Some Lower Bounds in Dynamic Networks with Oblivious Adversaries*†

Irvan Jahja[1], Haifeng Yu[2], and Yuda Zhao[‡3]

1 **National University of Singapore, Singapore**
irvan@comp.nus.edu.sg
2 **National University of Singapore, Singapore**
haifeng@comp.nus.edu.sg
3 **Grab, Singapore**
yudazhao@gmail.com

Abstract

This paper considers several closely-related problems in synchronous dynamic networks with *oblivious adversaries*, and proves novel $\Omega(d + \text{poly}(m))$ lower bounds on their time complexity (in rounds). Here d is the dynamic diameter of the dynamic network and m is the total number of nodes. Before this work, the only known lower bounds on these problems under oblivious adversaries were the trivial $\Omega(d)$ lower bounds. Our novel lower bounds are hence the first non-trivial lower bounds and also the first lower bounds with a $\text{poly}(m)$ term. Our proof relies on a novel reduction from a certain two-party communication complexity problem. Our central proof technique is unique in the sense that we consider the communication complexity with a special *leaker*. The leaker helps Alice and Bob in the two-party problem, by disclosing to Alice and Bob certain "non-critical" information about the problem instance that they are solving.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases dynamic networks, oblivious adversary, adaptive adversary, lower bounds, communication complexity

Digital Object Identifier 10.4230/LIPIcs.DISC.2017.29

1 Introduction

Dynamic networks [22] is a flourishing topic in recent years. We consider a synchronous setting where the m (fixed) nodes in the network proceed in synchronous rounds. Each node has a unique id of size $O(\log m)$, and the messages are of size $O(\log m)$ as well. The nodes never fail. The topology of the dynamic network can change from round to round, as determined by an *adversary*, subject to the only constraint that the topology in each round must be a connected and undirected graph. The *time complexity* of a protocol is the number of rounds needed for all nodes to generate the final output, over the worst-case adversary, worst-case initial values, and average coin flips of the protocol. We consider a number of fundamental distributed computing problems within such a context:

* The authors of this paper are alphabetically ordered. This work is partly supported by the research grant MOE2014-T2-2-030 from Singapore Ministry of Education Academic Research Fund Tier-2.
† The full version of this paper is available at http://www.comp.nus.edu.sg/~yuhf/oblivious-disc17-technicalreport.pdf.
‡ This work was done while this author was in National University of Singapore.

- Consensus: Each node has a binary input. The nodes aim to achieve a consensus (with the standard agreement, validity, and termination requirements) and output the final decision.
- LeaderElect: Each node should output the leader's id.
- ConfirmedFlood: A certain node ν aims to propagate a token of size $O(\log m)$ to all other nodes, and wants to further confirm that all nodes have received the token.[1] Formally, node ν's output is correct only if by the time that ν outputs, the token has already been received by all the nodes. (The value of the output is not important.) The remaining nodes can output any time.
- Aggregation: Each node has a value of $O(\log m)$ bits, and the nodes aim to compute a certain aggregation function over all these values. We consider two specific aggregation functions, Sum and Max.

Let d be the *(dynamic) diameter* (see definition later) of the dynamic network. (Note that since the topology is controlled by an adversary, the protocol never knows d beforehand.) Given an optimal protocol for solving any of the above problems, let $\mathrm{tc}(d, m)$ denote the protocol's time complexity, when it runs over networks with d diameter and m nodes. It is easy to see that $\mathrm{tc}(d, m)$ crucially depends on d, since we trivially have $\mathrm{tc}(d, m) = \Omega(d)$. Given such, this paper focus on the following central question:

Ignoring polylog(m) terms, is tc(d, m) independent of the network size m?

Answering this fundamental question will reveal whether the complexity of all these basic problems is due to the diameter or due to both the diameter and the network size.

Existing results. If the network were *static*, then building a spanning tree would solve all these problems in either $O(d)$ or $O(d \log m)$ rounds, implying a **yes** answer to the above question. In dynamic networks, the picture is more complex. In a dynamic network model without congestion (i.e., message size unlimited), Kuhn et al. [20] have proposed elegant upper bound protocols with $O(d)$ complexity for all these problems. Hence the answer is **yes** as well. For dynamic networks with congestion (i.e., message size limited to $O(\log m)$), Yu et al. [25] recently have proved that $\mathrm{tc}(d, m) = O(d \log m)$ for Consensus and LeaderElect, if the nodes know a *good* estimate on m.[2] Hence the answer is **yes** in such cases. On the other hand, if nodes' estimate on m is *poor*,[3] then Yu et al. [25] prove a lower bound of $\Omega(d + \mathrm{poly}(m))$ for Consensus and LeaderElect, implying a **no** answer. For ConfirmedFlood and Aggregation, they have also proved $\mathrm{tc}(d, m) = \Omega(d + \mathrm{poly}(m))$, even if the nodes know m. This implies a **no** answer for those two problems.

All the lower bound proofs in [25], however, critically relies on a powerful *adaptive adversary*: In each round, the adaptive adversary sees all the coin flip outcomes so far of the protocol $\mathcal{P}$ and manipulates the topology based on those. In particular, in each round the adversary sees whether each node will be sending (and can then manipulate the topology accordingly), *before* the nodes actually send their messages. Their proof breaks

[1] Such confirmation does not have to come from explicit acknowledgements, and can be via implicit means, such as counting the number of rounds.

[2] More precisely, if the nodes know m' such that $|\frac{m'-m}{m}| \leq \frac{1}{3} - c$ for some positive constant c. Obviously, this covers the case where the nodes know m itself.

[3] More precisely, if the nodes only knows m' such that $|\frac{m'-m}{m}|$ reaches $\frac{1}{3}$ or above. Obviously, this covers the case where the nodes do not have any knowledge about m.

under *oblivious adversaries*, which do not see $\mathcal{P}$'s coin flip outcomes and have to decide the topologies in all the rounds before $\mathcal{P}$ starts.[4]

In summary, our central question of whether $\text{tc}(d, m)$ is largely independent of the network size m has been answered in:

(i) static networks,
(ii) dynamic networks without congestion under both adaptive and oblivious adversaries, and
(iii) dynamic networks with congestion under adaptive adversaries.

Our results. This work gives the last piece of the puzzle for answering our central question. Specifically, we show that in dynamic networks with congestion and under oblivious adversaries, for Consensus and LeaderElect, the answer to the question is **no** when the nodes' estimate on m is poor. (If the nodes' estimate on m is good, results from [25] already implied a **yes** answer.) Specifically, we prove a novel $\Omega(d + \text{poly}(m))$ lower bound on Consensus under oblivious adversaries, when the nodes' estimate on m is poor. This is the first non-trivial lower bound and also the first lower bound with a $\text{poly}(m)$ term, for Consensus under oblivious adversaries. The best lower bound before this work was the trivial $\Omega(d)$ lower bound. Our Consensus lower bound directly carries over to LeaderElect since Consensus reduces to LeaderElect [25].

Our approach will also lead to a $\Omega(d + \text{poly}(m))$ lower bound under oblivious adversaries for ConfirmedFlood, which in turn reduces to Sum and Max [25]. Such a lower bound similarly gives a **no** answer for ConfirmedFlood and Aggregation. But since the lower bound proof for ConfirmedFlood is similar to and in fact easier than our Consensus proof, for clarity, we will not separately discuss it in this paper.

Different adversaries. In dynamic networks, different kinds of adversaries often require different algorithmic techniques and also yield different results. Hence it is common for researchers to study them separately. For example, lower bounds for information dissemination were proved separately, under adaptive adversaries [13] and then later under oblivious adversaries [1]. Dynamic MIS was investigated separately under adaptive adversaries [17] and later under oblivious adversaries [8]. Broadcasting was first studied under adaptive adversaries [18], and later under oblivious adversaries [14].

Our approach. Our novel Consensus lower bound under oblivious adversaries is obtained via a reduction from a two-party communication complexity (CC) problem called *Gap Disjointness with Cycle Promise* or Gdc. Our reduction partly builds upon the reduction in [25] for adaptive adversaries, but has two major differences. In fact, these two novel aspects also make our central proof technique rather unique, when compared with other works that use reductions from CC problems [9, 12, 21].

The first novel aspect is that we reduce from Gdc with a special *leaker* that we design. The leaker is an oracle in the Gdc problem, and is separate from the two parties Alice and Bob . It helps Alice and Bob, by disclosing to them certain "non-critical" information in the following way. For a CC problem Π, let $\Pi_n(X, Y)$ be the answer to Π for length-n inputs X and Y. Let x_i and y_i denote the i-th character of X and Y, respectively. A pair (a, b) is defined to be a *leakable pair* if for all n, X, Y, and $i \in [0, n]$, $\Pi_n(x_1 x_2 \dots x_n, y_1 y_2 \dots y_n) =$

[4] Note however that all upper bounds, from [20] and [25], will directly carry over to oblivious adversaries.

$\Pi_{n+1}(x_1x_2...x_iax_{i+1}x_{i+2}...x_n, y_1y_2...y_iby_{i+1}y_{i+2}...y_n)$. Intuitively, inserting or removing a leakable pair does not impact the answer to Π. For each index i where (x_i, y_i) is leakable, independently with probability $\frac{1}{2}$, our leaker *leaks* the index i, by letting both Alice and Bob know for free the value of i and the value of the pair (x_i, y_i), before Alice and Bob start running their protocol.

Our reduction from GDC (with our leaker) to CONSENSUS still does not allow us to directly use an oblivious adversary. Instead, as the second novel aspect, we will use a special kind of adaptive adversaries which we call *sanitized adaptive adversaries*. These adversaries are still adaptive, but their adaptive decisions have been "sanitized" by taking XOR with independent coin flips. We then show that a sanitized adaptive adversary is no more powerful than an oblivious adversary, in terms of incurring the cost of a protocol.

2 Related Work

This section discusses related works beyond those already covered in the previous section.

Related work on Consensus and LeaderElect. Given the importance of CONSENSUS and LEADERELECT in dynamic networks, there is a large body of related efforts and we can only cover the most relevant ones. In dynamic networks without congestion, Kuhn et al. [20] show that the *simultaneous consensus* problem has a lower bound of $\Omega(d + \text{poly}(m))$ round. In this problem, the nodes need to output their consensus decisions simultaneously. Their knowledge-based proof exploits the need for simultaneous actions, and does not apply to our setting. Some other researchers (e.g., [3, 4]) have studied CONSENSUS and LEADERELECT in a dynamic network model where the set of nodes can change and where the topology is an *expander*. Their techniques (e.g., using random walks) critically reply on the expander property of the topology, and hence do not apply to our setting. Augustine et al. [2] have proved an upper bound of $O(d \log m)$ for LEADERELECT in dynamic networks while assuming d is known to all nodes. This does not contradict with our lower bound, since we do not assume the knowledge of d. Certain CONSENSUS and LEADERELECT protocols (e.g., [15]) assume that the network's topology eventually stops changing, which is different from our setting where the change does not stop. CONSENSUS and LEADERELECT have also been studied in *directed* dynamic networks (e.g., [11, 23]), which are quite different from our undirected version. In particular, lower bounds there are mostly obtained by exploiting the lack of guaranteed bidirectional communication in directed graphs. Our AGGREGATION problem considers the two aggregation functions SUM and MAX. Cornejo et al. [10] considers a different aggregation problem where the goal is to collect distributed tokens (without combining them) to a small number of nodes. Some other research (e.g., [6]) on AGGREGATION assumes that the topology is each round is a (perfect) matching, which is different from our setting where the topology must be connected.

Related work on reductions from CC. Reducing from two-party CC problems to obtain lower bounds for distributed computing problem has been a popular approach in recent years. For example, Kuhn et al. [21] and Das Sarma et al.[12] have obtained lower bounds on the *hear-from* problem and the *spanning tree verification* problem, respectively, by reducing from DISJOINTNESS. In particular, Kuhn et al.'s results suggest that the *hear-from* problem has a lower bound of $\Omega(d + \sqrt{m}/\log m)$ in *directed static networks*. Chen et al.'s work [9] on computing SUM in *static networks with node failures* has used a reduction from the $\text{GDC}_n^{1,q}$ problem. Our reduction in this paper is unique, in the sense that none of these previous reductions use the two key novel techniques in this work, namely CC with our leaker and sanitized adaptive adversaries.

Related work on CC. To the best of our knowledge, we are the first to exploit the CC with a leaker in reductions to distributed computing problems such as Consensus. Our leaker serves to allow oblivious adversaries. Quite interestingly, for completely different purposes, the notions of leakable pairs and a leaker have been extensively (but implicitly) used in proofs for obtaining direct sum results on the information complexity (IC) (e.g., [5, 7, 24]) of various communication problems: First, leakable pairs have been used to construct a *collapsing input*, for the purpose of ensuring that the answer to the problem Π is entirely determined by (x_i, y_i) at some index i. Second, an (implicit) leaker has often been used (e.g., in [7, 24]) to enable Alice and Bob to draw $(\mathbf{X}, \mathbf{Y})$ from a non-product distribution.

Because of the fundamentally different purposes of leaking, our leaker differs from those (implicit) leakers used in works on IC, in various specific aspects. For example in our work, all leakable pairs are subject to leaking, while in the works on IC, there is some index i that is never subject to leaking. Also, when our leaker leaks index j, it discloses both $\mathbf{x}_j$ and $\mathbf{y}_j$ to both Alice and Bob. In comparison, in works on IC, the (implicit) leaking is usually done differently: For example, Alice and Bob may use public coins to draw $\mathbf{x}_j$ and Bob may use his private coins to draw $\mathbf{y}_j$. Doing so (implicitly) discloses $\mathbf{x}_j$ to both Alice and Bob and (implicitly) discloses $\mathbf{y}_j$ *only* to Bob.

A key technical step in our work is to prove a lower bound on the CC of $\textsc{Gdc}_n^{g,q}$ with our leaker. For simpler problems such as Disjointness (which is effectively $\textsc{Gdc}_n^{1,2}$), we believe that such a lower bound could alternatively be obtained by studying its IC with our leaker. But the gap promise and the cycle promise in $\textsc{Gdc}_n^{g,q}$ make IC arguments rather tricky. Hence we will (in Section 8) obtain our intended lower bound by doing a direct reduction from the CC of $\textsc{Gdc}_{n'}^{g,q}$ without the leaker to the CC of $\textsc{Gdc}_n^{g,q}$ with the leaker.

3 Model and Definitions

Conventions. All protocols in this paper refer to Monte Carlo randomized algorithms. We always consider public coin protocols, which makes our lower bounds stronger. All log is base 2, while ln is base e. Upper case fonts (e.g., X) denote strings, vectors, sets, etc. Lower case fonts (e.g., x) denote scalar values. In particular, if X is a string, then x_i means the i-th element in X. Bold fonts (e.g., $\mathbf{X}$ and $\mathbf{x}$) refer to random variables. Blackboard bold fonts (e.g., $\mathbb{D}$) denote distributions. We write $\mathbf{x} \sim \mathbb{D}$ if $\mathbf{x}$ follows the distribution $\mathbb{D}$. Script fonts (e.g., $\mathcal{P}$ and $\mathcal{Q}$) denote either protocols or adversaries.

Dynamic networks. We consider a synchronous dynamic network with m fixed nodes, each with a unique id of $\Theta(\log m)$ bits. A protocol in such a network proceeds in synchronous rounds, and starts executing on all nodes in round 1. (Clearly such simultaneous start makes our lower bound stronger.) In each round, each node v first does some local computation, and then chooses to either send a single message of $O(\log m)$ size or receive. All nodes who are v's neighbors in that round and are receiving in that round will receive v's message at the end of the round. A node with multiple neighbors may receive multiple messages.

The topology of the network may change arbitrarily from round to round, as determined by some *adversary*, except that the topology in each round must be a connected undirected graph. (This is the same as the 1-interval model [19].) A node does not know the topology in a round. It does not know its neighbors either, unless it receives messages from them in that round. Section 1 already defined *oblivious adversaries* and *adaptive adversaries*. In particular in each round, an adaptive adversary sees all $\mathcal{P}$'s coin flip outcomes up to and including the current round, and manipulates the topology accordingly, before $\mathcal{P}$ uses the current round's coin flip outcomes.

We use the standard definition for the *(dynamic) diameter* [22] of a dynamic network: Intuitively, the diameter of a dynamic network is the minimum number of rounds needed for every node to influence all other nodes. Formally, we say that $(\omega, r) \to (v, r+1)$ if either ω is v's neighbor in round r or $\omega = v$. The *diameter* d of a dynamic network is the smallest d such that $(\omega, r) \rightsquigarrow (v, r+d)$ for all ω, v, and r, where "$\rightsquigarrow$" is the transitive closure of "$\to$". Since the topology is controlled by an adversary, a protocol never knows d beforehand.

Communication complexity. In a two-party communication complexity (CC) problem Π_n, Alice and Bob each hold input strings X and Y respectively, where each string has n *characters*. A character here is q-ary (i.e., an integer in $[0, q-1]$) for some given integer $q \geq 2$. For any given i, we sometimes call (x_i, y_i) as a *pair*. Alice and Bob aim to compute the value of the binary function $\Pi_n(X, Y)$. Given a protocol $\mathcal{P}$ for solving Π (without a leaker), we define $\text{cc}(\mathcal{P}, X, Y, \mathbf{C}_\mathcal{P})$ to be the communication incurred (in terms of number of bits) by $\mathcal{P}$, under the input (X, Y) and $\mathcal{P}$'s coin flip outcomes $\mathbf{C}_\mathcal{P}$. Note that $\mathbf{C}_\mathcal{P}$ is a random variable while cc() is a deterministic function. We similarly define $\text{err}(\mathcal{P}, X, Y, \mathbf{C}_\mathcal{P})$, which is 1 if $\mathcal{P}$'s output is wrong, and 0 otherwise. We define the *communication complexity* of $\mathcal{P}$ to be $\text{cc}(\mathcal{P}) = \max_X \max_Y E_{\mathbf{C}_\mathcal{P}}[\text{cc}(\mathcal{P}, X, Y, \mathbf{C}_\mathcal{P})]$, and the *error* of $\mathcal{P}$ to be $\text{err}(\mathcal{P}) = \max_X \max_Y E_{\mathbf{C}_\mathcal{P}}[\text{err}(\mathcal{P}, X, Y, \mathbf{C}_\mathcal{P})]$. We define the δ-error $(0 < \delta < \frac{1}{2})$ *communication complexity* of Π_n to be $\mathfrak{R}_\delta(\Pi_n) = \min \text{cc}(\mathcal{P})$, with the minimum taken over all $\mathcal{P}$ where $\text{err}(\mathcal{P}) \leq \delta$. For convenience, we define $\mathfrak{R}_\delta(\Pi_0) = 0$ and $\mathfrak{R}_\delta(\Pi_a) = \mathfrak{R}_\delta(\Pi_{\lfloor a \rfloor})$ for non-integer a.

We define similar concepts for CC with our leaker. Section 1 already defined leakable pairs and how our leaker works. Given $\mathcal{P}$ for solving Π with our leaker, $\text{cc}(\mathcal{P}, X, Y, \mathbf{C}_\mathcal{P}, \mathbf{C}_\mathcal{L})$ is the communication incurred by $\mathcal{P}$, under the input (X, Y), $\mathcal{P}$'s coin flip outcomes $\mathbf{C}_\mathcal{P}$, and the leaker's coin flip outcomes $\mathbf{C}_\mathcal{L}$. Here (X, Y) and $\mathbf{C}_\mathcal{L}$ uniquely determine which indices get leaked. We define $\text{cc}(\mathcal{P}) = \max_X \max_Y E_{\mathbf{C}_\mathcal{L}} E_{\mathbf{C}_\mathcal{P}}[\text{cc}(\mathcal{P}, X, Y, \mathbf{C}_\mathcal{P}, \mathbf{C}_\mathcal{L})]$. We similarly define $\text{err}(\mathcal{P}, X, Y, \mathbf{C}_\mathcal{P}, \mathbf{C}_\mathcal{L})$ and $\text{err}(\mathcal{P})$. Finally, we define the δ-error $(0 < \delta < \frac{1}{2})$ *communication complexity* of Π_n with our leaker, denoted as $\mathfrak{L}_\delta(\Pi_n)$, to be $\mathfrak{L}_\delta(\Pi_n) = \min \text{cc}(\mathcal{P})$, with the minimum taken over all $\mathcal{P}$ such that $\mathcal{P}$ solves Π_n with our leaker and $\text{err}(\mathcal{P}) \leq \delta$. Note that we always have $\mathfrak{L}_\delta(\Pi_n) \leq \mathfrak{R}_\delta(\Pi_n)$.

4 Preliminaries on Gap Disjointness with Cycle Promise

The section defines the two-party GDC problem and describes some basic properties of GDC.

▶ **Definition 1** (Gap Disjointness with Cycle Promise)**.** In *Gap Disjointness with Cycle Promise*, denoted as $\text{GDC}_n^{g,q}$, Alice and Bob have input strings X and Y, respectively. X and Y each have n characters, and each character is an integer in $[0, q-1]$. Alice and Bob aim to compute $\text{GDC}_n^{g,q}(X, Y)$, defined to be 1 if (X, Y) contains no $(0, 0)$ pair, and 0 otherwise. The problem comes with the following two promises:

- **Gap promise:** (X, Y) contains either no $(0, 0)$ pair or at least g such pairs.
- **Cycle promise [9]:** For each index i, x_i and y_i satisfy exactly one of the following four conditions: i) $x_i = y_i = 0$, ii) $x_i = y_i = q-1$, iii) $x_i = y_i + 1$, or iv) $x_i = y_i - 1$.

One can easily verify that the cycle promise is trivially satisfied when $q = 2$. It is also easy to see $\text{GDC}_n^{1,2}$ degenerates to the classic DISJOINTNESS problem. The gap promise and the cycle promise start to impose material restrictions when $g \geq 2$ and $q \geq 3$, respectively. For example for $g = 2$ and $q = 4$, $X = 02103$ and $Y = 03003$ satisfy both the two promises, where (X, Y) contains 2 pairs of $(0, 0)$, at indices 1 and 4. For GDC, all $(0, 0)$ pairs are

non-leakable, while all other pairs are leakable. For example for $X = 02103$ and $Y = 03003$, those 3 pairs at index 2, 3, and 5 are leakable. The following result (proven in the full version [16] of this paper) on the CC of GDC is a simple adaption from the result in [9]:

▶ **Theorem 2.** *For any δ where $0 < \delta < 0.5$, there exist constants $c_1 > 0$ and $c_2 > 0$ such that for all n, g, and q, $\mathfrak{R}_\delta(\text{GDC}_n^{g,q}) \geq \frac{c_1 n}{gq^2} - c_2 \log \frac{n}{g}$.*

The proof of Theorem 2 also showed that $\mathfrak{R}_\delta(\text{GDC}_n^{g,q}) \geq \mathfrak{R}_\delta(\text{GDC}_{n/g}^{1,q})$. It is important to note that $\mathfrak{L}_\delta(\text{GDC}_n^{g,q}) \geq \mathfrak{L}_\delta(\text{GDC}_{n/g}^{1,q})$ does *not* hold in general (see [16] for more discussion). Hence when later proving the lower bound on $\mathfrak{L}_\delta(\text{GDC}_n^{g,q})$, we will have to work with the gap promise directly, instead of obtaining the lower bound via $\mathfrak{L}_\delta(\text{GDC}_{n/g}^{1,q})$.

5 Review of Existing Proof under Adaptive Adversaries

This section gives an overview of the recent CONSENSUS lower bound proof [25] under adaptive adversaries. That proof is quite lengthy and involved, hence we will stay at the high-level, while focusing on aspects that are more relevant to this paper.

Overview. Consider any oracle CONSENSUS protocol $\mathcal{P}$ with $\frac{1}{10}$ error. Let $\text{tc}(d, m)$ be $\mathcal{P}$'s time complexity, when running over dynamic network controlled by adaptive adversaries and with d diameter and m nodes. The proof in [25] is mainly for proving $\text{tc}(8, m) = \Omega(\text{poly}(m))$. The proof trivially extends to $\text{tc}(d, m)$ for all $d \geq 8$. Combining with the trivial $\Omega(d)$ lower bound will lead to the final lower bound of $\Omega(d + \text{poly}(m))$.

To prove $\text{tc}(8, m) = \Omega(\text{poly}(m))$, [25] uses a reduction from $\text{GDC}_n^{g,q}$ to CONSENSUS. To solve $\text{GDC}_n^{g,q}(X, Y)$, Alice knowing X and Bob knowing Y simulate the CONSENSUS protocol $\mathcal{P}$ in the following way: In the simulation, the input (X, Y) is mapped to a dynamic network. Roughly speaking, if $\text{GDC}_n^{g,q}(X, Y) = 1$, the resulting dynamic network will have a diameter of 8. Hence $\mathcal{P}$ should decide within $r_1 = \text{tc}(8, m)$ rounds on expectation. If $\text{GDC}_n^{g,q}(X, Y) = 0$, then the resulting dynamic network will have a diameter of roughly $\frac{q}{2}$. It is then shown [25] that $\mathcal{P}$ must take $r_2 = \Omega(q)$ rounds to decide in dynamic networks with such a diameter. The value of q is chosen, as a function of $\text{tc}(8, m)$, such that $r_2 > 10r_1$. Alice and Bob determine the answer to GDC based on when $\mathcal{P}$ decides: If $\mathcal{P}$ decides within $10r_1$ rounds, they claim that $\text{GDC}_n^{g,q}(X, Y) = 1$. Otherwise they claim $\text{GDC}_n^{g,q}(X, Y) = 0$.

To solve GDC using the above simulation, Alice and Bob need to simulate $\mathcal{P}$ for $10r_1 = 10\text{tc}(8, m)$ rounds. In each round, to enable the simulation to continue, Alice and Bob will need to incur $O(\log m)$ bits of communication. Hence altogether, they incur $10\text{tc}(8, m) \cdot O(\log m)$ bits for solving $\text{GDC}_n^{g,q}$. The lower bound on the CC of $\text{GDC}_n^{g,q}$ then immediately translates to a lower bound on $\text{tc}(8, m)$.

Crux of the proof. When solving GDC, Alice only knows X and not Y. This means that Alice does *not* actually have the full knowledge of the dynamic network, which is a function of (X, Y). Hence the proof's central difficulty is to design the dynamic network in such a way that Alice can nevertheless still properly simulate $\mathcal{P}$ over that dynamic network. The proof in [25] overcomes this key difficulty by i) leveraging the cycle promise in GDC, and ii) using an *adaptive* adversary — in particularly, using an adaptive adversary is highlighted [25] as a key technique. We give a concise review below.

Given (X, Y), the dynamic network constructed in [25] has one *chain* for each index $i \in [1, n]$. Each chain has 3 node in a line (Figure 1). Consider as an example the i-th chain where $x_i = 0$. Since $x_i = 0$, y_i must be either 0 or 1 (by the cycle promise). The set of edges

(a) ν is sending in round $t_i + 1$ **(b)** ν is receiving in round $t_i + 1$

Figure 1 The adaptive decisions of the adversary in [25].

on this chain will be different depending on whether y_i is 0 or 1 — this serves to make the diameter of the dynamic network different when $\text{GDC} = 1$ and when $\text{GDC} = 0$, as discussed earlier. The difficulty for Alice, is that she does not know y_i, and hence does not know the exact set of edges on this chain. This prevents her from properly simulating those nodes that she need to simulate for this chain. Similar difficulty applies to Bob.

To overcome this difficulty, if a pair (x_i, y_i) is not $(0, 0)$, the adversary in [25] will make an adaptive decision for manipulating the edges on the i-th chain,[5] to help enable Alice (and also Bob) to simulate. The cycle promise already tells us that for given x_i (e.g., 0), there are two possibilities for y_i (e.g., 0 and 1). The adaptive decisions of the adversary will have the following end effects: Under the topology resulted from such adaptive decisions, the behavior of those nodes that Alice needs to simulate will depend only on x_i and no longer depend on y_i. A similar property holds for Bob.

The details on why those adaptive decisions can achieve such end effects are complex, and are related to the fundamental fact that a node does not know its neighbors in a round until it receives messages from them. At the same time, those details are entirely orthogonal to this work. Hence due to space limitations, we refer interested readers to [25] for such details. Here we will only describe the specifics of all the adaptive decisions made by the adversary, which is needed for our later discussion: Consider any i where (x_i, y_i) is not $(0, 0)$. At the beginning of round $t_i + 1$ where t_i is some function of x_i and y_i, the adversary examines the coin flip outcomes of $\mathcal{P}$ and determines whether the middle node ν on the i-th chain is sending or receiving in round $t_i + 1$ (see Figure 1). If ν is sending, the adversary removes a certain edge e that is incidental to ν, immediately in round $t_i + 1$. Otherwise the adversary will remove the edge e in round $t_i + 2$. Except these adaptive decisions, the adversary does not make any other adaptive decisions. In particular, the adversary does not need to make adaptive decisions for chains corresponding to $(0, 0)$.

6 Roadmap for Lower Bound Proof under Oblivious Adversaries

This section provides the intuition behind, and the roadmap for, our novel proof of Consensus lower bound under oblivious adversaries. To facilitate discussion, we define a few simple concepts. Consider the i-th chain in the previous section where (x_i, y_i) is not $(0, 0)$, and the middle node ν on the chain. We define binary random variable $\mathbf{z} = 0$ if ν is sending in round $t_i + 1$, and define $\mathbf{z} = 1$ otherwise. We use $\mathcal{A}'$ to denote the adaptive adversary described in the previous section. We define $\lambda_{\mathcal{A}'}$ to be the adaptive decision made by $\mathcal{A}'$, where $\mathcal{A}'$ removes the edge e in round $t_i + 1 + \lambda_{\mathcal{A}'}$. With these concepts, $\mathcal{A}'$ essentially sets its decision $\lambda_{\mathcal{A}'}$ to be $\lambda_{\mathcal{A}'} = \mathbf{z}$.

[5] In the actual proof, the adversary only needs to make adaptive decisions for a subset (usually a constant fraction) of such chains. But it is much easier to understand if we simply let the adversary make an adaptive decision on all of them. Doing so has no impact on the asymptotic results.

Making guesses. $\mathcal{A}'$ is adaptive since $\lambda_{\mathcal{A}'}$ depends on $\mathbf{z}$, and $\mathbf{z}$ in turn is a function of $\mathcal{P}$'s coin flips. An oblivious adversary $\mathcal{A}$ cannot have its decision $\lambda_{\mathcal{A}}$ depend on $\mathbf{z}$. At the highest level, our idea of allowing $\mathcal{A}$ in the reduction is simple: We let $\mathcal{A}$ make a blind guess on whether ν is sending. Specifically, imagine that $\mathcal{A}$ itself flips a fair coin $\mathbf{c}$, and then directly set its decision to be $\lambda_{\mathcal{A}} = \mathbf{c}$. Same as $\mathcal{A}'$, $\mathcal{A}$ still removes the edge e in round $t_i + 1 + \lambda_{\mathcal{A}}$, except that now $\lambda_{\mathcal{A}} = \mathbf{c}$. Some quick clarifications will help to avoid confusion here. First, such a guess $\mathbf{c}$ may be either correct (i.e., $\mathbf{c} = \mathbf{z}$) or wrong (i.e., $\mathbf{c} = \bar{\mathbf{z}}$). $\mathcal{A}$ itself cannot tell whether the guess is correct, since $\mathcal{A}$ (being oblivious) does not know $\mathbf{z}$. Alice and Bob, however, can tell if the guess is correct, because they are simulating both the protocol $\mathcal{P}$ and the adversary $\mathcal{A}$, and hence know both $\mathbf{z}$ and $\mathbf{c}$. But they cannot interfere with the guess even if they know it is wrong.

Now if the guess is correct, then the decision of $\mathcal{A}$ will be exactly the same as $\mathcal{A}'$, and everything will work out as before. But if the guess is wrong, then $\mathcal{A}$ can no longer enable Alice to simulate without knowing Y. More specifically, if the guess is wrong, then for the i-th chain, the behavior of those nodes that Alice needs to simulate will depend on the value of y_i, and Alice does not know y_i. To overcome this main obstacle, our key idea is to add a special *leaker* entity in the two-party CC problem, which should be viewed as an oracle that is separate from Alice and Bob. If the guess is wrong for the i-th chain, the leaker will disclose for free to Alice and Bob the pair (x_i, y_i). The knowledge of y_i then immediately enables Alice to infer the exact behavior of the nodes that she needs to simulate. Similar arguments apply to Bob.

Roadmap. There are two non-trivial technical issues remaining in the above approach: i) when to make guesses, and ii) how the leaker impacts the CC of GDC. Overcoming them will be the main tasks of Section 7 and 8, respectively. Section 9 will present our final CONSENSUS lower bound, whose lengthy and somewhat tedious proof is deferred to the full version [16] of this paper.

7 Sanitized Adaptive Adversaries

The difficulty. It turns out that it does not quite work for Alice and Bob to approach the leaker for help when they feel needed. Consider the following example $\text{GDC}_6^{2,4}$ instance with $X = 000000$ and $Y = 111100$. As explained in Section 5, the dynamic network corresponding to this instance has six chains. For all i, we say that the i-th chain is an "$|_b^a$ chain" if $x_i = a$ and $y_i = b$. The first four chains in the dynamic network are thus all $|_1^0$ chains, while the remaining two are $|_0^0$ chains. The adaptive adversary $\mathcal{A}'$ in [25] (see Section 5) will make adaptive decisions for all $|_1^0$ chains, but does not need to do so for $|_0^0$ chains. Applying the idea from Section 6, the oblivious adversary $\mathcal{A}$ should thus make guesses for those four $|_1^0$ chains. Note that $\mathcal{A}$ needs to be simulated by Alice and Bob. The difficulty is that Alice does not know for which chains a guess should be made, since she does not know which chains are $|_1^0$ chains. In fact if she knew, she would have already solved GDC in this instance. Similar arguments apply to Bob.

A naive fix is to simply make a guess for each of the six chains. Imagine now that the guess turns out to be wrong for the last chain, which is a $|_0^0$ chain. Alice and Bob will then ask the leaker to disclose (x_6, y_6). Such disclosure unfortunately directly reveals the answer to the GDC instance. This in turn, reduces the CC of GDC to 0, rendering the reduction meaningless. (Refusing to disclose (x_6, y_6) obviously does not work either, since the refusal itself reveals the answer.)

Our idea. To overcome this, we do not let Alice and Bob decide for which chains the adversary $\mathcal{A}$ should make a guess. Instead, we directly let our leaker decide which indices should be leaked: For every i where $(x_i, y_i) \neq (0, 0)$, the leaker leaks the pair (x_i, y_i) with half probability, to both Alice and Bob. In the earlier example, the leaker will leak each of the indices 1 through 4 independently with half probability.

For any given i, define binary random variable $\mathbf{s} = 1$ iff the leaker leaks index i. If $\mathbf{s} = 1$, then Alice and Bob will "fabricate" a wrong guess for the adversary $\mathcal{A}$ that they are simulating, so that the guess of $\mathcal{A}$ is wrong (and hence index i needs to be leaked). Specifically, Alice and Bob examine the coin flip outcomes of the protocol $\mathcal{P}$ to determine the value of $\mathbf{z}$, and then set the guess $\mathbf{c}$ of $\mathcal{A}$ to be $\mathbf{c} = \bar{\mathbf{z}}$. (Recall that $\mathbf{z}$ indicates whether the middle node is sending in round $t_i + 1$.) In such a case, the decision $\lambda_{\mathcal{A}}$ of $\mathcal{A}$ will be $\lambda_{\mathcal{A}} = \mathbf{c} = \bar{\mathbf{z}}$. On the other hand, if $\mathbf{s} = 0$ (meaning that index i is not leaked), then Alice and Bob let $\mathcal{A}$ behave exactly the same as the adaptive adversary $\mathcal{A}'$ in Section 5. In particular, if $\mathcal{A}'$ makes an adaptive decision $\lambda_{\mathcal{A}'} = \mathbf{z}$ for this chain, then the decision $\lambda_{\mathcal{A}}$ of $\mathcal{A}$ will also be $\lambda_{\mathcal{A}} = \mathbf{z}$ (i.e., as if $\mathcal{A}$ guessed correctly). Combining the two cases gives $\lambda_{\mathcal{A}} = \mathbf{z} \oplus \mathbf{s}$.

Obviously $\mathcal{A}$ here is no longer oblivious (since $\lambda_{\mathcal{A}}$ now depends on $\mathbf{z}$), which seems to defeat the whole purpose. Fortunately, this adaptive adversary $\mathcal{A}$ is special in the sense that all the adaptivity (i.e., $\mathbf{z}$) has been "sanitized" by taking XOR with the independent coin of $\mathbf{s}$. Intuitively, this prevents $\mathcal{A}$ from effectively adapting. The following discussion will formalize and prove that such an $\mathcal{A}$ is no more powerful than an oblivious adversary, in terms of incurring the cost of a protocol.

Formal framework and results. Without loss of generality, we assume that an adversary makes *binary decisions* that fully describe the behavior of the adversary. An adversary is *deterministic* if its decisions are fixed given the protocol's coin flip outcomes, otherwise it is *randomized*. Consider any deterministic adaptive adversary $\mathcal{A}'$. A decision $\lambda_{\mathcal{A}'}$ made by $\mathcal{A}'$ is called *adaptive* if $\lambda_{\mathcal{A}'}$ can be different under different coin flip outcomes of the protocol. A randomized adaptive adversary $\mathcal{A}$ is called a *sanitized version* of $\mathcal{A}'$, if $\mathcal{A}$ behaves the same as $\mathcal{A}'$ except that $\mathcal{A}$ *sanitizes* all adaptive decisions made by $\mathcal{A}'$ and also an arbitrary (possibly empty) subset of the non-adaptive decisions made by $\mathcal{A}'$. Here $\mathcal{A}$ *sanitizes* a decision $\lambda_{\mathcal{A}'}$ made by $\mathcal{A}'$ by setting its own decision $\lambda_{\mathcal{A}}$ to be $\lambda_{\mathcal{A}} = \lambda_{\mathcal{A}'} \oplus \mathbf{s}$, where $\mathbf{s}$ is a separate fair coin and is independent of all other coins. We also call the above $\mathcal{A}$ as a *sanitized adaptive adversary*. In our discussion above, $\lambda_{\mathcal{A}'} = \mathbf{z}$, while $\lambda_{\mathcal{A}} = \mathbf{z} \oplus \mathbf{s} = \lambda_{\mathcal{A}'} \oplus \mathbf{s}$. The following simple theorem, proven in the full version [16] of this paper, confirms that $\mathcal{A}$ is no more powerful than an oblivious adversary:

▶ **Theorem 3.** *Let* $\mathrm{cost}(\mathcal{P}, \mathcal{A}, C_{\mathcal{P}}, C_{\mathcal{A}})$ *be any deterministic function (which the adversary aims to maximize) of the protocol* $\mathcal{P}$*, the adversary* $\mathcal{A}$*, the coin flip outcomes* $C_{\mathcal{P}}$ *of* $\mathcal{P}$*, and the coin flip outcomes* $C_{\mathcal{A}}$ *(if any) that may also influence the behavior of* $\mathcal{A}$*. For any protocol* $\mathcal{P}$*, any deterministic adaptive adversary* $\mathcal{A}'$*, and its sanitized version* $\mathcal{A}$*, there exists a deterministic oblivious adversary* $\mathcal{B}$ *such that* $E_{\mathbf{C}_{\mathcal{P}}}[\mathrm{cost}(\mathcal{P}, \mathcal{B}, \mathbf{C}_{\mathcal{P}}, -)] \geq E_{\mathbf{C}_{\mathcal{P}}, \mathbf{C}_{\mathcal{A}}}[\mathrm{cost}(\mathcal{P}, \mathcal{A}, \mathbf{C}_{\mathcal{P}}, \mathbf{C}_{\mathcal{A}})]$*. Furthermore, for every* $C_{\mathcal{P}}$ *in the support of* $\mathbf{C}_{\mathcal{P}}$*, there exists* $C_{\mathcal{A}}$ *in the support of* $\mathbf{C}_{\mathcal{A}}$*, such that* $\mathcal{B}$*'s decisions are exactly the same as the decisions made by* $\mathcal{A}$ *under* $C_{\mathcal{P}}$ *and* $C_{\mathcal{A}}$*.*

Summary of this section. Recall that $\mathcal{A}'$ denotes the adaptive adversary used in [25] and reviewed in Section 5. Based on the discussion in this section, our reduction from GDC (with a leaker) to CONSENSUS will use a sanitized adaptive adversary $\mathcal{A}$ for the dynamic network. $\mathcal{A}$ behaves exactly the same as $\mathcal{A}'$ except: For each i-th chain where $\mathcal{A}'$ makes an adaptive decision $\lambda_{\mathcal{A}'}$ for that chain, $\mathcal{A}$ sets its own decision $\lambda_{\mathcal{A}}$ for that chain to be

$\lambda_{\mathcal{A}} = \lambda_{\mathcal{A}'} \oplus \mathbf{s}$. Here $\mathbf{s}$ denotes whether index i is leaked by the leaker. Theorem 3 confirms that the consensus protocol $\mathcal{P}$'s end guarantees, even though $\mathcal{P}$ was designed to work against oblivious adversaries instead of adaptive adversaries, will continue to hold under $\mathcal{A}$.

8 Communication Complexity with The Leaker

To get our final Consensus lower bound , the next key step is to prove a lower bound on the CC of Gdc with the leaker. At first thought, one may think that having the leaker will not affect the CC of Gdc much, since i) the leakable pairs have no impact on the answer to the problem and are hence "dummy" parts, and ii) the leaker only leaks about half of such "dummy" parts. As a perhaps surprising example, Lemma 1 in the full version [16] of this paper shows that having the leaker reduces the CC of $\text{Gdc}_n^{16\sqrt{n}\ln\frac{1}{\delta},2}$ from $\Omega(\sqrt{n})$ to 0. This implies that the impact of the leaker is more subtle than expected. In particular, without a careful investigation, it is not even clear whether the CC of Gdc with our leaker is large enough to translate to our intended $\Omega(d + \text{poly}(m))$ lower bound on Consensus.

This section will thus do a careful investigation and eventually establish a formal connection between the CC with the leaker ($\mathfrak{L}_\delta$) and the CC without the leaker ($\mathfrak{R}_\delta$):

▶ **Theorem 4.** *For any constant $\delta \in (0, \frac{1}{2})$, there exist constants $c_1 > 0$ and $c_2 > 0$ such that for all n, g, q, and $n' = c_2\sqrt{n}/(q^{1.5}\log q)$, $\mathfrak{L}_\delta(\text{Gdc}_n^{g,q}) \geq c_1\mathfrak{R}_\delta(\text{Gdc}_{n'}^{g,q})$.*

Later we will see that the lower bound on Gdc with our leaker as obtained in the above theorem (combined with Theorem 2) is sufficient for us to get a final $\Omega(d + \text{poly}(m))$ lower bound on Consensus. The theorem actually also holds for many other problems beyond Gdc, though we do not present the general form here due to space limitations.

8.1 Our Approach and Key Ideas

While we will only need to prove Theorem 4 for Gdc, we will consider general two-party problem Π, since the specifics of Gdc are not needed here. We will prove Theorem 4 via a reduction: We will construct a protocol $\mathcal{Q}$ for solving $\Pi_{n'}$ without the leaker, by using an oracle protocol $\mathcal{P}$ for solving Π_n with the leaker, where n' is some value that is smaller than n. Such a reduction will then lead to $\mathfrak{R}_{\delta'}(\Pi_{n'}) = O(\mathfrak{L}_\delta(\Pi_n))$.

We will call each kind of leakable pairs as a *leakable pattern*. For example, $\text{Gdc}_n^{1,2}$ has leakable patterns of $(1,1)$, $(0,1)$, and $(1,0)$. Note that leakable patterns are determined by the problem Π and not by an instance of the problem. We use $k \in [0, q^2]$ to denote the total number of leakable patterns for Π whose inputs are q-ary strings. For $\text{Gdc}_n^{g,q}$, $k = 2q - 1$.

Simulating the leaker via padded pairs. The central difficulty in the reduction is that Alice and Bob running $\mathcal{Q}$ need to simulate the leaker, in order to invoke the oracle protocol $\mathcal{P}$. (Note that $\mathcal{P}$ here is the two-party protocol, and has nothing to do with the Consensus protocol.) This is difficult because each party only knows her/his own input. Our first step to overcome this difficulty is to pad known characters to the inputs and then leak *only* those padded characters, as explained next.

Let (X', Y') be the given input to $\mathcal{Q}$. Assume for simplicity that $(2,1)$ is the only leakable pattern in Π, and consider the problem instance in Figure 2 where $X' = 02$ and $Y' = 01$. Alice and Bob will append/pad a certain number of occurrences of each leakable pattern to (X', Y'). Let (X, Y) denote the resulting strings after the padding. In the example in Figure 2, Alice and Bob append 1 occurrence of $(2,1)$ to (X', Y') — or more specifically,

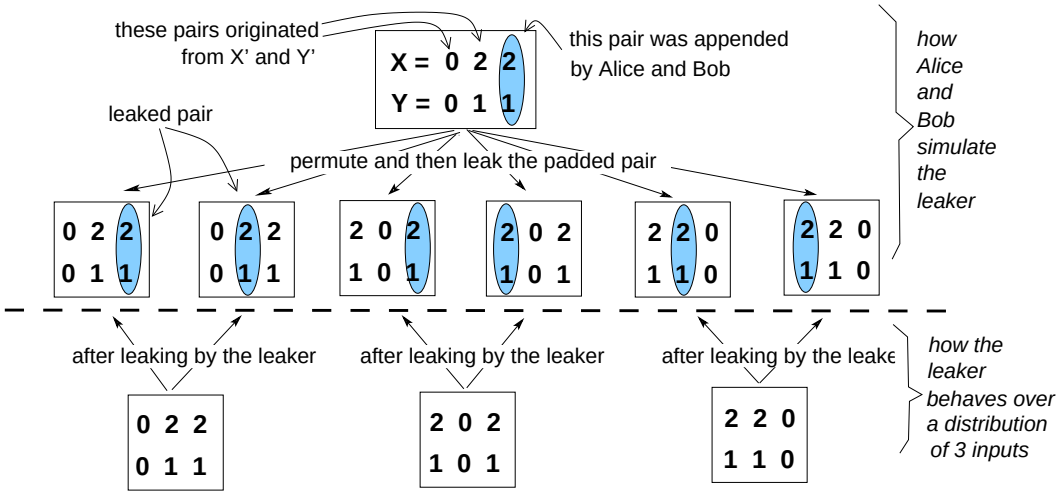

Figure 2 How padding and permutation enable Alice and Bob to simulate the leaker. In this example $X' = 02$, $Y' = 01$, $X = 022$, and $Y = 011$. Here to help understanding, we assume that the leaker leaks *exactly* half of all the leakable pairs.

Alice appends 2 to X' and Bob appends 1 to Y'. Doing so gives $X = 022$ and $Y = 011$. Note that doing so does not involve any communication, since the leakable patterns are publicly known. Imagine that Alice and Bob now invoke $\mathcal{P}$ using (X, Y), where $X = 022$ and $Y = 011$. Note that the two-party protocol $\mathcal{P}$ assumes the help from our leaker. Alice and Bob can easily simulate the leaking of (x_3, y_3), since (x_3, y_3) is the padded pair and they both know that the pair is exactly $(2, 1)$. However, (x_2, y_2) is also a leakable pair. Alice and Bob still cannot simulate the leaking of this pair, since this pair originated from (X', Y') and they do not know the value of this pair.

To overcome this, Alice and Bob use public coins to generate a random permutation, and then use the permutation to permute X and Y, respectively (Figure 2). This step does not involve communication. For certain problems Π (e.g., for GDC), one can easily verify that such permutation will not affect the answer to Π. Such permutation produces an interesting effect, as illustrated in Figure 2. The upper part of Figure 2 plots the 6 possible outcomes after the permutation, for our earlier example of $X = 022$ and $Y = 011$. Before the permutation, the last pair in (X, Y) is a padded pair. Imagine that Alice and Bob leak this pair. Now after the permutation, this leaked pair will occupy different indices in the 6 outcomes of the permutation.

The bottom part of Figure 2 illustrates the (real) leaker's behavior over certain inputs. To help understanding, assume here for simplicity that the leaker leaks *exactly* half of all the leakable pairs. Now consider 3 different inputs $(022, 011)$, $(202, 101)$, and $(220, 110)$. One can see that the behavior of the leaker over these 3 inputs (see Figure 2) exactly matches the result of permutation as done by Alice and Bob. Hence when Alice and Bob feed the result of the permutation into $\mathcal{P}$ while leaking the padded pair, it is as if $\mathcal{P}$ were invoked over the previous 3 inputs (each chosen with $1/3$ probability) together with the real leaker. This means that $\mathcal{P}$'s correctness and CC guarantees should continue to hold, when Alice and Bob invoke $\mathcal{P}$ while leaking only the padded pair.

How many pairs to leak. Imagine that (X', Y') contain o pairs of $(2, 1)$, and Alice and Bob pad p pairs of $(2, 1)$ to (X', Y'). The result of the padding, (X, Y), will contain $o + p$ pairs of $(2, 1)$. Let $\mathbf{f}$ be the number of $(2, 1)$ pairs in (X, Y) that should be leaked, which

Protocol 1: Our δ'-error protocol $\mathcal{Q}$ for solving $\Pi_{n'}$ without our leaker. $\mathcal{Q}$ invokes the δ-error oracle two-party protocol $\mathcal{P}$ that solves Π_n with our leaker. The following only shows Alice's part of $\mathcal{Q}$. Bob's part of $\mathcal{Q}$ can be obtained similarly.

Input : X', n, n', δ, δ', where $\delta < \delta'$

1 $s \leftarrow \frac{\mathfrak{R}_{\delta'}(\Pi_{n'})}{4 \log q}$; **foreach** $j = 1, \ldots, k$ **do** $\mathbf{v}_j \leftarrow 0$;

2 **repeat** s **times do**

3 draw a uniformly random integer $i \in [1, n']$ using public coins;

4 send x'_i to Bob and receive y'_i from Bob ;

5 **foreach** $j = 1, \ldots, k$ **do if** (x'_i, y'_i) *equals the* j*-th leakable pattern* **then** $\mathbf{v}_j \leftarrow \mathbf{v}_j + \frac{n'}{s}$;

6 **end**

/*** Here h_j is the number of times that the j-th leakable pattern is padded to (X', Y'). ***/

7 $h \leftarrow 2n' + \frac{500}{(\delta'-\delta)^2}(k^2 + \frac{kn'^2}{2s} \ln \frac{24k}{\delta'-\delta})$;

8 **foreach** $j = 1, \ldots, k-1$ **do** $h_j \leftarrow h$;

9 $h_k \leftarrow n - n' - (k-1)h$; **if** $h_k < h$ **then** generate an arbitrary output and exit;

10 **foreach** $j = 1, \ldots, k$ **do**

11 draw an integer $\mathbf{b}_j$ from the binomial distribution $\mathbb{B}(\frac{h_j + \mathbf{v}_j}{2})$ using public coins ;
// $\mathbb{B}(\mu)$ is the distribution for the number of heads obtained when flipping 2μ fair coins.

12 **if** $\mathbf{b}_j > h_j$ **then** $\mathbf{b}_j \leftarrow h_j$;

13 let (a, b) be the j-th leakable pattern ;

14 append h_j copies of a to X', and flag the first $\mathbf{b}_j$ indices of these h_j indices as "to be leaked";

15 **end**

16 generate a uniformly random permutation $\mathbf{M}$ using public coins;

17 $\mathbf{X} \leftarrow \mathbf{M}(X')$ /* the flags in X' will be treated as part of X' and be permuted as well. */;

18 invoke $\mathcal{P}$ (together with the other party) using $\mathbf{X}$ as input, while leaking all those indices that are flagged, until either $\mathcal{P}$ outputs or $\mathcal{P}$ has incurred $(\frac{6}{\delta'-\delta})cc(\mathcal{P})$ bits of communication ;
/* when leaking index i, both x'_i and y'_i will be given to $\mathcal{P}$ — this can be done since a leaked index here must correspond to a padded pair at Line 14 */;

19 **if** $\mathcal{P}$ *has incurred* $(\frac{6}{\delta'-\delta})cc(\mathcal{P})$ *bits of communication* **then** exit with an arbitrary output ;

20 **else** output $\mathcal{P}$'s output and exit ;

obviously follows a binomial distribution with a mean of $\frac{o+p}{2}$. Ideally, Alice and Bob should draw $\mathbf{f}$ from the above binomial distribution, and then simulate the leaking of $\mathbf{f}$ pairs of $(2, 1)$. (They can do so as long as $\mathbf{f} \leq p$ — with proper p, we easily throw $\Pr[\mathbf{f} > p]$ into the error.) The difficulty, however, is that Alice and Bob do not know o, and hence cannot draw $\mathbf{f}$ with the correct mean of $\frac{o+p}{2}$.

To overcome this, Alice and Bob will estimate the value of o by sampling: For each sample, they use public coin to choose a uniformly random $i \in [1, n']$, and then send each other the values of x'_i and y'_i. They will spend total $\frac{\mathfrak{R}_{\delta'}(\Pi_{n'})}{2}$ bits for doing this, so that such sampling is effectively "free" and does not impact the asymptotic quality of the reduction. Alice and Bob will nevertheless still not obtain the exact value of o. This means that the distribution they use to draw $\mathbf{f}$ will be different from the distribution that the (real) leaker uses. Our formal proof will carefully take into account such discrepancy.

8.2 Formal Reduction and Final Guarantees

Pseudo-code. Protocol 1 presents the protocol $\mathcal{Q}$ for solving $\Pi_{n'}$ without our leaker, as run by Alice. $\mathcal{Q}$ internally invokes the oracle two-party protocol $\mathcal{P}$, where $\mathcal{P}$ solves Π_n with our leaker. At Line 1–6, Alice and Bob first exchange sampled indices to estimate the occurrences of each leakable pattern. Next Line 7–9 calculate the amount of padding needed. Line 10–15 do the actual padding, and then for each leakable pattern, flag a certain number of padded pairs as "to be leaked". At Line 16–20, Alice and Bob do a random permutation to obtain $(\mathbf{X}, \mathbf{Y})$, and then invoke $\mathcal{P}$ on $(\mathbf{X}, \mathbf{Y})$ while leaking all those flagged pairs.

Final properties of $\mathcal{Q}$. The full version [16] of this paper will prove that $\mathcal{Q}$ solves Π without our leaker, with an error of $\delta + \frac{11}{12}(\delta' - \delta)$, while incurring $\frac{\mathfrak{R}_{\delta'}(\Pi_{n'})}{2} + 5.5\mathrm{cc}(\mathcal{P})$ bits of communication. This will eventually lead to the proof of Theorem 4 (see the full version [16] of this paper).

9 Consensus Lower Bound under Oblivious Adversaries

Following is our final theorem on Consensus under oblivious adversaries:

▶ **Theorem 5.** *If the nodes only know a poor estimate m' for m such that $|\frac{m'-m}{m}|$ reaches $\frac{1}{3}$ or above, then a $\frac{1}{10}$-error* Consensus *protocol for dynamic networks with oblivious adversaries must have a time complexity of $\Omega(d + m^{\frac{1}{12}})$ rounds.*

Our proof under oblivious adversaries partly builds upon the previous proof under adaptive adversaries [25], as reviewed in Section 5. The key difference is that we reduce from Gdc *with our leaker* to Consensus. The complete proof is lengthy and tedious as it needs to build upon the lengthy proof in [25]. Since Section 7 and 8 already discussed the key differences between our proof and [25], we leave the full proof to the full version [16] of this paper, and only provide an overview here on how to put the pieces from Section 7 and 8 together.

Consider any oracle Consensus protocol $\mathcal{P}$ with $\frac{1}{10}$ error. Let $\mathrm{tc}(d, m)$ denote $\mathcal{P}$'s time complexity when running over dynamic networks controlled by oblivious adversaries and with d diameter and m nodes. As explained in Section 5, the crux will be to prove $\mathrm{tc}(8, m) \geq m^{\frac{1}{12}}$. To do so, we consider $\mathrm{Gdc}_n^{g,q}$ with $n = \frac{m-4}{3}$, $q = 20\mathrm{tc}(8, m) + 20$, and $g = 15q \ln q$. To solve $\mathrm{Gdc}_n^{g,q}(X, Y)$, Alice and Bob simulate $\mathcal{P}$ in the following way: In the simulation, the input (X, Y) is mapped to a *sanitized* adaptive adversary $\mathcal{A}$ that determines the topology of the dynamic network. Roughly speaking, if $\mathrm{Gdc}_n^{g,q}(X, Y) = 1$, the resulting dynamic network will have a diameter of 8. Even though $\mathcal{A}$ is an adaptive adversary, by Theorem 3 in Section 7, $\mathcal{P}$'s time complexity should remain $\mathrm{tc}(d, m)$ under $\mathcal{A}$. Hence $\mathcal{P}$ should decide within $\mathrm{tc}(8, m)$ rounds on expectation. If $\mathrm{Gdc}_n^{g,q}(X, Y) = 0$, then the resulting dynamic network will have a diameter of $\Theta(q)$. For $\mathcal{P}$ to decide in this dynamic network, we prove that it takes at least roughly $\frac{q}{2}$ rounds. Note that $\frac{q}{2} > 10\mathrm{tc}(8, m)$ — in other words, it takes longer for $\mathcal{P}$ to decide if $\mathrm{Gdc}_n^{g,q}(X, Y) = 0$. Alice and Bob do not know the other party's input, and hence does not have full knowledge of the dynamic network. But techniques from [25], together with the help from our leaker, enable them to still properly simulate $\mathcal{P}$'s execution. Finally, if $\mathcal{P}$ decides within $10\mathrm{tc}(8, m)$ rounds, Alice and Bob claim that $\mathrm{Gdc}_n^{g,q}(X, Y) = 1$. Otherwise they claim $\mathrm{Gdc}_n^{g,q}(X, Y) = 0$. Our proof will show that to solve $\mathrm{Gdc}_n^{g,q}$ with our leaker, using the above simulation, Alice and Bob incur $\Theta(\mathrm{tc}(8, m) \cdot \log n)$ bits of communication. We thus have $\Theta(\mathrm{tc}(8, m) \log n) \geq \mathfrak{L}_\delta(\mathrm{Gdc}_n^{g,q})$. Together with the lower bound on $\mathfrak{L}_\delta(\mathrm{Gdc}_n^{g,q})$ from Theorem 4 in Section 8 (and Theorem 2 in Section 4), this will lead to a lower bound on $\mathrm{tc}(8, m)$.

References

1 J. Augustine, C. Avin, M. Liaee, G. Pandurangan, and R. Rajaraman. Information spreading in dynamic networks under oblivious adversaries. In *DISC*, 2016.

2 J. Augustine, T. Kulkarni, P. Nakhe, and P. Robinson. Robust leader election in a fast-changing world. In *Workshop on Foundations of Mobile Computing*, 2013.

3 J. Augustine, G. Pandurangan, and P. Robinson. Fast byzantine leader election in dynamic networks. In *DISC*, October 2015.

4 John Augustine, Gopal Pandurangan, and Peter Robinson. Fast byzantine agreement in dynamic networks. In *PODC*, July 2013.

5 Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, June 2004.

6 Q. Bramas, T. Masuzawa, and S. Tixeuil. Distributed online data aggregation in dynamic graphs. In *ICDCS*, 2016.

7 M. Braverman. Interactive information complexity. *SIAM Journal on Computing*, 44(6):1698–1739, 2015.

8 K. Censor-Hillel, E. Haramaty, and Z. Karnin. Optimal dynamic distributed MIS. In *PODC*, 2016.

9 Binbin Chen, Haifeng Yu, Yuda Zhao, and Phillip B. Gibbons. The cost of fault tolerance in multi-party communication complexity. *Journal of the ACM*, 61(3), May 2014.

10 Alejandro Cornejo, Seth Gilbert, and Calvin Newport. Aggregation in dynamic networks. In *PODC*, July 2012.

11 E. Coulouma, E. Godard, and J. Peters. A characterization of oblivious message adversaries for which consensus is solvable. *Theoretical Computer Science*, 584:80–90, June 2015.

12 A. Das Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg, and R. Wattenhofer. Distributed verification and hardness of distributed approximation. In *STOC*, 2011.

13 C. Dutta, G. Pandurangan, R. Rajaraman, Z. Sun, and E. Viola. On the complexity of information spreading in dynamic networks. In *SODA*, January 2013.

14 Mohsen Ghaffari, Nancy Lynch, and Calvin Newport. The cost of radio network broadcast for different models of unreliable links. In *PODC*, July 2013.

15 R. Ingram, P. Shields, and J. Walter. An asynchronous leader election algorithm for dynamic networks. In *IPDPS*, 2009.

16 I. Jahja, H. Yu, and Y. Zhao. Some lower bounds in dynamic networks with oblivious adversaries. Technical Report TRA7/17, School of Computing, National University of Singapore, July 2017. Also available at `http://www.comp.nus.edu.sg/~yuhf/oblivious-disc17-technicalreport.pdf`.

17 M. Konig and R. Wattenhofer. On local fixing. In *OPODIS*, 2013.

18 Fabian Kuhn, Nancy Lynch, Calvin Newport, Rotem Oshman, and Andrea Richa. Broadcasting in unreliable radio networks. In *PODC*, July 2010.

19 Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *STOC*, June 2010.

20 Fabian Kuhn, Yoram Moses, and Rotem Oshman. Coordinated consensus in dynamic networks. In *PODC*, June 2011.

21 Fabian Kuhn and Rotem Oshman. The complexity of data aggregation in directed networks. In *DISC*, September 2011.

22 Fabian Kuhn and Rotem Oshman. Dynamic networks: Models and algorithms. *SIGACT News*, 42(1):82–96, March 2011.

23 U. Schmid, B. Weiss, and I. Keidar. Impossibility results and lower bounds for consensus under link failures. *SIAM Journal on Computing*, 38(5):1912–1951, 2009.

24 O. Weinstein. Information complexity and the quest for interactive compression. *SIGACT News*, 46(2):41–64, June 2015.

25 H. Yu, Y. Zhao, and I. Jahja. The cost of unknown diameter in dynamic networks. In *SPAA*, July 2016. (Journal version under submission).