Distributed Download from an External Data Source in Byzantine Majority Settings

John Augustine
Indian Institute of Technology Madras, India

Soumyottam Chatterjee
CISPA Helmholtz Center for Information Security, Saarbrückem, Germany

Valerie King
University of Victoria, Canada

Manish Kumar
Indian Institute of Technology Madras, India

Shachar Meir
Weizmann Institute of Science, Rehovot, Israel

David Peleg
Weizmann Institute of Science, Rehovot, Israel

Abstract

We consider the Download problem in the Data Retrieval Model, introduced in DISC'24, where a distributed set of peers, some of which may be Byzantine, seek to learn n bits of data stored at a trustworthy external data source. Each bit of data can be learned by a peer either through a direct and *costly* query of the source or through other peers that have already learned it; the goal is to design a collaborative protocol that reduces the query complexity defined as the maximum number of bits queried by any honest peer.

We begin with a randomized protocol for the Download problem that achieves optimal query complexity, up to a logarithmic factor. For a stronger "dynamic" adversary that can change the set of Byzantine peers from one round to the next, we achieve optimality (within log factors) for both query complexity (in expectation) and time complexity, but with larger messages. In broadcast communication, where all peers (including Byzantine peers) are required to send the same message to all peers, we achieve (up to log factors) an optimal trade-off between query complexity, time complexity, and message size with the dynamic adversary. All of our protocols can tolerate any constant fraction $\beta < 1$ of Byzantine peers.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Byzantine Fault Tolerance, Blockchain Oracle, Data Retrieval Model, Distributed Download

Digital Object Identifier 10.4230/LIPIcs.DISC.2025.9

Related Version *Full Version*: `https://arxiv.org/abs/2412.19649` [5]

Funding *John Augustine*: Supported by the Centre for Cybersecurity, Trust and Reliability (CyStar), IIT Madras.
Soumyottam Chatterjee: Work done while at IIT Madras, supported by the Centre for Cybersecurity, Trust and Reliability (CyStar), IIT Madras.
Manish Kumar: Supported by the Centre for Cybersecurity, Trust and Reliability (CyStar), 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.

1 Introduction

Almost 45 years ago, the Byzantine Generals Problem, which launched the study of Byzantine agreement, was posed as follows:

> "We imagine that several divisions of the Byzantine army are camped outside an enemy city, each division commanded by its own general. The generals can communicate with one another only by messenger. After observing the enemy, they must decide upon a common plan of action." [18]

One component of the problem is that each general v must first evaluate the situation and make observations about the enemy that are used to guide its choice of action $x(v)$ ("attack" or "retreat"). The common formal definition of the agreement problem, as given in the literature, abstracts away the observation stage, and considers the action choice $x(v)$ of each general v as its *input bit* (allowing those bits to differ even among the honest generals).

The Data Retrieval (DR) model [4] "zooms in" on the stage of collecting the observations. Suppose that there are n questions of interest about the enemy, which the generals are required to answer to reach a correct action choice. Denote the answers to these questions (conveniently assumed to be Boolean) by the *observation bits* $b_1, \ldots, b_n$. Making all n observations and learning the entire *observations vector* $\bar{b}$ is possible but expensive for any single general v. The DR model addresses this by allowing the generals to collaborate in order to make all n observations and share the bits b_i correctly. Consequently, each general v can compute its choice $x(v)$ based on the entire observations vector $\bar{b}$.

The original Byzantine Generals problem was introduced to model protocols that function correctly despite the failure of components of a computer system, where perhaps there is no external information that can be independently verified. In contrast, our modified variant of the Byzantine Generals problem can be thought of as a metaphor for scenarios (that are common these days) where there is a need to ascertain factual information that is spread across many sources, and the cost for an individual to validate every fact would be prohibitively expensive. This raises a need for a collaborative protocol to share these costs, where, as in the original Byzantine Generals Problem, there are untrustworthy peers who might communicate false findings.

Formally, our model represents the process of making observations as a trustworthy external "read-only" data source. Making observation i is modeled as sending a query on b_i to this data source. Note that in the DR model, making the right choice is *always possible* (albeit at a very high cost); each peer can learn each observation bit b_i by querying the source directly. Hence, a significant advantage of this approach is that impossibility concerns are eliminated; reaching an agreement is always possible, for any number of corrupt peers. Instead, the focus shifts to efficiency: the peers must collaborate to share the cost of learning the observation bits from the source.

The DR model consists of a peer-to-peer network and an external data source in the form of an n-bit array $\mathbf{X}$. There are k peers, some of which may be Byzantine. They can query information from the array and communicate with each other through messages. The peers initially do not know the array's contents, but are required to compute some function of the n bits. The peers can learn information about the bits either by querying the data source directly or from other peers. The primary goal is to minimize the maximum number of queries by any honest peer.

The DR model gives rise to a natural class of problems, called *retrieval problems*. In a retrieval problem, each peer needs to output $f(\mathbf{X})$ for some computable function f of the input. In this work, we focus on the specific problem where f is the *identity* function,

hereafter referred to as the Download problem. We view this as 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 $f(\mathbf{X})$ locally at no additional query, communication, or time cost. Hence, the query cost of the Download problem serves as a baseline against which to compare the costs of other specialized protocols for specific problems. Observe that a lower bound on query complexity for computing any Boolean function on $\mathbf{X}$ serves as a lower bound for Download as well.

The DR model was introduced in [4] and was inspired by distributed oracle networks (DONs) which are a part of blockchain systems. In such networks, peers are tasked with retrieving information from external data sources such as stock prices via well-defined *Application Interfaces* (APIs). Calls to these APIs incur a cost that increases with the oracle's usage. Thus, reducing the number of API calls also reduces the overall cost of the DON.

The Download problem is easily solved in a query-balanced manner in the absence of failures. This problem becomes challenging in the presence of faults. We consider a synchronous[1] failure-prone setting where up to βk peers are Byzantine and at least $\gamma \geq 1-\beta$ fraction are honest. For such systems, a tight bound on query complexity is established in [4] for deterministic Download, complemented by two randomized protocols that solve Download w.h.p.[2] The first can tolerate any constant fraction $\beta < 1$ of Byzantine faults but has non-optimal query complexity of $O\left(\frac{n}{\gamma k} + \sqrt{n}\right)$, while the second has optimal query complexity of $\tilde{O}\left(\frac{n}{\gamma k}\right)$ but can only tolerate up to $\beta < \frac{1}{3}$ fraction of Byzantine faults[3]. In this paper, we explore both the query and time complexity of the Download problem, with our main contribution being a novel randomized synchronous protocol that combines the best of both results, achieving optimal query complexity while tolerating any fraction $\beta < 1$ of Byzantine faults.

1.1 The Model

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

Clique network and communication mode. In a clique (complete) network, the k peers are identified by unique IDs, assumed to be from the range $[1, k]$. In each round, every peer can send a message of up to b bits to each of the other peers. The common variant of this communication mechanism, referred to as *peer-to-peer message passing* communication, allows a peer to send in each round a *different* message to each of the other peers. However, we also discuss (in Sect. 4.3) a variant termed *broadcast* communication, where each peer (including a Byzantine one) can send at most one message per round, and that message is delivered to all other peers. This variant can be implemented in a peer-to-peer network using Byzantine Reliable Broadcast, albeit at additional time and message costs. However, there are communication networks, such as radio networks, where the behavior of communication matches the description of our broadcast model.

[1] The companion paper [6] considers the *asynchronous* setting, which behaves rather differently.

[2] Throughout this paper, we use the term "with high probability" to mean the following: with probability $\geq 1 - n^{-c}$, for some constant $c \geq 1$. We often abbreviate the phrase "with high probability" as "whp" or "w.h.p."

[3] We use the $\tilde{O}(\cdot)$ notation to hide β factors and polylogarithms in n and k.

The source. The n-bit input array[4] $\mathbf{X} = \{b_1, \ldots, b_n\}$ is stored in the source. It allows peers to retrieve that data through queries of the form $\texttt{Query}(i)$, for $1 \leq i \leq n$. The answer returned by the source would then be b_i, the true i^{th} element in the array. This type of communication is referred to as *source-to-peer* communication. Note that we assume the source to be a completely passive entity, and thus it cannot perform computations (such as error correcting codes) as part of the execution of protocols.

Network delay and rounds. We consider a synchronous setting where peers share a global clock and time is partitioned into rounds as follows. Each round consists of three sub-rounds:

1. The *query sending sub-round*, in which each peer can send q queries ($0 \leq q \leq n$) of the form $\texttt{Query}(\cdot)$ to the source.
2. The *query response sub-round*, in which the source responds to all the queries.
3. The *message-passing sub-round* of peer-peer communication, which consists of message exchange between peers. Every message is of size $O(\log n)$ unless otherwise stated.

We assume that local computation is instantaneous and is performed at the beginning of each sub-round. We also assume that a peer can choose to ignore (not process) messages received from another peer during the execution. Such messages incur no communication cost to the recipient peer.

The adversarial settings. The behavior of the environment in which our protocols operate is modeled by an adversary $\mathcal{A}$ that selects the input data, and determines the peers' failure pattern, but it cannot corrupt the communication with the source. In executing a protocol, a *Byzantine* peer can deviate from the protocol arbitrarily (controlled by $\mathcal{A}$). The adversary $\mathcal{A}$ can corrupt at most βk peers for some given[5] $\beta \in [0, 1)$. A peer is considered *honest* if it obeys the protocol throughout the execution. 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 $\mathcal{F}$. (resp., $\mathcal{H}$).

The Byzantine adversary can select peers to corrupt at the start of each round. We consider two types of Byzantine adversaries. Under the *fixed adversary*, a corrupted peer remains corrupted for the rest of the execution. The *dynamic* adversary can decide on the set of corrupt peers arbitrarily at the start of any round, or more explicitly, it can make a peer v Byzantine on one round and honest on the next. In both cases, the total number of corrupted peers at any given time is bounded by βk, and the adversary can adaptively decide which peers to corrupt.

As in [4], we assume a strong adversarial model. At the beginning of each round t, $\mathcal{A}$ has knowledge of all the local random bits generated up to round $t - 1$ and all peer-peer and source-peer communications up to round $t - 1$.

Definitions. The following complexity measures and parameters are used in our analysis.

Query Complexity ($\mathcal{Q}$): the maximum number of bits queried by an honest peer during the execution,

Round Complexity ($\mathcal{T}$): the number of rounds (or *time*) it takes for the protocol to terminate,

Message Complexity ($\mathcal{M}$): the total number of messages sent by honest peers during the execution,

Message_size (ϕ): the maximum number of bits sent in one message by an honest peer during the execution.

---

[4] Throughout this paper, we assume $n \geq k$. In typical applications, $n \gg k$.

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

Table 1 Our main results (with β treated as *any* positive constant < 1). Here, * indicates results that only hold in expectation and # indicates results that hold in worst-case.

Comparison of the Existing and Developed Results for Byzantine Fault					
Adversary	**Theorem**	**Resiliency**	**Query**	**Time**	**Message Size**
Fixed Byzantine	[4]	$\beta < 1$	$\tilde{O}(n/k + \sqrt{n})$	$O(n)$	$O(\log n)$
	[4]#	$\beta < 1/3$	$\tilde{O}(n/k)$	$O(n)$	$O(\log n)$
	Thm 5	$\beta < 1$	$\tilde{O}(n/k)$	$O(n \log k)$	1
Dynamic Byzantine	Thm 8	$\beta < 1$	$\tilde{O}(n/k + \sqrt{n})$	2	$\tilde{O}(n/k + \sqrt{n})$
	Thm 10	$\beta < 1$	$\tilde{O}(n/k)$*	$O(\log k)$	$O(n)$
Dynamic Byzantine and Broadcast	Cor 11	$\beta < 1$	$\tilde{O}(n/k)$*	$O(\log n)$	$O(n/k)$
	Thm 12	$\beta < 1$	$\tilde{O}(n/k)$#	$O(\log^2 n)$	$\tilde{O}(n/k)$

Given that queries to the source generally incur higher costs, we focus on minimizing the query complexity $\mathcal{Q}$. Note that 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.

The Download problem. Consider a DR network with k peers, where at most βk can be Byzantine, and a source that stores a bit array $\mathbf{X} = [b_1, \ldots, b_n]$. The Download problem requires each peer to learn $\mathbf{X}$. Formally, each honest peer μ outputs a bit array res_μ, and it is required that, upon termination, $res_\mu[i] = b_i$ for every $i \in \{1, \cdots, n\}$ and $\mu \in \mathcal{H}$, where $\mathcal{H}$ is the set of honest peers.

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 $\mathcal{M} = \tilde{O}(nk)$, assuming small messages of size $\tilde{O}(1)$, and the 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.

1.2 Related Work

Following the model introduced in [4], our work studies a new class of fault-tolerant problems that is heavily inspired by Blockchain oracles. Our focus is on the synchronous setting, but we also explore the asynchronous version, which is quite different, in our companion paper [6]. There are multiple classic *Byzantine Fault-Tolerant (BFT)* problems (e.g., agreement, broadcast, and state machine replication) that provide insight and inspiration when considering the Download problem, cf. [9, 14, 19, 23, 24]. We were also influenced by protocols developed for Oracle networks, such as OCR and DORA [10, 13]. Our model and algorithms have a similarity to some work on recommendations systems, in particular [7]. See Sect. 5 for a more detailed comparative discussion.

1.3 Our Contributions

We explore the Download problem under various adversarial and network models and present several deterministic and randomized protocols and a lower bound for Download. 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.

Query Optimality with Synchronous Point-to-point Communication and Byzantine Failures. We start with closing the gap left open in [4]. The model studied in that paper involves a synchronous point-to-point communication network and Byzantine failures. In this model, for deterministic protocols, the Download problem turns out to be expensive, requiring $\Omega(\beta n)$

queries in the worst case (a matching upper bound is shown which works under asynchrony as well). Every peer essentially has to query the entire input array for itself. However, [4] gives a randomized protocol 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 $k > \sqrt{n}$. It was shown in [4] that as long as $\beta < 1/3$, one 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 $\mathcal{M} = \tilde{O}(nk^2)$. In Section 2 we close the gap and show the existence of a randomized Download protocol with query complexity $\mathcal{Q} = O\left(\frac{n}{\gamma k}\right)$, $\mathcal{T} = O(n \log k)$, and $\mathcal{M} = O(nk^2)$ for $\beta \in [0, 1)$. In that protocol, we use a *blacklisting* technique, i.e., 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 can be blacklisted for a variety of reasons, e.g., if it is directly "caught" in a lie about some bit value, or stops sending messages while they are expected of it, or sends more messages than it is expected to. The protocol is carefully constructed such that the amount of "noise" that can be caused by Byzantine peers, is bounded, and does not cause a large overhead for the honest peers.

Faster Query-optimal Solutions with Synchronous Communication and Byzantine failures. We next ask whether Download can be achieved faster than linear time. To exclude the extreme end of the scale, we show (in Sect. 3) that hoping for a single round Download (assuming arbitrarily large messages can be sent in a single round) is too ambitious. We show that every randomized protocol in which each peer queries no more than $(n - 1)$ bits fails to solve Download with constant probability greater than 0 (for $\beta \approx 1/2$).

Nevertheless, we next derive faster protocols (in Section 4) than the one in Section 2. Specifically, in Subsection 4.1 we show how Download can be achieved in *two* rounds. This solution enjoys the additional advantage that it can cope with a stronger type of adversary, termed *dynamic adversary*, which can change the set of Byzantine peers from one round to the other, provided that this set never exceeds a size of βk. To achieve this result, during the execution of the protocol, we consider a multi-set of the bit strings proposed by different peers that purport their respective strings to be equal to a particular interval of the input bit array. If it is known that at least $t > 0$ proposers were honest peers that correctly know the interval, we can discard all strings in the multi-set that do not appear at least t times, and conclude that the remaining set of distinct strings contains a correct string. A *decision tree* for a set S of strings is a rooted binary tree. Each internal node x is labeled by an index i of the input array $\mathbf{X}$ and each leaf is labeled by a string s such that if a root-to-leaf path goes to the left subtree of a node labeled i, then the i^{th} bit of s is 0, else it is 1. Given a set of strings S of which one is consistent with $\mathbf{X}$, one can build a decision tree with $|S| - 1$ nodes and determine the correct leaf with $|S| - 1$ simultaneous queries.

Unfortunately, this protocol no longer attains query-optimality; its query complexity is $O(n/(\gamma k) + \sqrt{n})$. We improve this query complexity in Subsection 4.2 where we describe an iterated version of the 2-step protocol with *expected* query complexity $O(n \log n/(\gamma k))$ and $O(\log n)$ time. Note that in some settings, large messages can be simulated by several rounds of smaller messages, but here, in a randomized setting with an adaptive adversary, it may not be possible. An adversary can observe the smaller messages that arrive piecemeal over multiple rounds and may be able to adaptively corrupt peers more effectively.

Finally, in Subsection 4.3 we show how the same nearly optimal results can be achieved, with worst case query complexity and small message size, provided we assume *broadcast* communication among peers rather than point-to-point. That is, in each round, each peer must send the same message to all other peers, and this applies also to the Byzantine peers. In this model, we get a protocol with the worst case query complexity $O((1/\gamma)\log^2 n)$ and $O((1/\gamma)\log^2 n)$ time, and message size $O(\log n/\gamma)$. A lower bound in Section D shows that this protocol demonstrates an optimal trade-off among these parameters, up to $\log n$ factors.

2 Query-optimal Download

In this section, we show how to optimize the query complexity of Download up to a factor of $\log n$, achieving $\mathcal{Q} = O\left(\frac{n \log n}{\gamma k}\right) = \tilde{O}\left(\frac{n}{\gamma k}\right)$ for any $\beta < 1$. This provides an improvement over the best previously known results for Download [4], which either guaranteed a query complexity of $\mathcal{Q} = \tilde{O}\left(\frac{n}{\gamma k} + \sqrt{n}\right)$ or imposed the additional restriction of $\beta < \frac{1}{3}$.

The protocol described next works when $2\gamma k > 2^{\delta} \cdot \lg^2 n$, for some constant δ[6]. Note that, for smaller values of k, the desired bound of $\tilde{O}\left(\frac{n}{\gamma k}\right)$ on the query complexity holds trivially: when $k \leq \left(\frac{2^{\delta}}{2\gamma}\right) \cdot \lg^2 n$, this bound is $\tilde{O}\left(\frac{n}{\gamma k}\right) = \tilde{O}(n)$, which is attainable by the trivial protocol where every peer queries the entire input vector.

Our protocol runs in n *epochs*. For $1 \leq i \leq n$, an honest peer μ dedicates the i^{th} epoch to learning the i^{th} input bit. Furthermore, each epoch i is divided into *phases*. During each phase j of the epoch i, μ attempts to learn the i^{th} input bit b_i in one of two ways:

(i) based on messages received from the other peers in earlier phases, or

(ii) by querying the source directly (which happens with gradually increasing probability).

μ relies on the messages of its peers for learning $b_i = b$ in phase j only if the number of peers who sent it the value b exceeds a specified threshold (depending on j) *and* the number of peers who sent it the value $(1 - b)$ is below that threshold. If μ fails to learn the bit in phase j, then it proceeds to phase $(j + 1)$, in which it doubles its probability of querying. This is repeated until μ successfully learns b_i. (Note that a head is always thrown when 2^j exceeds $\frac{\gamma k}{\lg n}$).

Once μ learns b_i, it broadcasts it and blacklists any peers that sent contradictory values. In subsequent phases, μ never relies on bits sent by blacklisted peers. See Algorithm 1 for the pseudocode. Note that in each epoch i, μ learns the value of the i^{th} bit either by *gossip learning* (i.e., receiving messages with a decisive majority, in line 9 of the protocol) or by *query learning* (in line 19). We name the epoch accordingly as a *gossip* epoch or as a *query* epoch. Furthermore, in every epoch i, μ performs a "learning step" in *exactly one* phase, hereafter referred to as the *learning phase* of epoch i and denoted $\ell(i)$.

Correctness. Let $P_j = 1 - \left(1 - \frac{1}{\gamma k}\right)^{2^j}$ be the probability that when flipping 2^j independent random coins, each with bias $\frac{1}{\gamma k}$ toward head, at least one turns heads. The following technical lemma provides upper and lower bounds on P_j.

▶ **Lemma 1.** *For $j \in [f, f^{end}]$ (see line number 6 in Algorithm 1), $P_j \in \left(\frac{2^j}{\gamma k}\left(1 - \frac{1}{2\lg n}\right), \frac{2^j}{\gamma k}\right)$.*

[6] For any positive real number x, we use the notation $\lg x$ to denote $\log_2 x$.

Proof. The binomial expansion of $(1-\varepsilon)^m$ yields the bounds $1-m\varepsilon < (1-\varepsilon)^m < 1-m\varepsilon+\frac{m(m-1)}{2}\varepsilon^2$. Setting $\varepsilon = 1/\gamma k$ and $m = 2^j$, we get $1-2^j/\gamma k < (1-1/\gamma k)^{2^j} < 1-2^j/\gamma k + 2^j(2^j-1)/(2(\gamma k)^2)$, or

$$\frac{2^j}{\gamma k} \;>\; P_j \;>\; \frac{2^j}{\gamma k} - \frac{2^j(2^j-1)}{2(\gamma k)^2} \;>\; \frac{2^j}{\gamma k}\left(1-\frac{1}{2}\frac{2^j}{\gamma k}\right) \;>\; \frac{2^j}{\gamma k}\left(1-\frac{1}{2\lg n}\right),$$

where the last inequality follows from the assumption that $2^j < \gamma k/\lg n$. ◀

To prove the correctness of the protocol, we need to show that all honest peers μ compute $\mathbf{X}^\mu = \mathbf{X}$ correctly with high probability. This is shown by inductively proving that, for any honest peer μ and any bit i, $1 \le i \le n$, $b_i^\mu = b_i$ with high probability. The induction is on i for a fixed honest peer μ. We prove the following two invariant statements inductively.

Blacklisting Statements (BL_i). All peers blacklisted by some honest peer μ until the end of the i-th epoch (i.e., the i^{th} iteration of the outer `for` loop, line 2 of Algorithm 1) are indeed Byzantine, with high probability. For convenience, BL_0 refers to the empty blacklist before the execution enters the loop and is clearly true.

To capture the inner `for` loop (i.e., the phases) as well (see Algorithm 1, line 6), we use $BL_{i,j}$, for $j \in [f, f^{end}]$ and $1 \le i < n$, to refer to the statement that all peers blacklisted by some honest peer μ until the end of the j^{th} inner `for` loop of the $(i+1)$-th epoch are indeed Byzantine with high probability. For convenience, we may use $BL_{i,f-1}$ to refer to BL_i. Also, note that $BL_{i,f^{end}}$ implies BL_{i+1}.

Correctness Statement (C_i). All honest peers have executed the first i epochs correctly, with high probability, i.e., $b_{i'}^\mu = b_{i'}$ for all $1 \le i' \le i$ and all honest peers μ. Again, C_0 refers to the execution being vacuously correct before any bit is processed by the peers. Furthermore, $C_{i,j}$, $1 \le i < n$ and $j \in [f, f^{end}]$, refers to the following statement at the end of phase j in epoch $(i+1)$: for all honest peers μ, $b_{i'}^\mu = b_{i'}$ for all $1 \le i' \le i$ and $b_{i+1}^\mu = b_{i+1}$ for all honest peers that have set their respective I_{voted} bit to 1, with high probability. For convenience, we sometimes use $C_{i,f-1}$ to refer to C_i. Also, note that $C_{i,f^{end}}$ implies C_{i+1}.

Observe that peer μ performs a query if (a) it is the first phase of that particular epoch and it has obtained "heads", or (b) it is phase $j > f$, it is the first time μ has drawn "heads", and $COUNT_i^b \ge \nu 2^j$ for *both* $b = 0$ and $b = 1$.

The basis for the induction is provided by the statements BL_0 and C_0, which are both true. The inductive step is given by the following lemma.

▶ **Lemma 2.** *Consider some epoch $i \in [1, n]$ and assume that conditions BL_{i-1} and C_{i-1} hold at the start of epoch i. For any fixed $c \ge 1$, there is a suitably small fixed choice for the constant δ such that the statements BL_i and C_i hold with probability at least $1 - \frac{1}{n^{c+1}}$ at the end of the epoch i.*

Proof. Fix $c \ge 1$. To prove the claim, we show that $BL_{i-1,j}$ and $C_{i-1,j}$ hold for all phases $j \in [f, f^{end}]$. By definition, $BL_{i-1,f^{end}} = BL_i$ and $C_{i-1,f^{end}} = C_i$. For the sake of contradiction, suppose statement C_i does not hold, and let j^* be the first phase j in epoch i when statement $C_{i-1,j}$ is false, namely, some peer μ tosses heads and goes on to set its I_{voted} bit to 1, but incorrectly assigns $b_i^\mu = 1 - b_i$. Since j^* is the first such occurrence of incorrect behavior, BL_{i-1,j^*-1} and C_{i-1,j^*-1} are true. If $j^* = f$, then μ will explicitly query the bit, so $b_i^\mu = b_i$, a contradiction. So, we focus on the case where $j^* > f$.

Without loss of generality, let $b_i = 0$. We claim that the number of votes received by μ in favor of $b_i = 0$ will be at least $\nu \cdot 2^{j^*} = 2^{j^*-2}$ with probability at least $1 - \frac{1}{n^c}$. Since BL_{i-1,j^*-1} and C_{i-1,j^*-1} are true, all honest peers that tossed heads in earlier phases $j < j^*$ would have correctly voted for $b_i = 0$. To establish the required contradiction, we show that the number of such votes is at least $\nu \cdot 2^{j^*} = 2^{j^*-2}$. Let G_{old} be the set of votes received by μ from honest peers in epoch i during phases j for $j \leq j^* - 2$ (if such phases exist). Let G_{recent} be the set of votes received by μ from honest peers in phase $j' = j^* - 1$; we know this phase exists as $j^* > f$. Our goal is to show that $X = G_{old} \cup G_{recent}$ has cardinality $|X| \geq 2^{j^*-2}$ with high probability. Let G denote the set of all honest peers; $|G| \geq \gamma k$. Note that for every peer in $G \setminus G_{old}$, the probability of joining G_{recent} is $P_{j'} = P_{j^*-1}$ by Lemma 1, so the expected size of X is

$$\mathbb{E}[|X|] \;\geq\; |G| \cdot P_{j^*-1} \;\geq\; \gamma k \cdot \frac{2^{j^*-1}}{\gamma k}\left(1 - \frac{1}{2\lg n}\right) \;=\; 2^{j^*-1}\left(1 - \frac{1}{2\lg n}\right) \;\geq\; 0.9 \cdot 2^{j^*-1},$$

for sufficiently large n (say, $n \geq 32$). Thus, applying Chernoff bound, we get

$$\begin{aligned}
\Pr[|X| \;<\; 2^{j^*-2}] \;&=\; \Pr\left[|X| < 0.5 \cdot 2^{j^*-1}\right] \;\leq\; \Pr\left[|X| < (1 - 4/9)\mathbb{E}[|X|]\right] \\
&<\; e^{-\frac{(4/9)^2}{2}\mathbb{E}[|X|]} \leq\; e^{-\frac{(4/9)^2}{2}\cdot 0.9 \cdot 2^{j^*-1}} \leq\; e^{-\frac{4}{45}\cdot 2^f} \;=\; e^{-\frac{4}{45}\cdot 2^{\delta + \lg\lg n}} \;=\; e^{-\frac{4}{45}\cdot 2^{\delta}\cdot \lg n} \\
&<\; n^{-\frac{4}{45}\cdot 2^{\delta}} \;<\; \frac{1}{n^{c+2}},
\end{aligned}$$

where the last inequality holds when $\delta \geq \lg(45(c+2)) - 2$).

Thus, the number of votes received by μ for the correct bit value (0, as per our assumption wlog) must be at least $\nu \cdot 2^{j^*}$ relying on the fact that by the inductive assumption BL_{i-1,j^*-1}, no honest peer was added to B. From the else part (see line 15 of Algorithm 1), if both bit values received $\nu \cdot 2^j$ votes (or more), then μ will query bit i and this will ensure C_{i-1,j^*}. On the other hand, if the number of votes for bit value 1 is less than $\nu 2^{j^*}$, then, μ will vote for 0, which will be the correct bit. Thus, the contradiction is established, thereby implying C_i with probability at least $1 - \frac{1}{n^{c+1}}$ (applying the union bound over all peers μ).

If C_i holds, then all other honest peers $\mu' \neq \mu$ also vote correctly for 0, so μ will not blacklist any of them. This implies that the invariant BL_i also holds. ◀

Applying Lemma 2 and taking the union bound over all n epochs, we get the following.

▶ **Corollary 3.** *For δ fixed as in Lemma 2, Algorithm 1 ensures that all honest peers μ correctly compute $\boldsymbol{X}^\mu = \boldsymbol{X}$ with probability at least $1 - \frac{1}{n^c}$.*

Query complexity analysis. In the absence of Byzantine peers, the total number of (necessary) queries is $O(n)$, so the average cost per peer is $\mathcal{Q} = O\left(\frac{n}{\gamma k}\right)$. The reason for the additional wasteful queries is two-fold. First, the fact that the protocol is randomized and must succeed in learning all bits with high probability requires some redundancy in querying. Second, Byzantine peers spread fake information, forcing honest peers to perform queries to blacklist the culprits and clarify the true values of $\mathbf{X}$.

Let $\mathbf{R}(\mu)$ denote the sequence of independent random coins flipped by peer μ during execution (each with a bias of $\frac{1}{\gamma k}$ towards heads). For notational convenience, we will often write $\mathbf{R}$ to mean $\mathbf{R}(\mu)$, when the underlying peer μ is clear from the context. Let $\mathbf{R}_{i,j}$ denote the subsequence of $\mathbf{R}$ that was drawn at the beginning of phase j of the epoch i, and let $\mathbf{R}_i$ denote the subsequence of $\mathbf{R}$ that was drawn during the entire epoch i. Respectively, let $R = |\mathbf{R}|$, $R_i = |\mathbf{R}_i|$ and $R_{i,j} = |\mathbf{R}_{i,j}|$.

The variable $S_{i,j}$ used in the protocol denotes the number of coins of $\mathbf{R}_{i,j}$ that turned heads. Similarly, let S_i denote the number of coins of $\mathbf{R}_i$ that turned heads.

▶ **Lemma 4.** *For any fixed $c' \geq 1$, there is a suitably small fixed choice for the constant δ such that Algorithm 1 has query complexity $\mathcal{Q} = O\left(\frac{n \log n}{\gamma k}\right)$ with probability $1 - \frac{1}{n^{c'}}$.*

Proof. Throughout the proof, we consider an honest peer μ.

For every $1 \leq i \leq n$, let $B(i)$ denote the number of peers that were blacklisted by μ in epoch i and let $C(i)$ denote the number of queries performed by μ in epoch i. (Note that $C(i)$ is 0 if i is a *gossip epoch* and 1 if i is a *query epoch*.)

As $C(i) = 0$ for a gossip epoch i, the total cost of gossip epochs is

$$C^{gossip} \;=\; \sum_{\text{gossip } i} C^{gossip}(i) \;=\; 0. \tag{1}$$

Hence, we only need to analyze query epochs. We bound the number of queries in these epochs by first bounding the number of random coins flipped in these epochs.

Consider such an epoch i, with learning phase $\ell = \ell(i)$. There are two cases. The first is when $\ell = f$. In this case, perhaps no Byzantine peers were blacklisted, so $B(i) \geq 0$ and the number of random Coins flipped in this epoch is $R_i = R_{i,f} = 2^f$.

The second case is when $\ell > f$. The fact that μ had to query in phase ℓ implies that gossip learning was not possible, so both $COUNT_i^0 \geq \nu 2^j = 2^{\ell-2}$ and $COUNT_i^1 \geq 2^{\ell-2}$. This, in turn, implies that the number of Byzantine peers that μ gets to blacklist in this epoch is $B(i) \geq 2^{\ell-2}$. On the other hand, the number of Coins used during this epoch is $R_i = \sum_{j=f}^{\ell} R_{i,j} = 2^{\ell+1} - 2^f$.

Combining both cases, we get that for every epoch i, $B(i) \geq (R_i - 2^f)/8$. Summing over all i, we get

$$\beta k \;\geq\; B \;=\; \sum_i B(i) \;\geq\; \sum_i \frac{R_i - 2^f}{8} \;=\; \frac{R - 2^f n}{8}\,.$$

Rewriting, and recalling that $f = \lceil \delta + \lg \lg n \rceil$, we get

$$n \cdot 2^\delta \lg n \;\leq\; R \;\leq\; 8\beta k + 2^f n \;\leq\; 8\beta k + n \cdot 2^\delta \lg n, \tag{2}$$

where the first inequality follows from the fact that in every epoch i, $R_i \geq R_{i,f} \geq 2^\delta \lg n$.

Note that $C(i) \leq S_i$ for every i, hence

$$C^{query} \;=\; \sum_{\text{query } i} C(i) \leq \sum_{\text{query } i} S_i \;=\; S. \tag{3}$$

Recalling that $\mathbf{R}$ is a sequence of R independent Bernoulli variables with probability $\frac{1}{\gamma k}$ for heads, we have that $\mathbb{E}[S] = \frac{R}{\gamma k}$, and applying Chernoff bound we get

$$\Pr[S > 2R/\gamma k] \;=\; \Pr[S > 2\mathbb{E}[S]] \;\leq\; e^{-\mathbb{E}[S]/3} \;=\; e^{-R/3\gamma k} \;\leq\; e^{-n2^\delta \lg n/3\gamma k} \;\leq\; \frac{1}{n^{c'}}\,, \tag{4}$$

where the penultimate inequality relies on the left side of Eq. (2) and the last one holds when $\delta \geq \lg(3\gamma c')$, also relying on the assumption that $k \leq n$. Combining Equations (3), (4), and the right side of (2), we get that with probability at least $1 - \frac{1}{n^{c'}}$,

$$C^{query} \;\leq\; S \;\leq\; \frac{2R}{\gamma k} \;\leq\; \frac{2(8\beta k + cn \lg n)}{\gamma k} \;=\; O\left(\frac{n \log n}{\gamma k}\right). \tag{5}$$

Finally, the lemma follows by Equations (1) and (5). ◀

Each epoch lasts $O(\log(\gamma k))$ phases, so the time complexity is $O(n \log(\gamma k))$. Each peer sends at most n bits to each of the other machines, so the total number of messages sent is $O(nk^2)$. Moreover, the messages sent in line numbers 10 and 20 of Algorithm 1 can be encoded as one-bit messages as μ and i are known from context. Thus, we have the following theorem.

▶ **Theorem 5.** *In the synchronous point-to-point model with Byzantine failures, there is a randomized protocol for* Download *such that, with high probability,* $\mathcal{Q} = O\left(\frac{n \log n}{\gamma k}\right)$, $\mathcal{T} = O(n \log(\gamma k))$, *and* $\mathcal{M} = O(nk^2)$. *Moreover, it only requires single bit messages.*

3 Lower Bounds

Having established the existence of a query-optimal protocol for Download with an arbitrary bound $\beta < 1$ on the fraction of Byzantine peers, we turn to the issue of time complexity and derive faster Download protocols. In this section, we first establish that any *single*-round protocol is inefficient, requiring $\mathcal{Q} = n$. Then, we utilize information theory to show a lower bound on the trade-off between query complexity, time complexity and messages size.

Indistinguishability arguments typically require showing that two carefully constructed executions with different outputs "look the same" from the point of view of at least one peer μ that is honest in both of them. Hence, the peer μ will behave the same (and will output the same value) in both executions. In the well-studied Byzantine agreement context, the prover must show that a peer has the same input and will receive the same messages in both executions. In the case of randomized protocols, this becomes more complicated. Even under a fixed adversarial strategy, there is a probability space of executions over the random choices of all the peers. Therefore, we carefully construct probabilistic arguments over these execution spaces to show indistinguishability. In the DR model, these arguments are further complicated by the additional element of queries. Even if we show that the input of an honest peer and the messages it received are the same in both executions if the queries made by this peer in the different executions are not the same, then those executions do not "look the same" from its point of view and we can't say that it will act the same in the different executions. We obtain the following lower bound.

▶ **Theorem 6.** *Consider an n peer system with external data, for odd n, and let the fraction of Byzantine peers be bounded above by $\beta = (n-1)/2n$. For every protocol $A \in \mathcal{A}(n-1, 1)$, there is an adversarial strategy such that A fails to learn all the input bits with probability at least $(1 - e^{-1})/4$. The lower bound holds even in the broadcast communication model.*

Due to space considerations, readers interested in the proof details are referred to the full version of the paper [5].

4 Faster Download with Near Optimal Query Cost

In this section, we show how to reduce the time complexity to $O(\log n)$. Unlike the previous protocol, the protocols here do not use blacklisting and do not require fixed IDs from round to round; only that there is a known lower bound on the number of honest peers present in each round.

We first describe a key definition and a subroutine in these protocols. In a typical step of these protocols, we fix a parameter φ depending on the round, and the input vector $\mathbf{X}$ is partitioned into $\mathcal{K} = \lceil n/\varphi \rceil$ contiguous subsets of size φ, $\mathbf{X}[\ell, \varphi] = (b_{(\ell-1)\cdot\varphi+1}, b_{(\ell-1)\cdot\varphi+2}, \ldots, b_{\ell\cdot\varphi})$, for $\ell \in [1, \mathcal{K}]$, with the last interval, $(b_{(\mathcal{K}-1)\cdot\varphi+1}, \ldots, b_n))$, being possibly shorter. Each honest

peer queries all the bits of some interval $\mathbf{X}[\ell, \varphi]$ and broadcasts its findings to all other peers, in the form of a pair $\langle \ell, s \rangle$, where s is the bit string obtained from the queries. Hence at the end of a round, every peer receives a collection of strings for different intervals. Let $s[j]$ denotes the j-th bit of the string s. Note that a message $\langle \ell, s \rangle$ can possibly be received in multiple copies, from different peers, and the protocol keeps all copies.

Let S be a multiset of strings for a particular interval ℓ received by peer μ at the current round. Define the set of *t-frequent* strings at a peer μ as

$$\mathsf{FS}(S, t) = \{s \in S \mid s \text{ appears } \geq t \text{ times in } S\}$$

Note that the FS function gets a *multiset* and returns a *set* of t-frequent strings.

A *decision-tree* T is a rooted ordered binary tree where leaves are labeled with strings and each internal node v is labeled by an index i of the input array such that all the strings s in the left (respectively, right) sub-tree of v has $s[i] = 0$ (resp., $s[i] = 1$). Given a set of strings FS of which one is consistent with the input array (i.e., it appears exactly in the input array at the specified indices), we can build a decision tree to determine which one of the strings in FS is consistent with the input array with a cost of $|\mathsf{FS}| - 1$ queries, by querying the indices associated with all the internal nodes of the decision tree at the same time. The path consistent with the results of these queries reaches a leaf that is labeled by the correct string for the interval. See Algorithm 2.

4.1 A Simple 2-step Protocol

To illustrate the ideas of the following protocols in this section, we first give a 2-step protocol with $O(n/\sqrt{\gamma k})$ queries which succeeds with high probability. Note that if γk is very small, every honest peer queries every bit and the protocol is always correct. Otherwise:

1. Split the n bit string into $\mathcal{K}$ equal length intervals of φ bits each. Each peer picks an interval uniformly at random, queries all its bits, and broadcasts the discovered string s together with the identifier $\ell \in [1, \mathcal{K}]$ of its chosen interval.
2. Let FS_ℓ be the set of strings, each of which was received in at least $t = \gamma k/(2\mathcal{K})$ messages from distinct honest peers for interval ℓ, disregarding any string sent by a peer which sends more than one string. Formally, $\mathsf{FS}_\ell = \mathsf{FS}(S_\ell, t)$ where $S_\ell = \{s \mid \langle \ell, s \rangle \text{ received from some peer}\}$.
   In parallel, for each interval $\ell \in [1, \mathcal{K}]$, build a decision tree T_ℓ from FS_ℓ and invoke Procedure DETERMINE$(\cdot)$, letting each peer determine the correct string for interval ℓ.

Correctness. Consider an execution of the protocol, and denote the number of honest peers that pick the interval ℓ by k_ℓ. The protocol succeeds if $k_\ell \geq t$ for every interval ℓ, since every decision tree T_ℓ will contain a leaf with the correct string for the interval ℓ returned by Procedure DETERMINE.

▷ Claim 7. For constant $c \geq 1$, if $t \geq 8(c+1)\ln n$, then $k_\ell \geq t$ for every interval $\ell \in [1, \mathcal{K}]$, with probability at least $1 - 1/n^c$.

Proof. Fix $\ell \in [1, \mathcal{K}]$. The expected number of honest peers that pick the interval ℓ is $\mathbb{E}[k_\ell] = \gamma k/\mathcal{K} = 2t$. Therefore, applying Chernoff bounds, $\Pr[k_\ell < t] \leq e^{-t/8} \leq 1/n^{c+1}$ for $t \geq 8(c+1)\ln n$. Taking a union bound over all intervals, the probability that $k_\ell < t$ for *any* interval ℓ is less than $1/n^c$. ◁

Query complexity. The analysis of the query complexity is based on the fact that the query cost in Step 2 for interval ℓ is the number of internal nodes of the decision tree, which is $|\mathsf{FS}_\ell| - 1$. We defer the full analysis to Appendix B

Message size: The message size in this protocol is the number of bits required to describe the first index of the interval chosen and its bits in Step 1, i.e., $O(\varphi + \log n) = O(\varphi) = O(\sqrt{n/\gamma} + n \log n/(\gamma k))$. In other words, the message size is not more than the query complexity in the first round which is not more than $\mathcal{Q}$, i.e., $O(\mathcal{Q})$.

▶ **Theorem 8.** *There is a 2-round randomized protocol for* Download *in the point-to-point model with* $\mathcal{Q} = O(n \log n/(\gamma k) + \sqrt{n/\gamma})$ *and message size* $O(n \log n/(\gamma k) + \sqrt{n/\gamma})$.

4.2 Download with $O(n \log n/(\gamma k))$ Expected Queries and $O(\log \gamma k)$ time

The 2-step protocol presented in the previous subsection, while fast and simple, is not query-optimal, with its main source of overhead being that every peer, after its initial query of the randomly selected interval, builds a decision tree for every other interval and determines the correct leaf (and hence the correct string) by performing additional queries. In this subsection, we extend the previous protocol, achieving optimal query complexity at the cost of going from $O(1)$ to $O(\log n)$ rounds.

The protocol is based on the following idea. Let $\varphi = \lceil (n/(\gamma k))(8(c+1)\ln n) \rceil$. In Step i, for $i \in [0, \ldots, \lceil \lg(n/\varphi) \rceil]$, the n bits are partitioned into $\mathcal{K}_i = \lceil n/\varphi_i \rceil$ intervals of size $\varphi_i = 2^i \varphi$. Each interval in step i consists of the concatenation of two consecutive intervals from step $i-1$, beginning with the first two intervals. In each step, each honest peer determines a random interval of increasing size by splitting it into its two interval parts and using the technique described in the 2-step protocol above on both parts, until the entire input vector is determined. This significantly reduces the number of decision trees constructed by each peer, from $\mathcal{K}$ to $O(\log n)$, which allows the improved query complexity.

See Algorithm 4 for a formal description.

Correctness. We defer some of the correctness proof to Appendix C, which is a straightforward extension of the argument in Section 4.1.

Denote by ℓ_μ^i the interval ID picked by peer μ at step i.

▶ **Lemma 9.** *In every step* $i \in [0, \lceil \lg(n/\varphi) \rceil]$*, every honest peer* μ *learns the correct value of* $\boldsymbol{X}[\ell_\mu^i, \varphi_i]$ *w.h.p.*

The correctness of the protocol follows from observing that at the last round $\ell = \lg(n/\varphi)$, the intervals are of size $\varphi_\ell = 2^\ell \varphi = n$. Hence, by Lemma 9, every peer learns the entire input.

Query complexity. Given an interval ℓ in Step $i+1$, let x be the total number of strings received for the subintervals in Step i that compose interval, i.e., $x = x_L + x_R$, where x_u for $u \in \{L, R\}$ denotes the number of strings received for subinterval ℓ_u. Let m_x be the number of intervals in Step $i+1$ such that the total number of strings received for that interval equals x. Formally,

$$m_x = |\{\ell \mid \text{received exactly } x \text{ messages of the form } \langle \ell_L, s, i \rangle \text{ or } \langle \ell_R, s, i \rangle\}|$$

The probability of picking an interval in Step $i+1$ with x strings is m_x divided by the total number of intervals, or $m_x/\mathcal{K}_{i+1}$. The expected cost of querying in Step $i+1$ is therefore

$$\sum_x \frac{m_x}{\mathcal{K}_{i+1}} \cdot \frac{x}{t_i} \;=\; \sum_x \frac{m_x}{\mathcal{K}_{i+1}} \cdot \frac{2x}{\gamma k/\mathcal{K}_i} \;=\; \frac{2\mathcal{K}_i}{\gamma k \cdot \mathcal{K}_{i+1}} \sum_x m_x \cdot x \;\le\; \frac{4}{\gamma}\,,$$

where the last inequality follows since each peer broadcasts at most one string, so $\sum_x m_x \cdot x \le k$.

Step 0 requires $\varphi = O(n \log n/(\gamma k))$ queries. The expected cost of querying is $O(1/\gamma)$ per step $i > 0$ per peer, and there are fewer than $\log n$ steps, so the total expected query cost is at most $O(n \log n/(\gamma k))$.

Message size. The maximum message size is that of the last round or $n/2 + 1$.

▶ **Theorem 10.** *There is a* $O\left(\log\left(\frac{\gamma k}{\log n}\right)\right)$*-round protocol which w.h.p. computes* Download *in the point-to-point model with expected query complexity* $\mathcal{Q} = O(n \log n/(\gamma k))$ *and message size* $O(n)$*.*

4.3 Broadcast Model: Worst-case $O((1/\gamma)\log^2 n)$ Queries and Time, and Message Size $O(\log n/\gamma)$

Here we start by showing that in the Broadcast model, where every peer (including those controlled by the adversary) must send the same message to all peers in every round, one can drastically reduce the message size in Algorithm 4. Next, we show that the bound on expected query complexity can be improved to a bound on the worst case query complexity because of the "common knowledge" property guaranteed to the peers by the broadcast medium. Specifically, our main results for the broadcast model are the following.

▶ **Lemma 11.** *There is a* $O(\log n)$*-round protocol that w.h.p. performs* Download *in the Broadcast model with expected query complexity and message size* $O(n \log n/(\gamma k))$*.*

▶ **Theorem 12.** *In the Broadcast model, there is a protocol with worst case* $O(\log^2 n)$ *time,* $O((n/\gamma k)\log^2 n)$ *queries, and* $O((n/\gamma k)\log n)$ *message size.*

Again, readers interested in the details are referred to the full paper [5].

5 Discussion of Related Work

Byzantine fault tolerance was first introduced by Pease, Shostak, and Lamport [23, 19] in 1980 and has since played a pivotal role in distributed computing. Throughout the 80's, the focus was on Byzantine Agreement [9, 19, 23, 24] along with some closely related problems like Byzantine Reliable Broadcast (BRB) [1, 14] and Collective Coin Tossing [11, 20]. Subsequently, Lamport introduced state machine replication (SMR) [17], which was extended to Byzantine resilience by Castro and Liskov [12]. SMR lies at the heart of distributed ledgers and blockchains, technologies that have gained popularity in the recent years.

In Byzantine Agreement, depending on the model assumptions and parameters, the Byzantine adversary can use various strategies to misrepresent or suppress critical information and foil agreement. In fact, even if communication is via broadcast, when $\beta \ge 1/2$, half of the input bits can be misrepresented, paving the way for a simple indistinguishability argument that shows that Byzantine Agreement is impossible. When communication is point-to-point

(and cryptographic primitives are not used), the Byzantine peers can equivocate, i.e., send inconsistent messages. Exploiting equivocation, a more subtle argument (see Chapter 5 in [3]) can be used to prove that Byzantine Agreement is impossible when $\beta \geq 1/3$.

A natural question then is to ask whether there are reasonable circumstances under which any fixed $\beta < 1$ can be tolerated. The answer is affirmative in a version of the Byzantine Resilient Broadcast [14] that allows peers to digitally sign messages. In the BRB problem, we have a designated leader, known to all peers, that has a message. All other peers must learn that message. The safety requirement of BRB, mandates that every honest peer outputs the same message, while the validity requirement demands that all honest peers must output the leader's message when the leader is honest. This problem is challenging because the leader may be Byzantine and can try to trick peers into having different views of the input message. Without using cryptographic primitives, BRB can only be solved when $\beta < 1/3$. However, BRB can be solved for any $\beta < 1$ when peers can digitally sign messages, a well-known result since the early days [14]. Intuitively, BRB with digital signatures makes it difficult for the Byzantine peers to suppress information or equivocate.

The Download problem first introduced in [4] also admits algorithms for any $\beta < 1$. The source is accessible to all nodes in the network, which makes it difficult for Byzantine nodes to suppress crucial information or equivocate without being caught. It is worth noting, however, that the Download problem is fundamentally different from BRB and other traditional problems. Unlike those problems, the Download problem admits a trivial protocol, tolerating any $\beta < 1$ even without the use of cryptographic primitives. In fact, each peer can directly query all the bits, but spending that many queries is prohibitively wasteful. Moreover, BRB protocols require active engagement from the leader, for signing messages, applying error-correcting codes and collision resistant hash functions, etc. In contrast, the source in the DR model is a read-only data source with no computational power, and therefore cannot be expected to execute such primitives. Thus, solutions for traditional problems such as BRB do not translate to solutions for Download. Importantly, with impossibility a non-issue for Download, our focus shifts to ensuring optimal query complexity (and other complexities as well) without sacrificing fault tolerance. This work shows that one can achieve both the optimal query complexity (within log factors) and resilience of up to $\beta < 1$ simultaneously. It is also worth noting that while using primitives like BRB for solving Download is possible, our solutions are significantly more efficient.

Recommendation Systems. The paper [7] studies the problem of determining the *preferences* (represented as Boolean values) of each peer for each of n objects. In that paper, in a single round, peers probe objects at unit cost per object and then post their results with their identity on a public billboard. For each peer, there is a presumed lower bound on the number of peers that have the same preferences, and this fact may be used, like in our algorithms, to determine preferences for objects that have not been probed. The model assumed in [7] is stronger than ours, with (1) a joint billboard, which facilitates broadcast (one-to-all) communication among peers, (2) global randomness, and (3) a static adversary that fixes its strategy before the start of the execution and cannot dynamically change it according to the random coins generated by the algorithm. In contrast, our model generally assumes point-to-point communication, uses only private randomness, and allows adaptive Byzantine behavior. Subsequent papers on recommendation systems consider variants such as lowering the cost by allowing the output to approximate the real preferences [2, 15, 21, 22], or considering the problem in a limited asynchronous environment where message delays (and order of probes) are fixed before the start of the execution [8]. These follow-up works use the same stronger model mentioned above.

Oracle networks. As mentioned above, in Oracle networks, a set of peers is assigned the task of bringing external off-chain data to the network, where a subset of these peers can be Byzantine. Generally, one can describe the operation of an Oracle network as follows. The network generates a report containing the observations made by some (sufficiently many) peers. Once a report is successfully generated, one or several peers transmit the report to an intermediary program (known as a smart contract) that runs on the blockchain, verifies the validity of the report, derives a final value from it (e.g., the median), and then exposes the value for consumption on-chain. Since the traditional usage of these networks is to track exchange rates (e.g USD-ETH) that change with time, studies on Oracle networks focus on the problem of creating a report where the derived value (say the median) must be acceptable (in the sense that it does not deviate much from the set of honest observations) while keeping the costs (e.g., of sending reports to the smart contract) low and tolerating as many Byzantine peers as possible, even at the expense of higher communication and computation off-chain. The Off-Chain Reporting (OCR) protocol [10] solves this problem with $\beta < 1/3$ by running a BA protocol to agree on $2f + 1$ values, then a designated *leader* generates a report and sends it to the contract (the leader acts as an aggregator). The Distributed Oracle Agreement (DORA) protocol [13] takes it a step further by using an approximate agreement scheme, the inherent ability of a Blockchain to act as an ordering service, and multiple aggregators. They improve results to sustain $\beta < 1/2$ and $\beta < 1/3$ w.h.p when the size of the Oracle network is significantly smaller than the size of the entire system. In both of these works, every peer reads all the external data and goes on to participate in the report generation. Our work complements the OCR and DORA protocols and focuses on how to efficiently read the off-chain data while minimizing the number of bits read per peer (as reading from an external source is also more costly than off-chain communication). Our approach would drastically reduce cost when the Oracle network keeps track of a large number of (static) variables (e.g., financial information). It could be used as a black box in the OCR and DORA protocols, and save the need to query all of the n values individually by each peer. Note that both OCR and DORA use cryptographic primitives, whereas we do not. A more in-depth description of the relation between the Download problem and Blockchain oracles is provided in the companion paper [6].

6 Conclusions and Future Work

In this work, we studied the Data Retrieval model, introduced in [4], and improved the previous results on the Download problem by achieving both optimal query complexity and optimal resiliency of any fraction $\beta < 1$ of Byzantine peers. In addition, we presented several new results, including a protocol with $O(\log n)$ time complexity and near-optimal expected query complexity in a model with a dynamic adaptive adversary, and, in the Broadcast model, near-optimal worst case time, query, and message complexity. We also established a lower bound for single-round protocols, demonstrating that it is necessary to query every bit to solve the Download problem within a single round.

Our work is useful in the context of blockchain oracles, specifically, as a sub-routine for data extraction from multiple data sources, where the data sources have some (possibly probabilistic) guarantee of being trustworthy. A relevant question in this context involves handling data that changes over time, such as stock prices or exchange rates. We leave this question open for future exploration, as these temporal changes must be formalized carefully and may require adjustments to the model.

References

1 N. Alhaddad, S. Das, S. Duan, L. Ren, M. Varia, Z. Xiang, and H. Zhang. Balanced byzantine reliable broadcast with near-optimal communication and improved computation. In *Proc. ACM Symp. on Principles of Distributed Computing*, pages 399–417, 2022.

2 N. Alon, B. Awerbuch, Y. Azar, and B. Patt-Shamir. Tell me who I am: An interactive recommendation system. *Theory Comput. Syst.*, 45:261–279, 2009. doi:10.1007/S00224-008-9100-7.

3 H. Attiya and J. Welch. *Distributed computing: fundamentals, simulations, and advanced topics*. Wiley, 2004.

4 J. Augustine, J. Biju, S. Meir, D. Peleg, S. Ramachandran, and A. Thiruvengadam. Byzantine Resilient Distributed Computing on External Data. In *38th Symp. on Distributed Computing (DISC)*, pages 3:1–3:23, 2024.

5 J. Augustine, S. Chatterjee, V. King, M. Kumar, S. Meir, and D. Peleg. Distributed download from an external data source in faulty majority settings. *CoRR*, abs/2412.19649, 2024. doi:10.48550/arXiv.2412.19649.

6 J. Augustine, S. Chatterjee, V. King, M. Kumar, S. Meir, and D. Peleg. Brief Announcement: Distributed Download from an External Data Source in Asynchronous faulty settings. In *DISC*, 2025. These proceedings.

7 B. Awerbuch, Y. Azar, Z. Lotker, B. Patt-Shamir, and M. R. Tuttle. Collaborate with strangers to find own preferences. *Theory Comput. Syst.*, 42:27–41, 2008. doi:10.1007/S00224-007-9016-7.

8 B. Awerbuch, A. Nisgav, and B. Patt-Shamir. Asynchronous active recommendation systems. In *Proc. 11th Conf. on Principles of Distributed Systems OPODIS 2007*, pages 48–61, 2007.

9 G. Bracha. Asynchronous byzantine agreement protocols. *Inf. & Computat.*, 75:130–143, 1987. doi:10.1016/0890-5401(87)90054-X.

10 L. Breidenbach, C. Cachin, A. Coventry, A. Juels, and A. Miller. Chainlink off-chain reporting protocol. Technical report, Chainlink Labs, 2021.

11 A.Z. Broder and D. Dolev. Flipping coins in many pockets (byzantine agreement on uniformly random values). In *25th Symp. on Foundations of Computer Science*, pages 157–170, 1984.

12 M. Castro and B. Liskov. Practical byzantine fault tolerance. In *3rd Symp. on Operating Systems Design and Implementation*, OSDI, pages 173–186. USENIX Assoc., 1999.

13 P. Chakka, S. Joshi, A. Kate, J. Tobkin, and D. Yang. DORA: distributed oracle agreement with simple majority. *CoRR*, abs/2305.03903, 2023. doi:10.48550/arXiv.2305.03903.

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

15 Seth Gilbert, Rachid Guerraoui, Faezeh Malakouti Rad, and Morteza Zadimoghaddam. Collaborative scoring with dishonest participants. In *Proceedings of the Twenty-Second Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '10, pages 41–49, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1810479.1810488.

16 R. Impagliazzo and R. Williams. Communication complexity with synchronized clocks. In *25th IEEE Conf. on Computational Complexity*, pages 259–269, 2010.

17 L. Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16:133–169, 1998. doi:10.1145/279227.279229.

18 L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4:382–401, July 1982. doi:10.1145/357172.357176.

19 L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4:382–401, 1982. doi:10.1145/357172.357176.

20 S. Micali and T. Rabin. Collective coin tossing without assumptions nor broadcasting. In *CRYPTO*, pages 253–266, 1991.

21 A. Nisgav and B. Patt-Shamir. Finding similar users in social networks. *Theor. Comp. Sys.*, 49:720–737, 2011. doi:10.1007/S00224-010-9307-2.

22 A. Nisgav and B. Patt-Shamir. Improved collaborative filtering. In *22nd ISAAC*, pages 425–434, 2011.

23 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.

24 M. O. Rabin. Randomized byzantine generals. In *24th FOCS*, pages 403–409, 1983.

A Supplementary Pseudocode

Algorithm 1 Resilient Download for any fixed known $\beta < 1$; code for peer μ.

$B \leftarrow \emptyset$ ▷ *Set of blacklisted peers*
for i from 1 to n **do** ▷ *Epoch i deals with input bit b_i*
  $I_{voted} \leftarrow 0$ ▷ *Indicator of learning the i-th bit and voting on it*
  $COUNT_i^0 \leftarrow 0$; $COUNT_i^1 \leftarrow 0$; ▷ *Counters for the number of peers that voted 0/1 for bit i*
  $f \leftarrow \lceil \delta + \lg\lg n \rceil$; $f^{end} \leftarrow \lceil \lg(\gamma k) - \lg\lg n \rceil$; ▷ *See Lem. 2 & 4 for choice of constant δ.*
  for j from f to f^{end} **do** ▷ *Phases within each epoch.*
    if $I_{voted} = 0$ (μ has not voted for bit i yet) **then**
      if $COUNT_i^{1-b} < \nu \cdot 2^j$ for $\nu = 1/4$ and for some $b \in \{0,1\}$ AND $(j \neq f)$ **then**
        $b_i^\mu \leftarrow b$ ▷ *Perform a "gossip learning" step*
        Broadcast a vote $\langle \mu, i, b_i^\mu = b \rangle$
        Set $I_{voted} \leftarrow 1$
      else
        if $j = f^{end}$ **then**
          Set coinflip to heads
        else
          Toss 2^j independent Coins with bias $1/\gamma k$ towards heads.
          Let $S_{i,j}$ be the number of Coins that turned heads. Set coinflip to heads if $S_{i,j} \geq 1$.
        if heads **then** ▷ *Perform a "query learning" step*
          $b_i^\mu \leftarrow b \leftarrow query(i)$; ▷ *$\mu$ learns $b_i^\mu = b$ by query*
          Broadcast a vote $\langle \mu, i, b_i^\mu = b \rangle$ ▷ *In practice, sending one bit suffices.*
          Set $I_{voted} \leftarrow 1$
      Receive messages from other peers.
      for $b \in \{0,1\}$ **do**
        $COUNT_i^b \leftarrow |\{\mu' \notin B \mid \mu$ received message $\langle \mu', i, b_i^{\mu'} = b \rangle$ during epoch $i\}|$
          ▷ *count voters for $b \in \{0,1\}$; ignore peers blacklisted in earlier epochs*
    if $I_{voted} = 1$ **then** ▷ *Including the case when I_{voted} changed during the current phase*
      $B \leftarrow B \cup \{\mu' \mid \mu'$ sent message with $b_i^{\mu'} \neq b_i^\mu\}$ ▷ *peers with contradictory vote*

Algorithm 2 CONST_DECISION_TREE(S)
input: Set of strings S
output: Node labeled tree T.

Create a root node v
if $|S| > 1$ **then**
  Set $i \leftarrow$ smallest index of bit on which at least two strings in S differ
  $label(v) \leftarrow i$
  Set $S^b \leftarrow \{s \in S \mid s[i] = b\}$ for $b \in \{0, 1\}$
  $T^0 \leftarrow$ CONST_DECISION_TREE(S^0)
  $T^1 \leftarrow$ CONST_DECISION_TREE(S^1)
  Let T be tree rooted at v with left-child(v) $\leftarrow T^0$ and right-child(v) $\leftarrow T^1$
else
  $label(v) \leftarrow s$, where $S = \{s\}$ ▷ $|S| = 1$, v *is a leaf*
  Let T be a tree consisting of the singleton v.
Return T

procedure DETERMINE(T)
  $J = \{j \mid \exists u \in T$ s.t. $j = label(u)\}$
  for all $j \in J$ **do** in parallel
    $query(j)$
  Let v be the root of T
  while v is not a leaf **do**
    Let $j = label(v)$.
    if $b_j = 0$ **then**
      Set $v \leftarrow$ left-child(v)
    else
      Set $v \leftarrow$ right-child(v)
  Return $label(v)$

Algorithm 3 2-Round Download with $\beta < 1$; Code for peer μ.

if $k \leq 32(c+1)\ln n/\gamma$ **then** query every bit and return
if $k < \sqrt{n/\gamma}\ln n$ **then** $\varphi \leftarrow \lceil 32(c+1)n\ln n/(\gamma k)\rceil$
else $\varphi \leftarrow \lceil 32(c+1)\sqrt{n/\gamma}\rceil$
$t \leftarrow \gamma k/(2\mathcal{K})$
Randomly select $\ell^\mu \in [1, \mathcal{K}]$
Set string $s^\mu \leftarrow query(\mathbf{X}[\ell^\mu, \varphi])$
Broadcast $\langle \ell^\mu, s^\mu \rangle$
for $\ell = 1$ to $\mathcal{K}$ **do** in parallel
  Construct the multiset $S_\ell \leftarrow \{s \mid \langle \ell, s\rangle$ received$\}$
  $T_\ell \leftarrow$ CONST_DECISION_TREE(FS(S_ℓ, t))
  $s^\ell \leftarrow$ DETERMINE(T_ℓ)
Output $s^1 s^2 \cdots s^{\mathcal{K}}$

B Query complexity analysis of Algorithm 3

The cost of querying in Step 2 for interval ℓ is the number of internal nodes of the decision tree, which is $|\mathsf{FS}_\ell| - 1$. Let x_ℓ be the number of strings received for interval ℓ in Step 1 (including copies). Then $|\mathsf{FS}_\ell| \leq x_\ell/t$. Since each peer sends no more than one string overall, $\sum x_\ell = k$. Hence, the cost of determining all intervals is $\sum_\ell |\mathsf{FS}_\ell| \leq \sum_\ell x_\ell/t \leq k/t$.

Algorithm 4 Download Protocol with $\tilde{O}(n/\gamma k)$ Expected Queries and $O(\log \gamma k)$ Time $\beta < 1$.

for $i = 0$ to $\lceil \lg(n/\varphi) \rceil$ **do**
  Randomly pick an interval $\ell \in [1, \mathcal{K}_i]$ of size φ_i
  if $i = 0$ **then**
    Set string $s = query(\mathbf{X}[\ell, \varphi])$ and broadcast $\langle \ell, s, 0 \rangle$
  else
    $\ell_L \leftarrow 2\ell - 1; \ell_R \leftarrow 2\ell$
    for $u \in \{\ell_L, \ell_R\}$ **do** in parallel
      Construct the multiset $S(u) \leftarrow \{s \mid$ received a message of the form $\langle u, s, i-1 \rangle\}$
      $t_{i-1} = 2^{i-2}\varphi(\gamma k/n)$
      $T_u \leftarrow$ Const_Decision_Tree($\mathsf{FS}(S(u), t_{i-1})$)
      $s_u \leftarrow$ Determine(T_u)
    Set s_ℓ to $s_{\ell_L} s_{\ell_R}$ and broadcast $\langle \ell, s_\ell, i \rangle$
return the determined string for interval $[1, \ldots, n]$

The query cost per peer, $\mathcal{Q}$, is the cost of determining every interval using decision trees plus the initial query cost; hence $\mathcal{Q} \leq k/t + \varphi$. Since $t = \gamma k/(2\mathcal{K})$, $\mathcal{Q} \leq 2\lceil n/\varphi \rceil (1/\gamma) + \varphi$. To satisfy the premise of Claim 7, it is required that $t = \gamma k/(2\mathcal{K}) \geq 8(c+1)\ln n$. To set the value of φ we consider the following cases.

Case 1 ($k \geq \sqrt{n/\gamma}\ln n$). Set $\varphi = \lceil 32(c+1)\sqrt{n/\gamma} \rceil$. Since $t = \frac{\gamma k}{2\lceil \frac{n}{\varphi} \rceil} \geq \frac{\gamma k}{4n/\varphi} \geq 8(c+1)\ln n$, then t satisfies the premise of Claim 7 and $\mathcal{Q} \leq k/t + \varphi = \frac{k}{\frac{\gamma k}{2\lceil \frac{n}{\varphi} \rceil}} + \varphi = \frac{2\lceil n/\varphi \rceil}{\gamma} + \varphi = O(\sqrt{n/\gamma})$.

Case 2 ($32(c+1)\ln n/\gamma < k < \sqrt{n/\gamma}\ln n$). Set $\varphi = \lceil 32(c+1)n\ln n/(\gamma k) \rceil$. Then $t = \frac{\gamma k}{2\lceil \frac{n}{\varphi} \rceil} \geq \frac{\gamma k}{4n/\varphi} \geq 8(c+1)\ln n$, satisfying Claim 7. Also $k/t \leq \sqrt{n/\gamma}\ln n/t = O(\sqrt{n/\gamma})$. Thus, $\mathcal{Q} \leq k/t + \varphi = O(\sqrt{n/\gamma} + \frac{n\ln n}{\gamma k})$.

Case 3 ($k \leq 32(c+1)\ln n/\gamma$). Each peer queries all n bits, resulting in $\mathcal{Q} = n$.

Since the protocol can check to see which case it is in (by inspecting k, n, and γ), the overall query complexity is $\mathcal{Q} = O(\min\{\sqrt{n/\gamma} + n\ln n/(\gamma k), n\})$.

C Proof of Lemma 9

The proof of Lemma 9 follows the next lemma.

▶ **Lemma 13.** *For every step $i \in [0, \lceil \lg(n/\varphi) \rceil]$, every interval in step i (those of size φ_i) is picked by at least t_i honest peers w.h.p*

Proof. Fix Step i. The number of intervals in step i is $\mathcal{K}_i$. The expected number of honest peers which pick a given interval at Step i is $E_i = \gamma k/\mathcal{K}_i = (\gamma k/(n/(2^i\varphi))) \geq 2^i(8(c+1)\ln n)$. Setting $t_i = E_i/2$, the probability of failure for any one interval, as given by Chernoff bounds (see previous subsection) is no more than $e^{-E_i/8} \leq n^{-c+1}$.

Taking a union bound over the $\sum_{i=0}^{\lg(n/\varphi)} \mathcal{K}_i < \sum_{i=0}^{\infty} \lceil \frac{n}{\varphi} \rceil \frac{1}{2^i} < n$ intervals over all steps i, the probability of any failure in any step is less than $n \cdot n^{-(c+1)} \leq n^{-c}$ for $c \geq 1$. ◀

Proof of Lemma 9. By induction on the steps. The base case, step 0, is trivial since every honest peer that picks an interval queries it completely.

For step $i > 0$, consider the interval $\ell = \ell_\mu^i$ picked by peer μ in step i. During step i, μ splits ℓ into two subintervals ℓ_L, ℓ_R of size φ_{i-1}. By Lemma 13, the intervals ℓ_L, ℓ_R were each picked by a least t_{i-1} honest peers w.h.p. during step $i-1$, and by the inductive hypothesis, those peers know (and broadcast) the correct strings s_L, s_R respectively. For $u \in \{L, R\}$, let $\mathsf{FS}_u = \mathsf{FS}(S(\ell_u), t_{i-1})$ denote the t_{i-1} frequent sets constructed in step i for ℓ_u. Then, $s_L \in \mathsf{FS}_L$ and $s_R \in \mathsf{FS}_R$. Hence, the decision trees built for ℓ_L and ℓ_R will contain leaves with labels s_L, s_R respectively, which will then be returned correctly by Procedure Determine, implying that μ learns the correct bit string for the interval ℓ, $s_\ell = s_L s_R$. ◀

D Trade-off among time, queries, and message size

Here we use information theory to prove a lower bound on the trade-offs among these parameters, as in [16].

▶ **Lemma 14.** *A lower bound on the trade-off among time t, number of non-faulty peers γk, number of queries (per peer) Q and maximum message size (per round) ϕ for a deterministic protocol is given by the following:* $(\gamma k t)[\lceil \lg \phi \rceil + \phi] + Q \geq n$.

Proof. We observe that there are 2^n possible outputs; hence n bits of information must be communicated to any peer. Assume that the adversarially controlled peers do not send any messages. Since an honest peer receives no more than ϕ bits (per round) from each of the other $\gamma k - 1$ honest peers over t rounds, the information received by the peer can be represented by a sequence of $(\gamma k - 1)t$ consecutive intervals, one for each message received, ordered by peer identity and time. Each interval contains the bits of the message, preceded by a sequence of $\lceil \lg \phi \rceil$ bits giving the exact number of bits in the message. Note that the $\lceil \lg \phi \rceil$ bits that represent the exact number of bits in the message are required to be able to know where every interval starts and ends. Hence, the total information received by a peer can be described by no more than $((\gamma k - 1)t)(\lceil \lg \phi \rceil + \phi) + Q$ bits, and this must be at least n. ◀

▶ **Corollary 15.** *If $((\gamma k - 1)t)(\lceil \lg \phi \rceil + \phi) + Q \leq n - 1$ then a deterministic protocol outputs incorrect results for at least half of the possible 2^n possible inputs.*

For randomized protocols: Assume the randomness is global. Any random protocol can be viewed as tossing the random coins first and then executing a deterministic protocol selected by the random coins.

▷ Claim 16. Let A be a randomized protocol. If all the information that the peer learns which depends on the input can be described in a sequence of only $n-1$ bits, then there is an input X such that $Pr(A(X) = X) \leq 1/2$.

Proof of Claim. Suppose to the contrary that for every input X, $Pr(A(X) = X) > 1/2$. Let p_i be the probability that A chooses deterministic protocol A_i. Let $1_{X,i} = 1$ if $A_i(X) = X$. Then $\sum_{X \in \{0,1\}^n} \sum_i p_i 1_{X,i} > 2^n * (1/2) = 2^{n-1}$.

But from the Corollary above, $|\{X \mid A_i(X) = X\}| \leq 2^{n-1}$. Rearranging the terms,

$$\sum_{X \in \{0,1\}^n} \sum_i p_i 1_{X,i} = \sum_i p_i \sum_X 1_{X,i} \leq \sum_i p_i 2^{n-1} = 2^{n-1},$$

giving a contradiction. ◁

It is easy to see that if one assumes global randomness, then the description length is the same as what's needed for a deterministic protocol. Hence, the same lower bound on the trade-off applies. That is,

▶ **Theorem 17.** *Suppose there is a randomized protocol with time t, number of non-faulty peers γk, number of queries (per peer) Q, and maximum message size (per round) ϕ. If on every input, its output is correct with a probability greater than 1/2, then*

$$((\gamma kt) - 1)[\lceil \lg \phi \rceil + \phi] + Q \geq n - 1.$$

Note that this holds true for a protocol with public randomness and a fortiori, with private randomness. Furthermore, the same analysis holds in the Broadcast model. Hence the result of Theorem 12 is optimal up to $\log n$ factors in terms of the trade-off among queries, time, and message size.