for requesting unnecessary work:** Some of our protocols maintain a known-to-all list of bits. If M' claims that a certain bit x_i is unknown to it and requests to learn it, M can check if x_i is listed at M as *known to all.* If so, M knows that M' must be Byzantine.
- **Blacklisting for over-activity:** Lemma 1 implies that the number of honest peers in our construction of a private ρ-representative committee is bounded from above w.h.p. M' can be blacklisted as Byzantine for being *over-active*, namely, claiming to have been randomly selected to many more committees than expected.

3 Results on the Download Problem

3.1 Deterministic Setting

We first note that Download can be solved trivially by having each peer query all n bits directly from the source. This protocol incurs $\mathcal{Q} = n$, $\mathcal{T} = 1$ and $\mathcal{M} = 0$ and works for $\beta < 1$. However, we can improve the query complexity for $\beta < 1/2$. (Some proofs are deferred to Appendix A.)

▶ **Theorem 2.** *When $\beta < 1/2$, there is a deterministic protocol for Download with $\mathcal{Q} = O(\beta n)$, $\mathcal{T} = \tilde{O}(\beta n)$ and $\mathcal{M} = \tilde{O}(\beta n k^2)$*

The following theorem establishes that one cannot hope to improve the query complexity.

▶ **Theorem 3.** *Any deterministic protocol for the Download problem has $\mathcal{Q} = \Omega(\beta n)$.*

3.2 Randomized setting

Near Query-Optimal Randomized Protocol for $\beta < 1$

We start with a simple randomized algorithm that works for any $\beta < 1$. The problem posed by the randomized model is that the adversary can fail peers online in the randomized setting based on the protocol's progress. This implies that if the protocol appoints some random peer M to query a bit x_i on some round t of the execution but communicate the bit to other peers at a *later* round t', then we cannot rely on the hope that the randomly selected M will be honest, say, with probability $1 - \beta = \varepsilon$, since the adversary gets an opportunity to learn the identity of the chosen M on round t and subsequently corrupt it before round t'. Hence in order for us to benefit from the fact that some peer M is randomly chosen for some sub-task on round t, it is imperative that M completes that sub-task *on the same round.*

The idea used to overcome this difficulty is as follows. Sequentially, for n rounds, do the following. At round i we query bit x_i from the source to a private ρ-representative committee (see Algorithm 1) $\mathcal{C}_i$, i.e., x_i is queried by each (honest) peer in $\mathcal{C}_i$. Then (still on the same round), each peer in $\mathcal{C}_i$ sends the value of x_i to every other peer. Peers not in $\mathcal{C}_i$ might receive incorrect values from the Byzantine peers in $\mathcal{C}_i$. However, if strictly fewer than ρ incorrect values are received, each peer can be confident of the majority as the right answer (w.h.p). In case at least ρ peers sent an incorrect value, or more precisely, in case an honest peer receives at least ρ zeros and at least ρ ones, then peers resort to querying the source for the answer, forcing at least ρ Byzantine peers to reveal themselves as being Byzantine. Choosing ρ optimally results in a query complexity of $O(\frac{n \log n}{\gamma k} + \sqrt{n})$. See Algorithm 2 for the pseudocode.

Algorithm 2 Algorithm `Blacklist_Download` model, Code for peer M.

```
    Output: Array $res$ such that $res[i] = x_i$ for $i = 1, 2, ...n$
 1: $\mathcal{B} \leftarrow \emptyset$                                          ▷ Peers known to be faulty
 2: for $i = 1, 2, \ldots n$ (in separate rounds) do
 3:     Form a private $\rho$-representative committee $\mathcal{C}_i$.       ▷ Parameter $\rho$ is fixed later
 4:     if $M \in \mathcal{C}_i$ then
 5:         $res[i] \leftarrow$ Query($i$), send (vote, $res[i]$) to all peers.
 6:     $S_j \leftarrow$ set of peers not in $\mathcal{B}$ that voted $j$ for $j \in \{0, 1\}$
 7:     if $\min(|S_0|, |S_1|) > \rho$ then
 8:         $res[i] \leftarrow$ Query($i$).
 9:         $\mathcal{B} \leftarrow \mathcal{B} \cup S_{1-res[i]}$
10:     else  $res[i] \leftarrow \arg\max_{j=0,1} |S_j|$.
11: return $res$
```

▶ **Theorem 4.** *When $\beta < 1$, Protocol* `Blacklist_Download` *solves the Download problem w.h.p. with $\mathcal{Q} = O\left(\frac{n \log n}{\gamma k} + \sqrt{n}\right)$, $\mathcal{T} = O(n)$ and $\mathcal{M} = O(kn \log n + k^2\sqrt{\gamma n})$.*

Proof. The correctness follows from the observation that for each bit x_i, each honest peer either (i) heard fewer than ρ votes for one value in $\{0, 1\}$ or (ii) queried x_i. Since $\mathcal{C}_i$ is ρ-representative (w.h.p), the correct bit value would have been reported by at least ρ peers, and we can conclude that an honest peer can verify the correct value of x_i in both cases. Next, We analyze the query complexity of Algorithm 2 and choose ρ optimally.

Queries are made in lines 5 and 8 of Algorithm 2. For peer M, the expected number of queries in line 5 is np where p is the probability of joining a committee. As $n > \gamma k$, we have $np \geq 9 \ln n$. By Chernoff bounds, w.h.p there are no more than $2np$ queries, similar to the proof of Lemma 1. Every time a peer reaches Line 8 and queries the source, the size of its local set $\mathcal{B}$ increases by ρ. Therefore, these queries are performed at most $\frac{\beta k}{\rho}$ times. Therefore, the total number of queries for peer M (w.h.p.) is at most $\mathcal{Q} = \frac{18 n \ln n}{\gamma k} + \frac{8 n \rho}{\gamma k} + \frac{\beta k}{\rho}$, and choosing $\rho = \max\left\{1, k\sqrt{\frac{\gamma\beta}{8n}}\right\}$, we get $\mathcal{Q} = O\left(\frac{n \log n}{\gamma k} + \sqrt{\frac{\beta}{\gamma} \cdot n}\right)$.

By the description of the protocol. the time complexity $\mathcal{T}$ is clearly $O(n)$, the number of iterations. The message complexity $\mathcal{M}$ is calculated as the product of the number of honest peers that join each $\mathcal{C}_i$ (which is $O(\log n + k/\sqrt{n})$) times $O(k)$ (the number of messages sent by each honest peer in $\mathcal{C}_i$) time n (the number of iterations). ◀

Observe that there is a trivial $\Omega(\frac{n}{\gamma k})$ lower bound on $\mathcal{Q}$ (in the case where Byzantine peers crash and do not participate in the execution).

The additional $\sqrt{n}$ term can be neglected whenever $k < \sqrt{n}$ since it is smaller than the lower bound in these cases. Thus, in a wide range of cases, the above protocol is "near-optimal". It is also tolerant against the strongest form of Byzantine adversary, one that even has knowledge of random bits sampled up until the previous round.

Query-Optimal Randomized Protocol for $\beta < 1/3$

The Download protocol of the previous section works when $\beta < 1$ but falls short of yielding optimal query complexity. This section presents a query-optimal protocol for Download when $\beta < 1/3$. For a complete analysis see Appendix A.

The Protocol

Let us first give an overview of the approach. The protocol proceeds in $J_0 = \left\lceil \log_{1/\alpha} \frac{k}{c \log n} \right\rceil$ phases, whose goal is to reduce the number of unknown bits by a *shrinkage factor* $\alpha < 1$. The protocol maintains a number of set variables, updated in each phase, including the following. $\mathcal{K}_M$ (respectively, $\mathcal{U}_M$) is the set of indices i whose value $res[i]$ is already *known* (resp., still *unknown*) to M. At any time during the execution, $\mathcal{K}_M \cup \mathcal{U}_M = \{1, \dots, n\}$. $res[i] = x_i$ is the Boolean value of x_i for every $i \in \mathcal{K}_M$. (Slightly abusing notation for convenience, we sometimes treat $\mathcal{K}_M$ as a set of *pairs* $(i, res[i])$, i.e., we write $\mathcal{K}_M$ where we actually mean $\mathcal{K}_M \circ res_M$.) Each peer also identifies a set KTA_M of *known-to-all* bits and $\mathcal{I}_M$ of unknown indices for at least one peer. Each phase contains four subroutines, each with a specific goal in mind. First, the `Committee_Work` subroutine forms private committees where each peer M joins committee i if $i \in \mathcal{I}_M$ with some probability. Each member of committee i then reports x_i, and each peer decides whether to accept some or no value (updating $\mathcal{K}_M$ and $\mathcal{U}_M$ accordingly). There is also a blacklisting component in which if a peer belongs to too many committees, it is deemed Byzantine and ignored for the rest of the execution. Second, the `Gossip` subroutine has every peer M report its $\mathcal{K}_M$ to all other peers. If a peer receives at least $\beta k + 1$ reports of the same value for x_i, it accepts it. Third, the second invocation of `Gossip` repeats the reporting of $\mathcal{K}_M$ for every peer M, but this time, in addition to the update of $\mathcal{K}_M$, if a value is reported $2\beta k + 1$ times, it adds it to KTA_M. The motivation behind this second invocation is that if a value is reported $2\beta k + 1$ times, then at least $\beta k + 1$ of those reports are from non-faulty peers. Thus, all peers will accept that value (and add it to $\mathcal{K}_M$). Last, the `Collect_Requests` subroutine is meant to update $\mathcal{I}_M$, i.e., to know which indices are unknown to at least one peer. This subroutine also has a blacklisting component in which if a peer sends a request for index i but $i \in \mathrm{KTA}_M$, M blacklists the requesting peer.

▶ Remark 5. The communication performed in the various steps of the protocol takes more than one time unit in the CONGEST model. Hence, the protocol must also ensure that the different steps are synchronized and that all peers start each step only after the previous step is completed. Relying solely on reports from each peer concerning its progress might lead to deadlocks caused by the Byzantine peers. Hence, the scheduling must be based on the fact that the duration of each step is upper-bounded by the maximum amount of communication the step involves. We omit this aspect from the description of the algorithm.

We next detail the code of the main algorithm and its procedures. (Hereafter, we omit the superscript J when clear from the context.) We denote by *update(i, b)* the function that sets $res[i] = b$, removes i from $\mathcal{U}_m$ and adds it to $\mathcal{K}_M$. We denote by $BlacklistOverWork(w_{max}, M)$ the function that checks the number x of committees M reported to belong to and adds M to $\mathcal{B}$ if $x > w_{max}$. We refer to the first and second invocations of Procedure `Gossip` as `Gossip`(1) and `Gossip`(2) respectively.

Algorithm 3 Algorithm `Gossip_Download`, $\beta < 1/3$, code for peer M.

```
1: $\mathcal{K}_M \leftarrow \emptyset$                                   ▷ Indices of bits known to $M$
2: $\mathrm{KTA}_M \leftarrow \emptyset$                                 ▷ Indices of bits that are known-to-all
3: $\mathcal{U}_M \leftarrow \{1, \ldots, n\}$                           ▷ Indices of bits not known to $M$
4: $\mathcal{I}_M \leftarrow \mathcal{U}_M$                              ▷ Indices of bits not known to some non-blacklisted peers
5: $res \leftarrow \emptyset$                                            ▷ Values of bits known to $M$
6: $\mathcal{B} \leftarrow \emptyset$                                    ▷ Peers blacklisted by $M$ as Byzantine
7: $c \leftarrow Z/\gamma$                                               ▷ The parameter $Z$ will be fixed later.
8: $\alpha \leftarrow \frac{(1+\epsilon)\beta}{(1-\epsilon)(1-2\beta)}$  ▷ shrinkage factor, $\alpha < 1$. The parameter $\epsilon$ will be fixed later.
9: $J_0 \leftarrow \lceil \log_{1/\alpha} \frac{k}{c \log n} \rceil$     ▷ Number of phases
10: for $J = 0, 1, 2, \ldots, J_0 - 1$ (sequentially) do
11:     Invoke Committee_Work
12:     Invoke Gossip(1)
13:     Invoke Gossip(2)
14:     Invoke Collect_Requests.
15: for every $i \in \mathcal{U}_M$ do $res[i] \leftarrow$ Query($i$)    ▷ Querying the remaining unknown bits
16: return $res$
```

Partial Analysis

Sanity checks. Let us start with the two sanity checks needed to ensure the validity of the random selection step and the convergence of the protocol.

▶ **Observation 6.** *For β and ϵ satisfying*

$$\beta < \frac{1-\epsilon}{3-\epsilon} \tag{1}$$

(a) *the chosen shrinkage factor satisfies $\alpha < 1$, and*
(b) *the chosen probability satisfies $p < 1$ for every $0 \leq J \leq J_0 - 1$.*

Progress tracking variables. Next, we define the notation for the values of the main variables of the protocol during the different phases.

- Denote by $\mathcal{K}_M^J$ (respectively, $\mathcal{U}_M^J$) the value of the set $\mathcal{K}_M$ (resp., $\mathcal{U}_M$) at the beginning of phase J. (Note that it is also the value of $\mathcal{K}_M$ at the end of phase $J-1$)
- Denote by $\mathcal{K}_M^{J,mid}$ (resp., $\mathcal{U}_M^{J,mid}$) the value of the set $\mathcal{K}_M$ (resp., $\mathcal{U}_M$) at the end of the `Gossip`(1) step of phase J.
- Denote by $\mathcal{I}_M^J$ the value of the set $\mathcal{I}_M$ at the beginning of phase J.
- Denote by KTA_M^J the value of the set KTA_M at the end of the `Gossip`(2) step of phase J.

Algorithm 4 Sub routines, code for peer M.

```
 1: procedure Committee_Work
 2:     $\hat{\mathcal{I}}_M \leftarrow \emptyset$                                  ▷ Set of indices whose committees $M$ joins
 3:     $\rho \leftarrow (1-\epsilon) Z \log n / \alpha^J$                          ▷ Also $\rho = (1-\epsilon)p\gamma k$
 4:     $\mathsf{W}_{max} \leftarrow (1+\epsilon)c \log n \cdot \frac{n}{k}$        ▷ Blacklisting "over-active" Byzantine peers
 5:     for every $i = 1, \ldots, n$ sequentially do                                ▷ Setting up committees
 6:         if $i \in \mathcal{I}_M$ then
 7:             Join the private committee $\mathcal{C}_i$ at random with probability $p = \dfrac{c \log n}{\alpha^J k}$.
 8:             if $M$ was selected to $\mathcal{C}_i$ then
 9:                 $\hat{\mathcal{I}}_M \leftarrow \hat{\mathcal{I}}_M \cup \{i\}$.
10:                 if $i \in \mathcal{U}_M$ then
11:                     $update(i, \texttt{Query}(i))$                              ▷ Direct-verification
12:                 Send the message (vote ,$i$,$res[i]$) to every other peer.
13:             Collect votes sent by members of $\mathcal{C}_i$.                   ▷ Ignore messages on bits $i \notin \mathcal{I}_M$.
14:     for every other peer $M'$ do
15:         $BlacklistOverWork(\mathsf{W}_{max}, M')$
16:     for every $i \in \mathcal{I}_M$ do
17:         Let $\mathcal{C}_i^M$ be the remaining reduced committee.               ▷ Possibly $\mathcal{C}_i^M \neq \mathcal{C}_i^{M'}$ for $M \neq M'$.
18:     for every $i \in \mathcal{U}_M$ do                                          ▷ comm-verification
19:         for $b \in \{0, 1\}$ do
20:             $\psi_b(i) \leftarrow$ number of votes from $\mathcal{C}_i^M$ members for $x_i = b$.
21:         if $\psi_0(i) \geq \rho$ and $\psi_1(i) < \rho$ then
22:             $update(i, 0)$
23:         if $\psi_1(i) \geq \rho$ and $\psi_0(i) < \rho$ then
24:             $update(i, 1)$                  ▷ If both $\psi_0(i) \geq \rho$ and $\psi_1(i) \geq \rho$, then$i$ remains unknown
25: ──────────────────────────────────────────────
26: procedure Gossip(GossipNum)
27:     for every $i \in \mathcal{K}_M$ do
28:         send the message $(i, res[i])$ to all other peers.
29:     Receive a list $\mathcal{K}_{M'}$ from every other peer $M'$.
30:     for every $i \in \mathcal{U}_M$ do
31:         $\varphi_0(i) \leftarrow |\{M' \mid (i, 0) \in \mathcal{K}_{M'}\}|$.
32:         if $\varphi_0(i) \geq \beta k + 1$ then
33:             $update(i, 0)$
34:         $\varphi_1(i) \leftarrow |\{M' \mid (i, 1) \in \mathcal{K}_{M'}\}|$.
35:         if $\varphi_1(i) \geq \beta k + 1$ then
36:             $update(i, 1)$
37:         if GossipNum=2 and $(\varphi_0(i) \geq 2\beta k + 1$ or $\varphi_1(i) \geq 2\beta k + 1)$ then
38:             $\mathrm{KTA}_M \leftarrow \mathrm{KTA}_M \cup \{i\}$
39: ──────────────────────────────────────────────
40: procedure Collect_Requests
41:     Set $\mathcal{I}_M \leftarrow \mathcal{U}_M$
42:     Send $\mathcal{U}_M$ to all other peers.
43:     Collect lists $\mathcal{U}_{M'}$ from all other peers $M'$.
44:     for every $i = 1, \ldots, n$ do
45:         $R_U(i) \leftarrow \{M' \mid i \in \mathcal{U}_{M'}\}$.
46:         if $i \in \mathrm{KTA}_M$ then $\mathcal{B} \leftarrow \mathcal{B} \cup R_U(i)$     ▷ Blacklisting for requesting known-to-all bits
47:     $\mathcal{I}_M \leftarrow \mathcal{I}_M \cup \bigcup_{M' \notin \mathcal{B}} \mathcal{U}_{M'}$     ▷ Indices to be learned, including $\mathcal{U}_M$ of $M$ itself
```

Note that a bit x_i can be unknown for M and known for M' for two honest peers M and M'. We say that x_i is *unknown* in phase J, and the committee $\mathcal{C}_i$ is *necessary*, if $i \in \mathcal{U}^J_M$ for *some* honest peer M, or equivalently, if $i \in \mathcal{U}^J$, where

$$\mathcal{U}^J = \bigcup_{M \in \mathcal{H}} \mathcal{U}^J_M$$

is the set of indices i for which some honest peers request setting up a committee $\mathcal{C}_i$ and querying in the current phase. A bit x_i is *known* once $i \in \mathcal{K}_M$ for every honest peer M. Also let

$$\mathcal{U}^{J,mid} = \bigcup_{M \in \mathcal{H}} \mathcal{U}^{J,mid}_M \qquad \text{and} \qquad \mathrm{NKTA}^J_M = \{1, \ldots, n\} \setminus \mathrm{KTA}^J_M.$$

Bad events. In an execution ξ of the protocol, there are two types of *bad events*, whose occurrence might fail the protocol. Our analysis is based on bounding the probability of bad events, showing that with high probability, no bad events will occur in the execution, and then proving that in a *clean* execution, where none of the bad events occurred, the protocol succeeds with certainty. The bad events are as follows.

Bad event $\mathcal{EV}_1(J,i)$: In phase J, the committee $\mathcal{C}_i$ selected for an unknown bit x_i is not ρ-representative, for $\rho = \dfrac{(1-\epsilon)Z \log n}{\alpha^J}$, where Z is a parameter of the algorithm that must satisfy some constraints described in Lemmas 7 and 8. (If x_i is already known, then this bad event does not affect the correctness or query complexity of the honest peers, although it might increase the time and message complexity.)

Bad event $\mathcal{EV}_2(J,M)$: In phase J, an honest peer M has $|\hat{\mathcal{I}}^J_M| > \mathsf{W}_{max}$, namely, M joins more than $\mathsf{W}_{max} = \dfrac{(1+\epsilon)c \cdot n \log n}{k}$ committees, and subsequently gets blacklisted.

For an integer $J \geq 0$, call the execution ξ *J-clean* if none of the bad events $\mathcal{EV}_1(j,i)$ or $\mathcal{EV}_2(j,M)$ occurred in it for $0 \leq j \leq J$.

High probability of clean executions. We now argue that with the right choice of parameters ϵ and Z, the probability for the occurrence of any of the bad events is low.

▶ **Lemma 7.** *For any $J \geq 0$, if the execution ξ is $(J-1)$-clean, and the parameters ϵ and Z satisfy*

$$\epsilon^2 Z/2 \geq 2 + \lambda \tag{2}$$

for some constant $\lambda > 0$, then the probability that any of the bad events $\mathcal{EV}_1(J,i)$ occurred in ξ is at most $O(\frac{1}{n^{1+\lambda}})$.

▶ **Lemma 8.** *For any $J \geq 0$, if the execution ξ is $(J-1)$-clean, and the parameters ϵ and Z satisfy*

$$\frac{\epsilon^2}{2+\epsilon} \cdot Z \geq 2 + \lambda \tag{3}$$

for some constant $\lambda > 0$, then the probability that any of the bad events $\mathcal{EV}_2(J,M)$ occurred in ξ is at most $O(\frac{1}{n^{1+\lambda}})$.

The above two lemmas yield the following:

▶ **Corollary 9.** *Consider an execution ξ. If the parameters ϵ and Z satisfy*

$$Z \cdot \min\{\epsilon^2/2\ ,\ \epsilon^2/(2+\epsilon)\} \ \geq\ 2 + \lambda \tag{4}$$

for some constant $\lambda > 0$, then the probability that ξ is clean is at least $1 - O(\frac{\log n}{n^{1+\lambda}})$.

Convergence invariants.

▶ **Lemma 10.** *In a J-clean execution, assuming $\beta < 1/3$, for every honest M,*

$$\mathcal{I}_M^{J+1} \subseteq NKTA_M^J \subseteq \mathcal{U}^{J,mid} \subseteq \mathcal{U}^J \subseteq \mathcal{I}_M^J.$$

We remark that if x_i is known, hence $\mathcal{C}_i$ is not necessary, then the inviting peer is Byzantine, so it may invite only a few honest peers (or none) hence the constructed $\mathcal{C}_i$ is not guaranteed to be ρ-representative, but this will not hurt any honest peer, since, in this case, the honest peers already know x_i and will not listen to the committee.

▶ **Lemma 11.** *In a J-clean execution, for every $J \geq 0$ and every honest M,*
(1) $|\mathcal{U}^{J,mid}| \leq \alpha^{J+1}n$, (2) $|\mathcal{I}_M^J| \leq \alpha^J n$, (3) $|\mathcal{U}^J| \leq \alpha^J n$, (4) $|\mathcal{U}_M^J| \leq \alpha^J n$.

Using these convergence invariants, we get the following theorem.

▶ **Theorem 12.** *When $\beta < 1/3$, Protocol* `Gossip_Download` *solves the Download problem w.h.p. with*[4] $\mathcal{Q} = O\left(\frac{n \log^2 n}{\gamma k}\right)$, $\mathcal{T} = O\left(n \log_{\frac{1}{\beta}}\left(\frac{\gamma k}{\log n}\right)\right)$ *and* $\mathcal{M} = O\left(nk^2 \log_{\frac{1}{\beta}}\left(\frac{\gamma k}{\log n}\right)\right)$.

4 Results on the Disjunction Problem

In this section, we consider the problem of computing the Disjunction of the input bits. We first state some basic lower bounds and then present some upper bound results, along with an overview of the building blocks used to design protocols that match the upper bounds. For a complete formal presentation see the full version of the paper.

▶ **Theorem 13.** *When $\beta < 1$, any deterministic protocol for the Disjunction(δ) and the Explicit Disjunction(δ) problems has $\mathcal{Q} = \Omega\left(\beta \cdot \boldsymbol{\delta}^{-1} + \frac{(1-\delta)n}{\gamma k}\right)$.*

▶ **Theorem 14.** *Any randomized protocol for Disjunction(δ) that succeeds with constant probability has $\mathcal{Q} = \Omega(\frac{1}{\gamma k} \cdot \boldsymbol{\delta}^{-1})$ in expectation.*

The remainder of this section deals with efficient deterministic protocols for Disjunction and Explicit Disjunction under different settings. A key observation that we rely on is that single round algorithms exhibit similar properties to bipartite expanders. The connection is as follows. One can represent the access pattern of the peers to the input array $\mathcal{X}$ as a bipartite graph $G(L, R, E)$, where L represents the n input bits, R represents the k peers, and an edge $(i, j) \in E$ indicates that M_j queries $\mathcal{X}[i]$. We would like to ensure that if the number of bits set to 1 in $\mathcal{X}$ exceeds some value s, then no matter which set S of indices corresponds to these s 1s, the set $\Gamma(S)$ of neighbors of S in G will contain at least $\beta k + 1$ peers, guaranteeing that *at least one honest peer will query at least one of the set bits of S*. This can be ensured by taking G to be a *Large Set Expander (LSE)*, an expander variant defined formally later on. Not knowing the density δ in advance, we can search for it, starting with the hypothesis that δ is close to 1 (and hence using a sparse LSE and spending a small number of queries), and gradually trying denser LSE's (and spending more queries), until we reach the correct density level allowing some honest peer to discover and expose a set bit. Once the set bit is exposed, we have all the honest peers send the new bit to every other

[4] We remark that our focus was on optimizing query complexity. The $\mathcal{T}$ and $\mathcal{M}$ complexities can be improved further. For example, the current protocol requires the peers to send the entire set of known bits in each iteration, but clearly, it suffices to send the updates.

peer. The peers then query all the bits they received (one per peer) to confirm the answer. The total query complexity per peer is $\tilde{O}(n/k + \boldsymbol{\delta}^{-1} + k)$. Observe that it is near-optimal when $k < \sqrt{n}$. See Theorem 17. For $\beta < 1/2$, we obtain near-optimal query complexity $\tilde{O}(n/k + \boldsymbol{\delta}^{-1})$. The observation leading to this is that one can use expanders as before, and assign vertices to input bits such that for every possible input and every possible set of corrupt peers, strictly more than $k/2$ honest peers query a set bit. Subsequently, whenever more than $1/2$ of the peers found a 1, the remaining honest peers can conclude that the answer is 1, and because of the stronger guarantee, we no longer have to verify all the bits sent by the agents. This algorithm, however, only obtains the Disjunction of the input bits, not the actual index of a set bit. See Theorem 18.

Our definition of LSE ensures that for every possible input configuration of Disjunction with input density δ and every possible set of peers that can be corrupted by Byzantine agents, at least one honest peer reads a set bit. To the best of our knowledge, this exact definition of LSE has not been used in the literature. The definition of samplers [29] to construct asynchronous Byzantine agreement and leader election protocols is the closest to LSE. Roughly speaking, samplers ensure there are at most δ fraction of the input bits x such that their neighborhood has β fraction of Byzantine nodes, for every possible choice of corruptions that the adversary can make. Even though our definitions are different, we use similar techniques (the probabilistic method) to show their existence.

▶ **Definition 15** (Large Set Expander (LSE)). A bipartite graph $G(L, R)$ is an (n, k, β, δ)-*Large Set Expander* (or (n, k, β, δ)-LSE) if $n = |L|, k = |R|$ and $|\Gamma(S)| > \beta k$ for all $S \subseteq L$ with $|S| \geq n\delta$.

Informally, a large set expander is such that for every large enough subset S, i.e., $S \subseteq L$ and $|S| \geq \delta n$, its neighborhood cannot be covered fully by any subset of βk vertices, i.e., $|\Gamma(S)| > \beta k$. The definition of an LSE is similar to that of expander graphs and we use a similar probabilistic analysis to prove their existence. We formalize this in the lemma below.

▶ **Lemma 16.** *There exists a bipartite graph $G(L, R)$ that is a (n, k, β, δ) Large Set Expander such that, (1) Every vertex in L has degree at most d, and (2) Every vertex in R has degree at most $\frac{2nd}{k}$, for all d satisfying* $d > \max\left\{\frac{1 + \log(e \cdot \boldsymbol{\delta}^{-1})}{\log \frac{1}{\beta}} + \frac{\beta k}{\delta n} \cdot \frac{\log \frac{e}{\beta}}{\log \frac{1}{\beta}}, \frac{3k \ln 2k}{n}\right\}$.

We use the existence of large set expanders to design algorithms that achieves the results stated in Theorems 17 and 18.

▶ **Theorem 17.** *When $\beta < 1$, There exists a protocol that solves Disjunction with $\mathcal{Q} = O\left(\frac{n}{k} \cdot \left(\log_{\frac{1}{\beta}}(e^2\boldsymbol{\delta}^{-1}) + \log k\right) \cdot \log \boldsymbol{\delta}^{-1} + \boldsymbol{\delta}^{-1} \cdot (\beta \log_{\frac{1}{\beta}} \frac{e}{\beta}) + k\right)$, $\mathcal{T} = O(\log n)$ and $\mathcal{M} = O(\beta k^2 \log n)$.*

Ignoring log factors and constants dependent on β, the resulting query complexity is $\mathcal{Q} = \tilde{O}(n/k + \boldsymbol{\delta}^{-1} + k)$, essentially matching the lower bound (except for the additive k term). The constant factors increase as β gets closer to 1 and reduce to the naive algorithm when $\beta = 1 - 1/k$. We improve on that in the following result, albeit with the cost of β being at most $1/2$.

▶ **Theorem 18.** *When $\beta < 1/2$, There exists a protocol that solves Disjunction with $\mathcal{Q} = O\left(\frac{n \log n}{k \log(2/(2\beta+1))} + \frac{1}{\log(2/(2\beta+1))} \cdot \boldsymbol{\delta}^{-1} + \log^2 n\right)$, $\mathcal{T} = O(\log n)$ and $\mathcal{M} = O(\beta k^2 \log n)$.*

Allowing randomization in the protocol design, we achieve the following result.

▶ **Theorem 19.** *When $\beta < 1$, There exists a protocol that w.h.p. solves Disjunction with $\mathcal{Q} = O\left(\frac{\log n}{\gamma k} \cdot \boldsymbol{\delta}^{-1} + \frac{\log k \log n \log(1/\boldsymbol{\delta})}{\gamma}\right)$, $\mathcal{T} = O\left(\log k \cdot \log \boldsymbol{\delta}^{-1}\right)$ and $\mathcal{M} = O(k^2 \log k \cdot \log \boldsymbol{\delta}^{-1})$.*

5 Related Work

To the best of our knowledge, we are the first to study retrieval problems in the DR model, as defined above. We now provide a description of related studies.

As discussed earlier, Byzantine resilience research was largely limited to a few problems like Byzantine Agreement, Byzantine Broadcast, State Machine Replication, etc. More recently, we have seen many investigations of Byzantine resilience in other problems and models. Quite naturally, it has been explored in P2P settings to ensure robust membership sampling [9] and resilient P2P overlay design [21, 3]. Apart from that, Byzantine resilience was explored in the context of mobile agents [17, 10, 15] and graph algorithms [5]. In the last decade, there was quite a bit of interest in Byzantine resilient learning, starting with multi-armed bandit problems [6]. Finally, there was a recent flurry of works inspired by the popularity of Byzantine resilient optimization algorithms in federated and distributed learning [41, 8, 25, 19, 42, 20].

Byzantine Reliable Broadcast (BRB) was first introduced by Bracha [11]. In BRB, a designated sender holds a message M, and the goal is for every honest peer to output the same M' that must uphold $M' = M$ if the sender is honest. The Download problem can be viewed as a variant of BRB, where the sender is always honest but has no computational powers and is passive (read-only), and peers are always required to output the correct message M. These differences make solving Download different than solving BRB. One easy-to-see difference in results is that Download can be solved trivially even when $1/3 \leq \beta < 1$ and there are no authenticated messages, whereas BRB can not be solved under the same conditions [18]. Another difference is that state-of-the-art BRB protocols like [2] where the sender uses error-correcting codes and collision-resistant hash functions are inapplicable (when considering the source to be the sender). In optimal *balanced* BRB protocols like in [2], the sender sends $O(\frac{n}{k})$ bits to each peer whereas Theorem 3 shows that Download requires $\Omega(\beta n)$ queries (the difference stems from the inability of the source to perform computations).

Most works on Byzantine resilience have focused on models and problems where the data is integrated into the network, making it difficult to get Byzantine resilience past $\beta < 1/3$ or $\beta < 1/2$. However, there have been some exceptions that were observed quite early in the Byzantine resilience literature, like authenticated broadcast [18] that can be achieved for any $\beta < 1$. More recently, the power of decoupling data and computing came into play in the context of mobile agents. The *gathering* problem [17], where mobile agents must gather at one location, can be solved for all fixed $\beta < 1$. Crucially, the honest agents can explore every part of the graph. The Byzantine agents do not control any portion of the graph.

Our work can be viewed as a step towards understanding the power of oracles with the data source playing that role. The use of oracles (also called probes, queries, etc.) has been widespread in classical computing with references dating back to the early seventies [40, 39, 32, 31, 26]. See [28] for an excellent treatment of the various structural complexity theory results that have been obtained through oracles. The power of oracles has been explored in distributed computing as well in the context of overcoming challenges posed by failures in asynchronous settings [35]. On the broader algorithmic front, the property testing model [24] can be viewed as using oracles to access data that is only available through expensive queries.

In essence, we have proposed a hybrid combination of two communication technologies – querying the source and P2P message passing. Such hybrid combinations leading to overall improvements is not new [27]. Friedman et al. [23] studied distributed computing aided by an external entity that they called cloud. They studied asynchronous consensus with the cloud providing a common compare-and-swap (CAS) register access. More recently, Afek et al. [1] introduced the computing with cloud (CWC) model wherein traditional distributed computing models were augmented with one or more cloud nodes that are typically connected to several regular nodes.

The notion of an External Data Source that multiple peers can access is reminiscent of the PRAM model [22, 30] where all processors could access a shared memory. Unfortunately, there has been no work on Byzantine resilience in the PRAM setting. This is not surprising because the PRAM setting allows writing over the shared memory, and Byzantine processors can easily overwrite portions of the input, thereby making it impossible to solve problems in the exact sense.

6 Directions for future work

Our framework adds Byzantine resilience to standard distributed computing with the help of an External Data Source, an entity external to the network. We initiated this study through deterministic and randomized models, focusing on the Download and Disjunction problems, and developing several algorithms, tools, and techniques. Our emphasis was on optimizing the query complexity but also considered time and message complexities. Extending our work to other model variations and/or broader classes of problems like graph and geometric problems, data analytics and peer learning problems are natural next steps.

Our work has shown that this framework is well-suited for Byzantine resilience owing to decoupling of data and computation that lends well to "trust, but verify" techniques in an algorithmically rigorous manner. It will be interesting to see the limits to which Byzantine resilience can be pushed in this framework.

This framework can be interpreted in multiple ways and applied to a wide variety of contexts. Ideas from oracle based computation such as property testing [24] can be easily adapted to our context. One can also envision variants in which the External Data Source offers a richer set of services that may include computation or data re-organization at its end that the peers may need to pay for. Such dynamics can potentially uncover many algorithmic and game theoretic issues like pricing mechanisms and coalition formation. Our approach is thus relevant in contexts like blockchain oracles [7, 12] where a distributed set of peers wish to perform computation on multiple public data sources at different locations (like news outlets, government portals, think-tank reports, etc.) with disparate access costs, access controls and varying levels of trustworthiness. We therefore believe that our work will lead to several other follow-up work exploring all these variations.

In this paper we studied a strong adversarial model. If the source is allowed to provide also a source of global randomness, then our results may be improved further. Specifically, with such service, one can deploy committees guaranteed to have an honest majority w.h.p., which may lead to efficient algorithms for additional problems.

References

1 Yehuda Afek, Gal Giladi, and Boaz Patt-Shamir. Distributed computing with the cloud. In *23rd Int. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS)*. Springer, 2021. doi:10.1007/978-3-030-91081-5_1.

2 Nicolas Alhaddad, Sourav Das, Sisi Duan, Ling Ren, Mayank Varia, Zhuolun Xiang, and Haibin Zhang. Balanced byzantine reliable broadcast with near-optimal communication and improved computation. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, PODC'22, pages 399–417, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519270.3538475.

3 John Augustine, Soumyottam Chatterjee, and Gopal Pandurangan. A fully-distributed scalable peer-to-peer protocol for byzantine-resilient distributed hash tables. In *34th ACM SPAA*, pages 87–98, 2022. doi:10.1145/3490148.3538588.

4 John Augustine, Anisur Rahaman Molla, and Gopal Pandurangan. Byzantine agreement and leader election: From classical to the modern. In *ACM PODC*, pages 569–571, 2021. doi:10.1145/3465084.3467484.

5 John Augustine, Anisur Rahaman Molla, Gopal Pandurangan, and Yadu Vasudev. Byzantine Connectivity Testing in the Congested Clique. In *36th DISC*, pages 7:1–7:21, 2022. doi:10.4230/LIPICS.DISC.2022.7.

6 Baruch Awerbuch and Robert Kleinberg. Competitive collaborative learning. *JCSS*, 74(8):1271–1288, 2008. doi:10.1016/J.JCSS.2007.08.004.

7 Abdeljalil Beniiche. A study of blockchain oracles, 2020. arXiv:2004.07140.

8 Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *NeurIPS*, pages 119–129, 2017. URL: https://proceedings.neurips.cc/paper/2017/hash/f4b9ec30ad9f68f89b29639786cb62ef-Abstract.html.

9 Edward Bortnikov, Maxim Gurevich, Idit Keidar, Gabriel Kliot, and Alexander Shraer. Brahms: Byzantine resilient random membership sampling. *Computer Networks*, 53(13):2340–2359, 2009. doi:10.1016/J.COMNET.2009.03.008.

10 Sébastien Bouchard, Yoann Dieudonné, and Anissa Lamani. Byzantine gathering in polynomial time. *Distributed Comput.*, 35(3):235–263, 2022. doi:10.1007/S00446-022-00419-9.

11 Gabriel Bracha. Asynchronous byzantine agreement protocols. *Information & Computation*, 75:130–143, 1987. doi:10.1016/0890-5401(87)90054-X.

12 Giulio Caldarelli. Overview of blockchain oracle research. *Future Internet*, 14:175, June 2022. doi:10.3390/fi14060175.

13 Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *3rd Symp. on Operating Systems Design and Implementation*, OSDI, pages 173–186. USENIX Assoc., 1999. URL: https://dl.acm.org/citation.cfm?id=296824.

14 Soma Chaudhuri, Maurice Erlihy, Nancy A Lynch, and Mark R Tuttle. Tight bounds for k-set agreement. *J. ACM*, 47(5):912–943, 2000. doi:10.1145/355483.355489.

15 Arnhav Datar, Nischith Shadagopan M. N, and John Augustine. Gathering of anonymous agents. In *AAMAS*, pages 1457–1465, 2023. doi:10.5555/3545946.3598798.

16 Arnhav Datar, Arun Rajkumar, and John Augustine. Byzantine spectral ranking. In *NeurIPS*, volume 35, pages 27745–27756, 2022.

17 Yoann Dieudonné, Andrzej Pelc, and David Peleg. Gathering despite mischief. *ACM Trans. Algorithms*, 11(1), August 2014. doi:10.1145/2629656.

18 D. Dolev and H. R. Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Computing*, 12(4):656–666, 1983. doi:10.1137/0212045.

19 El-Mahdi El-Mhamdi, Sadegh Farhadkhani, Rachid Guerraoui, Arsany Guirguis, Lê-Nguyên Hoang, and Sébastien Rouault. Collaborative learning in the jungle (decentralized, byzantine, heterogeneous, asynchronous and nonconvex learning). In *NeurIPS*, pages 25044–25057, 2021. URL: https://proceedings.neurips.cc/paper/2021/hash/d2cd33e9c0236a8c2d8bd3fa91ad3acf-Abstract.html.

20 Sadegh Farhadkhani, Rachid Guerraoui, Lê Nguyên Hoang, and Oscar Villemaud. An equivalence between data poisoning and byzantine gradient attacks. In *ICML*, pages 6284–6323. PMLR, 2022. URL: https://proceedings.mlr.press/v162/farhadkhani22b.html.

21 Amos Fiat, Jared Saia, and Maxwell Young. Making chord robust to byzantine attacks. In *13th ESA*, pages 803–814. Springer, 2005. doi:10.1007/11561071_71.

22 Steven Fortune and James Wyllie. Parallelism in random access machines. In *Proc. Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 114–118, 1978. doi:10.1145/800133.804339.

23 Roy Friedman, Gabriel Kliot, and Alex Kogan. Hybrid distributed consensus. In *Proc. 17th Int. Conference on Principles of Distributed Systems, OPODIS 2013.*, pages 145–159, 2013. doi:10.1007/978-3-319-03850-6_11.

24 Oded Goldreich. *Introduction to Property Testing.* Cambridge Univ. Press, 2017.

25 Nirupam Gupta and Nitin H. Vaidya. Fault-tolerance in distributed optimization: The case of redundancy. In *ACM PODC*, pages 365–374, 2020. doi:10.1145/3382734.3405748.

26 Péter Hajnal. An ω (n 4/3) lower bound on the randomized complexity of graph properties. *Combinatorica*, 11:131–143, 1991.

27 Daniel Halperin, Srikanth Kandula, Jitendra Padhye, Paramvir Bahl, and David Wetherall. Augmenting data center networks with multi-gigabit wireless links. *SIGCOMM Comput. Commun. Rev.*, 41(4):38–49, August 2011. doi:10.1145/2018436.2018442.

28 Lane A Hemaspaandra and Mitsunori Ogihara. The complexity theory companion. *Acm Sigact News*, 32(4):66–68, 2001. doi:10.1145/568425.568436.

29 Bruce M. Kapron, David Kempe, Valerie King, Jared Saia, and Vishal Sanwalani. Fast asynchronous byzantine agreement and leader election with full information. In *Proc. 19th ACM-SIAM Symp. on Discrete Algorithms, SODA 2008*, pages 1038–1047, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347196.

30 Richard M Karp. *A survey of parallel algorithms for shared-memory machines.* University of California at Berkeley, 1988.

31 Valerie King. Lower bounds on the complexity of graph properties. In *Proc. 20th ACM Symposium on Theory of Computing*, STOC '88, pages 468–476, 1988. doi:10.1145/62212.62258.

32 Daniel J Kleitman and David Joseph Kwiatkowski. Further results on the aanderaa-rosenberg conjecture. *Journal of Combinatorial Theory, Series B*, 28(1):85–95, 1980. doi:10.1016/0095-8956(80)90057-X.

33 Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982. doi:10.1145/357172.357176.

34 Silvio Micali and Tal Rabin. Collective coin tossing without assumptions nor broadcasting. In *CRYPTO*, pages 253–266, Berlin, Heidelberg, 1991. Springer. doi:10.1007/3-540-38424-3_18.

35 Achour Mostefaoui, Eric Mourgaya, and Michel Raynal. An introduction to oracles for asynchronous distributed systems. *Future Generation Computer Systems*, 18(6):757–767, 2002. doi:10.1016/S0167-739X(02)00048-1.

36 M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, April 1980. doi:10.1145/322186.322188.

37 Michael O. Rabin. Randomized byzantine generals. In *24th FOCS*, pages 403–409, 1983. doi:10.1109/SFCS.1983.48.

38 R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978. doi:10.1145/359340.359342.

39 Ronald L. Rivest and Jean Vuillemin. On recognizing graph properties from adjacency matrices. *Theoretical Computer Science*, 3(3):371–384, 1976. doi:10.1016/0304-3975(76)90053-0.

40 Arnold L. Rosenberg. On the time required to recognize properties of graphs: a problem. *SIGACT News*, 5(4):15–16, October 1973. doi:10.1145/1008299.1008302.

41 Lili Su and Nitin H. Vaidya. Multi-agent optimization in the presence of byzantine adversaries: Fundamental limits. In *American Control Conf, ACC*, pages 7183–7188. IEEE, 2016. doi:10.1109/ACC.2016.7526806.

42 Lili Su and Nitin H. Vaidya. Byzantine-resilient multiagent optimization. *IEEE Trans. Autom. Control.*, 66(5):2227–2233, 2021. doi:10.1109/TAC.2020.3008139.

A Some missing proofs

Proof of Theorem 2

Proof. To perform Download deterministically, we use *public majority committees*. In this type of committee, *public* means that every peer knows the committee members, and *majority* that the committee is guaranteed to have a strict majority of honest members. The algorithm creates n public majority committees (one per input bit). The committee $\mathcal{C}_i$ is constructed by assigning it $2\beta k+1$ for $1 \le i \le n$ in a round robin fashion with wrap-around. See Algorithm 5. This ensures that

1. each committee gets $2\beta k+1$ members, thereby establishing majority, and
2. each peer appears in at most $O(\beta n + n/k)$= $O(\beta n)$ committees (since $\beta \ge 1/k$).

Algorithm 5 Elect Public Majority Committee $\mathcal{C}_i$.

```
1: for $0 \le j < 2\beta k + 1$ do
2:     Assign peer $(i-1)(2\beta k+1)+j \pmod{k}+1$, to $\mathcal{C}_i$.
```

The key observation is that it suffices if each bit i is queried by a public majority committee C_i since when such a committee sends votes on the value bit to every other peer, each other (honest) peer can trust the majority vote of the committee. Constructing public majority committees is done as described in Algorithm 5, and complexity measures follow from the properties of the construction (see Sect. 2). ◀

Proof of Theorem 3

Proof. To establish this, we prove a slightly stronger claim. Consider a deterministic protocol $\mathcal{P}$ for the Download problem. For an n-bit input $\mathcal{X}$, let $\mathcal{E}(\mathcal{X})$ denote the (unique) execution of $\mathcal{P}$ on $\mathcal{X}$ in which none of the peers has failed. Then, the following holds.

▶ **Lemma 20.** *For every $\mathcal{X}$, every bit x_i $(1 \le i \le n)$ is queried by at least $\beta k+1$ peers during the execution $\mathcal{E}(\mathcal{X})$.*

Proof. Towards contradiction, suppose there exists an input $\mathcal{X} = \{x_1, \ldots, x_n\}$ and an index $1 \le i \le n$ such that in the execution $\mathcal{E} = \mathcal{E}(\mathcal{X})$, the set $\hat{M}$ of peers that queried the bit x_i is of size $|\hat{M}| \le \beta k$. Without loss of generality, let $x_i = 0$.

The adversary can now apply the following strategy. It first simulates the protocol $\mathcal{P}$ on $\mathcal{X}$ and identifies the set $\hat{M}$. It now generates an execution $\mathcal{E}'$ similar to $\mathcal{E}$ except for the following changes: (a) The input $\mathcal{X}' = \{x'_1, \ldots, x'_n\}$ in $\mathcal{E}'$ is the same as $\mathcal{X}$ except that $x'_i = 1$. (b) The peers of $\hat{M}$ are Byzantine; all other peers are honest. (c) Each Byzantine peer $M \in \hat{M}$ behaves according to $\mathcal{P}$ except that it pretends that $x'_i = 0$, or in other words, it behaves as if the input is $\mathcal{X}$ (and the execution is $\mathcal{E}$).

One can verify (e.g., by induction on the rounds) that the honest peers cannot distinguish between the executions $\mathcal{E}$ and $\mathcal{E}'$. Therefore, they end up with the same output in both executions. This contradicts the fact that their output in $\mathcal{E}$ must be $\mathcal{X}$, and their output in $\mathcal{E}'$ must be $\mathcal{X}'$. ◀

The lemma implies that for every input $\mathcal{X}$, the total query complexity of the protocol is greater than βkn. Theorem 3 follows. ◀

In the remainder of this section we present the analysis of the main body of Theorem 12.

When M joins (in Procedure `Committee_Work`) the committee $\mathcal{C}_i$ for some $i \in \mathcal{U}_M^J$, M is required to *actively* query the source for the value of x_i. We then say that $\mathcal{C}_i$ is an *active committee* for M. (In contrast, when M joins a committee $\mathcal{C}_i$ for $i \in \mathcal{K}_M^J$, it costs it nothing since it already has the value of x_i stored in $res[i]$, so it does not need to spend another query.) We define the following size variables.

- Let $\tilde{n}_M^J$ denote the number of *active* committees for M in phase J.
- Let $\hat{n}_M^J = |\hat{\mathcal{I}}_M^J|$ denote the total number of committees that M joins by Procedure `Committee_Work` in phase J. (Note that $\tilde{n}_M^J \le \hat{n}_M^J$)
- Let $n_M^J = |\mathcal{I}_M^J|$ denote the total number of requests received by M by Procedure `Collect_Requests` in phase J.

▶ **Lemma 21.** *If some honest M adds i to its set KTA_M^J of known-to-all bits at the end of the* `Gossip`(2) *step of phase J, then $i \in \mathcal{K}_{M'}^{J+1}$ for every honest M'.*

▶ Note 22. the sets KTA_M might not be all equal. Namely, every honest peer might be aware of a different subset of the known-to-all bits. Note, however, that as shown later in Lemma 27, the sets KTA_M of all honest peers contain the set CORE discussed in the high-level overview, and the fast growth of CORE is essentially the cause for the fast shrinkage of the set of unknown bits.

Proof. Suppose $i \in \mathrm{KTA}_M^J$ for some honest M. Then in `Gossip`(2) of phase J, M counted at least $2\beta k + 1$ messages containing (i, b) (for $b \in \{0, 1\}$). At least $\beta k + 1$ of these messages were sent by honest peers, and therefore, in the `Gossip`(2) step of phase J, all honest peers will count at least $\beta k + 1$ messages containing (i, b). Consequently, every honest peer M' will move i to $\mathcal{K}_{M'}$ at that step, so $i \in \mathcal{K}_{M'}^{J+1}$. ◀

Properties of clean executions.

▶ **Observation 23.** *In a J-clean execution, if $i \in \mathcal{U}^J$ (i.e., x_i is still unknown in phase J), then for every honest peer M, the reduced committee $\mathcal{C}_i^M$ is ρ-representative.*

▶ Remark 24. Note that once a committee is selected, the adversary can corrupt all of its members in the very next round. By then, however, the committee had completed its querying and communication actions, so the fact that it is no longer representative does not harm the execution. Note also that the need to complete all committee actions in a single round is the reason why it is required to perform the querying sequentially, spending a round for each bit x_i. The querying operations of all committees could, in principle, be parallelized, but the subsequent communication step might require more than a single round in the CONGEST model, giving the adversary an opportunity to intervene and corrupt an entire committee before it has completed sending its messages.

Note that those bits that were not moved from $\mathcal{U}_M$ to $\mathcal{K}_M$ during the main phases J of the protocol were directly-verified in the final step of the protocol. This implies the following.

▶ **Observation 25.** *By the end of the execution, every honest peer has the value $res[i]$ for every bit x_i.*

It remains to show that for every x_i, the $res[i]$ value obtained by each honest peer is correct.

▶ **Lemma 26.** *In a J-clean execution, whenever an honest peer learns an input bit x_i in phases 0 to J, the learned value $res[i]$ is correct.*

Proof. Consider an input bit x_i. Order the honest peers that learned x_i during phases 0 to J according to the time by which they acquired x_i. the proof is by induction on this order.

For the induction basis, note that the first peer to acquire x_i must have directly verified it, so the value it has obtained is clearly correct.

Now consider the t-th peer M in this order, and suppose M knows that $x_i = b$. there are several cases to consider.

Case 1. M directly-verified x_i, either on the last step of the protocol or in Procedure `Committee_Work` during some phase J. Then again, $res[i]$ is clearly correct.

Case 2. M comm-verifies x_i, in Procedure `Committee_Work`. Then M found $\psi_b(i) \geq \rho$ and $\psi_{1-b}(i) < \rho$. Since the execution is J-clean, $\mathcal{C}_i^M$ is ρ-representative by Lemma 23. This implies that if $x_i = 1 - b$ then all the honest peers in $\mathcal{C}_i^M$ would return $1 - b$, and M would find $\psi_{1-b}(i) \geq \rho$, which did not happen. Hence, $x_i = b$.

Case 3. M gossip-verifies x_i, in the `Gossip`(1) or `Gossip`(2) step. Then M has received messages from $\beta k + 1$ or more peers stating that they already know that $x_i = b$. At least one of those peers, M', is honest, and it acquired x_i prior to M. Hence the inductive hypothesis applies to it, yielding that indeed $x_i = b$. ◀

Proofs of Convergence invariants

Proof of Lemma 10.

Proof. Consider a bit index $i \notin \mathrm{NKTA}_M^J$. Then x_i is marked known-to-all by M in the `Gossip`(2) step. Consequently, M ignores x_i in phase J even if it receives it in some request message in Procedure `Collect_Requests`. Hence $i \notin \mathcal{I}_M^{J+1}$. The first containment follows

Consider a bit index $i \in \mathrm{NKTA}_M^J$. Then x_i is not listed as known-to-all in M, i.e., $i \notin \mathrm{KTA}_M$, so M had $\varphi_0(i) \leq 2\beta k$ and $\varphi_1(i) \leq 2\beta k$.

Let $b = x_i$, i.e., the correct value of x_i, and let $0 \leq \delta \leq 1$ be the fraction of faulty peers that reported knowing i. Since the execution is J-clean, by Lemma 26, we know that $\varphi_{1-b}(i) \leq \delta\beta k$. Therefore $\varphi_0(i) + \varphi_1(i) \leq (2+\delta)\beta k$. Hence, the number of peers that informed M that they do not know x_i satisfies $k - (\varphi_0(i) + \varphi_1(i)) \geq (1-(2+\delta)\beta)k > (1-\delta)\beta k$, where the second inequality follows since $\beta < 1/3$.

Hence, there is at least one honest peer M' that did not send x_i as part of its $\mathcal{K}_{M'}^{J,mid}$, so $i \in \mathcal{U}_{M'}^{J,mid}$, and hence $i \in \mathcal{U}^{J,mid}$. The second containment follows.

The next containment follows from the fact that for an honest peer M, $\mathcal{U}_M$ is monotone, decreasing in time.

Consider an index $i \in \mathcal{U}^J$. Then some honest $M' \in \mathcal{H}$ has $i \in \mathcal{U}_{M'}^J$. This has two implications when $J \geq 1$. First, M' will send a request to learn i in Procedure `Collect_Requests` of phase $J-1$. Second, by Lemma 21 $i \notin \mathrm{KTA}_M^{J-1}$(otherwise $i \in \mathcal{K}_{M'}^J$). Hence M will respect the request by M' and add i to I_M^J. When $J = 0$, $\mathcal{U}^J = \{1, \ldots, n\} = \mathcal{U}_M^J = \mathcal{I}_M^J$. The fourth containment follows. ◀

Define the *core of 2-common-knowledge* after phase J as follows. For every index i, let $num_V^J(i)$ denote the number of honest peers M that comm-verified i and updated it in Procedure `Committee_Work` of phase J. Then

$$\mathsf{CORE}^J \; = \; \{i \mid num_V^J(i) \geq \beta k + 1\}.$$

The name is justified by the following lemma.

▶ **Lemma 27.** *If $i \in \mathsf{CORE}^J$ then, $i \in \mathcal{K}_M^{J,mid}$ and $i \in KTA_M^J$, for every honest peer M*

Proof. Consider an index $i \in \mathsf{CORE}^J$. By definition, x_i was comm-verified by at least $\beta k+1$ honest peers during Procedure `Committee_Work` of phase J. Each of these peers will send i (along with its value) to every other peer during the `Gossip`(1) step. Subsequently, at the end of this round, $i \in \mathcal{K}_M^{J,mid}$ for every honest M. Consequently, in `Gossip`(2) of phase J, *all* honest peers will report knowing x_i , so every honest peer M will add it to KTA_M^J ◀

Proof of Lemma 11.

Proof. We first prove part (1), by considering iteration $J \geq 0$ and bounding $|\mathcal{U}^{J,end}|$ at its end.

The purpose of blacklisting Byzantine peers that claim to participate in too many committees, via defining reduced committees, is to curb the influence of the Byzantine peers on votes, by bounding the extent of *Byzantine infiltration* into committees. For every honest peer M and Byzantine peer M', denote by $\mathsf{BI}_M(M')$ the number of reduced committees $\mathcal{C}_i^M$ that M' claimed to belong to. (Note that for peers M' that were not blacklisted, this value is the same as $\mathsf{Work}(M')$.) Denote the total number of Byzantine infiltrations into reduced committees of M by $\mathsf{BI}_M = \sum_{M' \in \mathcal{B}} \mathsf{BI}_M(M')$. Denote the total number of Byzantine infiltrations into reduced committees of honest peers by $\mathsf{BI} = \sum_{M \in \mathcal{H}} \mathsf{BI}_M$. By the way M constructs the reduced committees in Procedure `Committee_Work`, every peer appears in at most W_{max} reduced committees of M, hence $\mathsf{BI}_M \leq \beta k \cdot \mathsf{W}_{max}$, and therefore

$$\mathsf{BI} \;\leq\; \gamma k \cdot \mathsf{BI}_M \;\leq\; \gamma k \cdot \beta k \cdot \mathsf{W}_{max} \;=\; (1+\epsilon)c\beta\gamma \cdot kn\log n.$$

Consider a bit $x_i \in \mathcal{U}^J$. By the fourth containment of Lemma 10, $x_i \in \mathcal{I}_M^J$ for every honest peer M. Hence every honest M will set up a committee $\mathcal{C}_i$, which will be ρ-representative since the execution is J-clean.

A necessary condition for x_i to remain in $\mathcal{U}^{J,mid}$ is that at most βk honest peers directly verify it in Procedure `Committee_Work` of phase J. This is because otherwise,$i \in \mathsf{CORE}^J$ and by lemma 27, it will belong to $\mathcal{K}_M^{J,mid}$ for every honest M.

Hence, in order to keep i in $\mathcal{U}^{J,mid}$, the adversary must prevent at least $(1-2\beta)k$ honest peers from directly- or comm-verifying x_i. To achieve that, at least ρ Byzantine peers must infiltrate the reduced committee $\mathcal{C}_i^M$ for at least $(1-2\beta)k$ honest peers. This incurs at least $(1-2\beta)k\rho$ work. Hence, the number of bits x_i for which this can happen is at most

$$|\mathcal{U}^{J,mid}| \;\leq\; \frac{\mathsf{BI}}{(1-2\beta)k\rho} \;\leq\; \frac{(1+\epsilon)c\beta\gamma \cdot kn\log n}{(1-2\beta)k \cdot (1-\epsilon)Z\log n/\alpha^J} \;=\; \frac{(1+\epsilon)\beta Z \cdot \alpha^J n}{(1-2\beta)(1-\epsilon)Z} \;=\; \alpha \cdot \alpha^J n\,,$$

where the last equality is by the definition of α. This yields Part (1).

By Lemma 10, Part (2) follows from part (1) upon noting that $n_M^J = |\mathcal{I}_M^J| \leq |\mathcal{U}^{J-1,mid}|$, and Part (3) follows from part (2). Part (4) follows from part (3), noting that $\mathcal{U}_M^J \subseteq \mathcal{U}^J$. ◀

Proofs of high probability of clean executions

Proof of Lemma 7.

Proof. We first show that for every bit x_i, $\mathbb{P}[\mathcal{EV}_1(J,i)] \leq 1/n^{2+\lambda}$.

Consider an index $i \in \mathcal{U}^J$. By Lemma 10, $\mathcal{U}^J \subseteq \mathcal{I}_M^J$, and hence $i \in \mathcal{I}_M^J$,

Therefore, all honest peers join the committee $\mathcal{C}_i$ with probability p. Hence, denoting the number of honest peers in $\mathcal{C}_i$ by X,

$$\mathbb{E}[X] \;=\; p|\mathcal{H}| \;\geq\; p \cdot \gamma k \;=\; \frac{\gamma c \log n}{\alpha^J} = \frac{Z \log n}{\alpha^J}.$$

$$\mathbb{P}[\mathcal{EV}_1(J,i)] \;=\; \mathbb{P}[X<\rho] \;=\; \mathbb{P}[X \le (1-\epsilon)\cdot Z\log n/\alpha^J] \le \mathbb{P}[X \le (1-\epsilon)\mathbb{E}[X]] \qquad (5)$$

By Chernoff's bound,

$$\mathbb{P}[X \le (1-\epsilon)\mathbb{E}[X]] \le \exp\left(-\frac{\epsilon^2\mathbb{E}[X]}{2}\right) \le \exp\left(-\frac{\epsilon^2}{2}\cdot\frac{Z\log n}{\alpha^J}\right), \qquad (6)$$

and by Eq. (2) it follows that

$$\mathbb{P}[\mathcal{EV}_1(J,i)] \;\le\; \exp\left(-(2+\lambda)\cdot\frac{\log n}{\alpha^J}\right) \;\le\; n^{-2-\lambda}.$$

By the union bound, the probability that *any* bad event of type $\mathcal{EV}_1$ occurred in the execution is at most $O(\frac{1}{n^{1+\lambda}})$. ◀

Proof of Lemma 8.

Proof. We first show that for every honest peer M, $\mathbb{P}[\mathcal{EV}_2(J,M)] \le 1/n^{8/3}$. The bad event $\mathcal{EV}_2(J,M)$ occurs if $\hat{n}^J_M > \mathsf{W}_{max}$ in phase J. In Procedure `Committee_Work`, M tries (randomly) to join the committee $\mathcal{C}_i$ for every $x_i \in \mathcal{I}^J_M$, hence $\mathbb{E}[\hat{n}^J_M] = pn^J_M$. Applying Lemma 11(2), we get that

$$\mathbb{E}[\hat{n}^J_M] \;\le\; p\alpha^J n$$

We introduce a variable $X \in (0,1]$ such that

$$\mathbb{E}[\hat{n}^J_M] \;=\; X\cdot p\alpha^J n \;=\; X\cdot c\log n\cdot\frac{n}{k} \;=\; \frac{X\cdot\mathsf{W}_{max}}{1+\epsilon}.$$

We can see now that

$$\mathbb{P}[\mathcal{EV}_2(J,M)] \;=\; \mathbb{P}[\hat{n}^J_M > \mathsf{W}_{max}] \;\le\; \mathbb{P}\left[\hat{n}^J_M > \frac{1+\epsilon}{X}\cdot\mathbb{E}[\hat{n}^J_M]\right]$$

Using the variation of Chernoff's bound that says that, for $\delta>0$,

$$\mathbb{P}\left[A>(1+\delta)\mathbb{E}[A]\right] \le \exp\left(-\frac{\delta^2}{2+\delta}\cdot\mathbb{E}[A]\right)$$

and setting $\delta = \frac{1+\epsilon}{X}-1$, we get

$$\begin{aligned}\mathbb{P}[\mathcal{EV}_2(J,M)] \;&\le\; \exp\left(-\frac{(\frac{1+\epsilon-X}{X})^2}{2+\frac{1+\epsilon}{X}-1}\cdot\mathbb{E}[\hat{n}^J_M]\right) \;=\; \exp\left(-\frac{(1+\epsilon-X)^2}{X^2(\frac{1+\epsilon}{X}+1)}\cdot Xc\log n\cdot\frac{n}{k}\right)\\ &=\; \exp\left(-\frac{(1+\epsilon-X)^2}{X+1+\epsilon}\cdot c\log n\cdot\frac{n}{k}\right) \;=\; \exp\left(-f(X)\cdot c\log n\cdot\frac{n}{k}\right),\end{aligned}$$

where $f(x) = \frac{(1+\epsilon-x)^2}{x+1+\epsilon}$. It is easily verifiable that $f(x)$ is monotone decreasing in the range $[0,1]$, attaining a minimum value of $\frac{\epsilon^2}{2+\epsilon}$, i.e, $f(x) \ge \epsilon^2/2+\epsilon$ for every $x\in[0,1]$. Therefore, we get

$$\begin{aligned}\mathbb{P}\left[\hat{n}^J_M > \frac{1+\epsilon}{X}\cdot\mathbb{E}[\hat{n}^J_M]\right] \quad &\le \exp\left(-\frac{\epsilon^2}{2+\epsilon}\cdot c\log n\cdot\frac{n}{k}\right) \;\le\; n^{-c\epsilon^2/(2+\epsilon)}\\ &\le\; n^{-Z\epsilon^2/(2+\epsilon)} \;\le\; \frac{1}{n^{2+\lambda}},\end{aligned}$$

where the last inequality follows by Eq. (3). The lemma now follows by the union bound. ◀