Distributed Computing (2020) 33:1–40
https://doi.org/10.1007/s00446-019-00360-4

Some lower bounds in dynamic networks with oblivious adversaries

Irvan Jahja[1] · **Haifeng Yu**[1] · **Yuda Zhao**[2]

Published online: 7 August 2019

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 that communication complexity problem 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.

Keywords Dynamic networks · Oblivious adversary · Adaptive adversary · Lower bounds · Communication complexity

1 Introduction

Dynamic networks [24] 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:

- 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 v 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 v's output is correct only if by the time that v 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

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. This work was done while Y. Zhao was in National University of Singapore.

A preliminary conference version [18] of this paper is published in the 31st International Symposium on Distributed Computing, October 2017.

✉ Haifeng Yu
haifeng@comp.nus.edu.sg

[1] National University of Singapore, 15 Computing Drive, Singapore 117418, Republic of Singapore

[2] Grab, Singapore, Republic of Singapore

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

of the above problems, let $\text{tc}(d, m)$ denote the protocol's time complexity, when it runs over networks with d diameter and m nodes. It is easy to see that $\text{tc}(d, m)$ crucially depends on d, since we trivially have $\text{tc}(d, m) = \Omega(d)$. 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. [22] 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. [29] recently have proved that $\text{tc}(d, m) = O(d \log m)$ 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. [29] prove a lower bound of $\Omega(d + \text{poly}(m))$ for CONSENSUS and LEADERELECT, implying a **no** answer. For CONFIRMEDFLOOD and AGGREGATION, they have also proved $\text{tc}(d, m) = \Omega(d + \text{poly}(m))$, even if the nodes know m. This implies a **no** answer for those two problems.

All the lower bound proofs in [29], however, critically rely on a powerful *adaptive adversary*: In each round, the adaptive adversary sees all the coin flip outcomes so far of the protocol $\mathscr{P}$ 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 under *oblivious adversaries*, which do not see $\mathscr{P}$'s coin flip outcomes and have to decide the topologies in all the rounds before $\mathscr{P}$ 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 [29] 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 [29].

Our approach may also be extended to CONFIRMEDFLOOD, which in turn reduces to SUM and MAX [29]. 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 [14] and then later under oblivious adversaries [1]. Dynamic MIS was investigated separately under adaptive adversaries [19] and later under oblivious adversaries [9]. Broadcasting was first studied under adaptive adversaries [20], and later under oblivious adversaries [15].

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 essentially follows the existing proof framework under adaptive adversaries [29], 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 [10,13,23].

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 ith character of X

[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.

[4] Note however that all upper bounds, from [22,29], will directly carry over to oblivious adversaries.

and Y, respectively. We define (a, b) to be a *leakable pattern* if for all n, X, Y, and $i \in [0, n]$[5]:

$$\Pi_n(x_1x_2\ldots x_n, y_1y_2\ldots y_n) = \Pi_{n+1}(x_1x_2\ldots x_iax_{i+1}x_{i+2}\ldots x_n, y_1y_2\ldots y_iby_{i+1}y_{i+2}\ldots y_n)$$

Intuitively, for all (X, Y), the answer to Π does not change when an occurrence of a leakable pattern is either inserted into or removed from (X, Y). Note that since the property needs to hold for all n and for all (X, Y), the answer to Π will not change either when multiple occurrences of a leakable pattern (or multiple occurrences of multiple leakable patterns) are inserted or removed. For each index i where $x_i = a$ and $y_i = b$ for some leakable pattern (a, b), independently with probability $\frac{1}{2}$, our leaker *leaks* the index i. Here *leaking* the index i means that the leaker lets both Alice and Bob know for free the values of i, x_i, and y_i, before Alice and Bob start running their protocol.

We will mainly be concerned with the GDC problem with our leaker. Note that there are many possible ways of defining a leaker, and our specific definition above is not necessarily suitable in all other contexts. (For example, we require a leakable pattern to be "leakable" under all n, X, and Y. Alternatively, one could define this notion with respective to given X and Y.) Our goal is simply to facilitate the reduction from GDC to CONSENSUS under oblivious adversaries, rather than aiming for the best generality.

Even with our leaker, the reduction from GDC 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 "adaptivity" has 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.

Roadmap. At the technical level, this paper will eventually present two separate and completely independent reductions. The first reduction (elaborated in Sect. 8) is from the GDC problem without our leaker to the GDC problem with our leaker. In this reduction, we start with 2 entities: Alice and Bob. They aim to solve the GDC problem without our leaker (i.e., the standard GDC problem). They do so by simulating our leaker, and then invoking some black-box protocol that solves the GDC problem with our leaker.

The second reduction (elaborated in Sect. 9) is from the GDC problem with our leaker to the CONSENSUS problem. In this reduction, we start with three entities: Alice, Bob, and the leaker. The three entities together try to solve the GDC problem, by simulating some black-box CONSENSUS protocol. In this reduction, the leaker is given, and is not simulated by Alice and Bob.

[5] If $i = 0$, then (a, b) is prepended to (X, Y). Similarly if $i = n$, then (a, b) is appended to (X, Y).

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. [22] 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 rely 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., [17]) 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., [12,26]), 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. [11] 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., [7]) on AGGREGATION assumes that the topology in 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 and Oshman [23] and Das Sarma et al. [13] 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

Table 1 Key notations

Π	A two-party communication complexity problem
X	Alice's input string for Π
Y	Bob's input string for Π
n	Number of characters in X and Y
q	Each character in X and Y is an integer between 0 and $q-1$, inclusively
k	The number of leakable patterns
g	The minimum number of occurrences of the $(0,0)$ pattern in (X,Y) if $\text{GDC}_n^{g,q}(X,Y)=0$
$\mathscr{P}$, $\mathscr{Q}$	Protocols
$\mathscr{A}$, $\mathscr{B}$	Adversaries
$\mathbf{C}_{\mathscr{P}}$, $\mathbf{C}_{\mathscr{Q}}$	Coin flip outcomes of the protocol $\mathscr{P}$, $\mathscr{Q}$
$\mathbf{C}_{\mathscr{A}}$, $\mathbf{C}_{\mathscr{B}}$	Coin flip outcomes of the adversary $\mathscr{A}$, $\mathscr{B}$
$\mathbf{C}_{\mathscr{L}}$	Coin flip outcomes of the leaker
$\text{cc}(\mathscr{P},n)$	The communication complexity of $\mathscr{P}$
$\text{err}(\mathscr{P})$	The error of $\mathscr{P}$
$\mathfrak{R}_\delta(\Pi)$	The δ-error communication complexity of problem Π
$\mathfrak{L}_\delta(\Pi)$	The δ-error communication complexity of problem Π with our leaker
$\|_b^a(X,Y)$	The number of occurrences of the (a,b) pattern in (X,Y)
m	Number of nodes in the dynamic network
d	(Dynamic) diameter of the dynamic network
α, β, γ, λ	Special nodes in the dynamic network constructed by the reference adversary
τ, υ, ν, ω	Generic nodes in a dynamic network

al.'s work [10] 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 the GDC problem 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 for the GDC problem serves to allow oblivious adversaries. Quite interestingly, for completely different purposes, the notions of leakable patterns and a leaker have been extensively (but implicitly) used in proofs for obtaining direct sum results on the information complexity (IC) (e.g., [5,8,28]) of various communication problems: First, leakable patterns 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 [8,28]) 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 $\text{GDC}_n^{g,q}$ with our leaker. For simpler problems such as DISJOINTNESS (which is effectively $\text{GDC}_n^{1,2}$), 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 $\text{GDC}_n^{g,q}$ make IC arguments tricky. Hence we will (in Sect. 8) obtain our intended lower bound by doing a direct reduction from the CC of $\text{GDC}_{n'}^{g,q}$ without the leaker to the CC of $\text{GDC}_n^{g,q}$ with the leaker.

3 Model and definitions

Table 1 summarizes the key notations in this paper.

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 ith 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., $\mathscr{P}$ and $\mathscr{Q}$) 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. (In particular, we follow the standard convention in dynamic networks [24] that if v sends in a round, it will send the *same* message to all its neighbors.) 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. We emphasize that a node does not know its neighbors in each round beforehand—it can only infer such information based on the messages that it receives.

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 [21].) 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 $\mathscr{P}$'s coin flip outcomes up to and including the current round, and manipulates the topology accordingly, before $\mathscr{P}$ uses the current round's coin flip outcomes.

We use the standard definition for the *(dynamic) diameter* [24] 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 $(\omega, r) \rightarrow (v, r+1)$ 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 "$\rightarrow$". 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*. For any given integers $a \in [0, q-1]$ and $b \in [0, q-1]$, we will call (a, b) as a *pattern*. Alice and Bob aim to compute the value of the binary function $\Pi_n(X, Y)$. Given a protocol $\mathscr{P}$ for solving Π_n for all n (without a leaker), we define $\mathrm{cc}(\mathscr{P}, X, Y, \mathbf{C}_{\mathscr{P}})$ to be the communication incurred (in terms of number of bits) by $\mathscr{P}$, under the input (X, Y) and $\mathscr{P}$'s coin flip outcomes $\mathbf{C}_{\mathscr{P}}$. Note that $\mathbf{C}_{\mathscr{P}}$ is a random variable while cc() is a deterministic function. We similarly define $\mathrm{err}(\mathscr{P}, X, Y, \mathbf{C}_{\mathscr{P}})$, which is 1 if $\mathscr{P}$'s output is wrong, and 0 otherwise. In the following, $\max_X$ ($\max_Y$) is taken over all input strings X (Y) with n characters. We define the *communication complexity* of $\mathscr{P}$ to be $\mathrm{cc}(\mathscr{P}, n) = \max_X \max_Y E_{\mathbf{C}_{\mathscr{P}}}[\mathrm{cc}(\mathscr{P}, X, Y, \mathbf{C}_{\mathscr{P}})]$, and the *error* of $\mathscr{P}$ to be $\mathrm{err}(\mathscr{P}) = \max_n \max_X \max_Y E_{\mathbf{C}_{\mathscr{P}}}[\mathrm{err}(\mathscr{P}, X, Y, \mathbf{C}_{\mathscr{P}})]$. We define the δ-error ($0 < \delta < \frac{1}{2}$) *communication complexity* of Π_n to be $\mathfrak{R}_\delta(\Pi_n) = \min_{\mathscr{P}} \mathrm{cc}(\mathscr{P}, n)$, with the minimum taken over all $\mathscr{P}$ where $\mathrm{err}(\mathscr{P}) \leq \delta$. 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 patterns and how our leaker works. We sometimes call a pair (x_i, y_i) as a *leakable pair* if $x_i = a$ and $y_i = b$ for some leakable pattern (a, b). Given $\mathscr{P}$ for solving Π for all n with our leaker, we define $\mathrm{cc}(\mathscr{P}, X, Y, \mathbf{C}_{\mathscr{P}}, \mathbf{C}_{\mathscr{L}})$ be the communication incurred by $\mathscr{P}$, under the input (X, Y), $\mathscr{P}$'s coin flip outcomes $\mathbf{C}_{\mathscr{P}}$, and the leaker's coin flip outcomes $\mathbf{C}_{\mathscr{L}}$. Here (X, Y) and $\mathbf{C}_{\mathscr{L}}$ uniquely determine which indices get leaked. In the following, $\max_X$ ($\max_Y$) is taken over all input strings X (Y) with n characters. We define $\mathrm{cc}(\mathscr{P}, n) = \max_X \max_Y E_{\mathbf{C}_{\mathscr{L}}} E_{\mathbf{C}_{\mathscr{P}}}[\mathrm{cc}(\mathscr{P}, X, Y, \mathbf{C}_{\mathscr{P}}, \mathbf{C}_{\mathscr{L}})]$. We similarly define $\mathrm{err}(\mathscr{P}, X, Y, \mathbf{C}_{\mathscr{P}}, \mathbf{C}_{\mathscr{L}})$, and define $\mathrm{err}(\mathscr{P}) = \max_n \max_X \max_Y E_{\mathbf{C}_{\mathscr{L}}} E_{\mathbf{C}_{\mathscr{P}}}[\mathrm{err}(\mathscr{P}, X, Y, \mathbf{C}_{\mathscr{P}}, \mathbf{C}_{\mathscr{L}})]$. 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 $\mathfrak{L}_\delta(\Pi_n) = \min_{\mathscr{P}} \mathrm{cc}(\mathscr{P}, n)$, with the minimum taken over all $\mathscr{P}$ such that $\mathscr{P}$ solves Π_n with our leaker and $\mathrm{err}(\mathscr{P}) \leq \delta$. 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 $\mathrm{GDC}_n^{g,q}$, 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 $\mathrm{GDC}_n^{g,q}(X, Y)$, 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** [10] 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 on the CC of GDC is an adaptation from Theorem C.1 in [10]:

Theorem 1 *For any constant δ where $0 < \delta < 0.5$, there exist constants $c_1 > 0$ and $c_2 > 0$ such that for all n, g, and q, $\mathfrak{R}_\delta(\text{GDC}_n^{g,q}) \geq \frac{c_1 n}{gq^2} - c_2 \log \frac{n}{g}$.*

Proof First, we show $\mathfrak{R}_\delta(\text{GDC}_{n/g}^{1,q}) \leq \mathfrak{R}_\delta(\text{GDC}_n^{g,q})$, via a simple reduction: Given any protocol $\mathscr{P}$ for solving $\text{GDC}_n^{g,q}$, we will construct a protocol $\mathscr{Q}$ for solving $\text{GDC}_{n/g}^{1,q}$. In $\mathscr{Q}$, Alice replicates her length-(n/g) input g times to get a length-n input. Bob does the same. Alice and Bob then invoke $\mathscr{P}$ and output $\mathscr{P}$'s output. It is easy to verify the correctness of this trivial reduction. Next, the theorem directly follows from an existing result from Chen et al. [10] showing that $\mathfrak{R}_\delta(\text{GDC}_{n/g}^{1,q}) \geq \frac{c_1 n}{gq^2} - c_2 \log \frac{n}{g}$. □

The proof of Theorem 1 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. In particular, the previous reduction fails for $\mathfrak{L}_\delta$: After Alice replicates her length-(n/g) input g times, the leaker (over the length-n input) may leak different parts in each of the g segments, and Alice cannot simulate such behavior. 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 [29] 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 CONSENSUS protocol $\mathscr{P}$ with $\frac{1}{10}$ error. Let $\text{tc}(d, m)$ be $\mathscr{P}$'s time complexity, when running over dynamic networks controlled by adaptive adversaries and with d diameter and m nodes. The proof in [29] 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 $\Omega(d + \text{poly}(m))$.

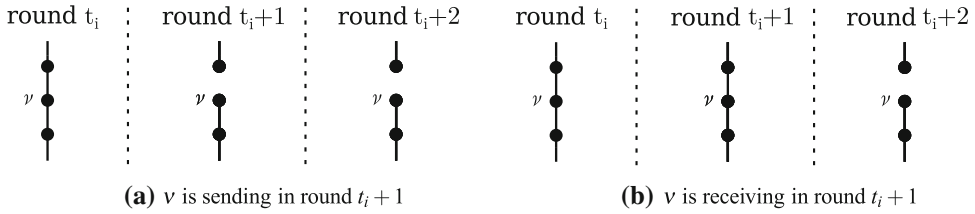
To prove $\text{tc}(8, m) = \Omega(\text{poly}(m))$, [29] uses a reduction from $\text{GDC}_n^{g,q}$ to CONSENSUS. To solve $\text{GDC}_n^{g,q}(X, Y)$, Alice knowing X and Bob knowing Y simulate the CONSENSUS protocol $\mathscr{P}$ in the following way: In the simulation, the input (X, Y) is mapped to 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 $\mathscr{P}$ 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 [29] that $\mathscr{P}$ 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 $\mathscr{P}$ decides: If $\mathscr{P}$ decides within $10r_1$ rounds, they claim that $\text{GDC}_n^{g,q}(X, Y) = 1$. Otherwise they claim that $\text{GDC}_n^{g,q}(X, Y) = 0$.

To solve GDC using the above simulation, Alice and Bob need to simulate $\mathscr{P}$ 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 $\mathscr{P}$ over that dynamic network. The proof in [29] 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 [29] as a key technique. We give a concise review below.

Given (X, Y), the dynamic network constructed in [29] has one *chain* for each index $i \in [1, n]$. Each chain has three nodes in a line (Fig. 1). Consider as an example the ith 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 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 [29] will make an adaptive decision for



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

Fig. 1 The adaptive decisions of the adversary in [29]

manipulating the edges on the ith chain,[6] 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 we refer interested readers to [29] 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 $\mathscr{P}$ and determines whether the middle node v on the ith chain is sending or receiving in round $t_i + 1$ (see Fig. 1). If v is sending, the adversary removes a certain edge e that is incidental to v, 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 our proof of the CONSENSUS lower bound under oblivious adversaries. To facilitate discussion, we define a few simple concepts. We use $\mathscr{A}'$ to denote the adaptive adversary described in the previous section. Consider the ith chain in the previous section where (x_i, y_i) is not $(0, 0)$, and the middle node v on that chain. Define binary random variable $\mathbf{z}_{\mathscr{A}'} = 0$ iff v is sending in round $t_i + 1$ in the execution of $\mathscr{P}$ against $\mathscr{A}'$. Recall that $\mathscr{A}'$ removes the edge e on this chain in round $t_i + 1 + \lambda_{\mathscr{A}'}$ where $\lambda_{\mathscr{A}'} = \mathbf{z}_{\mathscr{A}'}$.

Making guesses. The adversary $\mathscr{A}'$ is adaptive since $\lambda_{\mathscr{A}'} = \mathbf{z}_{\mathscr{A}'}$ and $\mathbf{z}_{\mathscr{A}'}$ in turn potentially depends on $\mathscr{P}$'s coin flips. ($\mathscr{A}'$ itself does not flip any coins.) Recall that we aim to obtain a lower bound under oblivious adversaries. But an oblivious adversary $\mathscr{A}$ cannot have its decision $\lambda_{\mathscr{A}}$ depend on $\mathscr{P}$'s coin flips. At the highest level, our idea of allowing $\mathscr{A}$ in the reduction is simple: We let $\mathscr{A}$ make a blind guess on whether v is sending. Specifically, imagine that $\mathscr{A}$ by itself sets either $\lambda_{\mathscr{A}} = 0$ or $\lambda_{\mathscr{A}} = 1$ with equal probability by flipping a fair coin. Similar to $\mathscr{A}'$, the adversary $\mathscr{A}$ still removes the edge e in round $t_i + 1 + \lambda_{\mathscr{A}}$, except that now $\lambda_{\mathscr{A}}$ is a fair coin. Some quick clarifications will help to avoid confusion here. Define binary random variable $\mathbf{z}_{\mathscr{A}} = 0$ iff v is sending in round $t_i + 1$ in the execution of $\mathscr{P}$ against $\mathscr{A}$. Note that $\mathbf{z}_{\mathscr{A}}$ potentially depends on $\mathscr{P}$'s coin flips. First, such a guess made by $\mathscr{A}$ can be either correct (i.e., $\lambda_{\mathscr{A}} = \mathbf{z}_{\mathscr{A}}$) or wrong (i.e., $\lambda_{\mathscr{A}} \neq \mathbf{z}_{\mathscr{A}}$). The adversary $\mathscr{A}$ itself cannot tell whether the guess is correct, since $\mathscr{A}$ (being oblivious) does not know $\mathbf{z}_{\mathscr{A}}$. Alice and Bob together can tell if the guess is correct, because they are simulating both the protocol $\mathscr{P}$ and the adversary $\mathscr{A}$, and hence know $\mathbf{z}_{\mathscr{A}}$. But they cannot interfere with the guess even if they know it is wrong.

Now if the guess is correct, then $\mathscr{A}$ will make the decision in the same way as $\mathscr{A}'$, and everything will work out as before. But if the guess is wrong, then $\mathscr{A}$ can no longer enable Alice to simulate without knowing Y. More specifically, if the guess is wrong, then for the ith 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. Now the CC problem has 3 separate and independent entities: Alice, Bob, and the leaker. Conceptually, when the guess is wrong for the ith chain, the leaker discloses for free to Alice and Bob the pair (x_i, y_i). (This is just intuition—in the actual proof, such disclosure from the leaker actually occurs at the very beginning of the simulation.) 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.

[6] 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.

Roadmap. There are two non-trivial technical issues remaining in the above approach: (i) for which chains to make guesses, and (ii) how the leaker impacts the CC of GDC. Overcoming them will be the main tasks of Sects. 7 and 8, respectively. Section 9 will present our final CONSENSUS lower bound.

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 Sect. 5, the dynamic network corresponding to this instance has six chains. For all i, we say that the ith 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 $\mathscr{A}'$ in [29] (see Sect. 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 Sect. 6, the oblivious adversary $\mathscr{A}$ should thus make guesses for those four $|_1^0$ chains. Note that $\mathscr{A}$ 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. The leaker then needs 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. (Not disclosing (x_6, y_6) obviously does not work either, since the non-disclosure itself reveals the answer.)

Our idea. To overcome this, we do not let Alice and Bob decide for which chains the adversary $\mathscr{A}$ 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.

We ultimately aim to design $\mathscr{A}$ such that $\lambda_{\mathscr{A}} = \mathbf{z}_{\mathscr{A}} \oplus \mathbf{s}$, where the random variable $\mathbf{s} = 1$ iff the leaker leaks index i. (Recall that $\mathbf{z}_{\mathscr{A}}$ indicates whether the middle node on the chain is sending in round $t_i + 1$.) To do so, if $\mathbf{s} = 1$, then the adversary $\mathscr{A}$ will intentionally use a wrong guess: Specifically, it examines the coin flip outcomes of the protocol $\mathscr{P}$ to determine $\mathbf{z}_{\mathscr{A}}$, and then set $\lambda_{\mathscr{A}} = \overline{\mathbf{z}_{\mathscr{A}}}$. On the other hand, if $\mathbf{s} = 0$ (meaning that index i is not leaked), then the adversary $\mathscr{A}$ will behave in the same way as the adaptive adversary $\mathscr{A}'$ in Sect. 5: Specifically, the adversary $\mathscr{A}$ simply sets $\lambda_{\mathscr{A}} = \mathbf{z}_{\mathscr{A}}$ (i.e., as if $\mathscr{A}$ guessed correctly).

Obviously $\mathscr{A}$ here is *not* oblivious (since $\lambda_{\mathscr{A}}$ now depends on $\mathbf{z}_{\mathscr{A}}$), which seems to defeat the whole purpose. Fortunately, this adaptive adversary $\mathscr{A}$ is special in the sense that all the adaptivity has been "sanitized" by taking XOR with the independent coin of $\mathbf{s}$. Intuitively, this prevents $\mathscr{A}$ from effectively adapting. The following discussion will formalize and prove that such an $\mathscr{A}$ is no more powerful than an oblivious adversary, in terms of incurring the cost of a protocol.

Formal results. Without loss of generality, we model an adversary as making a sequence of *binary decisions*. These binary decisions determine how the topology of the dynamic network changes. Consider any adaptive adversary $\mathscr{A}$, which may flips its own coins when making decisions. Given a protocol $\mathscr{P}$ and any initial inputs to $\mathscr{P}$, let $\{Z_1, Z_2, \ldots, Z_h\}$ be the set of all distinct sequences of decisions that $\mathscr{A}$ can possibly make under some coin flip outcomes $C_{\mathscr{P}}$ of $\mathscr{P}$ and some coin flip outcomes $C_{\mathscr{A}}$ of $\mathscr{A}$. Putting it another way, under any given $C_{\mathscr{P}}$ and $C_{\mathscr{A}}$, the sequence of decisions made by $\mathscr{A}$ will be Z_i (for some i). This adaptive adversary $\mathscr{A}$ is called a *sanitized adaptive adversary* if given the protocol $\mathscr{P}$, the initial inputs to $\mathscr{P}$, and $C_{\mathscr{P}}$, the probability (taken over $C_{\mathscr{A}}$) of the decision sequence of $\mathscr{A}$ being Z_i is $\frac{1}{h}$ for all i.

The following simple theorem confirms that a sanitized adaptive adversary $\mathscr{A}$ is no more powerful than an oblivious adversary.

Theorem 2 *Consider any given* CONSENSUS *protocol $\mathscr{P}$ and any given initial inputs to $\mathscr{P}$. Let $f_1(\mathscr{A}, C_{\mathscr{P}}, C_{\mathscr{A}})$ be the number of rounds needed for all nodes to output in $\mathscr{P}$ under the given input, the adversary $\mathscr{A}$, the coin flip outcomes $C_{\mathscr{P}}$ of $\mathscr{P}$, and the coin flip outcomes $C_{\mathscr{A}}$ of $\mathscr{A}$. Let $f_2(\mathscr{A}, C_{\mathscr{P}}, C_{\mathscr{A}}) = 0$ if $\mathscr{P}$'s outputs on all nodes are correct under the same settings as above, and 1 otherwise. For any sanitized adaptive adversary $\mathscr{A}$ and any $j \in \{1, 2\}$, there exists an oblivious adversary $\mathscr{B}_j$ such that:*

1. *$\mathscr{B}_j$ does not flip any coins itself.*
2. *$E_{\mathbf{C}_{\mathscr{P}}}[f_j(\mathscr{B}_j, \mathbf{C}_{\mathscr{P}}, -)] \geq E_{\mathbf{C}_{\mathscr{P}}, \mathbf{C}_{\mathscr{A}}}[f_j(\mathscr{A}, \mathbf{C}_{\mathscr{P}}, \mathbf{C}_{\mathscr{A}})]$.*
3. *For every $C_{\mathscr{P}}$, there exists $C_{\mathscr{A}}$ such that $\mathscr{B}_j$'s decisions are exactly the same as the decisions made by $\mathscr{A}$ under $C_{\mathscr{P}}$ and $C_{\mathscr{A}}$.*

Proof We will prove the theorem for $j = 1$. The proof can be trivially extended for $j = 2$. Recall the definition of $\{Z_1, Z_2, \ldots, Z_h\}$ from the earlier discussion. For all $i \in [1, h]$, let $\mathscr{B}_{Z_i}$ be the oblivious adversary that always make the sequence of decisions Z_i. Note that $\mathscr{B}_{Z_i}$ does not flip any coins itself.

For *given* $C_{\mathscr{P}}$, obviously some Z_i will maximize $f_1(\mathscr{B}_{Z_i}, C_{\mathscr{P}}, -)$, and will in turn make $f_1(\mathscr{B}_{Z_i}, C_{\mathscr{P}}, -) \geq E_{\mathbf{C}_{\mathscr{A}}}[f_1(\mathscr{A}, C_{\mathscr{P}}, \mathbf{C}_{\mathscr{A}})]$. However, this Z_i may be different for different $C_{\mathscr{P}}$, which prevents us from proving the theorem via this trivial argument. But a slightly more careful analysis, as following, will work.

By the definition of sanitized adaptive adversary, for any given $C_{\mathscr{P}}$ we have:

$$\underset{\mathbf{C}_{\mathscr{A}}}{E}[f_1(\mathscr{A}, C_{\mathscr{P}}, \mathbf{C}_{\mathscr{A}})] = \sum_{i=1}^{h} \frac{1}{h} f_1(\mathscr{B}_{Z_i}, C_{\mathscr{P}}, -)$$

Since $\mathbf{C}_{\mathscr{P}}$ and $\mathbf{C}_{\mathscr{A}}$ are independent, there must exists some $i_0 \in [1, h]$ such that:

$$\begin{aligned}\underset{\mathbf{C}_{\mathscr{P}}, \mathbf{C}_{\mathscr{A}}}{E}[f_1(\mathscr{A}, \mathbf{C}_{\mathscr{P}}, \mathbf{C}_{\mathscr{A}})] &= \underset{\mathbf{C}_{\mathscr{P}}}{E}\left[\sum_{i=1}^{h} \frac{1}{h} f_1(\mathscr{B}_{Z_i}, \mathbf{C}_{\mathscr{P}}, -)\right] \\ &= \frac{1}{h}\sum_{i=1}^{h} \underset{\mathbf{C}_{\mathscr{P}}}{E}[f_1(\mathscr{B}_{Z_i}, \mathbf{C}_{\mathscr{P}}, -)] \\ &\leq \underset{\mathbf{C}_{\mathscr{P}}}{E}[f_1(\mathscr{B}_{Z_{i_0}}, \mathbf{C}_{\mathscr{P}}, -)]\end{aligned}$$

We now let $\mathscr{B}_1 = \mathscr{B}_{Z_{i_0}}$. One can easily verify that $\mathscr{B}_1$ indeed satisfies the three properties needed by the lemma: The first and the second property directly follow from the discussion above. The third property requires that for every $C_{\mathscr{P}}$, there exists $C_{\mathscr{A}}$ such that $\mathscr{B}_1$'s decisions are the same as the decisions made by $\mathscr{A}$ under $C_{\mathscr{P}}$ and $C_{\mathscr{A}}$. From the definition of sanitized adaptive adversary, under the given $C_{\mathscr{P}}$, the adversary $\mathscr{A}$ will make the sequence of decisions Z_{i_0} with probability $\frac{1}{h}$. Hence under $C_{\mathscr{P}}$ and some $C_{\mathscr{A}}$, $\mathscr{A}$ will make the sequence of decisions Z_{i_0}. □

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 do not impact the answer to GDC and hence are "dummy" parts, and (ii) the leaker only leaks about half of such "dummy" parts. But as we will quickly see, the $\text{GDC}_n^{16\sqrt{n}\ln\frac{1}{\delta},2}$ problem suggests otherwise. Specifically, Theorem 1 earlier shows that the CC of $\text{GDC}_n^{16\sqrt{n}\ln\frac{1}{\delta},2}$ has a $\Omega(\sqrt{n})$ lower bound. On the other hand, Lemma 1 below indicates that having a leaker allows Alice and Bob to deduce the answer to $\text{GDC}_n^{16\sqrt{n}\ln\frac{1}{\delta},2}$ with zero CC. (They can actually infer the answer just based on the total number of leaked indices.) The proof of this lemma is deferred to "Appendix A".

Lemma 1 *For all constant $\delta \in (0, \frac{1}{2})$ and all $n \geq 1$, we have $\mathfrak{L}_\delta\left(\text{GDC}_n^{16\sqrt{n}\ln\frac{1}{\delta},2}\right) = 0$.*

Thus, having a leaker reduces the CC of $\text{GDC}_n^{16\sqrt{n}\ln\frac{1}{\delta},2}$ from $\Omega(\sqrt{n})$ to 0, implying 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 of GDC with the leaker ($\mathfrak{L}_\delta$) and the CC of GDC without the leaker ($\mathfrak{R}_\delta$)[7]:

Theorem 3 *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 $n' = c_2\sqrt{n}/(q^{1.5}\log q)$, we have $\mathfrak{L}_\delta(\text{GDC}_n^{g,q}) \geq c_1\mathfrak{R}_\delta(\text{GDC}_{n'}^{g,q})$.*

Directly combining Theorem 3 with Theorem 1, we have:

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, and q, we have $\mathfrak{L}_\delta(\text{GDC}_n^{g,q}) \geq \frac{c_1\sqrt{n}}{gq^{3.5}\log q} - c_2\log\frac{\sqrt{n}}{gq^{1.5}\log q}$.*

Later we will see that the above lower bound on GDC with our leaker is sufficient for us to get a final $\Omega(d + \text{poly}(m))$ lower bound on CONSENSUS.

8.1 Our approach and key ideas

While we will only need to prove Theorem 3 for GDC, we will consider general two-party problem Π, since the unique specifics of GDC are not needed here. We will prove Theorem 3 via a reduction: Using any given δ-error protocol $\mathscr{P}$ for solving Π_n with the leaker, we will construct a δ-error protocol $\mathscr{Q}$ for solving $\Pi_{n'}$ without the leaker, where n' is some value that is smaller than n. Such a reduction will then lead to $\mathfrak{R}_\delta(\Pi_{n'}) = O(\mathfrak{L}_\delta(\Pi_n))$.

Recall that we use leakable pattern to denote each kind of leakable pairs. 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 $\text{GDC}_n^{g,q}$, $k = 2q - 1$.

[7] Note that Lemma 1 does not contradict with Theorem 3. For $g = 16\sqrt{n}\ln\frac{1}{\delta}$, Lemma 1 shows that $\mathfrak{L}_\delta(\text{GDC}_n^{g,2}) = 0$, while Theorem 3 shows that $\mathfrak{L}_\delta(\text{GDC}_n^{g,2}) \geq c_1\mathfrak{R}_\delta(\text{GDC}_{n'}^{g,2})$ with $n' = \frac{c_2\sqrt{n}}{2\sqrt{2}}$. Lemma 1 and Theorem 3 do not contradict because $\mathfrak{R}_\delta(\text{GDC}_{n'}^{g,2})$ is actually 0. Specifically, for $c_2 < 32\sqrt{2}\ln\frac{1}{\delta}$, we have $g > n'$ and therefore the answer to the $\text{GDC}_{n'}^{g,2}$ problem is always 1—otherwise we would violate the gap promise. Hence $\mathfrak{R}_\delta(\text{GDC}_{n'}^{g,2}) = 0$.

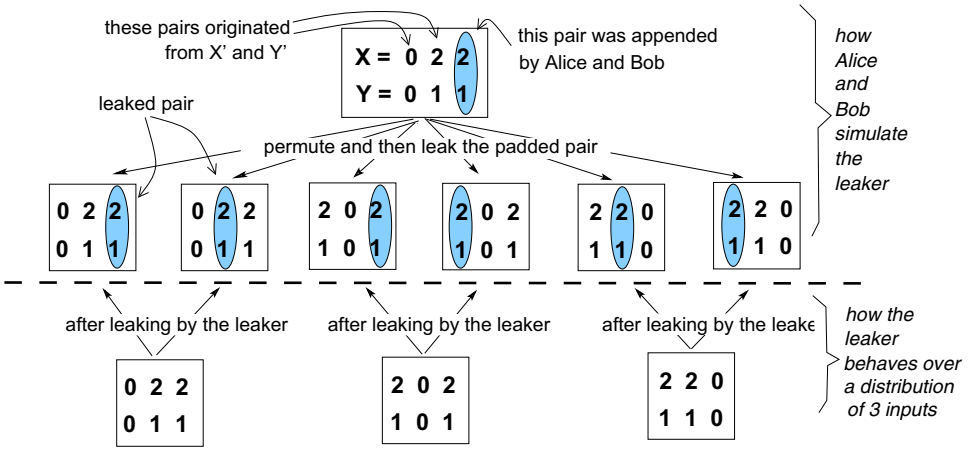


Fig. 2 How padding and permutation enable Alice and Bob to simulate the leaker. In this example $X' = 02$, $Y' = 01$, $X = 022$, and $Y = 011$. Here to help understanding, we assume that the leaker leaks *exactly* half of all the leakable pairs

Simulating the leaker via padded pairs. The central difficulty in the reduction is that Alice and Bob running $\mathscr{Q}$ need to simulate the leaker, in order to invoke the given protocol $\mathscr{P}$. (Note that $\mathscr{P}$ 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 $\mathscr{Q}$. Assume for simplicity that (2, 1) is the only leakable pattern in Π, and consider the problem instance in Fig. 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 Fig. 2, Alice and Bob append 1 occurrence of (2, 1) to (X', Y')—or more specifically, 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 $\mathscr{P}$ using (X, Y), where $X = 022$ and $Y = 011$. Note that the two-party protocol $\mathscr{P}$ 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 (Fig. 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 Fig. 2. The upper part of Fig. 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 Fig. 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 Fig. 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 $\mathscr{P}$ while leaking the padded pair, it is as if $\mathscr{P}$ were invoked over the previous 3 inputs (each chosen with 1/3 probability) together with the real leaker. This means that $\mathscr{P}$'s correctness and CC guarantees should continue to hold, when Alice and Bob invoke $\mathscr{P}$ 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 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

Input: $X', n, n', \delta, \delta'$, 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**
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**
6 **if** (x'_i, y'_i) *equals the* j*-th leakable pattern* **then**
7 $\mathbf{v}_j \leftarrow \mathbf{v}_j + \frac{n'}{s}$;
8 **end**
9 **end**
/*** Here h_j is the number of times that the j-th leakable pattern is padded to (X', Y'). ***/
10 $h \leftarrow 2n' + \frac{500}{(\delta'-\delta)^2}(k^2 + \frac{kn'^2}{2s} \ln \frac{24k}{\delta'-\delta})$;
11 **foreach** $j = 1, \ldots, k-1$ **do** $h_j \leftarrow h$;
12 $h_k \leftarrow n - n' - (k-1)h$; **if** $h_k < h$ **then** generate an arbitrary output and exit;
13 **foreach** $j = 1, \ldots, k$ **do**
14 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.
15 **if** $\mathbf{b}_j > h_j$ **then** $\mathbf{b}_j \leftarrow h_j$;
16 let (a, b) be the j-th leakable pattern ;
17 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";
18 **end**
19 generate a uniformly random permutation $\mathbf{M}$ using public coins;
20 $\mathbf{X} \leftarrow \mathbf{M}(X')$ /* the flags in X' will be treated as part of X' and be permuted as well. */;
21 invoke $\mathscr{P}$ (together with the other party) using $\mathbf{X}$ as input, while leaking all those indices that are flagged, until either $\mathscr{P}$ outputs or $\mathscr{P}$ has incurred $(\frac{6}{\delta'-\delta})cc(\mathscr{P}, n)$ bits of communication ;
/* when leaking index i, both x'_i and y'_i will be given to $\mathscr{P}$ — this can be done since a leaked index here must correspond to a padded pair at Line 17 */;
22 **if** $\mathscr{P}$ *has incurred* $(\frac{6}{\delta'-\delta})cc(\mathscr{P}, n)$ *bits of communication* **then** exit with an arbitrary output ;
23 **else** output $\mathscr{P}$'s output and exit ;

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

be different from the distribution that the (real) leaker uses. Our proof will carefully take into account such discrepancy.

8.2 Complete reduction and final guarantees

Pseudo-code. Protocol 1 presents the protocol $\mathscr{Q}$ for solving $\Pi_{n'}$ without our leaker, as run by Alice. $\mathscr{Q}$ internally invokes the given two-party protocol $\mathscr{P}$, where $\mathscr{P}$ solves Π_n with our leaker. At Line 1–9, Alice and Bob first exchange sampled indices to estimate the occurrences of each leakable pattern. Next Line 10–12 calculate the amount of padding needed. Line 13–18 do the actual padding, and then for each leakable pattern, flag a certain number of padded pairs as "to be leaked". At Line 19–23, Alice and Bob do a random permutation to obtain $(\mathbf{X}, \mathbf{Y})$, and then invoke $\mathscr{P}$ on $(\mathbf{X}, \mathbf{Y})$ while leaking all those flagged pairs.

We will prove various properties of Protocol 1, which will ultimately lead to the proof for Theorem 3. These properties include Lemmas 2–6 and Theorems 5–6. In particular, we aim to prove that Protocol 1 is correct for all two-party permutation-invariant problem Π. For length-n string X, define $M(X) = x_{m_1} x_{m_2} \ldots x_{m_n}$, where M is any given permutation of 1 through n, and m_i is the ith integer in M. A two-party problem Π is *permutation-invariant* iff for all X, Y, and M, $\Pi(X, Y) = \Pi(M(X), M(Y))$. Throughout this subsection, we assume that Π is permutation invariant, and when we mention a line number (e.g., Line 5), we refer to the corresponding line of Protocol 1.

We first quantify the estimation quality on the occurrence counts of each leakable pattern as done by the protocol. For $1 \le j \le k$, let w_j denote the occurrence count of the jth leakable pattern in (X', Y'). The $\mathbf{v}_j$'s in Protocol 1 are essentially estimates for w_j. We say that *Protocol 1's estimates are good* if immediately after Line 9, $\max_{1 \le j \le k} (\mathbf{v}_j - w_j)^2 \le \frac{n'^2}{2s} \ln \frac{24k}{\delta - \delta'}$.

Lemma 2 *Protocol 1's estimates are good with probability at least* $1 - \frac{\delta' - \delta}{12}$.

Proof Let $\epsilon = \sqrt{\frac{n'^2}{2s} \ln \frac{24k}{\delta'-\delta}}$.

By the definition of good, it suffices to prove:

$$\Pr\left[\max_{1 \le j \le k} |\mathbf{v}_j - w_j| \le \epsilon\right] \ge 1 - \frac{\delta' - \delta}{12}$$

For any $j \in [1, k]$, let $\mathbf{s}_j$ be the number of times $\mathbf{v}_j$ is incremented by $\frac{n'}{s}$ in Line 7 of Protocol 1. Each time Line 3 through 7 is executed, $\mathbf{v}_j$ is incremented only when (x'_i, y'_i) is the jth leakable pattern. Since i is drawn uniformly at random from $[1, n']$ and since there exists exactly w_j indices $i \in [1, n']$ such that (x'_i, y'_i) is the jth leakable pattern, each time Line 3 through 7 is executed $\mathbf{v}_j$ is incremented with probability exactly $\frac{w_j}{n'}$. Since Line 3 through 7 is executed s times, $\mathbf{s}_j$ is the sum of s independent and identical Bernoulli random variables, with each Bernoulli trial having a success probability of $\frac{w_j}{n'}$.

We will apply the Chernoff–Hoeffding bound [16] for absolute error, which states for any $0 \le \frac{w_j}{n'} \le 1$ and $a \ge 0$,

$$\Pr\left(\frac{\mathbf{s}_j}{s} \ge \frac{w_j}{n'} + a\right) \le e^{-2a^2 s}$$
$$\Pr\left(\frac{\mathbf{s}_j}{s} \le \frac{w_j}{n'} - a\right) \le e^{-2a^2 s}$$

$\mathbf{v}_j$ is modified only in Line 1, where it is initially set to zero, and then in Line 7. Thus, $\mathbf{v}_j = \frac{\mathbf{s}_j n'}{s}$. Hence we have:

$$\begin{aligned}\Pr[|\mathbf{v}_j - w_j| > \epsilon] &= \Pr\left[|\frac{\mathbf{s}_j n'}{s} - w_j| > \epsilon\right] \\ &= \Pr\left[|\frac{\mathbf{s}_j}{s} - \frac{w_j}{n'}| > \frac{\epsilon}{n'}\right] \\ &\leq 2\exp\left(-2s\left(\frac{\epsilon}{n'}\right)^2\right) = \frac{\delta' - \delta}{12k}\end{aligned}$$

Finally, taking a union bound for j from 1 through k, we have:

$$\Pr\left[\max_{1\leq j\leq k} |\mathbf{v}_j - w_j| > \epsilon\right] \leq k \times \frac{\delta' - \delta}{12k} \leq \frac{\delta' - \delta}{12}$$

□

Protocol 1 has (X', Y') as its inputs to Alice and Bob, respectively. It internally converts (X', Y') to a randomized input $(\mathbf{X}, \mathbf{Y})$. For any given (X, Y), conditioned upon $(\mathbf{X}, \mathbf{Y}) = (X, Y)$, we define $\hat{\mathbb{T}}(X, Y)$ to be the distribution of the *leaked sets*, as induced by Protocol 1 at Line 21. Here a *leaked set* is the set $\{(i, x_i, y_i) \mid \text{index } i \text{ is leaked}\}$. Define $\mathbb{T}(X, Y)$ to be the distribution of the leaked sets that would have resulted, if (X, Y) were subjected to the (real) leaker. Alice is using $\hat{\mathbb{T}}(X, Y)$ to approximate $\mathbb{T}(X, Y)$. We thus want to quantify the distance between these two distributions.

Consider any two distributions $\mathbb{D}$ and $\hat{\mathbb{D}}$ over the same sample space $\mathscr{D}$. We define their L_1 *distance* (denoted as $||\mathbb{D} - \hat{\mathbb{D}}||$ and also called *variation distance*) to be $\int_{x\in\mathscr{D}} |f_{\mathbb{D}}(x) - f_{\hat{\mathbb{D}}}(x)|dx$ if $\mathscr{D}$ is continuous, and $\sum_{x\in\mathscr{D}} |f_{\mathbb{D}}(x) - f_{\hat{\mathbb{D}}}(x)|$ if $\mathscr{D}$ is discrete. Here $f_{\mathbb{D}}$ and $f_{\hat{\mathbb{D}}}$ are the density functions of the two distributions, respectively. The following lemma (whose proof is in "Appendix B") quantifies the L_1 distance between $\hat{\mathbb{T}}(X, Y)$ and $\mathbb{T}(X, Y)$.

Lemma 3 *If Protocol 1's estimates are good and if it does not exit at Line 12, then for all (X, Y), we have $||\hat{\mathbb{T}}(X, Y) - \mathbb{T}(X, Y)|| \leq \frac{9(\delta'-\delta)}{12}$.*

Lemma 4 *If Protocol 1's estimates are good and if it does not exit at Line 12, then for all (X, Y) in the support of $(\mathbf{X}, \mathbf{Y})$, we have:*

$$\begin{aligned}&E_{\mathbf{C}_{\mathscr{Q}}}[err(\mathscr{Q}, X', Y', \mathbf{C}_{\mathscr{Q}})|(\mathbf{X}, \mathbf{Y}) = (X, Y)] \\ &\quad\leq \delta + \frac{11}{12}(\delta' - \delta) \qquad (1) \\ &E_{\mathbf{C}_{\mathscr{Q}}}[cc(\mathscr{Q}, X', Y', \mathbf{C}_{\mathscr{Q}})|(\mathbf{X}, \mathbf{Y}) = (X, Y)] \\ &\quad\leq \frac{\mathfrak{R}_{\delta'}(\Pi_{n'})}{2} + 5.5cc(\mathscr{P}, n) \qquad (2)\end{aligned}$$

Proof Recall that given input (X', Y'), Protocol 1 invokes $\mathscr{P}$ internally. When $(\mathbf{X}, \mathbf{Y}) = (X, Y)$, Protocol 1 invokes $\mathscr{P}$ with input (X, Y). The input (X, Y) is obtained by (i) padding some extra leakable patterns to (X', Y') at Line 17, and (ii) doing a permutation at Line 20.

– Proof for Inequality 1. Consider any (X, Y) in the support of $(\mathbf{X}, \mathbf{Y})$. By the definition of leakable patterns and permutation-invariant functions, $\Pi((X', Y')) = \Pi((X, Y))$. Hence Protocol 1's result must be correct if (i) Protocol 1 does not exit at Line 12, (ii) Protocol 1 does not exit in Line 23 due to $\mathscr{P}$ incurring more than $\frac{6}{\delta'-\delta}cc(\mathscr{P}, n)$ bits of communication, and (iii) $\mathscr{P}$ gives the correct result for (X, Y). Recall that $\hat{\mathbb{T}}(X, Y)$ is defined to be the distribution of the leaked set fed into $\mathscr{P}$ by Protocol 1, while $\mathbb{T}(X, Y)$ is defined to be the distribution of the leaked set that would have been generated by the leaker for (X, Y). For clarity, we write them as $\hat{\mathbb{T}}$ and $\mathbb{T}$. Define $\mathbb{C}_{\mathscr{Q}}$ and $\mathbb{C}_{\mathscr{P}}$ to be the distribution of coin flips made by $\mathscr{Q}$ and $\mathscr{P}$, respectively. Define $cc(\mathscr{P}, X, Y, \mathbf{C}_{\mathscr{P}}, \mathbf{T})$ to be the communication incurred (in terms of number of bits) by $\mathscr{P}$, under the input (X, Y), protocol's coin flip outcomes $\mathbf{C}_{\mathscr{P}}$, and leaked set $\mathbf{T}$. Note that $\mathbf{T}$ captures all coins flipped by the leaker. We similarly define $\text{err}(\mathscr{P}, X, Y, \mathbf{C}_{\mathscr{P}}, \mathbf{T})$, which is 1 if the $\mathscr{P}$'s output is wrong, and 0 otherwise. From the condition of the lemma, we already know that Protocol 1 does not exit at Line 12. We have:

$$\begin{aligned}&\Pr_{\mathbf{C}_{\mathscr{Q}}\sim\mathbb{C}_{\mathscr{Q}}}[\text{err}(\mathscr{Q}, X', Y', \mathbf{C}_{\mathscr{Q}}) = 1|(\mathbf{X}, \mathbf{Y}) = (X, Y)] \\ &= \Pr_{\mathbf{C}_{\mathscr{P}}\sim\mathbb{C}_{\mathscr{P}}, \mathbf{T}\sim\hat{\mathbb{T}}}\Big[(\text{err}(\mathscr{P}, X, Y, \mathbf{C}_{\mathscr{P}}, \mathbf{T}) = 1) \text{ or} \\ &\quad (cc(\mathscr{P}, X, Y, \mathbf{C}_{\mathscr{P}}, \mathbf{T}) > \frac{6}{\delta' - \delta}cc(\mathscr{P}, n))\Big] \\ &\leq ||\hat{\mathbb{T}} - \mathbb{T}|| + \Pr_{\mathbf{C}_{\mathscr{P}}\sim\mathbb{C}_{\mathscr{P}}, \mathbf{T}\sim\mathbb{T}}\Big[(\text{err}(\mathscr{P}, X, Y, \mathbf{C}_{\mathscr{P}}, \mathbf{T}) = 1) \text{ or} \\ &\quad (cc(\mathscr{P}, X, Y, \mathbf{C}_{\mathscr{P}}, \mathbf{T}) > \frac{6}{\delta' - \delta}cc(\mathscr{P}, n))\Big] \\ &\leq ||\hat{\mathbb{T}} - \mathbb{T}|| + \Pr_{\mathbf{C}_{\mathscr{P}}\sim\mathbb{C}_{\mathscr{P}}, \mathbf{T}\sim\mathbb{T}}[\text{err}(\mathscr{P}, X, Y, \mathbf{C}_{\mathscr{P}}, \mathbf{T}) = 1] \\ &\quad + \Pr_{\mathbf{C}_{\mathscr{P}}\sim\mathbb{C}_{\mathscr{P}}, \mathbf{T}\sim\mathbb{T}}\Big[cc(\mathscr{P}, X, Y, \mathbf{C}_{\mathscr{P}}, \mathbf{T}) \\ &\quad > \frac{6}{\delta' - \delta}cc(\mathscr{P}, n)\Big] \\ &\leq ||\hat{\mathbb{T}} - \mathbb{T}|| + \delta + \frac{\delta' - \delta}{6} \\ &\leq \delta + \frac{11}{12}(\delta' - \delta) \quad \text{(by Lemma 3)}\end{aligned}$$

– Proof for Inequality 2. Protocol 1's communication only involves two parts. The first part is for taking $\frac{\mathfrak{R}_{\delta'}(\Pi_{n'})}{4\log q}$ samples in Step 1, which incurs at most $\frac{\mathfrak{R}_{\delta'}(\Pi_{n'})}{4\log q} \times 2\log q = \frac{\mathfrak{R}_{\delta'}(\Pi_{n'})}{2}$ bits. The second part is for

invoking $\mathscr{P}$, incurring no more than $\frac{6}{\delta'-\delta}\mathrm{cc}(\mathscr{P},n)$ bits. We have:

$$\begin{aligned}
&E_{\mathbf{C}_{\mathscr{Q}}}[\mathrm{cc}(\mathscr{Q},X',Y',\mathbf{C}_{\mathscr{Q}})|(\mathbf{X},\mathbf{Y})=(X,Y)]\\
&\le \frac{\mathfrak{R}_{\delta'}(\Pi_{n'})}{2}\\
&\quad +E_{\mathbf{C}_{\mathscr{P}}\sim\mathbb{C}_{\mathscr{P}},\mathbf{T}\sim\hat{\mathbb{T}}}\left[\min(\mathrm{cc}(\mathscr{P},X,Y,\mathbf{C}_{\mathscr{P}},\mathbf{T}),\right.\\
&\quad \left.\frac{6}{\delta'-\delta}\mathrm{cc}(\mathscr{P},n))\right]\\
&\le \frac{\mathfrak{R}_{\delta'}(\Pi_{n'})}{2}+||\hat{\mathbb{T}}-\mathbb{T}||\times\frac{6}{\delta'-\delta}\mathrm{cc}(\mathscr{P},n)\\
&\quad +E_{\mathbf{C}_{\mathscr{P}}\sim\mathbb{C}_{\mathscr{P}},\mathbf{T}\sim\mathbb{T}}[\mathrm{cc}(\mathscr{P},X,Y,\mathbf{C}_{\mathscr{P}},\mathbf{T})]\\
&\le \frac{\mathfrak{R}_{\delta'}(\Pi_{n'})}{2}+\frac{9(\delta'-\delta)}{12}\\
&\quad \times\frac{6}{\delta'-\delta}\mathrm{cc}(\mathscr{P},n)+\mathrm{cc}(\mathscr{P},n)\\
&= \frac{\mathfrak{R}_{\delta'}(\Pi_{n'})}{2}+5.5\mathrm{cc}(\mathscr{P},n)
\end{aligned}$$

We invoked Lemma 3 to obtain the last inequality. □

Theorem 5 *For all permutation-invariant problem Π, all constants δ and δ' such that $0<\delta<\delta'<0.5$, and all n and n' such that $n\ge n'+2kn'+\frac{500}{(\delta'-\delta)^2}\times\left(k^3+\frac{2k^2n'^2}{\mathfrak{R}_{\delta'}(\Pi_{n'})}(\log q)(\ln\frac{24k}{\delta'-\delta})\right)$, we have $\mathfrak{L}_{\delta}(\Pi_n)\ge\frac{1}{14}\mathfrak{R}_{\delta'}(\Pi_{n'})$.*

Proof For any given protocol $\mathscr{P}$ for solving Π_n with the leaker and with error δ, we construct a protocol $\mathscr{Q}$ for solving $\Pi_{n'}$ without our leaker and with error δ' as in Protocol 1. It is easy to verify that n is large enough such that Protocol 1 does not exit at Line 12:

$$\begin{aligned}
n &\ge n'+2kn'\\
&\quad +\frac{500}{(\delta'-\delta)^2}\left(k^3+\frac{2k^2n'^2}{\mathfrak{R}_{\delta'}(\Pi_{n'})}(\log q)\left(\ln\frac{24k}{\delta'-\delta}\right)\right)\\
&= n'\\
&\quad +k\left(2n'+\frac{500}{(\delta'-\delta)^2}\left(k^2+\frac{2kn'^2}{\mathfrak{R}_{\delta'}(\Pi_{n'})}(\log q)\left(\ln\frac{24k}{\delta'-\delta}\right)\right)\right)\\
&= n'+kh
\end{aligned}$$

Denote z as the event that Protocol 1's estimates are good. By Lemma 2, $\Pr[z]\ge 1-\frac{\delta'-\delta}{12}$. Consider any given input (X',Y') to our reduction protocol in Protocol 1 and the corresponding random variables $(\mathbf{X},\mathbf{Y})$ obtained at Line 20 of Protocol 1. By Lemma 4, we have:

$$\begin{aligned}
&\Pr[\mathrm{err}(\mathscr{Q},X',Y',\mathbf{C}_{\mathscr{Q}})=1|z]\\
&= \sum_{(X,Y)}\Pr[\mathrm{err}(\mathscr{Q},X',Y',\mathbf{C}_{\mathscr{Q}})=1|(\mathbf{X},\mathbf{Y})=(X,Y),z]\\
&\quad\times\Pr[(\mathbf{X},\mathbf{Y})=(X,Y)|z]\\
&\le \sum_{(X,Y)}\left(\delta+\frac{11}{12}(\delta'-\delta)\right)\times\Pr[(\mathbf{X},\mathbf{Y})=(X,Y)|z]\\
&= \delta+\frac{11}{12}(\delta'-\delta)E_{\mathbf{C}_{\mathscr{Q}}}[\mathrm{cc}(\mathscr{Q},X',Y',\mathbf{C}_{\mathscr{Q}})|z]\\
&= \sum_{(X,Y)}E_{\mathbf{C}_{\mathscr{Q}}}[\mathrm{cc}(\mathscr{Q},X',Y',\mathbf{C}_{\mathscr{Q}})|(\mathbf{X},\mathbf{Y})=(X,Y),z]\\
&\quad\times\Pr[(\mathbf{X},\mathbf{Y})=(X,Y)|z]\\
&\le \sum_{(X,Y)}\left(\frac{\mathfrak{R}_{\delta'}(\Pi_{n'})}{2}+5.5\mathrm{cc}(\mathscr{P},n)\right)\\
&\quad\times\Pr[(\mathbf{X},\mathbf{Y})=(X,Y)|z]\\
&= \frac{\mathfrak{R}_{\delta'}(\Pi_{n'})}{2}+5.5\mathrm{cc}(\mathscr{P},n)
\end{aligned}$$

Now we consider the case where z does not hold, i.e., the protocol's estimates are not good. Although most our previous technical lemmas no longer hold, we still know that the error probability is at most 1, and the communication cost, by our protocol design, is at most $\frac{\mathfrak{R}_{\delta'}(\Pi_{n'})}{4\log q}\cdot 2\log q+\frac{6}{\delta'-\delta}\mathrm{cc}(\mathscr{P},n)$ bits. Hence we have:

$$\begin{aligned}
&\Pr[\mathrm{err}(\mathscr{Q},X',Y',\mathbf{C}_{\mathscr{Q}})=1]\\
&\quad\le \Pr[\mathrm{err}(\mathscr{Q},X',Y',\mathbf{C}_{\mathscr{Q}})=1|z]+(1-\Pr[z])\\
&\quad\le \delta+\frac{11}{12}(\delta'-\delta)+\frac{1}{12}(\delta'-\delta)=\delta'\\
&E_{\mathbf{C}_{\mathscr{Q}}}[\mathrm{cc}(\mathscr{Q},X',Y',\mathbf{C}_{\mathscr{Q}})]\\
&\quad\le E_{\mathbf{C}_{\mathscr{Q}}}[\mathrm{cc}(\mathscr{Q},X',Y',\mathbf{C}_{\mathscr{Q}})|z]\\
&\qquad +(1-\Pr[z])\left(\frac{\mathfrak{R}_{\delta'}(\Pi_{n'})}{4\log q}\cdot 2\log q+\frac{6}{\delta'-\delta}\mathrm{cc}(\mathscr{P},n)\right)\\
&\quad\le 5.5\mathrm{cc}(\mathscr{P},n)+\frac{\mathfrak{R}_{\delta'}(\Pi_{n'})}{2}\\
&\qquad +\frac{\delta'-\delta}{12}\left(\frac{\mathfrak{R}_{\delta'}(\Pi_{n'})}{4\log q}\cdot 2\log q+\frac{6}{\delta'-\delta}\mathrm{cc}(\mathscr{P},n)\right)\\
&\quad\le 6\mathrm{cc}(\mathscr{P},n)+\frac{25}{48}\mathfrak{R}_{\delta'}(\Pi_{n'})
\end{aligned}$$

Since $\mathscr{Q}$ solves $\Pi_{n'}$ with at most δ' error, we have $6\mathrm{cc}(\mathscr{P},n)+\frac{25}{48}\mathfrak{R}_{\delta'}(\Pi_{n'})\ge\mathrm{cc}(\mathscr{Q},n')\ge\mathfrak{R}_{\delta'}(\Pi_{n'})$. Let $\mathscr{P}$ be the optimal protocol for solving Π_n with the leaker and with error δ, we have $\mathfrak{L}_{\delta}(\Pi_n)=\mathrm{cc}(\mathscr{P},n)\ge\frac{1}{14}\mathfrak{R}_{\delta'}(\Pi_{n'})$. □

Lemma 5 *For any given two-party problem Π, any given constants δ and δ' such that $0<\delta<\delta'<0.5$, we have $\mathfrak{R}_{\delta}(\Pi)\le\frac{\ln(1/\delta)}{2(0.5-\delta')^2}\mathfrak{R}_{\delta'}(\Pi)$.*

Proof Given a protocol $\mathscr{P}$ for Π with error δ', we will construct a protocol with error at most δ as follows: we invoke $\mathscr{P}$ for $\frac{\ln(1/\delta)}{2(0.5-\delta')^2}$ times, and take the majority of these outputs as the final output. Let random variable $\mathbf{z}$ denote the fraction of correct outputs. Since $E[\mathbf{z}]\ge 1-\delta'$, by the Chernoff–Hoeffding bound [16] for absolute error, we have:

$$\Pr[\mathbf{z} \le 0.5] \le \Pr[\mathbf{z} \le (E[\mathbf{z}] - (1 - \delta' - 0.5))] \le \exp\left(-2\frac{\ln(1/\delta)}{2(0.5-\delta')^2}(1-\delta'-0.5)^2\right) = \delta$$

□

Theorem 6 *For all permutation-invariant problem Π, all constants $\delta \in (0, \frac{1}{2})$, all n and n' such that $n \ge n' + 2kn' + \frac{500}{(0.25-0.5\delta)^4}(\ln\frac{1}{\delta})\left(k^3 + \frac{k^2 n'^2}{\mathfrak{R}_\delta(\Pi_{n'})}(\log q)\left(\ln\frac{48k}{0.5-\delta}\right)\right)$, we have $\mathfrak{L}_\delta(\Pi_n) \ge \frac{(0.25-0.5\delta)^2}{7\ln(1/\delta)}\mathfrak{R}_\delta(\Pi_{n'})$.*

Proof Obviously, $0 < \delta < 0.5\delta + 0.25 < 0.5$. Apply Lemma 5 (with the values of δ and δ' in Lemma 5 being set to δ and $0.5\delta + 0.25$, respectively) and we have:

$$\begin{aligned}
n &\ge n' + 2kn' + \frac{500}{(0.25-0.5\delta)^4} \times \left(\ln\frac{1}{\delta}\right)\left(k^3 + \frac{k^2 n'^2}{\mathfrak{R}_\delta(\Pi_{n'})}(\log q)\left(\ln\frac{48k}{0.5-\delta}\right)\right) \\
&\ge n' + 2kn' + \frac{500}{(0.25-0.5\delta)^2} \times \left(k^3 + \frac{k^2 n'^2 \ln\frac{1}{\delta}}{(0.25-0.5\delta)^2\mathfrak{R}_\delta(\Pi_{n'})}(\log q)\left(\ln\frac{48k}{0.5-\delta}\right)\right) \\
&> n' + 2kn' + \frac{500}{(0.25-0.5\delta)^2} \times \left(k^3 + \frac{2k^2 n'^2 \ln\frac{1}{\delta}(\log q)}{2(0.5-(0.5\delta+0.25))^2\mathfrak{R}_\delta(\Pi_{n'})}\left(\ln\frac{48k}{0.5-\delta}\right)\right) \\
&\ge n' + 2kn' + \frac{500}{((0.5\delta+0.25)-\delta)^2} \times \left(k^3 + \frac{2k^2 n'^2}{\mathfrak{R}_{0.5\delta+0.25}(\Pi_{n'})}(\log q)\left(\ln\frac{24k}{(0.5\delta+0.25)-\delta}\right)\right)
\end{aligned}$$

The above inequality shows that n satisfies the condition needed for Theorem 5. Invoke Theorem 5 and we have $\mathfrak{L}_\delta(\Pi_n) \ge \frac{1}{14}\mathfrak{R}_{0.5\delta+0.25}(\Pi_{n'})$. Applying Lemma 5 a second time (again with the values of δ and δ' in Lemma 5 being set to δ and $0.5\delta + 0.25$, respectively) yields $\mathfrak{L}_\delta(\Pi_n) \ge \frac{1}{14}\mathfrak{R}_{0.5\delta+0.25}(\Pi_{n'}) \ge \frac{(0.25-0.5\delta)^2}{7\ln(1/\delta)}\mathfrak{R}_\delta(\Pi_{n'})$. □

Lemma 6 *For all $q \ge 2$ and all δ where $0 < \delta < 0.5$, we have $2(\ln\frac{140}{0.5-\delta})(\log q) > \ln(\frac{48q^2}{0.5-\delta})$.*

Proof

$$\begin{aligned}
&\ln\left(\frac{48q^2}{0.5-\delta}\right) \\
&= \ln q^2 + \ln\left(\frac{48}{0.5-\delta}\right) = \left(\ln q^2\right)\left(1 + \frac{\ln\left(\frac{48}{0.5-\delta}\right)}{\ln q^2}\right) \\
&\le 2(\ln q)\left(1 + \ln\left(\frac{48}{0.5-\delta}\right)\right) \\
&< 2\left(\ln\frac{140}{0.5-\delta}\right)(\ln q) < 2\left(\ln\frac{140}{0.5-\delta}\right)(\log q)
\end{aligned}$$

□

Theorem 3 *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 $n' = c_2\sqrt{n}/(q^{1.5}\log q)$, we have $\mathfrak{L}_\delta(\text{GDC}_n^{g,q}) \ge c_1\mathfrak{R}_\delta(\text{GDC}_{n'}^{g,q})$.*

Proof Let $c_1 = \frac{(0.25-0.5\delta)^2}{7\ln(1/\delta)}$, and let c_2 be the positive constant such that:

$$\frac{1}{2{c_2}^2} = 3 + \frac{4000}{(0.25-0.5\delta)^4}\left(\ln\frac{1}{\delta}\right)\left(\ln\frac{140}{0.5-\delta}\right)$$

We will show that c_1 and c_2 satisfy the desired properties in the theorem. Consider any n, g, q, and $n' = c_2\sqrt{n}/(q^{1.5}\log q)$. If $n' < 1$, then trivially $\mathfrak{L}_\delta(\text{GDC}_n^{g,q}) \ge c_1\mathfrak{R}_\delta(\text{GDC}_{n'}^{g,q}) = 0$.

Otherwise we aim to invoke Theorem 6 with $\Pi_n = \text{GDC}_n^{g,q}$. The GDC problem is obviously permutation-invariant. From the cycle promise in $\text{GDC}_n^{g,q}$, we further know that $k \le 2q$. We thus have:

$$\begin{aligned}
n &= \frac{q^3 n'^2 \log^2 q}{{c_2}^2} > \frac{q^3 + q^2 n'^2 \log^2 q}{2{c_2}^2} \\
&\ge \frac{1}{2{c_2}^2}\left(q^3 + \frac{q^2 n'^2 \log^2 q}{\mathfrak{R}_\delta(\Pi_{n'})}\right) \\
&> n' + 2q^2 n' + \frac{4000}{(0.25-0.5\delta)^4}\left(\ln\frac{1}{\delta}\right)\left(\ln\frac{140}{0.5-\delta}\right)\left(q^3 + \frac{q^2 n'^2 \log^2 q}{\mathfrak{R}_\delta(\Pi_{n'})}\right) \\
&> n' + 2q^2 n' + \frac{500}{(0.25-0.5\delta)^4}\left(\ln\frac{1}{\delta}\right)\left(\ln\frac{140}{0.5-\delta}\right)\left(8q^3 + \frac{8q^2 n'^2 \log^2 q}{\mathfrak{R}_\delta(\Pi_{n'})}\right) \\
&> n' + 2kn' + \frac{500}{(0.25-0.5\delta)^4}\left(\ln\frac{1}{\delta}\right)\left(k^3 + \frac{k^2 n'^2 \log^2 q}{\mathfrak{R}_\delta(\Pi_{n'})} \times \left(2\ln\frac{140}{0.5-\delta}\right)\right) \\
&> n' + 2kn' + \frac{500}{(0.25-0.5\delta)^4}\left(\ln\frac{1}{\delta}\right)\left(k^3 + \frac{k^2 n'^2 \log q}{\mathfrak{R}_\delta(\Pi_{n'})} \times \ln\left(\frac{48q^2}{0.5-\delta}\right)\right) \\
&> n' + 2kn' + \frac{500}{(0.25-0.5\delta)^4}\left(\ln\frac{1}{\delta}\right)\left(k^3 + \frac{k^2 n'^2 \log q}{\mathfrak{R}_\delta(\Pi_{n'})}\ln\left(\frac{48k}{0.5-\delta}\right)\right)
\end{aligned}$$

The second last inequality in the above is by Lemma 6. The above shows that n satisfies the condition needed by Theorem 6. Invoking Theorem 6 then immediately gives $\mathfrak{L}_\delta(\text{GDC}_n^{g,q}) \geq c_1 \mathfrak{R}_\delta(\text{GDC}_{n'}^{g,q})$. □

9 CONSENSUS lower bound under oblivious adversaries

This section will ultimately prove our final theorem on CONSENSUS under oblivious adversaries, as follows:

Theorem 7 *If the nodes only know a poor estimate m' for m where $|\frac{m'-m}{m}|$ is at least $\frac{1}{3}$, 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.*

We emphasize that the only interface between this section and all previous sections is Theorems 2 and 4—we will only apply those two theorems as black boxes. In particular, Protocol 1 and its analysis will no longer be relevant, and it will be convenient for the reader to forget about those.

In the remainder of this section, we will sometimes refer to round 0, where the CONSENSUS protocol does nothing and where every node is in the receiving state.

9.1 Proof overview

Consider any CONSENSUS protocol $\mathscr{P}$ with $\frac{1}{10}$ error. Let $\text{tc}(d, m)$ denote $\mathscr{P}$'s time complexity when running over dynamic networks controlled by oblivious adversaries and with d diameter and m nodes. As explained in Sect. 5, the crux will be to prove $\text{tc}(8, m) \geq m^{\frac{1}{12}}$. To do so, we consider the $\text{GDC}_n^{g,q}$ problem with our leaker, and set $n = \frac{m-4}{3}$, $q = 20\text{tc}(8, m) + 21$, and $g = 15q \ln q$. To solve the $\text{GDC}_n^{g,q}(X, Y)$ problem with the help from our leaker, Alice and Bob simulate $\mathscr{P}$ in the following way: In the simulation, the input (X, Y), together with the leaked information (given by the leaker), is mapped to a *sanitized* adaptive adversary $\mathscr{A}$ that determines the topology of the dynamic network. Roughly speaking, if $\text{GDC}_n^{g,q}(X, Y) = 1$, the resulting dynamic network will have a diameter of 8. Even though $\mathscr{A}$ is an adaptive adversary, by Theorem 2 in Sect. 7, $\mathscr{P}$'s time complexity should remain $\text{tc}(d, m)$ under $\mathscr{A}$. Hence $\mathscr{P}$ should decide within $\text{tc}(8, m)$ rounds on expectation. If $\text{GDC}_n^{g,q}(X, Y) = 0$, then the resulting dynamic network will have a diameter of $\Theta(q)$. For $\mathscr{P}$ to decide in this dynamic network, we prove that it takes at least roughly $\frac{q}{2}$ rounds. Note that $\frac{q}{2} > 10\text{tc}(8, m)$—in other words, it takes longer for $\mathscr{P}$ to decide if $\text{GDC}_n^{g,q}(X, Y) = 0$. Alice and Bob do not know the other party's input, and hence do not have full knowledge of the dynamic network. But the help from our leaker enables them to still properly simulate $\mathscr{P}$'s execution. Finally, if $\mathscr{P}$ decides within $10\text{tc}(8, m)$ rounds, Alice and Bob claim that $\text{GDC}_n^{g,q}(X, Y) = 1$. Otherwise they claim $\text{GDC}_n^{g,q}(X, Y) = 0$. Our proof will show that to solve $\text{GDC}_n^{g,q}$ with our leaker, using the above simulation, Alice and Bob incur $\Theta(\text{tc}(8, m) \cdot \log n)$ bits of communication. We thus have $\Theta(\text{tc}(8, m) \log n) \geq \mathfrak{L}_\delta(\text{GDC}_n^{g,q})$. Together with the lower bound on $\mathfrak{L}_\delta(\text{GDC}_n^{g,q})$ from Theorem 4 in Sect. 8, this will lead to a lower bound on $\text{tc}(8, m)$.

Roadmap for the remainder of this section. We will reduce $\text{GDC}_n^{g,q}$ with our leaker to CONSENSUS. Here we start with three separate and independent parties: Alice, Bob, and the leaker. Alice and Bob are trying to solve $\text{GDC}_n^{g,q}(X, Y)$, with the help from the leaker. Before Alice and Bob begin, for each leakable pair (x_i, y_i), with half probability the leaker leaks the index i. Recall that leaking the index i means that the leaker lets both Alice and Bob know for free the values of i, x_i, and y_i. Hence initially Alice knows X, as well as all the leaked information. Similarly Bob will initially knows Y, as well as all the leaked information. Once Alice and Bob begin, the leaker does nothing anymore.

In the reduction, Alice and Bob will effectively simulate (i) an adversary based on X, Y, and the leaked information, and (ii) the execution of some black-box CONSENSUS protocol $\mathscr{P}$ over a dynamic network determined by such an adversary. The details of the simulation are rather technical, and we will elaborate in the following steps:

- We first elaborate the adversary used in the simulation.
- We then describe the simulation done by Alice and Bob, and prove several guarantees of the simulation.
- Finally, we put everything together to prove Theorem 7.

9.2 Reference adversary

This section describes how we map X, Y, and the leaked information into the adversary used in the simulation, which we call the *reference adversary*.

Overview. We start with an overview. First, Alice and Bob convert the input (X, Y) into the *processed input* $(\mathbf{X}', \mathbf{Y}')$, using public coin flips and without any communication. Note that unlike (X, Y), the processed input $(\mathbf{X}', \mathbf{Y}')$ is a random variable, since its value depends on the public coin flips.

Next, the processed input $(\mathbf{X}', \mathbf{Y}')$ is mapped into a reference adversary. The reference adversary determines the dynamic network with the following properties:

- If $\text{GDC}_n^{g,q}(X, Y) = 0$, then the dynamic network depends on the pairs $(\mathbf{x}_1, \mathbf{y}_1)$ through $(\mathbf{x}_n, \mathbf{y}_n)$—namely, the entirety of $(\mathbf{X}', \mathbf{Y}')$. The resulting dynamic network will have $3n + 4$ nodes and $\Theta(q)$ diameter.
- If $\text{GDC}_n^{g,q}(X, Y) = 1$, then the dynamic network depends only on the pairs $(\mathbf{x}_1, \mathbf{y}_1)$ through $(\mathbf{x}_{\frac{n}{2}}, \mathbf{y}_{\frac{n}{2}})$—namely, the

first half of $(\mathbf{X}', \mathbf{Y}')$. The resulting dynamic network will have $\frac{3n}{2} + 2$ nodes and diameter of 8.

9.2.1 Preprocessing

Alice and Bob will process the input (X, Y) to obtain $(\mathbf{X}', \mathbf{Y}')$ in the following way, without involving any communication:

1. Alice and Bob use public coins to generate a uniformly random permutation π, and set $\mathbf{X}' = \pi(X)$ and $\mathbf{Y}' = \pi(Y)$, respectively.
2. Next for each i $(1 \leq i \leq n)$, Alice and Bob use public coins to draw an independent random integer $\mathbf{o}_i$ as an *offset*, such that[8] $\Pr[\mathbf{o}_i = 0] = \frac{1}{2}$ and $\Pr[\mathbf{o}_i = 2j] = \frac{1}{q-1}$ for $1 \leq j \leq \frac{q-1}{2}$. Alice and Bob then set $\mathbf{x}'_i = \min(\mathbf{x}'_i + \mathbf{o}_i, q-1)$ and $\mathbf{y}'_i = \min(\mathbf{y}'_i + \mathbf{o}_i, q-1)$, respectively. (We will explain later in this section why the offset is needed.)

Figure 3 gives two examples on how Alice and Bob process (X, Y) to obtain $(\mathbf{X}', \mathbf{Y}')$.

Intuition behind preprocessing. Let us define $\text{left}(\mathbf{X}', \mathbf{Y}') = (\mathbf{x}'_1 \dots \mathbf{x}'_{\frac{n}{2}}, \mathbf{y}'_1 \dots \mathbf{y}'_{\frac{n}{2}})$ and $\text{right}(\mathbf{X}', \mathbf{Y}') = (\mathbf{x}'_{\frac{n}{2}+1} \dots \mathbf{x}'_n, \mathbf{y}'_{\frac{n}{2}+1} \dots \mathbf{y}'_n)$. Our preprocessing aims to achieve the following properties in $(\mathbf{X}', \mathbf{Y}')$.

If $\text{GDC}_n^{g,q}(X, Y) = 0$, our dynamic network will have two parts. The first part corresponds to $\text{left}(\mathbf{X}', \mathbf{Y}')$, while the second part corresponds to $\text{right}(\mathbf{X}', \mathbf{Y}')$. To ensure that the network has a diameter of $\Theta(q)$, we want both $\text{left}(\mathbf{X}', \mathbf{Y}')$ and $\text{right}(\mathbf{X}', \mathbf{Y}')$ to satisfy the following properties (for brevity, the following will only discuss $\text{left}(\mathbf{X}', \mathbf{Y}')$):

- We need $\text{left}(\mathbf{X}', \mathbf{Y}')$ to contain at least q occurrences of the $(0, 0)$ pattern. By the gap promise, (X, Y) contains at least g occurrences of the $(0, 0)$ pattern. We will later set $g = 15q \ln q$. However, such g occurrences may not be spread evenly across the first half and the second half of (X, Y).
  As the first step in our preprocessing, Alice and Bob use public coins to generate a random permutation π, and set $\mathbf{X}' = \pi(X)$ and $\mathbf{Y}' = \pi(Y)$, respectively. We can later easily show that with good probability, doing so will make $\text{left}(\mathbf{X}', \mathbf{Y}')$ contain at least $4q \ln q$ occurrences of the $(0, 0)$ pattern, immediately after the permutation step.
- We further need $\text{left}(\mathbf{X}', \mathbf{Y}')$ to contain a $(2, 2)$ pair, a $(4, 4)$ pair, …, and a $(q-1, q-1)$ pair. (Note that $\text{left}(\mathbf{X}', \mathbf{Y}')$ does not need to satisfy the cycle promise, and hence such pairs are possible.) These pairs are needed to later ensure that the dynamic network constructed by the adversary remains connected in each round. In order to have such pairs in $\text{left}(\mathbf{X}', \mathbf{Y}')$, the second step in our preprocessing is for Alice and Bob to add some random *offsets* to each character in $\text{left}(\mathbf{X}', \mathbf{Y}')$.
  Recall that $\text{left}(\mathbf{X}', \mathbf{Y}')$ is likely to contain at least $4q \ln q$ occurrences of the $(0, 0)$ pattern, immediately after the permutation step. Our hope is to change at least one of these pairs to become a $(2, 2)$ pair, at least one of these pairs to become a $(4, 4)$ pair, and so on. At the same time, we still want a sufficient number of $(0, 0)$ pairs to remain unchanged, so that at the end of the process, we still have q pairs of $(0, 0)$. To achieve this, the offset $\mathbf{o}_i$ is chosen such that $\Pr[\mathbf{o}_i = 0] = \frac{1}{2}$ and $\Pr[\mathbf{o}_i = 2j] = \frac{1}{q-1}$ for $1 \leq j \leq \frac{q-1}{2}$. Alice and Bob then add $\mathbf{o}_i$ to $\mathbf{x}'_i$ and $\mathbf{y}'_i$, respectively. Note that since Alice and Bob do not know which pairs are $(0, 0)$ pairs, they will end up adding offsets to all other pairs as well. This will not cause any problem in our simulation later.
  Finally, for convenience in discussion (rather than correctness), we do not want the characters in $(\mathbf{X}', \mathbf{Y}')$ to be larger than $q-1$. Hence if adding the offset makes a character to be larger than $q-1$, we simply set the character to be $q-1$.

If $\text{GDC}_n^{g,q}(X, Y) = 1$, our dynamic network will only have one part, which corresponds to $\text{left}(\mathbf{X}', \mathbf{Y}')$. To ensure that the dynamic network is connected, we will need $\text{left}(\mathbf{X}', \mathbf{Y}')$ to contain at least one $(q-1, q-1)$ pair. This fortunately will have already been achieved by adding the offsets. Namely, regardless of (X, Y), some pair will likely become $(q-1, q-1)$ after adding the offset.

Formal properties of preprocessing. Define $|_b^a(X, Y)$, $|_b^a(\mathbf{X}', \mathbf{Y}')$, $|_b^a(\text{left}(\mathbf{X}', \mathbf{Y}'))$, and $|_b^a(\text{right}(\mathbf{X}', \mathbf{Y}'))$ to be the number of occurrences of the (a, b) pattern in (X, Y), $(\mathbf{X}', \mathbf{Y}')$, $\text{left}(\mathbf{X}', \mathbf{Y}')$, and $\text{right}(\mathbf{X}', \mathbf{Y}')$, respectively. We say that $(\mathbf{X}', \mathbf{Y}')$ is of *type*-0 if it satisfies all the following:

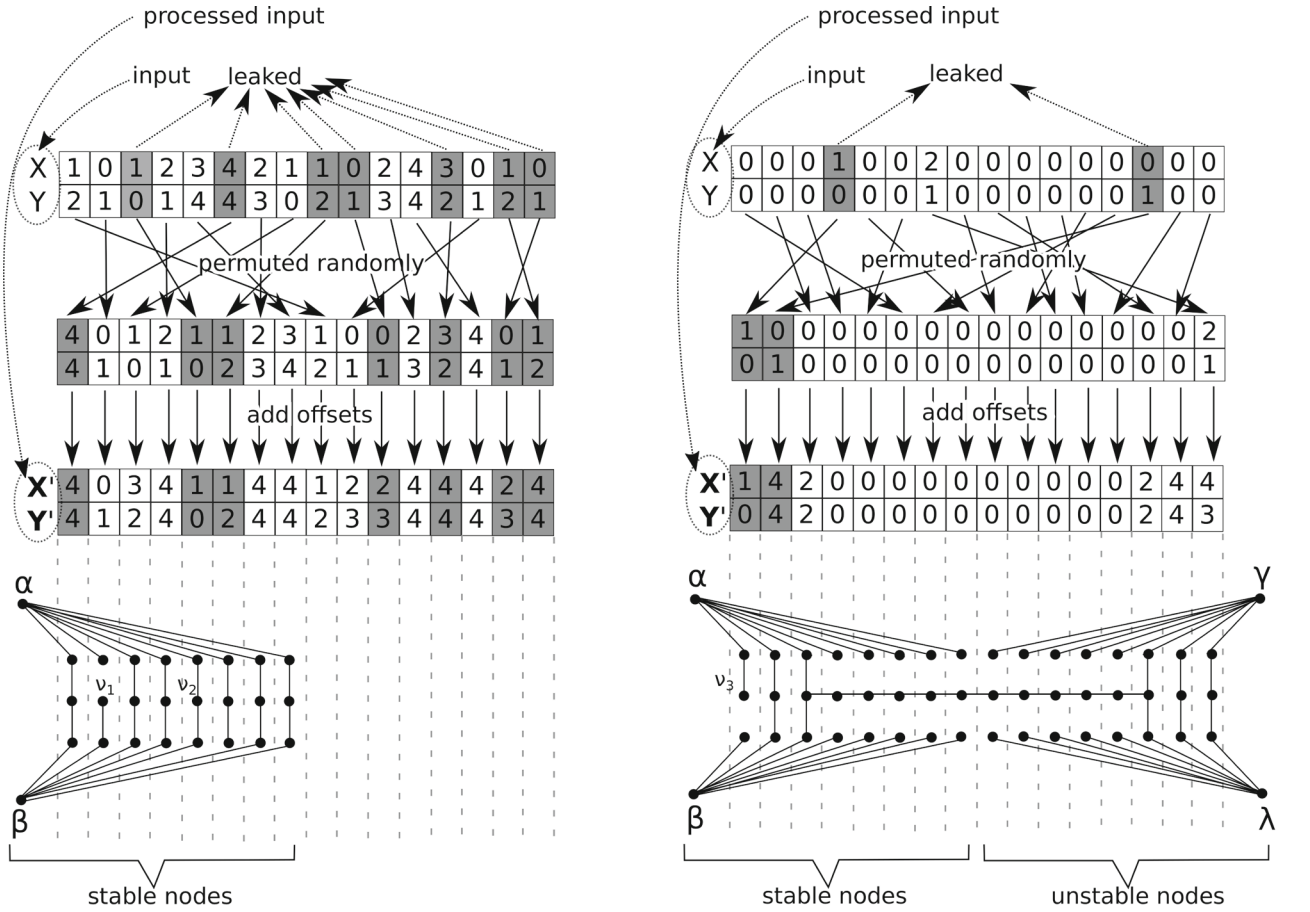
- $|_0^0(\text{left}(\mathbf{X}', \mathbf{Y}')) \geq q$ and $|_0^0(\text{right}(\mathbf{X}', \mathbf{Y}')) \geq q$.
- $|_{2j}^{2j}(\text{left}(\mathbf{X}', \mathbf{Y}')) \geq 1$ and $|_{2j}^{2j}(\text{right}(\mathbf{X}', \mathbf{Y}')) \geq 1$ for all j from 1 through $\frac{q-1}{2}$.

We say that $(\mathbf{X}', \mathbf{Y}')$ is of *type*-1 if it satisfies all the following:

- $|_{2j}^{2j}(\text{left}(\mathbf{X}', \mathbf{Y}')) = |_{2j}^{2j}(\text{right}(\mathbf{X}', \mathbf{Y}')) = 0$ for all j from 1 through $\frac{q-3}{2}$.
- $|_{q-1}^{q-1}(\text{left}(\mathbf{X}', \mathbf{Y}')) \geq 1$ and $|_{q-1}^{q-1}(\text{right}(\mathbf{X}', \mathbf{Y}')) \geq 1$.

It is possible that $(\mathbf{X}', \mathbf{Y}')$ is neither *type*-0 nor *type*-1. In such a case, we say that $(\mathbf{X}', \mathbf{Y}')$ is of *type*-$\perp$.

[8] Our reduction uses an odd value for q—hence $\frac{q-1}{2}$ is an integer.



(a) An example where $n = 16$, $g = 12$, $q = 5$, and $\text{GDC}_n^{g,q}(X, Y) = 1$. Here the processed input $(\mathbf{X}', \mathbf{Y}')$ happens to be of type-1. The dynamic network is constructed based only on left$(\mathbf{X}', \mathbf{Y}')$ (i.e., the first half of the processed input). In this example, v_1 and v_2 happen to be both sending in round 1

(b) An example where $n = 16$, $g = 12$, $q = 5$, and $\text{GDC}_n^{g,q}(X, Y) = 0$. Here the processed input $(\mathbf{X}', \mathbf{Y}')$ happens to be of type-0. The entire processed input is used to construct the dynamic network. In this example, v_3 happens to be receiving in round 1.

Fig. 3 Two examples of how the input (X, Y) is converted into the processed input $(\mathbf{X}', \mathbf{Y}')$, as well as the round 1 topologies of the resulting dynamic networks

Intuitively, if $\text{GDC}_n^{g,q}(X, Y) = 0$, we would hope $(\mathbf{X}', \mathbf{Y}')$ to be of *type*-0. Similarly, if $\text{GDC}_n^{g,q}(X, Y) = 1$, we would hope $(\mathbf{X}', \mathbf{Y}')$ to be of *type*-1. But since the preprocessing is a random process, there is no guarantee that this would happen. Still, the following lemma (proof deferred to "Appendix D") shows that we do get what we hope for, with at least $1 - \frac{1}{q}$ probability:

Lemma 7 *Consider any input* (X, Y) *of the* $\text{GDC}_n^{g,q}$ *problem and its corresponding processed input* $(\mathbf{X}', \mathbf{Y}')$. *For* $z \in \{0, 1\}$, *if* $q \geq 20$, $g \geq 15q \ln q$, $n \geq 4g$, *and* $\text{GDC}(X, Y) = z$, *then* $\Pr[(\mathbf{X}', \mathbf{Y}') \text{ is of type-}z] > 1 - \frac{1}{q}$.

9.2.2 The reference adversary based on processed input

We next define the *reference adversary* based on $(\mathbf{X}', \mathbf{Y}')$. Figure 3 illustrates the dynamic network determined by the reference adversary. We separately consider 3 cases, depending on the type of $(\mathbf{X}', \mathbf{Y}')$.

For type-1 $(\mathbf{X}', \mathbf{Y}')$. If $(\mathbf{X}', \mathbf{Y}')$ is of type-1, then the dynamic network will depend only on left$(\mathbf{X}', \mathbf{Y}')$ (i.e., the first half of $(\mathbf{X}', \mathbf{Y}')$). The dynamic network will have $\frac{3n}{2} + 2$ nodes, which are all called *stable nodes*. There are two special nodes α and β in the topology. For each index i $(1 \leq i \leq \frac{n}{2})$, there is a vertical chain consisting of three nodes (see Fig. 3a).

During round 0, on each chain, there is an edge connecting the top node and the middle node, and another edge connecting the middle node and the bottom node. The top nodes of all chains are connected to α, and the bottom nodes of all chains are connected to β (see Fig. 3). A chain for index i is called a $|_b^a$ chain if $\mathbf{x}'_i = a$ and $\mathbf{y}'_i = b$. If a is even, we call the top edge (i.e., the edge between the top node and the middle node on the chain) as an *even edge* on this chain. Similarly,

if b is even, the bottom edge is an *even edge*. We say a chain is *leaked* if the corresponding index is leaked by the leaker in the two-party GDC problem.

Starting from round 1, the adversary changes the topology in the following way:

- For every $|^{2t}_{2t-1}$ and $|^{2t-1}_{2t}$ chain ($1 \le t \le \frac{q-1}{2}$), the adversary removes the even edge on that chain at the beginning of round $t+1$.
- For every $|^{2t}_{2t+1}$ and $|^{2t+1}_{2t}$ chain ($1 \le t \le \frac{q-1}{2}$), the adversary removes the even edge on that chain at the beginning of round $t+1+(\mathbf{z} \oplus \mathbf{s})$.
  Here both $\mathbf{z}$ and $\mathbf{s}$ are random variables. We define $\mathbf{z}=1$ if the middle node on the chain is receiving in round $t+1$, and $\mathbf{z}=0$ otherwise. We define $\mathbf{s}=1$ if the chain is leaked, and $\mathbf{s}=0$ otherwise. Note that this is the only occasion where the reference adversary makes an adaptive decision. Specifically, the decision depends on $\mathbf{z}$, which in turn may depend on the coin flip outcomes of protocol.

It is easy to verify that this adaptive adversary is indeed a sanitized adaptive adversary (recall the definition from Sect. 7): Regardless of the coin flip outcomes $C_{\mathscr{P}}$ of $\mathscr{P}$, for the second type of chains above, the adversary removes the even edge at the beginning of round $t+1$ with half probability and at the beginning of round $t+2$ with the remaining half probability. Furthermore, these decisions are independent for different chains.

*For type-*0 $(\mathbf{X}', \mathbf{Y}')$. If $(\mathbf{X}', \mathbf{Y}')$ is of type-0, then the dynamic network will have two parts. The first part depends on left$(\mathbf{X}', \mathbf{Y}')$, while the second part depends on right$(\mathbf{X}', \mathbf{Y}')$. Each part has $\frac{3n}{2}+2$ nodes, and the network has total $3n+4$ nodes. All nodes in the first part are called *stable nodes*, while all nodes in second part are *unstable nodes*. The first part has two special nodes α and β, while the second part has two special nodes γ and λ.

During round 0, the topology among the stable nodes are exactly the same as in the case where $(\mathbf{X}', \mathbf{Y}')$ is of type-1. The topology among the unstable nodes are constructed in the same way as the stable nodes, except that we replace α and β with γ and λ, and except that we use right$(\mathbf{X}', \mathbf{Y}')$ to construct the $\frac{n}{2}$ chains (see Fig. 3b). Now γ (λ) is connected to all the top (bottom) nodes in the second part. In the next, a chain may refer to either a chain in the first part or a chain in the second part. Hence the network has total n chains.

Starting from round 1, the adversary changes the topology in the following way (the first two items below are the same as the case when $(\mathbf{X}', \mathbf{Y}')$ is of type-1):

- For every $|^{2t}_{2t-1}$ and $|^{2t-1}_{2t}$ chain ($1 \le t \le \frac{q-1}{2}$), the adversary removes the even edge on that chain at the beginning of round $t+1$.
- For every $|^{2t}_{2t+1}$ and $|^{2t+1}_{2t}$ chain ($1 \le t \le \frac{q-1}{2}$), the adversary removes the even edge on that chain at the beginning of round $t+1+(\mathbf{z} \oplus \mathbf{s})$. Here $\mathbf{z}$ and $\mathbf{s}$ have the same meaning as earlier.
- At the beginning of round 1, the adversary removes all the top edges and bottom edges on all the $|^0_0$ chains. Next, the adversary *arbitrarily* connects the middle nodes of all $|^0_0$ chains into a line such that all stable nodes are before the unstable nodes on the line. It then connects the stable node at one end of the line to the middle node of some $|^2_2$ chain of stable nodes, and the unstable node at the other end of the line to the middle node of some $|^2_2$ chain of unstable nodes (see Fig. 3b). This serves to keep the topology connected. Since $(\mathbf{X}', \mathbf{Y}')$ is of type-0, such $|^2_2$ chains must exist.
- At the beginning of round $t+1$ ($1 \le t < \frac{q-1}{2}$), for every $|^{2t}_{2t}$ chain, the adversary removes the top and bottom edges on that chain. At the same time, if such a chain consists of stable (unstable) nodes, then the adversary connects the middle node of this chain to the middle node of some arbitrary $|^{2t+2}_{2t+2}$ chain of stable (unstable) nodes. Again, this serves to keep the topology connected. Same as earlier, such $|^{2t+2}_{2t+2}$ chain must exist.

By same reasoning as for when $(\mathbf{X}', \mathbf{Y}')$ is of type-1, one can easily verify that when $(\mathbf{X}', \mathbf{Y}')$ is of type-0, the reference adversary is also a sanitized adaptive adversary.

For type-$\perp$ $(\mathbf{X}', \mathbf{Y}')$. Finally for completeness, we also need to define the reference adversary when $(\mathbf{X}', \mathbf{Y}')$ is of type-$\perp$. The specifics of how the reference adversary works in this case does not really matter, as long as the dynamic network remains connected in each round. Hence for simplicity, if $(\mathbf{X}', \mathbf{Y}')$ is of type-$\perp$, then the reference adversary is the same as for the case where $(\mathbf{X}', \mathbf{Y}')$ is of type-1, except that the adversary does not remove any edges. Thus the dynamic network is effectively a static network. Trivially, when $(\mathbf{X}', \mathbf{Y}')$ is of type-$\perp$, the reference adversary is a sanitized adaptive adversary.

9.3 Alice's and Bob's simulation

Overview. This section describes Alice's and Bob's simulation, and proves several properties of the simulation. We begin with an overview, from Alice's perspective. Alice's goal is to simulate $\mathscr{P}$'s execution against the reference adversary. Based only on her local knowledge, Alice does not know all the specifics of the reference adversary. Because of this, in any given round, Alice will only simulate $\mathscr{P}$'s execution on a subset of the nodes (which are called *non-spoiled* nodes). The set of non-spoiled nodes for Alice will shrink from round to round—namely, as the simulation goes on, Alice will give

```
1  Procedure Simulate_consensus_protocol()
2      foreach r = 1, ..., (q−1)/2 do
3          msg_pool ← ∅; msg_to_other_party ← ∅ ;
4          foreach node τ that was non-spoiled for me in round r − 1 do
5              determine whether τ is sending or receiving in round r ;
6          end
7          foreach node τ that was non-spoiled for me in round r − 1 and is sending in round r do
8              out_msg ← Advance_by_one_round(𝒫, C_𝒫 , τ, initial input to τ);
9              /* We assume a message always contains the id of the sender. */;
10             add out_msg to msg_pool;
11             if τ = α or τ = β then msg_to_other_party ← out_msg ;
12         end
13         send msg_to_other_party to the other party ;
14         receive msg_to_other_party from the other party, and add to msg_pool ;
15         foreach node τ that remains to be non-spoiled for me in round r and is receiving in round r do
16             in_msg ← {msg|msg ∈ msg_pool and the sender of msg is τ's neighbor in round r according to my rule in Section 9.3.3} ;
17             if in_msg is legal (/* see text in Section 9.3.2 for the definition of legal */) then
18                 Advance_by_one_round(𝒫, C_𝒫 , τ, initial input to τ, in_msg);
19             else
20                 abort;
21             end
22         end
23     end
24 end
```

Protocol 2: Simulation protocol executed by Alice and Bob to solve GDC. The `Advance_by_one_round()` procedure continues the simulation of the protocol $\mathscr{P}$ by one round. If $\mathscr{P}$ is sending a message in that round, then the procedure will have that message as its return value. Otherwise the procedure has no return value.

up simulating certain nodes due to the lack of sufficient information to do so.

Consider any given round and any non-spoiled node τ for Alice. To simulate the execution of $\mathscr{P}$ on τ in that round, among other things, Alice needs to be able to feed the incoming message to τ, if τ is receiving in that round. A unique challenge in our simulation is that since Alice does not know all the specifics of the reference adversary, Alice may not be able to precisely determine which nodes are τ's neighbors in the current round. Instead, Alice will use her own rule (based only on her local knowledge) to decide the neighbors of τ. We will later prove that such decisions are always good enough for the simulation to be correct.

In the following, we first define the notion of *spoiled* and *non-spoiled* nodes. Next we present the pseudo-code for the simulation, and then describe the rules used by Alice and Bob to decide the neighbors. Finally, we prove several properties of the simulation.

9.3.1 Spoiled and non-spoiled nodes

In each round, each node is either *spoiled* or *non-spoiled* for Alice. Roughly speaking, a node is non-spoiled for Alice in round r if, based solely on Alice's input X and all the messages sent by the special node β in the dynamic network so far, Alice can simulate the execution of $\mathscr{P}$ on this node against the reference adversary in round r. Formally (see Fig. 4 for an example), we define all unstable nodes as always spoiled for Alice, in all rounds. Among the stable nodes, we define α as always non-spoiled for Alice in all round, while β as always spoiled in all rounds. The remaining stable nodes are all on the chains. Consider any given chain with stable nodes, and let υ, ν, and ω be the three nodes on the chain, from the top to the bottom:

- A node on a chain that is leaked is always non-spoiled for Alice in all rounds.
- A node on a chain that is not leaked is non-spoiled for Alice until it becomes spoiled.
- If the chain is not leaked and is in the form of $|_*^{2t}$, then ν and ω become spoiled since the beginning of round $t+1$.
- If the chain is not leaked and is in the form of $|_*^{2t+1}$, then ω becomes spoiled since the beginning of round $t+1$.

In the above, "$*$" is a wildcard representing any integer.

We similarly define these concepts for Bob: All unstable nodes and the stable node α are always spoiled for Bob. The node β is never spoiled for Bob. For any chain with stable nodes υ, ν, and ω, from the top to the bottom:

- A node on a chain that is leaked is always non-spoiled for Bob in all rounds.
- A node on a chain that is not leaked is non-spoiled for Bob until it becomes spoiled.

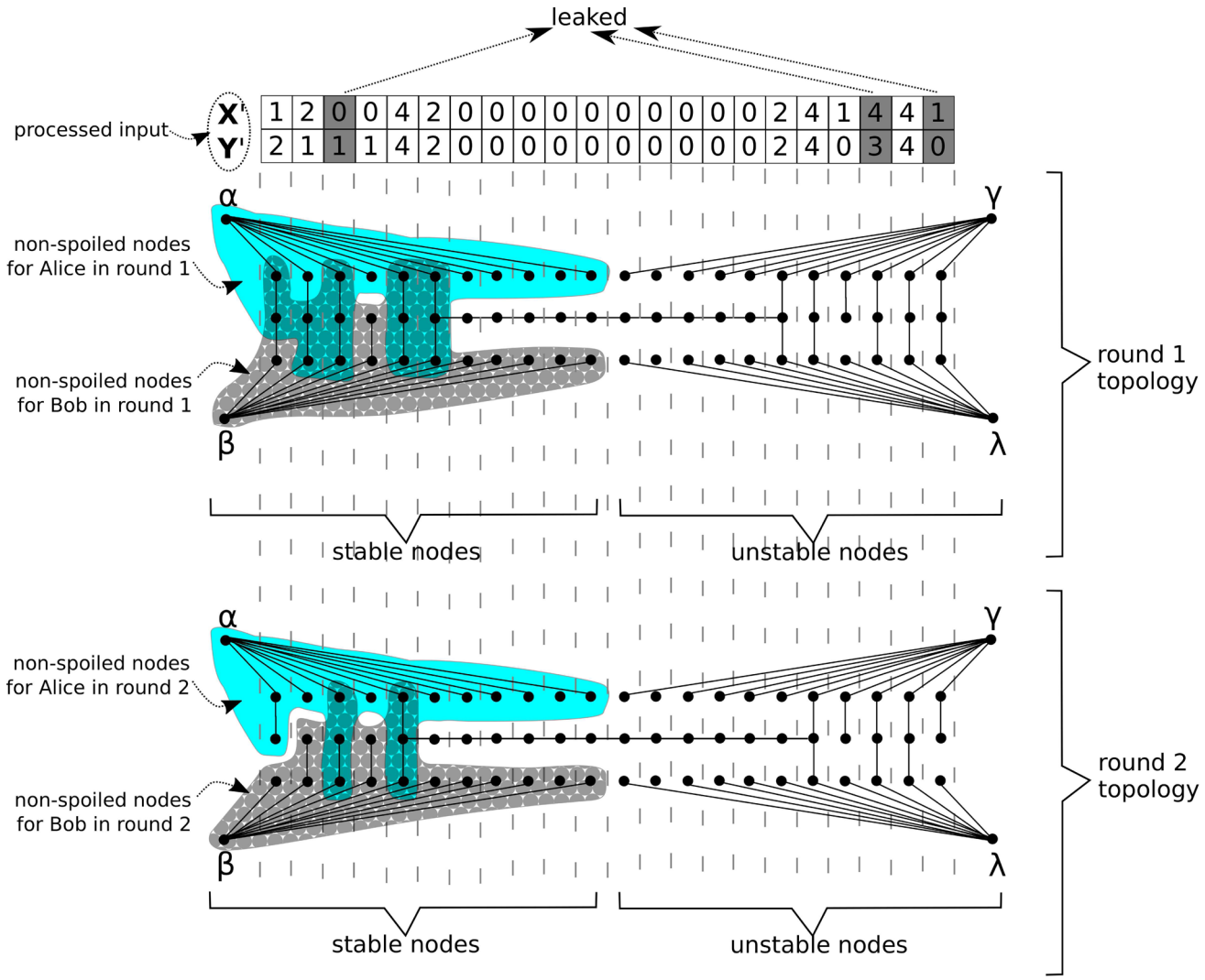


Fig. 4 An example illustrating which nodes are non-spoiled for Alice (Bob), in round 1 and 2, respectively. The middle nodes of all the chains happen to be sending in both rounds. Note that in any given round, some nodes may be non-spoiled for both Alice and Bob, or for exactly one of them, or for none of them. In the figure, the $|_0^0$ chains are all in the middle so that the figure does not become cluttered and messy. In general, the $|_0^0$ chains can be anywhere in the topology

– If the chain is not leaked and is in the form of $|_{2t}^*$, then υ and v become spoiled since the beginning of round $t+1$.
– If the chain is not leaked and is in the form of $|_{2t+1}^*$, then υ becomes spoiled since the beginning of round $t+1$.

9.3.2 The simulation

Pseudo-code for the simulation. Protocol 2 gives the pseudo-code that Alice and Bob use to simulate $\mathscr{P}$'s execution on the nodes that are non-spoiled for each of them. Alice and Bob will feed public coin flips into $\mathscr{P}$ in the simulation. It will be convenient to imagine that such public coin flips has already been done beforehand, with the outcomes being $C_{\mathscr{P}}$, so that $\mathscr{P}$ can be treated as deterministic given $C_{\mathscr{P}}$.

Protocol 2 is executed by both Alice and Bob, separately. We will explain Protocol 2 as it is executed by Alice. In Protocol 2, Alice simulates total $\frac{q-1}{2}$ rounds of $\mathscr{P}$'s execution. For each node in the dynamic network, Alice maintains the state for $\mathscr{P}$ running on that node. In each round r, Alice first checks all nodes that were non-spoiled for her in round $r-1$ and determines whether each of them is sending or receiving in round r. Note that if a node τ was non-spoiled in round $r-1$ but becomes spoiled in round r, we will later prove that Alice can still (i) determine whether τ is sending or receiving in round r, and (ii) simulate $\mathscr{P}$ on τ in round r if τ is sending in round r (since such a τ's behavior is not influenced by potential incoming messages in round r).

Next Alice processes all nodes that were non-spoiled for her in round $r-1$ and are sending in round r. For each such

node τ, Alice simulates and advances $\mathscr{P}$ running on that node by one round. To do so, Alice will need to know the initial input to τ, which may be used by the protocol. Note that incoming messages to τ in previous rounds have already been captured in the current state of the protocol, and there is no need for Alice to provide those again. Since τ is sending, Alice does not provide τ with any incoming message. $\mathscr{P}$ on τ will then generate an outgoing message, which Alice adds to the pool of messages to be delivered. Without loss of generality, we assume a message always contains the id of its sender. If $\tau = \alpha$, then Alice will further forward this message to Bob. Note that for Alice, β is always spoiled and hence $\tau \neq \beta$.

Finally Alice processes all nodes that remain non-spoiled for her in round r and are receiving in round r. For each such node τ, from the pool of messages to be delivered, Alice chooses all those messages that were sent by τ's neighbors to construct a set `in_msg`. Alice decides which nodes are τ's neighbors according to Alice's rule (described later in Sect. 9.3.3). If `in_msg` is *legal* (defined in the next paragraph), then Alice feeds `in_msg` into $\mathscr{P}$ running on τ, and advances $\mathscr{P}$ by one round at Line 18.

Checking whether incoming messages are legal. When the processed input $(\mathbf{X}', \mathbf{Y}')$ is of type-$\perp$, the simulated CONSENSUS protocol $\mathscr{P}$ on different nodes in Protocol 2 may be inconsistent, and may not correspond to the execution of $\mathscr{P}$ over any dynamic network. In such a case, the set `in_msg` of incoming message as constructed at Line 16 of Protocol 2 may be corrupted—namely, $\mathscr{P}$ never expects to receiving such a set of incoming messages. While we will not be concerned with the correctness of $\mathscr{P}$ when the processed input is of type-$\perp$, we do want to ensure that (i) Alice can complete simulation of $\mathscr{P}$ on τ at Line 18 within finite amount of time, and (ii) τ will not send excessively large messages in later rounds since Alice will need to forward τ's message to Bob when $\tau = \alpha$.

To ensure this, at Line 17, Alice check whether `in_msg` is *legal* in the following way: Alice exhaustively enumerate all possible dynamic networks with no more than $3n + 4$ nodes[9] and no more than r rounds, and all possible initial values of the nodes in the network. Alice next simulates the execution of $\mathscr{P}$ with $C_{\mathscr{P}}$ under each such setting. Note that all such simulations are done unilaterally by Alice and are completely independent of the simulation done by Alice and Bob together. Alice then checks whether `in_msg` matches any set of incoming messages to any node φ in any such simulation in round r, where φ has the same state as τ at the end of round $r - 1$ and has the same initial input as τ. Such checking is possible since communication complexity lower bounds hold irrespective of the computational power of Alice and Bob. If there is no such node φ, then Alice claims that `in_msg` is not legal and will abort Protocol 2.

[9] The dynamic network should contain either $3n + 4$ nodes or $\frac{3n+4}{2}$ nodes. But Alice does not know the total number of nodes in the dynamic network. Hence Alice simply tries all dynamic networks with no more than $3n + 4$ nodes. If a message is determined to be legal in this way, then it will already guarantee that (i) the local computation of P on τ takes finite amount of time, and (ii) any message sent by P on τ will not be too large.

9.3.3 Alice's rule and Bob's rule

Alice's rule. Consider any given round and any node τ that is non-spoiled for Alice in that round. To simulate $\mathscr{P}$ on τ, if τ is receiving, Alice needs to determine which nodes are τ's neighbors in that round, under the reference adversary. As mentioned earlier, Alice cannot determine this precisely based only on her local knowledge. Instead, in her simulation, Alice will decide τ's neighbors by following her own rule based solely on her local knowledge (i.e., the leaked information and $\mathbf{X}'$) in the following way:

- For $\tau = \alpha$: Under the reference adversary, node α has a fixed set of neighbors in all rounds, independent of the values of $\mathbf{X}'$ and $\mathbf{Y}'$. Hence Alice's rule will directly choose those nodes to be α's neighbors in her simulation.
- For any non-spoiled node τ on a leaked chain: Let i be the index of this leaked chain. Since the chain is leaked, Alice knows both $\mathbf{x}'_i$ and $\mathbf{y}'_i$. Alice will then determine the neighbors of τ under the reference adversary, while assuming $(\mathbf{X}', \mathbf{Y}')$ to be of type-0. Note that Alice has all necessary information to do so. Alice's rule will choose those nodes to be τ's neighbors in Alice's simulation.
  Obviously, Alice's assumption could be wrong. If $(\mathbf{X}', \mathbf{Y}')$ is of type-1, then given that τ is on a leaked chain, one can directly verify that under the reference adversary, the neighbors of τ will be the same as for the case where $(\mathbf{X}', \mathbf{Y}')$ is of type-0. Hence in this case, the neighbors chosen by Alice's rule will be the same as those under the reference adversary. If instead $(\mathbf{X}', \mathbf{Y}')$ is of type-$\perp$, then the neighbors decided by Alice's rule can be different from the neighbors under the reference adversary. This will however not cause any problem, since when $(\mathbf{X}', \mathbf{Y}')$ is of type-$\perp$, we only need our simulation to have some rather weak guarantees (e.g., terminating within finite amount of time).
- For any non-spoiled node τ on a non-leaked chain: Consider any given non-leaked chain, and let i be the index of this chain. Let the 3 nodes on the chain (from top to the bottom) be υ, ν, and ω. We will describe how Alice's rule decides the neighbors of these 3 nodes, which depends on whether $\mathbf{x}'_i$ is even or odd. We separately consider these two cases.
  The first case is $\mathbf{x}'_i = 2t$ for some integer t. Alice's rule

will choose $\{\alpha, \nu\}$ as the neighbors of υ, in rounds 1 through t. Starting from round $t+1$, Alice's rule will choose $\{\alpha\}$ as the neighbor of υ. Intuitively, this corresponds to the edge between υ and ν being removed at the beginning of round $t+1$. For nodes ν and ω, Alice's rule will always (in all rounds) choose $\{\upsilon, \omega\}$ and $\{\nu, \beta\}$ as their neighbors, respectively. (Note however that ν and ω both become spoiled for Alice starting from round $t+1$. Hence Alice's rule for ν and ω is only relevant for rounds 1 through t.)
The second case is $\mathbf{x}'_i = 2t-1$ for some integer t. Alice's rule will choose $\{\upsilon, \omega\}$ as the neighbors of ν, in rounds 1 through t. Starting from round $t+1$, Alice's rule will choose $\{\upsilon\}$ as the neighbor of ν. For nodes υ and ω, Alice's rule will always (in all rounds) choose $\{\alpha, \nu\}$ and $\{\nu, \beta\}$ as their neighbors, respectively.

Some intuition behind Alice's rule. As an example, consider a $|^2_3$ chain that is not leaked, and let τ and ν be the top node and the middle node of this chain, respectively. Note that τ is always non-spoiled for Alice.

Assume that ν is receiving in round 2. By Alice's rule, in round 2, ν is not a neighbor of τ. But under the reference adversary, the node ν is a neighbor of τ in round 2. Hence τ's neighbors as decided by Alice's rule are different from τ's neighbors under the reference adversary. One may suspect that this could cause Alice's simulation to be incorrect. But note that ν is receiving in round 2, and does not send any message. Hence τ will not receive any message from ν, and the simulation on τ will be the same, regardless of whether ν is a neighbor of τ. Intuitively, this is why despite Alice's rule not following the reference adversary precisely, Alice's simulation will still be correct.

On the other hand, if ν is sending in round 2, then under the reference adversary, the node ν is not a neighbor of τ in round 2. In this case, τ's neighbors as decided by Alice's rule will be the same as τ's neighbors under the reference adversary.

Bob's rule. Bob's rule is entirely symmetric to Alice's rule, and there is no fundamental difference between Bob's rule and Alice's rule. For completeness, the next fully describes Bob's rule. Consider any given round and any non-spoiled node τ for Bob in that round. In Bob's simulation, Bob will decide τ's neighbors by following his own rule based solely on his local knowledge (i.e., the leaked information and $\mathbf{Y}'$):

- For $\tau = \beta$: Under the reference adversary, node β has a fixed set of neighbors in all rounds, independent of $\mathbf{X}'$ and $\mathbf{Y}'$. Bob's rule will choose those nodes to be β's neighbors in his simulation.
- For any non-spoiled node τ on a leaked chain: Let i be the index of this leaked chain. Bob, knowing both $\mathbf{x}'_i$ and $\mathbf{y}'_i$, will determine the neighbors of τ under the reference adversary, while assuming $(\mathbf{X}', \mathbf{Y}')$ to be of type-0. Bob's rule will choose those nodes to be τ's neighbors in Bob's simulation.
- For any non-spoiled node τ on a non-leaked chain: Let i be the index of this chain, and let the 3 nodes on this chain (from top to the bottom) be υ, ν, and ω.
  If $\mathbf{y}'_i = 2t$ for some integer t, Bob's rule will choose $\{\nu, \beta\}$ as the neighbors of ω, in rounds 1 through t. Starting from round $t+1$, Bob's rule will choose $\{\beta\}$ as the neighbor of ω. For nodes υ and ν, Bob's rule will always (in all rounds) choose $\{\alpha, \nu\}$ and $\{\upsilon, \omega\}$ as their neighbors, respectively.
  If $\mathbf{y}'_i = 2t-1$ for some integer t, Bob's rule will choose $\{\upsilon, \omega\}$ as the neighbors of ν, in rounds 1 through t. Starting from round $t+1$, Bob's rule will choose $\{\omega\}$ as the neighbors of ν. For nodes υ and ω, Bob's rule will always (in all rounds) choose $\{\alpha, \nu\}$ and $\{\nu, \beta\}$ as their neighbors, respectively.

9.3.4 Performance of the simulation

We first prove that Alice's and Bob's simulation (using Protocol 2) will always terminate and will not incur too much communication, even when the processed input $(\mathbf{X}', \mathbf{Y}')$ is of type-$\perp$.

Lemma 8 *For any* CONSENSUS *protocol $\mathscr{P}$, there exists positive constant c such that for all n, q, and $C_{\mathscr{P}}$, Protocol 2 always terminates within finite amount of time and incurs at most $cq \log n$ bits of communication between Alice and Bob.*

Proof For any node τ and any round r, we say that the state of $\mathscr{P}$ running on τ (as maintained by Alice or Bob using Protocol 2) is *legal* if there exists some dynamic network of no more than $3n+4$ nodes, some initial values to the nodes in this dynamic network, and some node φ in this dynamic network whose initial value is the same as τ's, such that when running $\mathscr{P}$ on this dynamic network with $C_{\mathscr{P}}$, the state of $\mathscr{P}$ on φ in round r is exactly the same as the state of $\mathscr{P}$ on τ as maintained by Alice or Bob using Protocol 2.

We next prove via an induction that for all node τ and all round r, the state of $\mathscr{P}$ running on τ in round r is legal. The case for $r = 0$ is trivial. Assume the claim holds for round $r-1$, and consider any node τ. If τ is sending in round r, it is easy to see that the state of the CONSENSUS protocol running on τ will continue to be legal. If τ is receiving in round r, then Line 17 of Protocol 2 explicitly ensures that the state will be legal, before continuing.

Next since the state of $\mathscr{P}$ on all node τ are always legal[10] in all round r, it immediately means that simulated $\mathscr{P}$ running on τ will complete its execution for round r within finite amount of time at Line 8 and Line 18 of Protocol 2. Furthermore at Line 11, the size of `out_msg` (and hence the size of `msg_to_other_party`) must satisfy the maximum allowed message size (i.e., $O(\log n)$) for a network with $\Theta(n)$ nodes. The lemma follows since in each round, Alice and Bob only communicate once at Line 13 by sending `msg_to_other_party` to the other party. □

9.3.5 Correctness of the simulation

Overview. We next aim to prove that by using Protocol 2, Alice and Bob can indeed correctly simulate $\mathscr{P}$'s execution against the reference adversary on the non-spoiled nodes, when $(\mathbf{X}', \mathbf{Y}')$ is either of type-0 or type-1. We do not care about the case where $(\mathbf{X}', \mathbf{Y}')$ is of type-$\perp$, as this will only happen with probability less than $\frac{1}{q}$.

To make our claims rigorous, we need to first define the notion of *reference execution*. Consider any given CONSENSUS protocol $\mathscr{P}$, any given initial values for all the nodes, any given public coin flip outcomes $C_{\mathscr{P}}$ that Alice and Bob generate to feed into $\mathscr{P}$ in their simulation, any processed input $(\mathbf{X}', \mathbf{Y}')$ that is either type-0 or type-1, any given set of indices that are leaked by the leaker, and the corresponding reference adversary under such a setting. We define the *reference execution* to be the execution of $\mathscr{P}$ under such coin flips, such initial values of the nodes, and such adversary. (Such a reference execution must be deterministic since all coins have been flipped.)

We ultimately aim to prove that the behavior of each node in Alice's and Bob's simulation is exactly the same as in the reference execution. We do this via two steps, and the following provides some intuitions.

For the first step, as an example, let us consider any given round r and any given node τ that is receiving in round r in the reference execution. Assume that τ is non-spoiled for Alice, and hence is being simulated by Alice. Recall that in her simulation, Alice uses her own rule to decide the neighbors of τ in round r, which are later used to determine which messages should be fed into τ. These neighbors may be different from τ's neighbors in the reference execution. However, Lemma 9 later will prove that all nodes in the symmetric difference of these two sets of neighbors are all receiving in round r. Hence the difference will not impact the set of messages that τ receives in round r. Furthermore, Lemma 9 will also show that all of τ's sending neighbors (except potentially β)[11] must also be non-spoiled for Alice in round $r-1$, which allows Alice to generate the messages that she needs to feed into τ in round r.

Our second step will build upon Lemma 9. We will show (in Lemma 10) via an induction that in both Alice's and Bob's simulation (i.e., Protocol 2), the outgoing message of each node in each round of simulation is exactly the same as the outgoing message of the corresponding node in the corresponding round in the reference execution. Without loss of generality, assume that when a node decides in the CONSENSUS protocol $\mathscr{P}$, the node sends a special message. Hence a node will send such a special message in a round in the simulation if and only if it does so in the corresponding round in the reference execution. This will be the ultimate property that we later need.

Neighbors of nodes. The following lemma reasons about the neighbors of a node as decided by Alice's rule (Bob's rule), as compared to its neighbors under the reference execution:

Lemma 9 *Consider any given reference execution, any round $1 \leq r \leq \frac{q-1}{2}$, and any node τ in the reference execution that is receiving in the reference execution in round r and is non-spoiled for Alice (Bob) in round r. Let S be the set of nodes that are τ's neighbors according to Alice's (Bob's) rule in round r and that are sending in the reference execution in round r. Let S' be the set of nodes that are τ's neighbors under the reference adversary in round r and are sending in the reference execution in round r. Then,*

- $S = S'$.
- *For all $\varphi \in S$, either φ is non-spoiled in round $r-1$ for Alice (Bob) or $\varphi = \beta$ ($\varphi = \alpha$).*

Proof It suffices to prove the lemma for Alice. Throughout our proof, we will extensively leverage the fact that since we are considering a reference execution, the corresponding processed input $(\mathbf{X}', \mathbf{Y}')$ must be either type-0 or type-1. Hence whenever we consider φ's neighbors under the reference adversary, we should keep in mind that $(\mathbf{X}', \mathbf{Y}')$ is not of type-$\perp$.

Define T to be the set of τ's neighbors according to Alice's rule in round r, and T' to be the set of τ's neighbors under the reference adversary in round r. Obviously, $S \subseteq T$ and $S' \subseteq T'$. Note that S (S') consists solely of the nodes in T (T') that are sending in the reference execution. Hence $T = T'$ implies $S = S'$.

[10] The states of $\mathscr{P}$ on all the individual nodes being legal does not necessarily imply that the joint state across all the nodes corresponds to some execution of $\mathscr{P}$ over some dynamic network. However, note that the proof here does not rely on the properties on the joint state. The proof here only needs that (i) each node completes its execution for each round within finite amount of time, and (ii) messages sent are not excessively large.

[11] Node β is an exception as it is always spoiled for Alice, but this does not cause any problem since Bob will forward to Alice the message that β sends in each round.

Since τ is non-spoiled in round r, τ must be a stable node. Such τ can either be the special node α or can be a node on any of the chains consisting of stable nodes. If $\tau = \alpha$, then τ's neighbors according to Alice's rule are always exactly the same as τ's neighbors under the reference adversary, and all these neighbors are never spoiled for Alice. Hence the lemma holds when $\tau = \alpha$.

Next we consider the case where τ is on some chain consisting of stable nodes. If the chain is leaked, then regardless where τ is on the chain, we have $T = T'$. Furthermore, since nodes on a leaked chain are always non-spoiled, a node in T must be either β or some non-spoiled node. Hence the lemma holds.

The remainder of our proof covers the case where τ is on some non-leaked chain consisting of stable nodes. Let υ, ν, and ω be the three nodes, from top to the bottom, on any such chain. We exhaustively enumerate all possibilities, depending on what kind of chain it is. Let t be any integer where $0 \leq t \leq \frac{q-1}{2}$:

- For a $|^{2t}_{2t-1}$ chain or a $|^{2t}_{2t}$ chain, υ is always non-spoiled, and ν and ω are non-spoiled iff $r < t+1$:
  - For node υ, we exhaustively enumerate all scenarios: (i) If $r < t+1$, then $T = T' = \{\alpha, \nu\}$. By definition, both α and ν are non-spoiled in round $r-1$. (ii) If $r \geq t+1$, then $T = T' = \{\alpha\}$. By definition, α is non-spoiled in round $r-1$.
  - For node ν and $r < t+1$, we have $T = T' = \{\upsilon, \omega\}$, and both nodes are non-spoiled in round $r-1$.
  - For node ω and $r < t+1$, we have $T = T' = \{\nu, \beta\}$, where ν is non-spoiled in round $r-1$.
- For a $|^{2t}_{2t+1}$ chain, υ is always non-spoiled, and ν and ω are non-spoiled iff $r < t+1$:
  - For node υ, we exhaustively enumerate all scenarios: (i) If $r < t+1$, then $T = T' = \{\alpha, \nu\}$. By definition, both α and ν are non-spoiled in round $r-1$. (ii) If $r > t+1$, then $T = T' = \{\alpha\}$. By definition, α is non-spoiled in round $r-1$. (iii) If $r = t+1$ and ν is sending in round r, then $T = T' = \{\alpha\}$ and α is non-spoiled in round $r-1$. iv) If $r = t+1$ and ν is receiving in round r, then $T' = \{\alpha, \nu\}$ and $T = \{\alpha\}$. If α is receiving in round r, we have $S = S' = \emptyset$. Otherwise, $S' = \{\alpha\} = S$. By definition, α is non-spoiled in round $r-1$.
  - For node ν and $r < t+1$, we have $T = T' = \{\upsilon, \omega\}$, and both nodes are non-spoiled in round $r-1$.
  - For node ω and $r < t+1$, we have $T = T' = \{\nu, \beta\}$, and ν is non-spoiled in round $r-1$.
- For a $|^{2t-1}_{2t}$ chain, υ and ν are always non-spoiled, and ω is non-spoiled iff $r < t$:
  - For node υ, $T = T' = \{\alpha, \nu\}$. By definition, both α and ν are non-spoiled in round $r-1$.
  - For node ν, we exhaustively enumerate all scenarios: (i) If $r \leq t$, we have $T = T' = \{\upsilon, \omega\}$, and both nodes are non-spoiled in round $r-1$. (ii) If $r \geq t+1$, then $T = T' = \{\upsilon\}$. By definition, υ is non-spoiled in round $r-1$.
  - For node ω and $r < t$, we have $T = T' = \{\nu, \beta\}$, where ν is non-spoiled in round $r-1$.
- For a $|^{2t+1}_{2t}$ chain, υ and ν are always non-spoiled, and ω is non-spoiled iff $r < t+1$:
  - For node υ, $T = T' = \{\alpha, \nu\}$. By definition, both α and ν are non-spoiled in round $r-1$.
  - For node ν, we exhaustively enumerate all scenarios: (i) If $r < t+1$, we have $T = T' = \{\upsilon, \omega\}$, and both nodes are non-spoiled in round $r-1$. (ii) If $r > t+1$, then $T = T' = \{\upsilon\}$. By definition, υ is non-spoiled in round $r-1$. (iii) If $r = t+1$, recall that we only need to consider the case where ν is receiving in round r. Thus, $T = T' = \{\upsilon, \omega\}$, and both υ and ω are non-spoiled in round $r-1$.
  - For node ω and $r < t+1$, we have $T = T' = \{\nu, \beta\}$, where ν is non-spoiled in round $r-1$.

Hence the lemma holds in all above cases. □

Outgoing messages of nodes. We next aim to prove that the outgoing message of each node in each round of simulation is exactly the same as the corresponding outgoing message in the reference execution.

Consider any round r and any node τ. If τ is sending in round r, we say that the outgoing message from τ as determined in round r of Protocol 2 at Line 8 is *consistent with the reference execution* (or *consistent* in short), if it is exactly the same as the τ's outgoing message in the reference execution in round r. Similarly, if τ is receiving in round r, we say that the set of incoming messages fed into τ in round r of Protocol 2 at Line 16 is *consistent (with the reference execution)* if it is exactly the same as τ's set of incoming message in the reference execution in round r.

The lemma below actually proves more properties than we need in the end—however, we need those properties for the inductive proof to go through.

Lemma 10 *Consider any given reference execution, any node τ in the reference execution, and any r where $1 \leq r \leq \frac{q-1}{2}$.*

- *If τ was non-spoiled for Alice (Bob) in round $r-1$ and is sending in the reference execution in round r, then* (i) *τ will be determined as sending in round r by Alice (Bob) at Line 5 of Protocol 2, and* (ii) *τ's outgoing message as*

determined by round r of Alice's (Bob's) Protocol 2 at Line 8 is consistent with the reference execution.

– *If τ was non-spoiled for Alice (Bob) in round $r-1$ and is receiving in the reference execution in round r, then τ will be determined as receiving in round r by Alice (Bob) at Line 5 of Protocol 2. Furthermore if such a τ continues to be non-spoiled in round r, the set of τ's incoming messages as determined by round r of Alice's (Bob's) Protocol 2 at Line 16 is consistent with the reference execution.*

Furthermore, Line 20 in Protocol 2 will not be executed in round r.

Proof The last claim that Line 20 will not be executed does not need to be proved separately—as long as we can prove the other claims in the lemma, the last claim will directly follow. The reason is that Line 20 can only be executed when `in_msg` is not legal at Line 17. However, if the previous claims in the lemma hold, then `in_msg` must be legal. Thus we will not separately prove the last claim.

It suffices to prove the lemma for Alice. We prove via an induction on r. The induction base for $r = 0$ is trivial since τ by definition is receiving in that round, and the set of incoming messages is empty. For the inductive step, suppose that the lemma holds for all rounds before round r, and we prove the lemma for round r.

First, consider the case where τ is non-spoiled for Alice in round $r-1$ and is sending in the reference execution in round r. By definition, τ must be non-spoiled in round 0 through round $r-1$. By the inductive hypothesis, in Protocol 2 for every previous round where τ was receiving, the set of incoming messages fed into τ by Alice was consistent with the reference execution. Since everything is deterministic, at Line 5 Alice can determine that τ must be sending in round r, and the outgoing message from τ in round r as generated by Protocol 2 must be consistent as well.

Next consider the case where τ is non-spoiled for Alice in round $r-1$ and is receiving in the reference execution in round r. By same argument as earlier, at Line 5 Alice must be able to determine that τ is in a receiving state in round r.

If τ continues to be non-spoiled in round r, let S be the set of nodes that are τ's neighbors as decided by Alice's rule in round r and that are sending in the reference execution in round r. Consider the set `in_msg` of messages that Alice constructs as τ's incoming messages at Line 16. We claim that `in_msg` is the same as the set of the messages sent by all the nodes in S in the reference execution. It is easy to see that for any node $\varphi \notin S$, the outgoing message from φ will not be added to `in_msg` at Line 16 since by definition of S, φ is not τ's neighbor according to Alice's rule in round r. Hence to prove the claim, we only need to show that for any node $\varphi \in S$, the outgoing message from φ that Alice adds to `msg_pool` (at either Line 8 or Line 14) is consistent with the reference execution. As long as this message is in `msg_pool`, it will be later added to `in_msg` at Line 16 since φ is τ's neighbor according to Alice's rule.

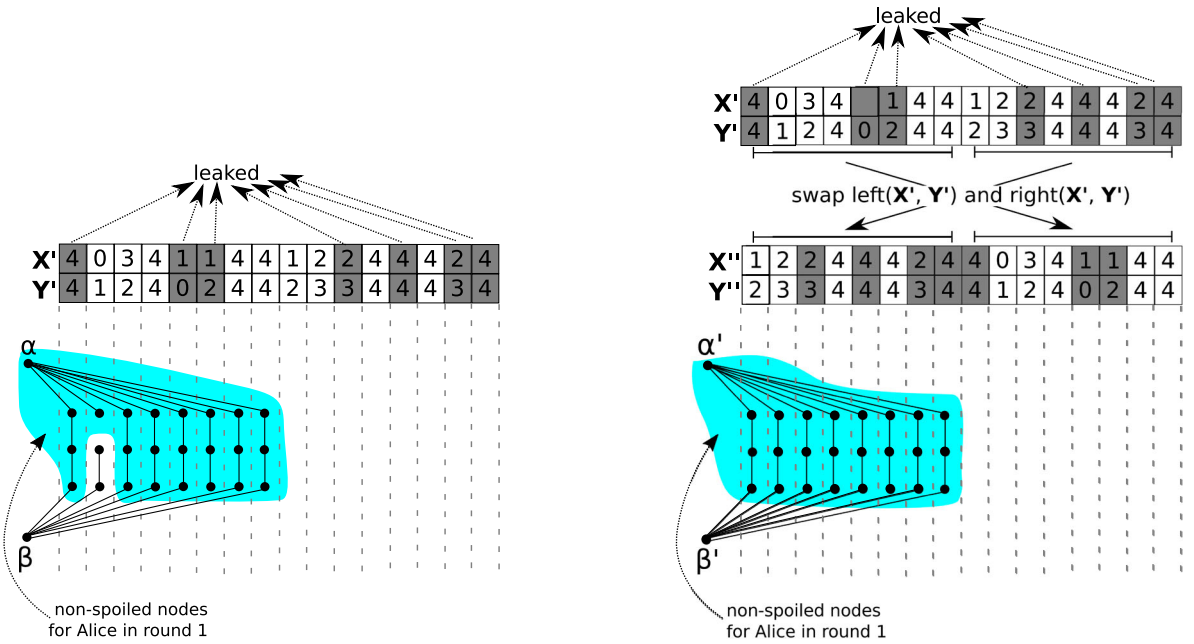
If $\varphi \in S$ and $\varphi = \beta$, then φ is sending in round r in the reference execution and φ is non-spoiled for Bob in round $r-1$. By our earlier argument, at Line 8, the outgoing message from φ generated by Bob in round r is consistent with reference execution. Such a message will then be forwarded to Alice at Line 13, and then added to `msg_pool` at Line 14. If $\varphi \in S$ and $\varphi \neq \beta$, by Lemma 9, φ must be non-spoiled in round $r-1$. Again by our earlier arguments, at Line 8, Alice will generate the consistent outgoing message from φ in round r, and add such a message to `msg_pool` at Line 8.

So far we have proved that `in_msg` is the same as the set of the messages sent by all the nodes in S in the reference execution. Let S' be the set of nodes that are τ's neighbors in the reference adversary in round r and are sending in the reference execution in round r. Lemma 9 tells us that $S = S'$, which immediately implies that `in_msg` is consistent with the reference execution and hence completes the proof. □

9.4 Proving Theorem 7 from the simulation

Two simulations. Before giving the proof for Theorem 7, we first highlight a tricky part in the proof, and provide intuition for that part. We will reduce $\textsc{Gdc}(X, Y)$ with leaker to Consensus by first converting (X, Y) to the processed input $(\mathbf{X}', \mathbf{Y}')$. The processed input $(\mathbf{X}', \mathbf{Y}')$, together with the the leaked information, will determine the reference adversary. Alice and Bob will effectively simulate the Consensus protocol $\mathscr{P}$'s execution against the reference adversary, and infer the answer to $\textsc{Gdc}(X, Y)$ by monitoring $\mathscr{P}$'s execution on the special node α. (Of course, other nodes still need to be simulated to enable the simulation of $\mathscr{P}$'s execution on α.) Roughly speaking, if $\textsc{Gdc}(X, Y) = 1$ (and hence $(\mathbf{X}', \mathbf{Y}')$ is likely to be of type-1), then the dynamic network will have a small diameter, implying that $\mathscr{P}$ will output quickly on node α. When $\textsc{Gdc}(X, Y) = 0$ (and hence $(\mathbf{X}', \mathbf{Y}')$ is likely to be of type-0), we instead want $\mathscr{P}$ to output in $\Omega(q)$ rounds on α.

The tricky part is that even though the dynamic network has $\Omega(q)$ diameter when $(\mathbf{X}', \mathbf{Y}')$ is of type-0, $\mathscr{P}$ may still output fast on the node α and take $\Omega(q)$ rounds to output on some other nodes. To overcome this challenge, we will actually do *two* simulations of $\mathscr{P}$, which are separate and independent. Having such two simulations is a tricky aspect of our proof for Theorem 7. The first simulation is based on the processed input $(\mathbf{X}', \mathbf{Y}')$ and the corresponding reference adversary. Next, let $\mathbf{X}''$ be the string obtained by swapping $\texttt{left}(\mathbf{X}')$ and $\texttt{right}(\mathbf{X}')$. Similarly define $\mathbf{Y}''$. With a slight abuse of notation, we also call $(\mathbf{X}'', \mathbf{Y}'')$ as a



(a) The round 1 topology of the dynamic network in the first simulation. This dynamic network is based on $(\mathbf{X}', \mathbf{Y}')$. Since $(\mathbf{X}', \mathbf{Y}')$ is of type-1, the reference adversary determines the dynamic network based only on left$(\mathbf{X}', \mathbf{Y}')$ (i.e., the first half of $(\mathbf{X}', \mathbf{Y}')$). The middle nodes of all the chains happen to be sending (as determined by the CONSENSUS protocol) in this round.

(b) The round 1 topology of the dynamic network in the second simulation. This dynamic network is based on $(\mathbf{X}'', \mathbf{Y}'')$. Since $(\mathbf{X}'', \mathbf{Y}'')$ is of type-1, the reference adversary determines the dynamic network based only on left$(\mathbf{X}'', \mathbf{Y}'')$ (i.e., the first half of $(\mathbf{X}'', \mathbf{Y}'')$) The middle nodes of all the chains happen to be sending (as determined by the CONSENSUS protocol) in this round.

Fig. 5 The two dynamic networks used in the two simulations, respectively. Here $(\mathbf{X}', \mathbf{Y}')$ and $(\mathbf{X}'', \mathbf{Y}'')$ are of type-1

processed input. To avoid notation collision, in this second reference adversary, we rename the nodes α, β, γ (if exists), and λ (if exists) to be α', β', γ', and λ', respectively.

Now if $(\mathbf{X}', \mathbf{Y}')$ is of type-1, then $(\mathbf{X}'', \mathbf{Y}'')$ must also be of type-1. The two dynamic networks in the two simulations will both have a small diameter (see Fig. 5). This means that $\mathscr{P}$ will output fast on both α in the first simulation and α' in the second simulation. If $(\mathbf{X}', \mathbf{Y}')$ is instead of type-0, then $(\mathbf{X}'', \mathbf{Y}'')$ must also be of type-0. In such case, the two dynamic networks in the two simulations will both have $\Omega(q)$ diameter (see Fig. 6). Via a coupling argument, if $\mathscr{P}$ does not err, we can show that at least one of the following two cases must hold: (i) in the first simulation $\mathscr{P}$ takes $\Omega(q)$ rounds to output on α, or (ii) in the second simulation $\mathscr{P}$ takes $\Omega(q)$ rounds to output on α'.

Putting everything together, if Alice observes that $\mathscr{P}$ outputs fast on both α in the first simulation and α' in the second simulation, then Alice will claim that GDC$(X, Y) = 0$. Otherwise Alice claims that GDC$(X, Y) = 1$. We now present the complete proof for Theorem 7:

Theorem 7 *If the nodes only know a poor estimate m' for m where $|\frac{m'-m}{m}|$ is at least $\frac{1}{3}$, 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.*

Proof Consider any given $\frac{1}{10}$-error CONSENSUS protocol $\mathscr{P}$ with time complexity of $\mathrm{tc}(d, m)$ rounds over average coin flips, when running over dynamic networks controlled by oblivious adversaries and with d diameter and m nodes. We aim to prove that $\mathrm{tc}(d, m) = \Omega(d + m^{\frac{1}{12}})$. To do so, we will prove that $\mathrm{tc}(8, m) \geq m^{\frac{1}{12}}$ for all sufficiently large m. This proof will trivially extend to $\mathrm{tc}(d, m)$ for all $d > 8$. Combining with the fact that $\mathrm{tc}(d, m) = \Omega(d)$ then completes the proof.

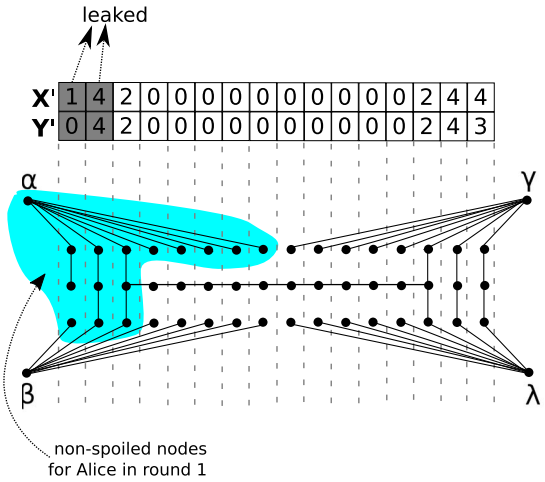
Consider the constants c_1 and c_2 in Theorem 4 (for $\delta = \frac{2}{5}$), the constant c in Lemma 8, and the following inequalities:

$$\frac{m-4}{3} \geq 60\left(20m^{\frac{1}{12}} + 21\right)\ln\left(20m^{\frac{1}{12}} + 21\right) \tag{3}$$

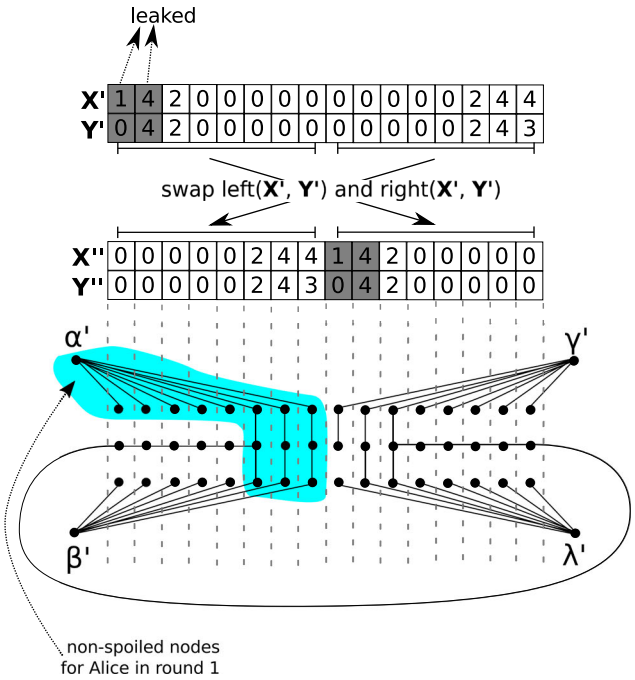
$$\frac{c_1\sqrt{\frac{m-4}{3}}}{15\left(20m^{\frac{1}{12}} + 21\right)^{4.5}\log^3 m} \geq 2c\left(20m^{\frac{1}{12}} + 21\right) + c_2 \tag{4}$$

It is easy to see that there must exist constant $c_3 > 0$ such that for all $m \geq c_3$, both inequalities hold. We will prove that $\mathrm{tc}(8, m) \geq m^{\frac{1}{12}}$ for all $m \geq c_3$.

Assume by contradiction that there exists some $m \geq c_3$ such that $\mathrm{tc}(8, m) < m^{\frac{1}{12}}$. We will proceed with the reduction from GDC and eventually obtain a contradiction. Let $n = \frac{m-4}{3}$, $q = 20\mathrm{tc}(8, m) + 21$, and $g = 15q\ln q$. We



(a) The round 1 topology of the dynamic network in the first simulation. This dynamic network is based on $(\mathbf{X}', \mathbf{Y}')$. Since $(\mathbf{X}', \mathbf{Y}')$ is of type-0, the reference adversary determines the dynamic network based on the entirety of $(\mathbf{X}', \mathbf{Y}')$. The middle nodes of all the chains happen to be receiving (as determined by the CONSENSUS protocol) in this round.



(b) The round 1 topology of the dynamic network in the second simulation. This dynamic network is based on $(\mathbf{X}'', \mathbf{Y}'')$. Since $(\mathbf{X}'', \mathbf{Y}'')$ is of type-0, the reference adversary determines the dynamic network based on the entirety of $(\mathbf{X}'', \mathbf{Y}'')$. The middle nodes of all the chains happen to be receiving (as determined by the CONSENSUS protocol) in this round.

Fig. 6 The two dynamic networks used in the two simulations, respectively. Here $(\mathbf{X}', \mathbf{Y}')$ and $(\mathbf{X}'', \mathbf{Y}'')$ are of type-0

later will need to invoke Lemma 7. Note that these parameters do satisfy the requirements in Lemma 7, since by Inequality 3:

$$n = \frac{m-4}{3} \geq 60(20m^{1/12} + 21)\ln(20m^{1/12} + 21) > 60q \ln q = 4g$$

Also note that since $n > 4g$, we have $n > q$, and hence the $\text{GDC}_n^{g,q}$ problem is well-defined.

To solve the $\text{GDC}_n^{g,q}(X, Y)$ problem with our leaker, Alice and Bob will simulate the execution of $\mathscr{P}$. Alice and Bob will first generate public coin flip outcomes (denoted as $\mathbf{C}_{\mathscr{P}}$) to feed into $\mathscr{P}$. This effectively makes $\mathscr{P}$ deterministic. Alice and Bob set $\hat{m} = \frac{2}{3}m = \frac{2}{3}(3n+4)$, and feeds $\hat{m}$ into $\mathscr{P}$ as an estimate of the total number of nodes, if $\mathscr{P}$ needs such an estimate. As we will quickly see, the number of nodes in the dynamic network will be either m or $m/2$. Hence obviously, such $\hat{m}$ satisfies both $|\frac{\hat{m}-m}{m}| = \frac{1}{3}$ and $|\frac{\hat{m}-m/2}{m/2}| = \frac{1}{3}$.

Alice and Bob will simulate $\mathscr{P}$ twice on two different dynamic networks, using the same $\mathbf{C}_{\mathscr{P}}$. The ids of the nodes in the dynamic network will be determined by the adversary and then given to $\mathscr{P}$ as inputs.

- **First simulation** The first simulation is based on the processed input $(\mathbf{X}', \mathbf{Y}')$. We first assign initial values and ids to the nodes under the corresponding reference adversary. All stable nodes has initial values 0. Order all the stable nodes into a total order by some arbitrary criterion, and then assign them ids from 1 to $\frac{3}{2}n + 2$. Note that Alice and Bob can determine the initial values and the ids of all the stable nodes without the need of communication, since these initial values and ids do not depend on $(\mathbf{X}', \mathbf{Y}')$.
  If there are unstable nodes (i.e., when $(\mathbf{X}', \mathbf{Y}')$ is of type-0), then they will all have initial values 1. The unstable nodes will have ids from $\frac{3}{2}n + 3$ to $3n + 4$, by the total ordering as described later for the stable nodes in the second simulation.
  Note that by our definition, if a node is non-spoiled in any round, then that node must be a stable node. Hence for any non-spoiled node in any round, by our above reasoning, Alice and Bob must know the initial values and id of that node. Alice and Bob then proceed with the first simulation using Protocol 2. By Lemma 8, such simulation must complete within finite time.
- **Second simulation** For the second simulation, we construct a second processed input $(\mathbf{X}'', \mathbf{Y}'')$ by swapping

left($\mathbf{X}'$, $\mathbf{Y}'$) and right($\mathbf{X}'$, $\mathbf{Y}'$). Specifically, we set $\mathbf{X}''_i = \mathbf{X}'_{i+\frac{n}{2}}$ and $\mathbf{Y}''_i = \mathbf{Y}'_{i+\frac{n}{2}}$ for $1 \le i \le \frac{n}{2}$, and $\mathbf{X}''_i = \mathbf{X}'_{i-\frac{n}{2}}$ and $\mathbf{Y}''_i = \mathbf{Y}'_{i-\frac{n}{2}}$ for $\frac{n}{2}+1 \le i \le n$.

It is trivial to see that ($\mathbf{X}''$, $\mathbf{Y}''$) and ($\mathbf{X}'$, $\mathbf{Y}'$) must be of the same type. The second simulation is based on the processed input ($\mathbf{X}''$, $\mathbf{Y}''$). In particular, if ($\mathbf{X}''$, $\mathbf{Y}''$) is of type-1, then the reference adversary will use left($\mathbf{X}''$, $\mathbf{Y}''$) to construct the topology. Recall that for clarity, we rename the nodes α, β, γ, and λ to be α', β', γ', and λ' in the second simulation.

We still need to assign initial values and ids to the nodes under the corresponding reference adversary. All stable nodes have initial values of 1, and all unstable nodes (if any) have initial values of 0. Order all the stable nodes into a total order by some arbitrary criterion. These nodes are then assigned ids from $\frac{3}{2}n+3$ to $3n+4$. Note that the initial topology among these stable nodes will be exactly the same as the initial topology among the unstable nodes in the first simulation. As mentioned earlier, we used the same total ordering used here to order the unstable nodes in the first simulation, if there were unstable nodes there.

If there are unstable nodes (i.e., when ($\mathbf{X}''$, $\mathbf{Y}''$) is of type-0), then again, the initial topology among these unstable nodes will be exactly the same as the initial topology among the stable nodes in the first simulation. We will use the same total ordering used in the first simulation to order these unstable nodes, and assign them ids from 1 to $\frac{3}{2}n+2$.

Again, in the second simulation, Alice and Bob know the initial values and ids of all their respective non-spoiled nodes. Alice and Bob then proceed with the second simulation using Protocol 2. By Lemma 8, such simulation must complete within finite time.

- **Generating an output** Alice monitors when α decides in the first simulation and when α' decides in the second simulation. If they *both* decide by round $10\text{tc}(8, m)$, Alice outputs 1 for the original GDC problem. Otherwise Alice outputs 0. Note that if either of the simulation aborts at Line 20 of Protocol 2, Alice will output 0 as well.
- **Correctness of Alice's output** If $\text{GDC}(X, Y) = 1$, then Lemma 7 tells us that ($\mathbf{X}'$, $\mathbf{Y}'$) is of type-1 with probability at least $1 - \frac{1}{q}$. Since ($\mathbf{X}''$, $\mathbf{Y}''$) and ($\mathbf{X}'$, $\mathbf{Y}'$) must be of the same type, with at least such probability, both of them are of type-1. When both of them are of type-1, Lemma 11 later proves that with probability at least $1 - \frac{1}{5}$, α in the first simulation and α' in the second simulation both decide within $10\text{tc}(8, m)$ rounds. This will make Alice generate the correct output 1. Hence Alice generates the correct output 1 with probability at least $(1 - \frac{1}{q})(1 - \frac{1}{5}) \ge (1 - \frac{1}{20})(1 - \frac{1}{5}) > 1 - \frac{2}{5}$.

  If $\text{GDC}(X, Y) = 0$, then by Lemma 7 and similar argument as before, we know that with at least $1 - \frac{1}{q}$ probability, both ($\mathbf{X}'$, $\mathbf{Y}'$) and ($\mathbf{X}''$, $\mathbf{Y}''$) are of type-0. When both of them are of type-0, Lemma 12 later proves that with probability at most $\frac{3}{10}$, α in the first simulation and α' in the second simulation both decide within $10\text{tc}(8, m)$ rounds. Hence Alice's output is correct with probability at least $(1 - \frac{1}{q})(1 - \frac{3}{10}) \ge (1 - \frac{1}{20})(1 - \frac{3}{10}) > 1 - \frac{2}{5}$.
- **From communication complexity to time complexity** We have proved so far that Alice and Bob can solve $\text{GDC}_n^{g,q}$ with $\frac{2}{5}$ error, by simulating $\mathscr{P}$ twice. Lemma 8 tells us that in each simulation, Alice and Bob never incur more than $cq \log n$ bits of communication. Hence Alice and Bob can solve $\text{GDC}_n^{g,q}$ with no more than $2cq \log n$ bits of communication. By the lower bound in Theorem 4, we know that there exist constants c_1 and c_2 such that all $\frac{2}{5}$-error protocols for solving $\text{GDC}_n^{g,q}$ have a communication complexity of at least $\frac{c_1\sqrt{n}}{gq^{3.5}\log q} - c_2 \log \frac{\sqrt{n}}{gq^{1.5}\log q}$ bits, over average coin flips. This implies:

$$
\begin{aligned}
2cq \log n &\ge \frac{c_1\sqrt{n}}{gq^{3.5}\log q} - c_2 \log \frac{\sqrt{n}}{gq^{1.5}\log q} \\
&\Rightarrow 2cq \ge \frac{c_1\sqrt{n}}{gq^{3.5}\log q \log n} \\
&\quad - \frac{c_2}{\log n}\left(\frac{1}{2}\log n - \log(gq^{1.5}\log q)\right) \\
&> \frac{c_1\sqrt{n}}{gq^{3.5}\log q \log n} - c_2 \\
&\Rightarrow 2cq + c_2 > \frac{c_1\sqrt{n}}{15q^{4.5}\ln q \log q \log n} \\
&\ge \frac{c_1\sqrt{n}}{15q^{4.5}\log^3 m} \quad (\text{since } q < n < m) \\
&\Rightarrow 2c(20\text{tc}(8, m) + 21) + c_2 \\
&> \frac{c_1\sqrt{\frac{m-4}{3}}}{15(20\text{tc}(8, m) + 21)^{4.5}\log^3 m} \\
&\Rightarrow 2c(20m^{\frac{1}{12}} + 21) + c_2 \\
&> \frac{c_1\sqrt{\frac{m-4}{3}}}{15(20m^{\frac{1}{12}} + 21)^{4.5}\log^3 m} \quad (\text{since } m^{\frac{1}{12}} > \text{tc}(8, m))
\end{aligned}
$$

The last inequality contradicts with Inequality 4, which completes our proof by contradiction. □

Lemma 11 *Consider the processed inputs* ($\mathbf{X}'$, $\mathbf{Y}'$) *and* ($\mathbf{X}''$, $\mathbf{Y}''$) *in the proof of Theorem 7, and the corresponding first simulation and second simulation. If both processed inputs are of type-*1*, then* α *in the first simulation and* α' *in the second simulation will both decide within* $10tc(8, m)$ *rounds with probability at least* $1 - \frac{1}{5}$*, where the probability is taken*

over the coin flips of both the protocol and the adversary.[12] *Furthermore, neither the first simulation nor the second simulation will abort at Line 20 of Protocol 2.*

Proof Consider the first simulation where the reference adversary $\mathscr{A}$ is based on $(\mathbf{X}', \mathbf{Y}')$. Since $(\mathbf{X}', \mathbf{Y}')$ is of type-1, it is easy to verify that the dynamic network as generated by $\mathscr{A}$ has a diameter of no more than 8, under all possible coin flips of the CONSENSUS protocol $\mathscr{P}$ and of the reference adversary $\mathscr{A}$. We want to increase the diameter of the dynamic network to exactly 8, so that it corresponds to $\text{tc}(8, m)$. Recall from Protocol 2 that $\mathscr{P}$ is only simulated for round 1 through $\frac{q-1}{2}$. Given this, increasing the diameter to exactly 8 is trivial: Starting from round $\frac{q-1}{2} + 1$, we let the dynamic network's topology to be some fixed topology such that the resulting (dynamic) diameter of the dynamic network is exactly 8. Since the simulation has already stopped by round $\frac{q-1}{2}$, whatever we do after that will not impact the simulation in any way. (If we want to reason about $\text{tc}(d, m)$ for $d > 8$, then we should increase the diameter to exactly d, which is also trivial to achieve using the above approach.) In the next, when we refer to $\mathscr{A}$ (which was originally defined only for the first $\frac{q-1}{2}$ rounds), we will include the above topology starting from round $\frac{q-1}{2} + 1$ as well.

Section 9.2.2 already explained that the reference adversary $\mathscr{A}$ is a sanitized adaptive adversary. Let the *cost* of $\mathscr{P}$ be the number of rounds before termination. By Theorem 2, we know that there exists some deterministic oblivious adversary $\mathscr{B}$ such that $\mathscr{P}$'s expected cost under $\mathscr{B}$ is no smaller than its expected cost under $\mathscr{A}$. Furthermore also by Theorem 2, we know that for any coin flip outcomes of $\mathscr{P}$, there exist coin flip outcomes of $\mathscr{A}$, such that the decisions made by $\mathscr{B}$ are the same as the decisions made by $\mathscr{A}$ under those coin flip outcomes. Thus since the dynamic network constructed by $\mathscr{A}$ always has a diameter of 8, we know that the dynamic network constructed by $\mathscr{B}$ has a diameter of 8 as well.

When running against any given oblivious adversary where the corresponding dynamic network has a diameter of 8 and has m nodes, $\mathscr{P}$ promises to terminate within $\text{tc}(8, m)$ rounds over average coin flips. Hence $\mathscr{P}$ must terminate within $\text{tc}(8, m)$ rounds over average coin flips when running against $\mathscr{B}$. In turn, $\mathscr{P}$ must terminate within $\text{tc}(8, m)$ rounds over average coin flips (of both $\mathscr{P}$ and $\mathscr{A}$) when running against $\mathscr{A}$. By Markov inequality, $\mathscr{P}$ terminates within $10\text{tc}(8, m)$ rounds with probability at least $\frac{9}{10}$ when running against $\mathscr{A}$.

Since $10\text{tc}(8, m) \leq \frac{q-1}{2}$ and since α is always non-spoiled for Alice, Lemma 10 tells us that at Line 8 of Protocol 2, the outgoing message of α as determined by Alice must be consistent (i.e., the same as the corresponding outgoing message in the reference execution). Without loss of generality, assume that when α decides, it sends a special message. Hence if α decides within $10\text{tc}(8, m)$ rounds in the reference execution, Alice must be able to observe that.

By same argument, since $(\mathbf{X}'', \mathbf{Y}'')$ is of type-1, $\mathscr{P}$ must terminate within $10\text{tc}(8, m)$ rounds with probability at least $\frac{9}{10}$ when running against our reference adversary in the second simulation. Again by Lemma 10, Alice can observe when α' decides. A simple union bound shows that with probability at least $1 - \frac{1}{5}$, Alice will be able to observe that both α and α' decide within $10\text{tc}(8, m)$ rounds.

Finally, Lemma 10 also confirms that neither the first simulation nor the second simulation will abort at Line 20 of Protocol 2. □

Lemma 12 *Consider the processed inputs* $(\mathbf{X}', \mathbf{Y}')$ *and* $(\mathbf{X}'', \mathbf{Y}'')$ *in the proof of Theorem 7, and the corresponding first simulation and second simulation. If both processed inputs are of type-*0*, then* α *in the first simulation and* α' *in the second simulation will both decide within* $10tc(8, m)$ *rounds with probability at most* $\frac{3}{10}$*, where the probability is taken over the coin flips of both the protocol and the adversary.*[13] *Furthermore, neither the first simulation nor the second simulation will abort at Line 20 of Protocol 2.*

Proof We first prove that when the CONSENSUS protocol $\mathscr{P}$ runs against our reference adversary in the first simulation, α and γ both decide within $10\text{tc}(8, m)$ rounds with probability at most $\frac{3}{10}$.

Let $\mathscr{A}$ be our reference adversary in the first simulation, and Sect. 9.2.2 already explained that $\mathscr{A}$ is a sanitized adaptive adversary. We will need to construct another sanitized adaptive adversary $\mathscr{B}$, in the following way. Intuitively, $\mathscr{B}$ generates the same dynamic network (regardless of the initial values to the nodes) as the dynamic network generated by $\mathscr{A}$ when the initial values to the nodes are the initial values assigned in the first simulation. More precisely, under *all* possible initial values to the nodes, when $\mathscr{P}$'s coin flip outcomes are $C_{\mathscr{P}}$, and when $\mathscr{B}$'s coin flip outcomes are $C_{\mathscr{B}}$, the adversary $\mathscr{B}$ generates the dynamic network $\mathscr{G}$. Here $\mathscr{G}$ is the (unique) dynamic network generated by $\mathscr{A}$ when the initial values to the nodes are the same as the initial values assigned in the first simulation, when $\mathscr{P}$'s coin flip outcomes are $C_{\mathscr{P}}$, and when the coin flip outcomes $\mathbf{C}_{\mathscr{A}}$ of $\mathscr{A}$ satisfies $\mathbf{C}_{\mathscr{A}} = C_{\mathscr{B}}$. It is easy to verify that since $\mathscr{A}$ is a sanitized adaptive adversary, $\mathscr{B}$ must be a sanitized adaptive adversary as well.

Consider any given initial inputs to $\mathscr{P}$. For coin flip outcomes $C_{\mathscr{P}}$ of $\mathscr{P}$ and coin flip outcomes $C_{\mathscr{A}}$ of $\mathscr{A}$, define $\text{cost}(\mathscr{P}, \mathscr{A}, C_{\mathscr{P}}, C_{\mathscr{A}})$ to be 0 if the $\mathscr{P}$'s output is correct when running against $\mathscr{A}$ under $C_{\mathscr{P}}$, $C_{\mathscr{A}}$ and the given

[12] The adversary here is the reference adversary. The coin flips of the reference adversary are the same as the coin flips of the leaker. Specifically, these coin flips are all the random variables $\mathbf{s}$ in Sect. 9.2.2.

[13] See footnote 12.

initial inputs, and 1 otherwise. Since $\mathscr{A}$ is a sanitized adaptive adversary, Theorem 2 tells us that there exists some deterministic oblivious adversary such that the protocol's expected cost (over average $C_{\mathscr{P}}$) under this deterministic oblivious adversary is no smaller than its expected cost under $\mathscr{A}$. On the other hand, when executing against any given oblivious adversary and with any initial values, $\mathscr{P}$ promises to have at most $\frac{1}{10}$ error over average coin flips. Hence when running against $\mathscr{A}$ and with any initial values, $\mathscr{P}$ must have at most $\frac{1}{10}$ error over average coin flips (of both $\mathscr{P}$ and $\mathscr{A}$). By same argument, when running against $\mathscr{B}$ and with any initial values, $\mathscr{P}$ must have at most $\frac{1}{10}$ error.

Let $\mathscr{I}$ denote the CONSENSUS instance in the first simulation. We will construct two additional CONSENSUS instances, in the following way. The CONSENSUS instance $\mathscr{I}_0$ is the same as $\mathscr{I}$ except that (i) all nodes in $\mathscr{I}_0$ have initial values of 0, and (ii) $\mathscr{I}_0$ is under adversary $\mathscr{B}$ instead of $\mathscr{A}$. We similarly construct $\mathscr{I}_1$ under adversary $\mathscr{B}$ where all nodes have initial values of 1. Now consider any given coin flip outcomes $C_{\mathscr{P}}$ of $\mathscr{P}$ and coin flip outcomes $C_{\mathscr{A}}$ of the adversary (which is either $\mathscr{A}$ or $\mathscr{B}$). Note that under given $C_{\mathscr{P}}$ and $C_{\mathscr{A}}$, the dynamic networks in the three instances as determined by their respective adversaries are exactly the same. We claim that if α and γ both decide within $10\text{tc}(8, m)$ rounds, then under $C_{\mathscr{P}}$ and $C_{\mathscr{A}}$, $\mathscr{P}$ must err in either $\mathscr{I}$ or $\mathscr{I}_0$ or $\mathscr{I}_1$.

To see why, we consider two cases. If $\mathscr{P}$ err in $\mathscr{I}$, we are done. If $\mathscr{P}$ does not err in $\mathscr{I}$, without loss of generality, let the decision value be 1. This means that both α and γ decide on 1 within $10\text{tc}(8, m)$ rounds in $\mathscr{I}$. Next consider α's behavior in $\mathscr{I}_0$. Note that $C_{\mathscr{P}}$ and $C_{\mathscr{A}}$ have all been fixed, and also that $\mathscr{I}$ and $\mathscr{I}_0$ have exactly the same dynamic network. The only difference between $\mathscr{I}$ and $\mathscr{I}_0$ is the initial values. Since $q \geq 10\text{tc}(8, m)$, by the way we construct $\mathscr{A}$ and $\mathscr{B}$, it is easy to verify that for all nodes τ where $(\tau, 0) \rightsquigarrow (\alpha, 10\text{tc}(8, m))$, τ has the same initial value of 0 in both $\mathscr{I}$ and $\mathscr{I}_0$. Only a node τ such that $(\tau, 0) \rightsquigarrow (\alpha, 10\text{tc}(8, m))$ may influence α's behavior by round $10\text{tc}(8, m)$. Thus for every node τ that can influence α's behavior by round $10\text{tc}(8, m)$, τ has the same initial value in $\mathscr{I}$ and $\mathscr{I}_0$. Hence α's behavior in $\mathscr{I}$ and $\mathscr{I}_0$ must be the same. Since α decides on 1 by round $10\text{tc}(8, m)$ in $\mathscr{I}$, it must also decide on 1 by round $10\text{tc}(8, m)$ in $\mathscr{I}_0$. But such a decision value is wrong in $\mathscr{I}_0$.

We have proved that for every $C_{\mathscr{P}}$ and $C_{\mathscr{A}}$, if α and γ in the first simulation both decide within $80\text{tc}(m)$ rounds, then $\mathscr{P}$ must err in one of the 3 instances. On the other hand, as shown earlier, in each of the instances, $\mathscr{P}$ must have at most $\frac{1}{10}$ error, over average $C_{\mathscr{P}}$ and $C_{\mathscr{A}}$. Hence α and γ in the first simulation both decide within $10\text{tc}(8, m)$ rounds with probability at most $\frac{3}{10}$.

So far we have proved that when $\mathscr{P}$ runs against our reference adversary in the first simulation, α and γ both decide within $10\text{tc}(8, m)$ rounds with probability at most $\frac{3}{10}$. We call this as the *first reference execution*. Next we consider running $\mathscr{P}$ against our reference adversary in the second simulation (which we call the *second reference execution*), and consider the node α' there. One can verify that when both $(\mathbf{X}', \mathbf{Y}')$ and $(\mathbf{X}'', \mathbf{Y}'')$ are of type-0, then under the same $C_{\mathscr{P}}$ and $C_{\mathscr{A}}$, the first reference execution and the second reference execution are "isomorphic": A node with a certain id in the first reference execution must have exactly the same behavior as the node with that id in the second reference execution. This means that the behavior of node α' in the second reference execution must be exactly the same as the behavior of node γ in the first reference execution. Together with our earlier arguments, this means that with probability at most $\frac{3}{10}$, α in the first reference execution and α' in the second reference execution both decide within $10\text{tc}(8, m)$ rounds.

Finally, since $10\text{tc}(8, m) \leq \frac{q-1}{2}$ and since α and α' are always non-spoiled for Alice, Lemma 10 tells us that at Line 8 of Protocol 2, the outgoing messages of α and α' as determined by Alice must be consistent (i.e., the same as the corresponding outgoing messages in the reference execution). Hence if α and α' decide within $10\text{tc}(8, m)$ rounds in the respective reference executions, Alice will observe that. Lemma 10 also confirms that neither the first simulation nor the second simulation will abort at Line 20 of Protocol 2. □

Acknowledgements We would like to thank the Distributed Computing anonymous reviewers and the DISC'17 anonymous reviewers for their helpful feedbacks on this paper.

Appendix A: Proof for Lemma 1

Lemma 1 *For all constant $\delta \in (0, \frac{1}{2})$ and all $n \geq 1$, we have* $\mathfrak{L}_\delta\left(\text{GDC}_n^{16\sqrt{n}\ln\frac{1}{\delta},2}\right) = 0$.

Proof Define $g = 16\sqrt{n}\ln\frac{1}{\delta}$, and we will prove $\mathfrak{L}_\delta(\text{GDC}_n^{g,2}) = 0$.

If $g < (16\ln\frac{1}{\delta})^2$ and $n = g^2/(16\ln\frac{1}{\delta})^2$, then $n < g$ and hence the only possible answer for $\text{GDC}_n^{g,2}$ is 1. Thus Alice and Bob can trivially solve the problem without any communication. Next if $g \geq (16\ln\frac{1}{\delta})^2$ and $n = g^2/(16\ln\frac{1}{\delta})^2$, we construct the following protocol for solving $\text{GDC}_n^{g,2}$ with our leaker: Alice and Bob output 1 if the total number of leaked indices is at least $\frac{n}{2} - 2\sqrt{n}\ln\frac{1}{\delta}$, and 0 otherwise. We next show that this protocol has at most δ error.

Let random variable $\mathbf{z}$ denote the number of leaked indices. If the answer to the $\text{GDC}_n^{g,2}$ problem is 1, then $\mathbf{z}$ is the number of heads obtained when flipping n independent fair coins. Using Chernoff bound, the protocol's error probability is bounded as follows:

$$\Pr\left[\mathbf{z} < \frac{n}{2} - 2\sqrt{n}\ln\frac{1}{\delta}\right] = \Pr\left[\mathbf{z} < \left(1 - \frac{4\ln\frac{1}{\delta}}{\sqrt{n}}\right)\cdot\frac{n}{2}\right] \le \exp\left(-\frac{1}{2}\cdot\frac{n}{2}\cdot\frac{16\ln^2\frac{1}{\delta}}{n}\right) < \delta$$

If the answer to the $\text{GDC}_n^{g,2}$ problem is 0, then $\mathbf{z}$ is the number of heads obtained when flipping $n-b$ independent fair coins where $b \ge g = 16\sqrt{n}\ln\frac{1}{\delta}$. Let $\mathbf{z}'$ be the number of heads obtained when flipping $n - 14\sqrt{n}\ln\frac{1}{\delta}$ independent fair coins. Using Chernoff bound, the protocol's error probability is bounded as follows:

$$\begin{aligned}
&\Pr\left[\mathbf{z} \ge \frac{n}{2} - 2\sqrt{n}\ln\frac{1}{\delta}\right] \\
&< \Pr\left[\mathbf{z}' \ge \frac{n}{2} - 2\sqrt{n}\ln\frac{1}{\delta}\right] \\
&\le \Pr\left[\mathbf{z}' \ge \frac{n}{2} - 2\sqrt{n}\ln\frac{1}{\delta} - 70\ln^2\frac{1}{\delta}\right] \\
&= \Pr\left[\mathbf{z}' \ge \left(1 + \frac{10\ln\frac{1}{\delta}}{\sqrt{n}}\right)\cdot\left(\frac{n}{2} - 7\sqrt{n}\ln\frac{1}{\delta}\right)\right] \\
&\le \exp\left(-\frac{1}{3}\cdot\left(\frac{n}{2} - 7\sqrt{n}\ln\frac{1}{\delta}\right)\cdot\frac{100\ln^2\frac{1}{\delta}}{n}\right) \\
&= \exp\left(-\ln\frac{1}{\delta}\cdot\left(\frac{100\ln\frac{1}{\delta}}{6} - \frac{700\ln^2\frac{1}{\delta}}{3\sqrt{n}}\right)\right) \\
&\le \exp\left(-\ln\frac{1}{\delta}\cdot\left(\frac{100\ln\frac{1}{\delta}}{6} - \frac{700\ln\frac{1}{\delta}}{48}\right)\right) \\
&< \delta
\end{aligned}$$

The second to the last inequality holds since $n = g^2/(16\ln\frac{1}{\delta})^2 \ge (16\ln\frac{1}{\delta})^2$. □

Appendix B: Proof for Lemma 3

The proof of Lemma 3 will need to use the following theorem. Let $\mathbb{B}(\mu)$ denote the binomial distribution corresponding to the number of heads in 2μ fair coin flips.

Theorem 8 *Consider any positive integer k, any μ_i and μ_i' where $2\mu_i$ and $2\mu_i'$ are all integers ($1 \le i \le k$). Let $\mu = \min_{1\le i\le k}(\min(\mu_i, \mu_i'))$. Let δ and δ' be any given constants where $0 < \delta' < \delta < 0.5$. Let product distribution $\mathbb{D} = \mathbb{B}(\mu_1)\times\mathbb{B}(\mu_2)\times\ldots\times\mathbb{B}(\mu_k)$ and $\mathbb{D}' = \mathbb{B}(\mu_1')\times\mathbb{B}(\mu_2')\times\ldots\times\mathbb{B}(\mu_k')$. If $\mu \ge \frac{250}{(\delta-\delta')^2}(k^2 + k\max_{1\le i\le k}(\mu_i - \mu_i')^2)$, then $||\mathbb{D} - \mathbb{D}'|| \le \frac{2(\delta-\delta')}{3}$.*

The proof of Theorem 8 is long and complex—we will present its proof separately in "Appendix C".

Lemma 3 *If Protocol 1's estimates are good and if it does not exit at Line 12, then for all (X, Y), we have $||\hat{\mathbb{T}}(X,Y) - \mathbb{T}(X,Y)|| \le \frac{9(\delta'-\delta)}{12}$.*

Proof Consider any (X, Y) in the support of $(\mathbf{X}, \mathbf{Y})$. For clarity, we write $\hat{\mathbb{T}}(X, Y)$ as $\hat{\mathbb{T}}$ and $\mathbb{T}(X, Y)$ as $\mathbb{T}$. Let random variables $\hat{\mathbf{T}} \sim \hat{\mathbb{T}}$ and $\mathbf{T} \sim \mathbb{T}$. Given a leaked set (e.g., $\mathbf{T}$), define its corresponding *leaked vector* as a vector of k integers, where the jth entry is the number of indices i such that (i, x_i, y_i) is in the leaked set and (x_i, y_i) is the jth leakable pattern. Define many-to-one mapping $\rho(\cdot)$, which maps each leaked set to its corresponding leaked vector. Define random variables $\hat{\mathbf{S}} = \rho(\hat{\mathbf{T}})$ and $\mathbf{S} = \rho(\mathbf{T})$. Let the distributions of $\hat{\mathbf{S}}$ and $\mathbf{S}$ be $\hat{\mathbb{S}}$ and $\mathbb{S}$, respectively. By the behavior of the leaker, it is obvious that $\mathbb{S} = \Pi_{j=1}^{k}\mathbb{B}(\frac{h_j+w_j}{2})$. On the other hand, due to Line 15 in Protocol 1, $\hat{\mathbb{S}}$ is different from $\Pi_{j=1}^{k}\mathbb{B}(\frac{h_j+\mathbf{v}_j}{2})$. Define $\rho^{-1}(S) = \{T|\rho(T) = S\}$. Let $\text{supp}(\cdot)$ denote the support of a distribution, and let $f_{\mathbb{D}}(\cdot)$ denote the probability density function of a distribution $\mathbb{D}$. It is easy to see that $\text{supp}(\hat{\mathbb{T}}) = \{T|T \in \rho^{-1}(S) \text{ and } S \in \text{supp}(\hat{\mathbb{S}})\}$ and $\text{supp}(\mathbb{T}) = \{T|T \in \rho^{-1}(S) \text{ and } S \in \text{supp}(\mathbb{S})\}$.

We will first prove that $||\hat{\mathbb{T}} - \mathbb{T}|| = ||\hat{\mathbb{S}} - \mathbb{S}||$. To do so, we observe that conditioned on $\hat{\mathbf{S}} = \mathbf{S}$, the distributions of $\hat{\mathbf{T}}$ and $\mathbf{T}$ are the same, which implies that for all $T \in \text{supp}(\hat{\mathbb{T}}) \cup \text{supp}(\mathbb{T})$ and all $S \in \text{supp}(\hat{\mathbb{S}}) \cap \text{supp}(\mathbb{S})$, $\Pr[\hat{\mathbf{T}} = T|\hat{\mathbf{S}} = S] = \Pr[\mathbf{T} = T|\mathbf{S} = S]$. To see why such an observation holds, let $\hat{\mathbf{s}}_j$ ($1 \le j \le k$) denote the jth entry in $\hat{\mathbf{S}}$. By the construction of Protocol 1, $\hat{\mathbf{T}}$ can be viewed as choosing $\hat{\mathbf{s}}_j$ uniformly random indices among all indices that corresponds to the jth leakable pattern, and then leak those indices. This holds because all the indices were permuted using a uniformly random permutation at Line 20 in Protocol 1. Similarly, by the behavior of the leaker, T can be viewed as being generated via such a process as well.

Thus we have:

$$\begin{aligned}
&||\hat{\mathbb{T}} - \mathbb{T}|| \\
&= \sum_{T\in\text{supp}(\hat{\mathbb{T}})\cup\text{supp}(\mathbb{T})} |f_{\hat{\mathbb{T}}}(T) - f_{\mathbb{T}}(T)| \\
&= \sum_{S\in\text{supp}(\hat{\mathbb{S}})\setminus\text{supp}(\mathbb{S})} \sum_{T\in\rho^{-1}(S)} f_{\hat{\mathbb{T}}}(T) \\
&\quad + \sum_{S\in\text{supp}(\mathbb{S})\setminus\text{supp}(\hat{\mathbb{S}})} \sum_{T\in\rho^{-1}(S)} f_{\mathbb{T}}(T)
\end{aligned}$$

$$+ \sum_{S \in \text{supp}(\hat{\mathbb{S}}) \cap \text{supp}(\mathbb{S})} \sum_{T \in \rho^{-1}(S)} |f_{\hat{\mathbb{S}}}(S) \Pr[\hat{\mathbf{T}} = T | \hat{\mathbf{S}} = S] - f_{\mathbb{S}}(S) \Pr[\mathbf{T} = T | \mathbf{S} = S]|$$

Leveraging the earlier claim that $\Pr[\hat{\mathbf{T}} = T|\hat{\mathbf{S}} = S] = \Pr[\mathbf{T} = T|\mathbf{S} = S]$, we can simplify the third term:

$$\begin{aligned}
&\sum_{S \in \text{supp}(\hat{\mathbb{S}}) \cap \text{supp}(\mathbb{S})} \sum_{T \in \rho^{-1}(S)} |f_{\hat{\mathbb{S}}}(S) \Pr[\hat{\mathbf{T}} = T|\hat{\mathbf{S}} = S] \\
&\quad - f_{\mathbb{S}}(S) \Pr[\mathbf{T} = T|\mathbf{S} = S]| \\
&= \sum_{S \in \text{supp}(\hat{\mathbb{S}}) \cap \text{supp}(\mathbb{S})} \sum_{T \in \rho^{-1}(S)} (|f_{\hat{\mathbb{S}}}(S) - f_{\mathbb{S}}(S)| \\
&\quad \times \Pr[\mathbf{T} = T|\mathbf{S} = S]|) \\
&= \sum_{S \in \text{supp}(\hat{\mathbb{S}}) \cap \text{supp}(\mathbb{S})} (|f_{\hat{\mathbb{S}}}(S) - f_{\mathbb{S}}(S)| \\
&\quad \times \sum_{T \in \rho^{-1}(S)} \Pr[\mathbf{T} = T|\mathbf{S} = S]|) \\
&= \sum_{S \in \text{supp}(\hat{\mathbb{S}}) \cap \text{supp}(\mathbb{S})} |f_{\hat{\mathbb{S}}}(S) - f_{\mathbb{S}}(S)|
\end{aligned}$$

Combining this result with the earlier equation, we have:

$$\begin{aligned}
&||\hat{\mathbb{T}} - \mathbb{T}|| \\
&= \sum_{S \in \text{supp}(\hat{\mathbb{S}}) \setminus \text{supp}(\mathbb{S})} \sum_{T \in \rho^{-1}(S)} f_{\hat{\mathbb{T}}}(T) \\
&\quad + \sum_{S \in \text{supp}(\mathbb{S}) \setminus \text{supp}(\hat{\mathbb{S}})} \sum_{T \in \rho^{-1}(S)} f_{\mathbb{T}}(T) \\
&\quad + \sum_{S \in \text{supp}(\hat{\mathbb{S}}) \cap \text{supp}(\mathbb{S})} |f_{\hat{\mathbb{S}}}(S) - f_{\mathbb{S}}(S)| \\
&= \sum_{S \in \text{supp}(\hat{\mathbb{S}}) \setminus \text{supp}(\mathbb{S})} f_{\hat{\mathbb{S}}}(S) + \sum_{S \in \text{supp}(\mathbb{S}) \setminus \text{supp}(\hat{\mathbb{S}})} f_{\mathbb{S}}(S) \\
&\quad + \sum_{S \in \text{supp}(\hat{\mathbb{S}}) \cap \text{supp}(\mathbb{S})} |f_{\hat{\mathbb{S}}}(S) - f_{\mathbb{S}}(S)| \\
&= ||\hat{\mathbb{S}} - \mathbb{S}||
\end{aligned}$$

We have proved that $||\hat{\mathbb{T}} - \mathbb{T}|| = ||\hat{\mathbb{S}} - \mathbb{S}||$. Let distribution $\mathbb{D} = \Pi_{j=1}^{k} \mathbb{B}(\frac{h_j + \mathbf{v}_j}{2})$. We will next prove that:

$$||\mathbb{S} - \mathbb{D}|| \leq \frac{8(\delta' - \delta)}{12} \tag{5}$$

$$||\hat{\mathbb{S}} - \mathbb{D}|| \leq \frac{\delta' - \delta}{12} \tag{6}$$

If these two inequalities do hold, then we will have $||\hat{\mathbb{T}} - \mathbb{T}|| = ||\hat{\mathbb{S}} - \mathbb{S}|| \leq ||\mathbb{S} - \mathbb{D}|| + ||\hat{\mathbb{S}} - \mathbb{D}|| \leq \frac{9(\delta' - \delta)}{12}$.

We first prove $||\mathbb{S} - \mathbb{D}|| \leq \frac{8(\delta' - \delta)}{12}$, by using Theorem 8. Recall that $\mathbb{S} = \Pi_{j=1}^{k} \mathbb{B}(\frac{h_j + w_j}{2})$. Let $\mu_j = \frac{h_j + w_j}{2}$ and $\hat{\mu}_j = \frac{h_j + \mathbf{v}_j}{2}$ for $1 \leq j \leq k$. Let $\mu = \min_{1 \leq j \leq k}(\min(\mu_j, \hat{\mu}_j)) \geq \min_{1 \leq j \leq k}(\frac{h_j}{2}) = \min(\frac{h}{2}, \frac{h_k}{2})$. Since Protocol 1 does not exit at Line 12, we have $h_k \geq h$. Hence $\mu \geq \frac{h}{2}$. Since Protocol 1's estimates are good, by the definition of estimates being good, we have:

$$\begin{aligned}
\mu \geq \frac{h}{2} &\geq n' + \frac{250}{(\delta' - \delta)^2} \left(k^2 + k \max_{1 \leq j \leq k} (\mathbf{v}_j - w_j)^2 \right) \\
&> \frac{250}{(\delta' - \delta)^2} \left(k^2 + 4k \max_{1 \leq j \leq k} (\mu_j - \hat{\mu}_j)^2 \right) \\
&> \frac{250}{(\delta' - \delta)^2} \left(k^2 + k \max_{1 \leq j \leq k} (\mu_j - \hat{\mu}_j)^2 \right)
\end{aligned}$$

Applying Theorem 8 immediately tell us that $||\mathbb{S} - \mathbb{D}|| \leq \frac{8(\delta' - \delta)}{12}$.

Next we prove $||\hat{\mathbb{S}} - \mathbb{D}|| \leq \frac{\delta' - \delta}{12}$. The difference between $\hat{\mathbb{S}}$ and $\mathbb{D}$ arises solely from Line 15 in Protocol 1. when $\mathbf{b}_j > h_j$ for some j. Hence:

$$||\hat{\mathbb{S}} - \mathbb{D}|| \leq \Pr[\exists j \text{ where } 1 \leq j \leq k, \text{such that } \mathbf{b}_j > h_j]$$

We trivially have $h \geq 2n' \geq 2\mathbf{v}_j$ for all j. Since Protocol 1 does not exit at Line 12, we have $h \leq h_j$ for all j. Hence $\mathbf{v}_j \leq \frac{h}{2} \leq \frac{h_j}{2}$. The random variable $\mathbf{b}_j$ is drawn from the binomial distribution $\mathbb{B}(\frac{h_j + \mathbf{v}_j}{2})$ at Line 14 of Protocol 1, and thus $\frac{1}{2}h_j \leq E[\mathbf{b}_j] \leq \frac{3}{4}h_j$. By Chernoff bound, for any given j, we have:

$$\begin{aligned}
\Pr[\mathbf{b}_j \geq h_j] &\leq \Pr\left[\mathbf{b}_j \geq \frac{4}{3}E[\mathbf{b}_j]\right] \leq \exp\left(-\frac{1}{3}\left(\frac{1}{3}\right)^2 E[\mathbf{b}_j]\right) \\
&\leq \exp\left(-\frac{1}{3}\left(\frac{1}{3}\right)^2 \frac{1}{2}h_j\right) = \exp\left(-\frac{1}{54}h_j\right) \\
&< \exp\left(-\frac{500k^2}{54(\delta' - \delta)^2}\right) \\
&< \frac{54(\delta' - \delta)^2}{500k^2} \quad (\text{since } \exp(-x) < 1/x \text{ for } x > 0) \\
&< \frac{\delta' - \delta}{12k} \quad \left(\text{since } 0 < \delta < \delta' < \frac{1}{2}\right)
\end{aligned}$$

Finally, taking a union bound for j from 1 through k gives:

$$\begin{aligned}
||\hat{\mathbb{S}} - \mathbb{D}|| &\leq \Pr[\exists j \text{ where } 1 \leq j \leq k, \text{such that } \mathbf{b}_j > h] \\
&< k \cdot \frac{\delta' - \delta}{12k} = \frac{\delta' - \delta}{12}
\end{aligned}$$

□

Appendix C: Proof for Theorem 8

The section proves Theorem 8. Theorem 8 is not a surprising result, and we do not claim it as a major contribution. We

include this proof here mainly for completeness—while the overall approach is quite natural, the proof does involve some tedious and complicated steps, in order to get a relatively strong result. In particular, a weaker form Theorem 8 (i.e., by requiring μ in the theorem to be much larger) can be proved via a less complicated approach. But this weaker form would negatively impact the final asymptotic results in this paper.

Notations. We introduce some additional notations to be used in this section. For any μ where 2μ is a positive integer, recall that $\mathbb{B}(\mu)$ is defined to be the binomial distribution describing the number of heads obtained when flipping 2μ independent fair coins. Define continuous distribution $\widetilde{\mathbb{B}}(\mu)$ to be the distribution whose density function is $\frac{1}{2^{2\mu}}\binom{2\mu}{\lfloor x \rfloor}$ for $0 \leq x < 2\mu + 1$, and 0 for other x values. It is easy to verify that $\widetilde{\mathbb{B}}(\mu)$ is indeed a distribution. Intuitively, $\widetilde{\mathbb{B}}(\mu)$ is the continuous version of $\mathbb{B}(\mu)$. Define $\mathbb{N}(\mu)$ to be the normal distribution whose mean is μ and whose variance is $\mu/2$. For any distribution $\mathbb{D}$, $f_{\mathbb{D}}$ is the distribution's density function.

Recall that for any two given distributions $\mathbb{D}$ and $\mathbb{D}'$, with $\mathscr{D}$ being the sample space of $\mathbb{D}$ and $\mathbb{D}'$, we use $||\mathbb{D} - \mathbb{D}'||$ to denote their L_1 *distance*. The L_1 distance is defined as $\int_{x\in\mathscr{D}} |f_{\mathbb{D}}(x) - f_{\mathbb{D}'}(x)|dx$ if $\mathscr{D}$ is continuous, and $\sum_{x\in\mathscr{D}} |f_{\mathbb{D}}(x) - f_{\mathbb{D}'}(x)|$ if $\mathscr{D}$ is discrete. We use $D_{KL}(\mathbb{D}||\mathbb{D}')$ to denote their *KL Distance*, defined as $\int_{x\in\mathscr{D}} f_{\mathbb{D}}(x) \ln \frac{f_{\mathbb{D}}(x)}{f_{\mathbb{D}'}(x)} dx$ if $\mathscr{D}$ is continuous, and $\sum_{x\in\mathscr{D}} f_{\mathbb{D}}(x) \ln \frac{f_{\mathbb{D}}(x)}{f_{\mathbb{D}'}(x)}$ if $\mathscr{D}$ is discrete.

Overview of the proof. Theorem 8 is concerned with $||\mathbb{D} - \mathbb{D}'||$, where $\mathbb{D} = \mathbb{B}(\mu_1) \times \mathbb{B}(\mu_2) \times \cdots \times \mathbb{B}(\mu_k)$ and $\mathbb{D}' = \mathbb{B}(\mu'_1) \times \mathbb{B}(\mu'_2) \times \cdots \times \mathbb{B}(\mu'_k)$. The overall approach in our proof is to first show that $||\mathbb{D} - \mathbb{D}'||$ is close to $||\mathbb{N} - \mathbb{N}'||$, where $\mathbb{N} = \mathbb{N}(\mu_1) \times \mathbb{N}(\mu_2) \times \cdots \times \mathbb{N}(\mu_k)$ and $\mathbb{N}' = \mathbb{N}(\mu'_1) \times \mathbb{N}(\mu'_2) \times \cdots \times \mathbb{N}(\mu'_k)$. Next we will use existing results to upper bound $D_{KL}(\mathbb{N}||\mathbb{N}')$, which translates to an upper bound on $||\mathbb{N} - \mathbb{N}'||$, and in turn an upper bound on $||\mathbb{D} - \mathbb{D}'||$.

It is not surprising that $||\mathbb{D} - \mathbb{D}'||$ is close to $||\mathbb{N} - \mathbb{N}'||$, since normal distribution can be used to approximate binomial distribution. For our proof, however, the complexity arises from need to quantify the approximation error. Since $\mathbb{D}$ and $\mathbb{D}'$ are not continuous distributions, we cannot directly compare them with $\mathbb{N}$ and $\mathbb{N}'$. Hence we first consider $\widetilde{\mathbb{D}}$ and $\widetilde{\mathbb{D}}'$, where $\widetilde{\mathbb{D}} = \widetilde{\mathbb{B}}(\mu_1) \times \widetilde{\mathbb{B}}(\mu_2) \times \cdots \times \widetilde{\mathbb{B}}(\mu_k)$ and $\widetilde{\mathbb{D}}' = \widetilde{\mathbb{B}}(\mu'_1) \times \widetilde{\mathbb{B}}(\mu'_2) \times \cdots \times \widetilde{\mathbb{B}}(\mu'_k)$. In other words, they are the continuous versions of $\mathbb{D}$ and $\mathbb{D}'$. It is easy to show that $||\mathbb{D} - \mathbb{D}'|| = ||\widetilde{\mathbb{D}} - \widetilde{\mathbb{D}}'||$. We then show that the continuous distributions $\widetilde{\mathbb{D}}$ and $\widetilde{\mathbb{D}}'$ are close to $\mathbb{N}$ and $\mathbb{N}'$, respectively. To do so, we will prove that $f_{\widetilde{\mathbb{B}}}(x)$ is close to $f_{\mathbb{N}}(x)$, and in turn that $\widetilde{\mathbb{B}}(\mu)$ is close to $\mathbb{N}(\mu)$.

C.1 Basic technical lemmas

We first cite a strong form of Stirling's formula, and then prove a few basic technical lemmas. All these will be useful later.

Lemma 13 [6] *For all positive integer i,*

$$i^i e^{-i}\sqrt{2i\pi}\sqrt{\frac{i}{i-0.149}} < i! < i^i e^{-i}\sqrt{2i\pi}\sqrt{\frac{i}{i-1/6}}$$

Lemma 14 *For all real number $x \in [0, 1)$, $\ln(1-x) \geq \frac{-x}{1-x}$.*

Proof Let $f(x) = \ln(1-x)$. From Taylor's theorem, we have $f(x) = f(0) + f'(\xi)x = \frac{-x}{1-\xi}$, where ξ is a real number between 0 and x. The lemma follows since $\frac{-x}{1-\xi} \geq \frac{-x}{1-x}$. □

Lemma 15 *For all real number $x \in (-1, 1)$:*

$$(1+x)\ln(1+x) + (1-x)\ln(1-x) \geq x^2 + \frac{1}{6}x^4$$

For all real number $x \in [-0.5, 0.5]$:

$$(1+x)\ln(1+x) + (1-x)\ln(1-x) \leq x^2 + \frac{1}{3}x^4$$

Proof For the first inequality, define $f(x) = (1+x)\ln(1+x) + (1-x)\ln(1-x) - x^2 - \frac{1}{6}x^4$. We have:

$$f'(x) = \ln\left(\frac{1+x}{1-x}\right) - 2x - \frac{4}{6}x^3$$
$$f''(x) = \frac{-2x^4}{x^2-1}$$

It is easy to verify that $\lim_{x\to -1} f'(x) \to -\infty$, $\lim_{x\to 1} f'(x) \to \infty$, $f''(0) = 0$, and $f''(x) > 0$ for $x \in (-1, 0)$ and for $x \in (0, -1)$. This means that $f'(x) = 0$ has a unique root, which is $f'(0) = 0$. Next since $f(0) = 0$ and $\lim_{x\to -1} f(x) = \lim_{x\to 1} f(x) = 2\ln(2) - 1 - \frac{1}{6} > 0$, we know that $f(x) \geq 0$ for $x \in (-1, 1)$.

For the second inequality, define $f(x) = (1+x)\ln(1+x) + (1-x)\ln(1-x) - x^2 - \frac{1}{3}x^4$. We have:

$$f'(x) = \ln\left(\frac{1+x}{1-x}\right) - 2x - \frac{4}{3}x^3$$
$$f''(x) = \frac{2x^2 - 4x^4}{x^2-1}$$

It is easy to verify that $f'(-0.5) > 0$, $f'(0.5) < 0$, $f''(0) = 0$, and $f''(x) < 0$ for $x \in (-1, 0)$ and for $x \in (0, -1)$. This means that $f'(x) = 0$ has a unique root, which is $f'(0) = 0$. Next since $f(0) = 0$ and $f(-0.5) = f(0.5) < 0$, we know that $f(x) \leq 0$ for $x \in [-0.5, 0.5]$. □

C.2 Proof for $f_{\widetilde{\mathbb{B}}}(x)$ being close to $f_{\mathbb{N}}(x)$

Lemma 16 below proves that $f_{\widetilde{\mathbb{B}}}(x)$ and $f_{\mathbb{N}}(x)$ are close to easy other, under certain conditions.

Lemma 16 *Let $f_{\widetilde{\mathbb{B}}}(x)$ and $f_{\mathbb{N}}(x)$ be the probability density function for $\widetilde{\mathbb{B}}(\mu)$ and $\mathbb{N}(\mu)$, respectively.*

For all integer i where $1 \le i \le 2\mu - 1$:

$$\frac{f_{\widetilde{\mathbb{B}}}(i)}{f_{\mathbb{N}}(i)} < \exp\left(-\frac{\delta^4}{6\mu^3} + \frac{\delta^2}{2(\mu^2-\delta^2)}\right) \quad \text{where } \delta = i - \mu$$

If $\mu \ge 16$, then for all integer i where $\mu - 2\sqrt{\mu} \le i \le \mu + 2\sqrt{\mu}$:

$$\frac{f_{\widetilde{\mathbb{B}}}(i)}{f_{\mathbb{N}}(i)} > \left(1 - \frac{6}{\mu}\right)$$

Proof Let $\delta = i - \mu$, $a = (\frac{\mu}{\mu+\delta})^{\mu+\delta}(\frac{\mu}{\mu-\delta})^{\mu-\delta}$, and $b = 1/\sqrt{1-\frac{\delta^2}{\mu^2}}$. We first derive a simple equation:

$$\begin{aligned}
&\frac{1}{2^{2\mu}} \frac{(2\mu)^{2\mu}e^{-2\mu}\sqrt{2\pi\cdot 2\mu}}{i^i e^{-i}\sqrt{2\pi i}(2\mu-i)^{(2\mu-i)}e^{-(2\mu-i)}\sqrt{2\pi(2\mu-i)}} \\
&= \left(\frac{2\mu}{2i}\right)^i \left(\frac{2\mu}{4\mu-2i}\right)^{(2\mu-i)} \sqrt{\frac{2\mu}{2\pi i(2\mu-i)}} \\
&= \left(\frac{\mu}{\mu+\delta}\right)^{\mu+\delta} \left(\frac{\mu}{\mu-\delta}\right)^{\mu-\delta} \sqrt{\frac{\mu}{\pi(\mu+\delta)(\mu-\delta)}} \\
&= \left(\frac{\mu}{\mu+\delta}\right)^{\mu+\delta} \left(\frac{\mu}{\mu-\delta}\right)^{\mu-\delta} \frac{1}{\sqrt{1-\frac{\delta^2}{\mu^2}}} \frac{1}{\sqrt{\pi\mu}} = \frac{ab}{\sqrt{\mu\pi}}
\end{aligned}$$

Next, we upper bound $\frac{f_{\widetilde{\mathbb{B}}}(i)}{f_{\mathbb{N}}(i)}$ for $1 \le i \le 2\mu - 1$. Apply Lemma 13, and we have:

$$\begin{aligned}
f_{\widetilde{\mathbb{B}}}(i) &= \frac{1}{2^{2\mu}}\binom{2\mu}{i} = \frac{1}{2^{2\mu}}\frac{(2\mu)!}{i!(2\mu-i)!} \\
&< \left(\frac{1}{2^{2\mu}} \frac{(2\mu)^{2\mu}e^{-2\mu}\sqrt{2\pi\cdot 2\mu}}{i^i e^{-i}\sqrt{2\pi i}(2\mu-i)^{(2\mu-i)}e^{-(2\mu-i)}\sqrt{2\pi(2\mu-i)}}\right) \\
&\quad \times \sqrt{\frac{2\mu}{2\mu-1/6}\cdot\frac{i-0.149}{i}\cdot\frac{2\mu-i-0.149}{2\mu-i}} \\
&= \frac{ab}{\sqrt{\mu\pi}} \times \sqrt{\frac{2\mu}{2\mu-1/6}\cdot\frac{\min(i,2\mu-i)-0.149}{\min(i,2\mu-i)}} \\
&\quad \times \sqrt{\frac{\max(i,2\mu-i)-0.149}{\max(i,2\mu-i)}} \\
&< \frac{ab}{\sqrt{\mu\pi}} \times \sqrt{\frac{2\mu}{2\mu-1/6}\frac{\min(i,2\mu-i)-0.149}{\min(i,2\mu-i)}} \\
&\le \frac{ab}{\sqrt{\mu\pi}} \times \sqrt{\frac{2\mu}{2\mu-1/6}\cdot\frac{\mu-0.149}{\mu}} < \frac{ab}{\sqrt{\mu\pi}}
\end{aligned}$$

Lemma 15 tells us that:

$$\begin{aligned}
\ln a &= (\mu+\delta)\ln\left(\frac{\mu}{\delta+\mu}\right) + (\mu-\delta)\ln\left(\frac{\mu}{\mu-\delta}\right) \\
&= -(\mu+\delta)\ln\left(1+\frac{\delta}{\mu}\right) - (\mu-\delta)\ln\left(1-\frac{\delta}{\mu}\right) \\
&\le -\mu\left(\left(\frac{\delta}{\mu}\right)^2 + \frac{1}{6}\left(\frac{\delta}{\mu}\right)^4\right) = -\frac{\delta^2}{\mu} - \frac{\delta^4}{6\mu^3}
\end{aligned}$$

For b, applying Lemma 14 yields $\ln b = -\frac{1}{2}\ln(1-\frac{\delta^2}{\mu^2}) \le \frac{1}{2}\cdot\frac{\frac{\delta^2}{\mu^2}}{1-\frac{\delta^2}{\mu^2}}$. Therefore:

$$\begin{aligned}
f_{\widetilde{\mathbb{B}}}(i) &< \frac{ab}{\sqrt{\mu\pi}} \le \exp\left(-\frac{\delta^2}{\mu} - \frac{\delta^4}{6\mu^3}\right)\exp\left(\frac{\frac{\delta^2}{\mu^2}}{2\left(1-\frac{\delta^2}{\mu^2}\right)}\right)\frac{1}{\sqrt{\mu\pi}} \\
&= f_{\mathbb{N}}(i)\exp\left(-\frac{\delta^4}{6\mu^3} + \frac{\delta^2}{2(\mu^2-\delta^2)}\right)
\end{aligned}$$

Finally, we lower bound $\frac{f_{\widetilde{\mathbb{B}}}(i)}{f_{\mathbb{N}}(i)}$ for $\mu - 2\sqrt{\mu} \le i \le \mu + 2\sqrt{\mu}$. Applying Lemma 13 again in a similar way as before, we have:

$$\begin{aligned}
f_{\widetilde{\mathbb{B}}}(i) &= \frac{1}{2^{2\mu}}\binom{2\mu}{i} = \frac{1}{2^{2\mu}}\frac{(2\mu)!}{i!(2\mu-i)!} \\
&> \left(\frac{1}{2^{2\mu}} \frac{(2\mu)^{2\mu}e^{-2\mu}\sqrt{2\pi\cdot 2\mu}}{i^i e^{-i}\sqrt{2\pi i}(2\mu-i)^{(2\mu-i)}e^{-(2\mu-i)}\sqrt{2\pi(2\mu-i)}}\right) \\
&\quad \times \sqrt{\frac{2\mu}{2\mu-0.149}\cdot\frac{\mu+\delta-1/6}{\mu+\delta}\cdot\frac{\mu-\delta-1/6}{\mu-\delta}} \\
&= \frac{ab}{\sqrt{\mu\pi}} \times \sqrt{\frac{2\mu}{2\mu-0.149}\cdot\frac{\mu+\delta-1/6}{\mu+\delta}\cdot\frac{\mu-\delta-1/6}{\mu-\delta}} \\
&> \frac{ab}{\sqrt{\mu\pi}} \times \sqrt{\frac{\mu+\delta-1/6}{\mu+\delta}\cdot\frac{\mu-\delta-1/6}{\mu-\delta}} \\
&= \frac{ab}{\sqrt{\mu\pi}} \times \sqrt{1+\frac{1-12\mu}{36\mu^2-36\delta^2}} \\
&\ge \frac{ab}{\sqrt{\mu\pi}} \times \sqrt{1+\frac{1-12\mu}{36\mu^2-144\mu}} \\
&\quad \text{(since } \delta^2 \le 4\mu \text{ and } \mu^2 > 4\mu\text{)} \\
&> \frac{ab}{\sqrt{\mu\pi}} \times \sqrt{1-\frac{12\mu}{36\mu^2-18\mu^2}} \quad \text{(since } u \ge 8\text{)} \\
&= \frac{ab}{\sqrt{\mu\pi}} \times \sqrt{1-\frac{2}{3\mu}} > \frac{ab}{\sqrt{\mu\pi}} \times \left(1-\frac{2}{3\mu}\right)
\end{aligned}$$

Next because $\delta \le 2\sqrt{\mu}$ and $\mu \ge 16$, we have $|\delta/\mu| \le 0.5$. Lemma 15 tells us that:

$$\begin{aligned}\ln a &= (\mu+\delta)\ln\left(\frac{\mu}{\delta+\mu}\right)+(\mu-\delta)\ln\left(\frac{\mu}{\mu-\delta}\right)\\ &= -(\mu+\delta)\ln\left(1+\frac{\delta}{\mu}\right)-(\mu-\delta)\ln\left(1-\frac{\delta}{\mu}\right)\\ &\geq -\mu\left(\left(\frac{\delta}{\mu}\right)^2+\frac{1}{3}\left(\frac{\delta}{\mu}\right)^4\right)=-\frac{\delta^2}{\mu}-\frac{\delta^4}{3\mu^3}\end{aligned}$$

Therefore:

$$\begin{aligned}f_{\widetilde{\mathbb{B}}}(i) &> \frac{ab}{\sqrt{\mu\pi}}\times\left(1-\frac{2}{3\mu}\right)\\ &\geq \exp\left(-\frac{\delta^2}{\mu}-\frac{\delta^4}{3\mu^3}\right)\cdot\frac{1}{\sqrt{1-\frac{\delta^2}{\mu^2}}}\cdot\left(1-\frac{2}{3\mu}\right)\cdot\frac{1}{\sqrt{\mu\pi}}\\ &> \exp\left(-\frac{\delta^2}{\mu}-\frac{\delta^4}{3\mu^3}\right)\cdot\left(1-\frac{2}{3\mu}\right)\cdot\frac{1}{\sqrt{\mu\pi}}\\ &\geq f_{\mathbb{N}}(i)\cdot\left(1-\frac{\delta^4}{3\mu^3}\right)\cdot\left(1-\frac{2}{3\mu}\right)\\ &\geq f_{\mathbb{N}}(i)\cdot\left(1-\frac{16\mu^2}{3\mu^3}\right)\cdot\left(1-\frac{2}{3\mu}\right)\quad(\text{since }\delta^2\leq 4\mu)\\ &\geq f_{\mathbb{N}}(i)\left(1-\frac{6}{\mu}\right)\end{aligned}$$

□

C.3 Upper bound $||\widetilde{\mathbb{B}}(\mu)-\mathbb{N}(\mu)||$

This section shows that $\widetilde{\mathbb{B}}(\mu)$ and $\mathbb{N}(\mu)$ are close, by leveraging the lemma proved in the previous section.

Lemma 17 *For $\mu\geq 50$ where 2μ is an integer, $||\widetilde{\mathbb{B}}(\mu)-\mathbb{N}(\mu)||<\frac{4.3}{\sqrt{\mu}}$.*

Proof Let $f_{\widetilde{\mathbb{B}}}(x)$ and $f_{\mathbb{N}}(x)$ be the probability density function for $\widetilde{\mathbb{B}}(\mu)$ and $\mathbb{N}(\mu)$, respectively.

We also define function $f_{\widetilde{\mathbb{N}}}(x)=f_{\mathbb{N}}(\lfloor x\rfloor)$. Note that $f_{\widetilde{\mathbb{N}}}$ is not necessarily a probability density function. Define $S=\{\text{integer } i \mid \mu-2\sqrt{\mu}\leq i\leq\mu+2\sqrt{\mu}\}$. We have:

$$\begin{aligned}&||\widetilde{\mathbb{B}}(\mu)-N(\mu)||\\ &=\int_{-\infty}^{\infty}|f_{\widetilde{\mathbb{B}}}(x)-f_{\mathbb{N}}(x)|dx\\ &=\int_{-\infty}^{\infty}|(f_{\widetilde{\mathbb{B}}}(x)-f_{\widetilde{\mathbb{N}}}(x))+(f_{\widetilde{\mathbb{N}}}(x)-f_{\mathbb{N}}(x))|dx\\ &\leq\int_{-\infty}^{\infty}|f_{\widetilde{\mathbb{N}}}(x)-f_{\mathbb{N}}(x)|dx+\int_{-\infty}^{\infty}|f_{\widetilde{\mathbb{B}}}(x)-f_{\widetilde{\mathbb{N}}}(x)|dx\\ &=\int_{-\infty}^{\infty}|f_{\widetilde{\mathbb{N}}}(x)-f_{\mathbb{N}}(x)|dx+\int_{\lfloor x\rfloor\in S}|f_{\widetilde{\mathbb{B}}}(x)-f_{\widetilde{\mathbb{N}}}(x)|dx\\ &\quad+\int_{\lfloor x\rfloor\notin S}|f_{\widetilde{\mathbb{B}}}(x)-f_{\widetilde{\mathbb{N}}}(x)|dx\end{aligned}$$

We will prove the following three inequalities, which will complete the proof:

$$\int_{-\infty}^{\infty}|f_{\widetilde{\mathbb{N}}}(x)-f_{\mathbb{N}}(x)|dx<\frac{1.13}{\sqrt{\mu}}\tag{7}$$

$$\int_{\lfloor x\rfloor\in S}|f_{\widetilde{\mathbb{B}}}(x)-f_{\widetilde{\mathbb{N}}}(x)|dx<\frac{1}{\sqrt{\mu}}\tag{8}$$

$$\int_{\lfloor x\rfloor\notin S}|f_{\widetilde{\mathbb{B}}}(x)-f_{\widetilde{\mathbb{N}}}(x)|dx<\frac{2.13}{\sqrt{\mu}}\tag{9}$$

– Proof for Inequality 7.

$$\begin{aligned}&\int_{-\infty}^{\infty}|f_{\widetilde{\mathbb{N}}}(x)-f_{\mathbb{N}}(x)|dx\\ &=\int_{-\infty}^{\infty}|f_{\mathbb{N}}(\lfloor x\rfloor)-f_{\mathbb{N}}(x)|dx\\ &\leq\sum_{i=-\infty}^{\infty}\left(\max_{x\in[i,i+1)}f_{\mathbb{N}}(x)-\min_{x\in[i,i+1)}f_{\mathbb{N}}(x)\right)\\ &\leq 2\times\left(\max_{x\in(-\infty,\infty)}f_{\mathbb{N}}(x)-\min_{x\in(-\infty,\infty)}f_{\mathbb{N}}(x)\right)\\ &\quad(\text{since } f_{\mathbb{N}}(x)\text{ is first increasing and then decreasing})\\ &<\frac{2}{\sqrt{\pi\mu}}<\frac{1.13}{\sqrt{\mu}}\end{aligned}$$

– Proof for Inequality 8. We first show that $f_{\widetilde{\mathbb{B}}}(\lfloor x\rfloor)$ is very close to $f_{\widetilde{\mathbb{N}}}(\lfloor x\rfloor)$. Let $\delta=\lfloor x\rfloor-\mu$. For $\lfloor x\rfloor\in S$, invoke the first inequality in Lemma 16 with $\mu\geq 9$ (implying that $\lfloor x\rfloor\in S\Rightarrow 1\leq\lfloor x\rfloor\leq 2\mu-1$), and we have:

$$\begin{aligned}&f_{\widetilde{\mathbb{B}}}(\lfloor x\rfloor)\\ &<f_{\mathbb{N}}(\lfloor x\rfloor)\cdot\exp\left(-\frac{\delta^4}{6\mu^3}+\frac{\delta^2}{2(\mu^2-\delta^2)}\right)\\ &=f_{\widetilde{\mathbb{N}}}(\lfloor x\rfloor)\cdot\exp\left(-\frac{\delta^4}{6\mu^3}+\frac{\delta^2}{2(\mu^2-\delta^2)}\right)\\ &\leq f_{\widetilde{\mathbb{N}}}(\lfloor x\rfloor)\cdot\exp\left(\frac{\delta^2}{2(\mu^2-\delta^2)}\right)\\ &\leq f_{\widetilde{\mathbb{N}}}(\lfloor x\rfloor)\cdot\exp\left(\frac{4\mu}{2(\mu^2-4\mu)}\right)\\ &\quad(\text{since }|\delta|\leq 2\sqrt{\mu}\text{ for }\lfloor x\rfloor\in S,u>4,\text{ and }\mu^2>4\mu)\\ &=f_{\widetilde{\mathbb{N}}}(\lfloor x\rfloor)\cdot\exp\left(\frac{2}{\mu-4}\right)\\ &\leq f_{\widetilde{\mathbb{N}}}(\lfloor x\rfloor)\left(1+\frac{5}{\mu}\right)\quad(\text{since } u\geq 10)\end{aligned}$$

Since $\mu\geq 16$, together with the second inequality in Lemma 16, this means that for $\lfloor x\rfloor\in S$:

$$\begin{aligned}&f_{\widetilde{\mathbb{N}}}(\lfloor x\rfloor)\left(1-\frac{6}{\mu}\right)<f_{\widetilde{\mathbb{B}}}(\lfloor x\rfloor)<f_{\widetilde{\mathbb{N}}}(\lfloor x\rfloor)\left(1+\frac{5}{\mu}\right)\\ &\Rightarrow|f_{\widetilde{\mathbb{B}}}(\lfloor x\rfloor)-f_{\widetilde{\mathbb{N}}}(\lfloor x\rfloor)|<\frac{6}{\mu}f_{\widetilde{\mathbb{N}}}(\lfloor x\rfloor)\\ &<\frac{6}{\mu-6}f_{\widetilde{\mathbb{B}}}(\lfloor x\rfloor)\end{aligned}$$

This enables us to prove Inequality 8:

$$\begin{aligned}
&\int_{\lfloor x \rfloor \in S} |f_{\widetilde{\mathbb{B}}}(x) - f_{\widetilde{\mathbb{N}}}(x)| dx \\
&< \int_{\lfloor x \rfloor \in S} \frac{6}{\mu - 6} f_{\widetilde{\mathbb{B}}}(\lfloor x \rfloor) dx \\
&< \frac{6}{\mu - 6} < \frac{1}{\sqrt{\mu}} \quad \text{(since } \mu \geq 50\text{)}
\end{aligned}$$

– Proof for Inequality 9. We will later prove that $f_{\widetilde{\mathbb{B}}}(i) < f_{\mathbb{N}}(i)$ for all $i \notin S$. If such a claim does hold, then we have:

$$\begin{aligned}
&\int_{\lfloor x \rfloor \notin S} |f_{\widetilde{\mathbb{B}}}(x) - f_{\widetilde{\mathbb{N}}}(x)| dx \\
&= \sum_{i \notin S} |f_{\widetilde{\mathbb{B}}}(i) - f_{\widetilde{\mathbb{N}}}(i)| = \sum_{i \notin S} |f_{\widetilde{\mathbb{B}}}(i) - f_{\mathbb{N}}(i)| \\
&= \sum_{i \notin S} (f_{\mathbb{N}}(i) - f_{\widetilde{\mathbb{B}}}(i)) \\
&= \left(\sum_{i=-\infty}^{\infty} f_{\mathbb{N}}(i) - \sum_{i \in S} f_{\mathbb{N}}(i) \right) \\
&- \left(\sum_{i=-\infty}^{\infty} f_{\widetilde{\mathbb{B}}}(i) - \sum_{i \in S} f_{\widetilde{\mathbb{B}}}(i) \right) \\
&= \sum_{i \in S} f_{\widetilde{\mathbb{B}}}(i) - \sum_{i \in S} f_{\mathbb{N}}(i) \\
&= \left(\sum_{i \in S} f_{\widetilde{\mathbb{B}}}(i) - \sum_{i \in S} f_{\widetilde{\mathbb{N}}}(i) \right) + \left(\sum_{i \in S} f_{\widetilde{\mathbb{N}}}(i) - \sum_{i \in S} f_{\mathbb{N}}(i) \right) \\
&\leq \int_{\lfloor x \rfloor \in S} |f_{\widetilde{\mathbb{B}}}(x) - f_{\widetilde{\mathbb{N}}}(x)| dx \\
&+ \int_{\lfloor x \rfloor \in S} |f_{\widetilde{\mathbb{N}}}(x) - f_{\mathbb{N}}(x)| dx \\
&< \frac{1}{\sqrt{\mu}} + \frac{1.13}{\sqrt{\mu}} \\
&= \frac{2.13}{\sqrt{\mu}} \quad \text{(by Inequality 7 and Inequality 8)}
\end{aligned}$$

The only thing left now is to show that $f_{\widetilde{\mathbb{B}}}(i) < f_{\mathbb{N}}(i)$ for all $i \notin S$. If $i \notin S$ and $i < 0$ (or $i \notin S$ and $i > 2\mu$), then $f_{\widetilde{\mathbb{B}}}(i) = 0 < f_{\mathbb{N}}(i)$. If $i \notin S$ and $i = 0$ (or $i \notin S$ and $i = 2\mu$), then:

$$\begin{aligned}
f_{\widetilde{\mathbb{B}}}(i) &= \frac{1}{2^{2\mu}} = \frac{1}{4^{\mu}} < \frac{1}{\sqrt{\pi \mu}} \exp(-\mu) \\
&= f_{\mathbb{N}}(i) \quad \text{(since } \mu \geq 50\text{)}
\end{aligned}$$

If $i \notin S$ and $i = 1$ (or $i \notin S$ and $i = 2\mu - 1$), then:

$$\begin{aligned}
f_{\widetilde{\mathbb{B}}}(i) &= \frac{1}{2^{2\mu}} 2\mu = \frac{2\mu}{4^{\mu}} < \frac{1}{\sqrt{\pi \mu}} \exp\left(-\frac{(\mu - 1)^2}{\mu} \right) \\
&= f_{\mathbb{N}}(i) \quad \text{(since } \mu \geq 50\text{)}
\end{aligned}$$

Finally, if $i \notin S$ and $2 \leq i \leq 2\mu - 2$, then let $\delta = i - \mu$ and leverage the first inequality in Lemma 16:

$$\begin{aligned}
f_{\widetilde{\mathbb{B}}}(i) &< f_{\mathbb{N}}(i) \cdot \exp\left(-\frac{\delta^4}{6\mu^3} + \frac{\delta^2}{2(\mu^2 - \delta^2)} \right) \\
&= f_{\mathbb{N}}(i) \cdot \exp\left(\frac{\delta^2}{6\mu^3(\mu^2 - \delta^2)} \left(3\mu^3 - \delta^2(\mu^2 - \delta^2) \right) \right)
\end{aligned}$$

We will prove that $\delta^2(\mu^2 - \delta^2) > 3\mu^3$ when $\mu \geq 50$. Since $i \notin S$ and $2 \leq i \leq 2\mu - 2$, we know that $4\mu \leq \delta^2 \leq (\mu - 2)^2$. Define $f(\delta^2) = \delta^2(\mu^2 - \delta^2)$ where $\delta^2 \in [4\mu, (\mu - 2)^2]$, and it is easy to verify that since $4\mu \leq \mu^2/2$ and $(\mu - 2)^2 \geq \mu^2/2$ (for $\mu \geq 8$), the minimum of $f(\delta^2)$ is reached at $f((\mu - 2)^2)$. Hence we have $f(\delta^2) \geq f((\mu - 2)^2) = (\mu - 2)^2(\mu^2 - (\mu - 2)^2) = 4(\mu - 2)^2(\mu - 1) > 3\mu^3$ for $\mu \geq 50$. Hence we have $f_{\widetilde{\mathbb{B}}}(i) < f_{\mathbb{N}}(i) \cdot \exp(0) = f_{\mathbb{N}}(i)$.

□

C.4 Upper bound $||\mathbb{N} - \mathbb{N}'||$

This section will prove an upper bound on $||\mathbb{N} - \mathbb{N}'||$. We do so by using some existing result to upper bound $D_{KL}(\mathbb{N}||\mathbb{N}')$, and then using another existing result to convert this upper bound to an upper bound on $||\mathbb{N} - \mathbb{N}'||$.

Define $\mathbb{N}(\mu, \Sigma)$ to be the multi-variate normal distribution whose mean vector is μ and whose covariance matrix is Σ. The following is a known result on the KL distance between two multi-variate normal distributions:

Lemma 18 [25] *Let $\mathbb{N}(\mu, \Sigma)$ and $\mathbb{N}'(\mu', \Sigma')$ be two arbitrary k-variate normal distributions. If Σ and Σ' are both non-singular matrices, then:*

$$\begin{aligned}
&D_{KL}(\mathbb{N}||\mathbb{N}') \\
&= \frac{1}{2} \left(\text{tr}\left(\Sigma'^{-1} \Sigma \right) + (\mu' - \mu)^{\top} \Sigma'^{-1} (\mu' - \mu) \right. \\
&\left. -k + \ln\left(\frac{\det \Sigma'}{\det \Sigma} \right) \right)
\end{aligned}$$

Applying this lemma to our setting yields:

Lemma 19 *For any given positive integers μ_1 through μ_k and μ'_1 through μ'_k, define distributions $\mathbb{N} = \mathbb{N}(\mu_1) \times \mathbb{N}(\mu_2) \times \cdots \times \mathbb{N}(\mu_k)$ and $\mathbb{N}' = \mathbb{N}(\mu'_1) \times \mathbb{N}(\mu'_2) \times \cdots \times \mathbb{N}(\mu'_k)$. We have:*

$$\begin{aligned}
&D_{KL}(\mathbb{N}||\mathbb{N}') \\
&= \frac{1}{2} \left(\sum_{i=1}^{k} \frac{\mu_i - \mu'_i}{\mu'_i} + 2 \sum_{i=1}^{k} \frac{(\mu_i - \mu'_i)^2}{\mu'_i} + \sum_{i=1}^{k} \ln \frac{\mu'_i}{\mu_i} \right)
\end{aligned}$$

Proof Obviously, $\mathbb{N}$ is a multi-variate normal distribution. Let μ be its mean vector and Σ be its covariance matrix. Similarly define μ' and Σ'. We have:

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \cdots \\ \mu_k \end{bmatrix}, \mu' = \begin{bmatrix} \mu'_1 \\ \mu'_2 \\ \cdots \\ \mu'_k \end{bmatrix}, \Sigma = \frac{1}{2}\begin{bmatrix} \mu_1 & 0 & \cdots & 0 \\ 0 & \mu_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mu_k \end{bmatrix},$$

$$\Sigma' = \frac{1}{2}\begin{bmatrix} \mu'_1 & 0 & \cdots & 0 \\ 0 & \mu'_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mu'_k \end{bmatrix}$$

Since Σ and Σ' are both diagonal matrices of size $k \times k$, we denote $\Sigma = \text{diag}(\mu_1, \mu_2, \ldots, \mu_k)$ and $\Sigma' = \text{diag}(\mu'_1, \mu'_2, \ldots, \mu'_k)$. Obviously, both Σ and Σ' are non-singular. Following are some useful properties of diagonal matrices:

$$(\text{diag}(\mu'_1, \mu'_2, \ldots, \mu'_k))^{-1} = \text{diag}({\mu'_1}^{-1}, {\mu'_2}^{-1}, \ldots, {\mu'_k}^{-1})$$

$$\text{diag}(\mu'_1, \mu'_2, \ldots, \mu'_k) \cdot \text{diag}(\mu_1, \mu_2, \ldots, \mu_k) = \text{diag}(\mu_1\mu'_1, \mu_2\mu'_2, \ldots, \mu_k\mu'_k)$$

$$(\mu'_1, \mu'_2, \ldots, \mu'_k) \cdot \text{diag}(\mu_1, \mu_2, \ldots, \mu_k) = (\mu_1\mu'_1, \mu_2\mu'_2, \ldots, \mu_k\mu'_k)$$

$$\text{tr}(\text{diag}(\mu_1, \mu_2, \ldots, \mu_k)) = \sum_{i=1}^{k} \mu_i$$

$$\det(\text{diag}(\mu_1, \mu_2, \ldots, \mu_k)) = \Pi_{i=1}^{k} \mu_i$$

The last 2 equations are directly from the definition of the trace and the determinant of a matrix. We therefore have:

$$\begin{aligned}
\text{tr}\left(\Sigma'^{-1}\Sigma\right) &= \text{tr}\left(\left(\frac{1}{2}\text{diag}(\mu'_1, \mu'_2, \ldots, \mu'_k)\right)^{-1} \cdot \frac{1}{2}\text{diag}(\mu_1, \mu_2, \ldots, \mu_k)\right) \\
&= \text{tr}(\text{diag}({\mu'_1}^{-1}, {\mu'_2}^{-1}, \ldots, {\mu'_k}^{-1}) \cdot \text{diag}(\mu_1, \mu_2, \ldots, \mu_k)) \\
&= \text{tr}(\text{diag}(\mu_1{\mu'_1}^{-1}, \mu_2{\mu'_2}^{-1}, \ldots, \mu_k{\mu'_k}^{-1})) \\
&= \sum_{i=1}^{k} \frac{\mu_i}{\mu'_i} \left(\mu' - \mu\right)^{\top} \Sigma'^{-1} (\mu' - \mu) \\
&= (\mu'_1 - \mu_1, \mu'_2 - \mu_2, \ldots, \mu'_k - \mu_k) \left(\frac{\text{diag}(\mu'_1, \mu'_2, \ldots, \mu'_k)}{2}\right)^{-1} (\mu' - \mu) \\
&= 2\left(\frac{\mu'_1 - \mu_1}{\mu'_1}, \frac{\mu'_2 - \mu_2}{\mu'_2}, \ldots, \frac{\mu'_k - \mu_k}{\mu'_k}\right) \begin{bmatrix} \mu'_1 - \mu_1 \\ \mu'_2 - \mu_2 \\ \cdots \\ \mu'_k - \mu_k \end{bmatrix} \\
&= 2\sum_{i=1}^{k} \frac{(\mu'_i - \mu_i)^2}{\mu'_i}
\end{aligned}$$

$$\det \Sigma = \det\left(\frac{1}{2}\text{diag}(\mu_1, \mu_2, \ldots, \mu_k)\right) = \Pi_{i=1}^{k} \frac{1}{2}\mu_i$$

$$\det \Sigma' = \det\left(\frac{1}{2}\text{diag}(\mu'_1, \mu'_2, \ldots, \mu'_k)\right) = \Pi_{i=1}^{k} \frac{1}{2}\mu'_i$$

Plug in all the above equations into Lemma 18, and we have:

$$\begin{aligned}
&D_{KL}(\mathbb{N}||\mathbb{N}') \\
&= \frac{1}{2}\left(\sum_{i=1}^{k} \frac{\mu_i}{\mu'_i} + 2\sum_{i=1}^{k} \frac{(\mu_i - \mu'_i)^2}{\mu'_i} - k + \sum_{i=1}^{k} \ln \frac{\mu'_i}{\mu_i}\right) \\
&= \frac{1}{2}\left(\sum_{i=1}^{k} \frac{\mu_i - \mu'_i}{\mu'_i} + 2\sum_{i=1}^{k} \frac{(\mu_i - \mu'_i)^2}{\mu'_i} + \sum_{i=1}^{k} \ln \frac{\mu'_i}{\mu_i}\right)
\end{aligned}$$

□

Next, the following lemma is the well-known *Pinsker's inequality*:

Lemma 20 [27] *For all distributions $\mathbb{D}$ and $\mathbb{D}'$,*

$$\frac{1}{2}||\mathbb{D} - \mathbb{D}'|| \leq \sqrt{\frac{1}{2} D_{KL}(\mathbb{D}||\mathbb{D}')}$$

From Lemmas 19 and 20, we trivially have:

Lemma 21 *For any given positive integers μ_1 through μ_k and μ'_1 through μ'_k, define distributions $\mathbb{N} = \mathbb{N}(\mu_1) \times \mathbb{N}(\mu_2) \times \cdots \times \mathbb{N}(\mu_k)$ and $\mathbb{N}' = \mathbb{N}(\mu'_1) \times \mathbb{N}(\mu'_2) \times \cdots \times \mathbb{N}(\mu'_k)$. We have:*

$$||\mathbb{N} - \mathbb{N}'|| \leq \sqrt{\sum_{i=1}^{k} \frac{\mu_i - \mu'_i}{\mu'_i} + 2\sum_{i=1}^{k} \frac{(\mu_i - \mu'_i)^2}{\mu'_i} + \sum_{i=1}^{k} \ln \frac{\mu'_i}{\mu_i} x}$$

C.5 Putting everything together

In this section, we first show a simple connection between the L_1 distance between two product distributions and the L_1 distances between the respective component distributions in the two product distributions. Next we will put everything together to prove Theorem 8.

Lemma 22 *Consider any positive integer k, and any two product distributions $\mathbb{D} = \mathbb{D}_1 \times \mathbb{D}_2 \times \cdots \times \mathbb{D}_k$ and $\mathbb{D}' = \mathbb{D}'_1 \times \mathbb{D}'_2 \times \cdots \times \mathbb{D}'_k$, where $\mathbb{D}_1$ through $\mathbb{D}_k$ and $\mathbb{D}'_1$ through $\mathbb{D}'_k$ are arbitrary continuous distributions. We have $||\mathbb{D} - \mathbb{D}'|| \leq \sum_{i=1}^{k} ||\mathbb{D}_i - \mathbb{D}'_i||$.*

Proof First, for all real numbers $a, b, c, d \in [0, 1]$, we always have:

$$|ab - cd| = |(a - c)b + (b - d)c| \le |a - c||b| + |b - d||c| \le |a - c| + |b - d|$$

Let $f_{\mathbb{D}}$ and $f_{\mathbb{D}'}$ be the probability density function for $\mathbb{D}$ and $\mathbb{D}'$, respectively. Similarly define $f_{\mathbb{D}_1}$ through $f_{\mathbb{D}_k}$ and $f_{\mathbb{D}'_1}$ through $f_{\mathbb{D}'_k}$.

For all vector $(x_1, x_2, \ldots x_k)$ in the domain of $\mathbb{D}$ and $\mathbb{D}'$, we have:

$$\begin{aligned}
&|f_{\mathbb{D}}(x_1, x_2, \ldots, x_k) - f'_{\mathbb{D}}(x_1, x_2, \ldots, x_k)| \\
&= |f_{\mathbb{D}_1}(x_1) f_{\mathbb{D}_2}(x_2) \cdots f_{\mathbb{D}_k}(x_k) \\
&\quad - f_{\mathbb{D}'_1}(x_1) f_{\mathbb{D}'_2}(x_2) \cdots f_{\mathbb{D}'_k}(x_k)| \\
&\le |f_{\mathbb{D}_1}(x_1) - f_{\mathbb{D}'_1}(x_1)| \\
&\quad + |f_{\mathbb{D}_2}(x_2) f_{\mathbb{D}_3}(x_3) \cdots f_{\mathbb{D}_k}(x_k) \\
&\quad - f_{\mathbb{D}'_2}(x_2) f_{\mathbb{D}'_3}(x_3) \cdots f_{\mathbb{D}'_k}(x_k)| \\
&\le |f_{\mathbb{D}_1}(x_1) - f_{\mathbb{D}'_1}(x_1)| + |f_{\mathbb{D}_2}(x_2) - f_{\mathbb{D}'_2}(x_2)| \\
&\quad + |f_{\mathbb{D}_3}(x_3) f_{\mathbb{D}_4}(x_4) \cdots f_{\mathbb{D}_k}(x_k) \\
&\quad - f_{\mathbb{D}'_3}(x_3) f_{\mathbb{D}'_4}(x_4) \cdots f_{\mathbb{D}'_k}(x_k)| \\
&\le \sum_{i=1}^{k} |f_{\mathbb{D}_i}(x_i) - f_{\mathbb{D}'_i}(x_i)|
\end{aligned}$$

Thus we have:

$$\begin{aligned}
&||\mathbb{D} - \mathbb{D}'|| \\
&= \int_{x_1, x_2, \ldots, x_k} |f_{\mathbb{D}}(x_1, x_2, \ldots, x_k) \\
&\quad - f'_{\mathbb{D}}(x_1, x_2, \ldots, x_k)| d(x_1, x_2, \ldots, x_k) \\
&\le \sum_{i=1}^{k} \int_{x_i} |f_{\mathbb{D}_i}(x_i) - f_{\mathbb{D}'_i}(x_i)| dx_i = \sum_{i=1}^{k} ||\mathbb{D}_i - \mathbb{D}'_i||
\end{aligned}$$

□

Theorem 8 *Consider any positive integer k, any μ_i and μ'_i where $2\mu_i$ and $2\mu'_i$ are all integers ($1 \le i \le k$). Let $\mu = \min_{1 \le i \le k}(\min(\mu_i, \mu'_i))$. Let δ and δ' be any given constants where $0 < \delta' < \delta < 0.5$. Let product distribution $\mathbb{D} = \mathbb{B}(\mu_1) \times \mathbb{B}(\mu_2) \times \ldots \times \mathbb{B}(\mu_k)$ and $\mathbb{D}' = \mathbb{B}(\mu'_1) \times \mathbb{B}(\mu'_2) \times \ldots \times \mathbb{B}(\mu'_k)$. If $\mu \ge \frac{250}{(\delta - \delta')^2}(k^2 + k \max_{1 \le i \le k}(\mu_i - \mu'_i)^2)$, then $||\mathbb{D} - \mathbb{D}'|| \le \frac{2(\delta - \delta')}{3}$.*

Proof Define $\widetilde{\mathbb{D}} = \widetilde{\mathbb{B}}(\mu_1) \times \widetilde{\mathbb{B}}(\mu_2) \times \cdots \times \widetilde{\mathbb{B}}(\mu_k)$ and $\widetilde{\mathbb{D}}' = \widetilde{\mathbb{B}}(\mu'_1) \times \widetilde{\mathbb{B}}(\mu'_2) \times \cdots \times \widetilde{\mathbb{B}}(\mu'_k)$. Note that for all vector $(x_1, x_2, \ldots x_k)$ where x_1 through x_k are integers, the probability density under $\mathbb{D}$ is exactly the same as the probability density under $\widetilde{\mathbb{D}}$. The same property holds for $\mathbb{D}'$ and $\widetilde{\mathbb{D}}'$. Hence $||\mathbb{D} - \mathbb{D}'|| = ||\widetilde{\mathbb{D}} - \widetilde{\mathbb{D}}'||$. Next, define product distribution $\mathbb{N} = \mathbb{N}(\mu_1) \times \mathbb{N}(\mu_2) \times \cdots \times \mathbb{N}(\mu_k)$ and $\mathbb{N}' = \mathbb{N}(\mu'_1) \times \mathbb{N}(\mu'_2) \times \cdots \times \mathbb{N}(\mu'_k)$. We have:

$$||\mathbb{D} - \mathbb{D}'|| = ||\widetilde{\mathbb{D}} - \widetilde{\mathbb{D}}'|| \le ||\widetilde{\mathbb{D}} - \mathbb{N}|| + ||\widetilde{\mathbb{D}}' - \mathbb{N}'|| + ||\mathbb{N} - \mathbb{N}'||$$

In the next, we upper bound each of the three terms.

Since $\mu \ge \frac{250}{(\delta - \delta')^2}(k^2 + k \max_{1 \le i \le k}(\mu_i - \mu'_i)^2) > 50$, by Lemmas 17 and 22, we have:

$$\begin{aligned}
||\widetilde{\mathbb{D}} - \mathbb{N}|| &\le \sum_{i=1}^{k} ||\widetilde{\mathbb{B}}(\mu_i) - \mathbb{N}(\mu_i)|| < \sum_{i=1}^{k} \frac{4.3}{\sqrt{\mu_i}} \le \sum_{i=1}^{k} \frac{4.3}{\sqrt{\mu}} \\
&\le 4.3k / \sqrt{\frac{250}{(\delta - \delta')^2}\left(k^2 + k \max_{1 \le i \le k}(\mu_i - \mu'_i)^2\right)} \\
&< \frac{4.3}{\sqrt{250}}(\delta - \delta')
\end{aligned}$$

Similarly we have $||\widetilde{\mathbb{D}}' - \mathbb{N}'|| < \frac{4.3}{\sqrt{250}}(\delta - \delta')$.

Let $a = \max_{1 \le i \le k} |\mu_i - \mu'_i|$. By Lemma 21, we have:

$$\begin{aligned}
&||\mathbb{N} - \mathbb{N}'|| \\
&\le \sqrt{\sum_{i=1}^{k} \frac{\mu_i - \mu'_i}{\mu'_i} + 2\sum_{i=1}^{k} \frac{(\mu_i - \mu'_i)^2}{\mu'_i} + \sum_{i=1}^{k} \ln \frac{\mu'_i}{\mu_i}} \\
&\le \sqrt{\sum_{i=1}^{k} \frac{a}{\mu'_i} + 2\sum_{i=1}^{k} \frac{a^2}{\mu'_i} + \sum_{i=1}^{k} \ln\left(1 + \frac{a}{\mu_i}\right)} \\
&\le \sqrt{\sum_{i=1}^{k} \frac{a}{\mu} + 2\sum_{i=1}^{k} \frac{a^2}{\mu} + \sum_{i=1}^{k} \frac{a}{\mu}} = \sqrt{\frac{k}{\mu}(2a^2 + 2a)} \\
&\le \sqrt{\frac{k}{\frac{250}{(\delta - \delta')^2}(k^2 + ka^2)}(2a^2 + 2a)} \\
&= \frac{\delta - \delta'}{\sqrt{250}} \sqrt{\frac{2a^2 + 2a}{a^2 + k}} \le \frac{\delta - \delta'}{\sqrt{250}} \sqrt{\frac{2a^2 + 2a}{a^2 + 1}}
\end{aligned}$$

It is easy to verify that $\frac{2a^2 + 2a}{a^2 + 1}$ is always smaller that 2.5 for $a \in [0, \infty)$. Hence $||\mathbb{N} - \mathbb{N}'|| < \frac{\sqrt{2.5}}{\sqrt{250}}(\delta - \delta')$.

Finally, put everything together:

$$\begin{aligned}
||\mathbb{D} - \mathbb{D}'|| = ||\widetilde{\mathbb{D}} - \widetilde{\mathbb{D}}'|| &\le ||\widetilde{\mathbb{D}} - \mathbb{N}|| + ||\widetilde{\mathbb{D}}' - \mathbb{N}'|| + ||\mathbb{N} - \mathbb{N}'|| \\
&< \frac{4.3}{\sqrt{250}}(\delta - \delta') + \frac{4.3}{\sqrt{250}}(\delta - \delta') \\
&\quad + \frac{\sqrt{2.5}}{\sqrt{250}}(\delta - \delta') < \frac{2(\delta - \delta')}{3}
\end{aligned}$$

□

Appendix D: Proof for Lemma 7

Lemma 7 *Consider any input* (X, Y) *of the* $\textsc{Gdc}_n^{g,q}$ *problem and its corresponding processed input* $(\mathbf{X}', \mathbf{Y}')$. *For* $z \in \{0, 1\}$, *if* $q \geq 20$, $g \geq 15q \ln q$, $n \geq 4g$, *and* $\textsc{Gdc}(X, Y) = z$, *then* $\Pr[(\mathbf{X}', \mathbf{Y}') \text{is of type-} z] > 1 - \frac{1}{q}$.

Proof To facilitate discussion, we first define a few concepts. We say that:

- $\text{left}(\mathbf{X}', \mathbf{Y}')$ is of *half-type*-0 if $|_0^0(\text{left}(\mathbf{X}', \mathbf{Y}')) \geq q$ and $|_{2j}^{2j}(\text{left}(\mathbf{X}', \mathbf{Y}')) \geq 1$ for all j from 1 through $\frac{q-1}{2}$.
- $\text{right}(\mathbf{X}', \mathbf{Y}')$ is of *half-type*-0 if $|_0^0(\text{right}(\mathbf{X}', \mathbf{Y}')) \geq q$ and $|_{2j}^{2j}(\text{right}(\mathbf{X}', \mathbf{Y}')) \geq 1$ for all j from 1 through $\frac{q-1}{2}$.
- $\text{left}(\mathbf{X}', \mathbf{Y}')$ is of *half-type*-1 if $|_{q-1}^{q-1}(\text{left}(\mathbf{X}', \mathbf{Y}')) \geq 1$ and $|_{2j}^{2j}(\text{left}(\mathbf{X}', \mathbf{Y}')) = 0$ for all j from 1 through $\frac{q-3}{2}$.
- $\text{right}(\mathbf{X}', \mathbf{Y}')$ is of *half-type*-1 if $|_{q-1}^{q-1}(\text{right}(\mathbf{X}', \mathbf{Y}')) \geq 1$ and $|_{2j}^{2j}(\text{right}(\mathbf{X}', \mathbf{Y}')) = 0$ for all j from 1 through $\frac{q-3}{2}$.

For $z \in \{0, 1\}$, note that $(\mathbf{X}', \mathbf{Y}')$ is of type-z iff $\text{left}(\mathbf{X}', \mathbf{Y}')$ and $\text{right}(\mathbf{X}', \mathbf{Y}')$ both are of half-type-z.

To prove Lemma 7, we separately consider two cases:

- $\textsc{Gdc}(X, Y) = 1$. By definition, $|_0^0(X, Y) = 0$. In turn, $|_0^0(\mathbf{X}', \mathbf{Y}') = |_2^2(\mathbf{X}', \mathbf{Y}') = \ldots = |_{q-3}^{q-3}(\mathbf{X}', \mathbf{Y}') = 0$. Next, since for each index $\Pr[\mathbf{o} = q-1] = \frac{1}{q-1}$, the probability that there does not exist any index from 1 to $\frac{n}{2}$ such that $\mathbf{o} = q - 1$ is:

$$\begin{aligned}\left(1 - \frac{1}{q-1}\right)^{\frac{n}{2}} &< \exp\left(-\frac{1}{q-1} \cdot \frac{n}{2}\right) < \exp\left(-\frac{n}{2q}\right) \\ &\leq \exp\left(-\frac{4g}{2q}\right) \leq \exp(-30 \ln q) \\ &= \frac{1}{q^{30}} < \frac{1}{2q}\end{aligned}$$

  As long as there exists some $\mathbf{o} = q - 1$, $\text{left}(\mathbf{X}', \mathbf{Y}')$ will be of half-type-1. The probability of $\text{right}(\mathbf{X}', \mathbf{Y}')$ being half-type-1 is the same. A simple union bound then shows that with probability at least $1 - \frac{1}{q}$, both $\text{left}(\mathbf{X}', \mathbf{Y}')$ and $\text{right}(\mathbf{X}', \mathbf{Y}')$ are half-type-1.
- $\textsc{Gdc}(X, Y) = 0$. By definition, $|_0^0(X, Y) \geq g$. We first show that immediately after the permutation and before adding the offsets in the preprocessing step, with probability at least $1 - \frac{1}{q^2}$, $|_0^0(\text{left}(\mathbf{X}', \mathbf{Y}')) \geq 4q \ln q$. To see why, we view the permutation as obtained by assigning each index between 1 and n in (X, Y), one by one, to a new position between 1 and n after the permutation. Each position can only accommodate one index. Hence when we assign an index, we will choose a uniformly random position among all remaining unoccupied positions. Furthermore, we can imagine that we assign those indices corresponding to the $|_0^0$ patterns in (X, Y) first, before assigning other indices. There are at least g indices corresponding to the $|_0^0$ pattern, and let us consider the first g of them. For each such index, regardless what happened prior to our assigning this index, the probability of this index being assigned to the first half of the positions (i.e., to some position between 1 and $\frac{n}{2}$) must be no smaller than $\frac{1}{3}$. The reason is that there will always be at least $\frac{n}{2} - g \geq \frac{n}{4}$ unoccupied positions in the first half of the positions, and at most $\frac{n}{2}$ unoccupied positions in the second half. Consider the sum $\mathbf{z}$ of $15q \ln q$ independent Bernoulli random variables each taking a value of 1 with probability $\frac{1}{3}$. We have, via a simple coupling argument and Chernoff bound:

$$\begin{aligned}&\Pr[|_0^0(\text{left}(\mathbf{X}', \mathbf{Y}')) < 4q \ln q] \\ &\quad \leq \Pr\left[\mathbf{z} < \left(1 - \frac{1}{5}\right) 5q \ln q\right] \\ &\quad \leq \exp\left(-\frac{1}{2} \times \frac{1}{25} \times 5q \ln q\right) \\ &\quad \leq \frac{1}{q^2}\end{aligned}$$

  Next, conditioned upon the event that $|_0^0(\text{left}(\mathbf{X}', \mathbf{Y}')) \geq 4q \ln q$ before adding the offsets in the preprocessing step, we will show that after adding the offsets in the preprocessing step, $\Pr[\text{left}(\mathbf{X}', \mathbf{Y}') \text{ is half-type-0}] > 1 - \frac{1}{q^3} - \frac{1}{q^5}$. Consider any given j where $1 \leq j \leq \frac{q-1}{2}$. For each index corresponding to the $|_0^0$ pattern in $\text{left}(\mathbf{X}', \mathbf{Y}')$ immediately after the permutation, Alice and Bob will choose an offset $\mathbf{o}$ where $\Pr[\mathbf{o} = 2j] = \frac{1}{q-1}$. Hence the probability that $\mathbf{o} = 2j$ for none of the $4q \ln q$ indices is:

$$\begin{aligned}\left(1 - \frac{1}{q-1}\right)^{4q \ln q} &< \exp\left(-\frac{4q \ln q}{q-1}\right) \\ &< \exp(-4 \ln q) = \frac{1}{q^4}\end{aligned}$$

  There are total $\frac{q-1}{2}$ such j's. Hence by a union bound, with probability at least $1 - \frac{1}{q^3}$, after adding the offsets, $|_{2j}^{2j}(\text{left}(\mathbf{X}', \mathbf{Y}')) \geq 1$ for all j. Next, after adding the offsets, by a Chernoff bound, we also have:

$$\begin{aligned}Pr[|_0^0(\text{left}(\mathbf{X}', \mathbf{Y}')) < q] &< \Pr[|_0^0(\text{left}(\mathbf{X}', \mathbf{Y}')) \\ &< \left(1 - \frac{1}{2}\right) 2q \ln q] \\ &\leq \exp\left(-\frac{1}{2} \times \frac{1}{4} \times 2q \ln q\right) \\ &\leq \frac{1}{q^5}\end{aligned}$$

Taking a union bound thus shows that conditioned upon the event that $|_0^0(\text{left}(\mathbf{X}', \mathbf{Y}')) \geq 4q \ln q$ before adding the offsets, after adding the offsets, $\Pr[\text{left}(\mathbf{X}', \mathbf{Y}')$ is half-type-0$] > 1 - \frac{1}{q^3} - \frac{1}{q^5}$.

Putting everything together, $\Pr[\text{left}(\mathbf{X}', \mathbf{Y}')$ is half-type-0$] > \left(1 - \frac{1}{q^2}\right)\left(1 - \frac{1}{q^3} - \frac{1}{q^5}\right) > 1 - \frac{3}{q^2}$. By the same argument, we similarly have $\Pr[\text{right}(\mathbf{X}', \mathbf{Y}')$ is half-type-0$] > 1 - \frac{3}{q^2}$. By union bound we then have $\Pr[(\mathbf{X}', \mathbf{Y}')$ is type-0$] > 1 - \frac{6}{q^2} > 1 - \frac{1}{q}$. □

References

1. Augustine, J., Avin, C., Liaee, M., Pandurangan, G., Rajaraman, R.: Information spreading in dynamic networks under oblivious adversaries. In: DISC (2016)
2. Augustine, J., Kulkarni, T., Nakhe, P., Robinson, P.: Robust leader election in a fast-changing world. In: Workshop on Foundations of Mobile Computing (2013)
3. Augustine, J., Pandurangan, G., Robinson, P.: Fast byzantine agreement in dynamic networks. In: PODC (2013)
4. Augustine, J., Pandurangan, G., Robinson, P.: Fast byzantine leader election in dynamic networks. In: DISC (2015)
5. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D.: An information statistics approach to data stream and communication complexity. J. Comput. Syst. Sci. **68**(4), 702–732 (2004)
6. Batir, N.: Sharp inequalities for factorial n. Proyecciones **27**(1), 97–102 (2008)
7. Bramas, Q., Masuzawa, T., Tixeuil, S.: Distributed online data aggregation in dynamic graphs. In: ICDCS (2016)
8. Braverman, M.: Interactive information complexity. SIAM J. Comput. **44**(6), 1698–1739 (2015)
9. Censor-Hillel, K., Haramaty, E., Karnin, Z.: Optimal dynamic distributed MIS. In: PODC (2016)
10. Chen, B., Yu, H., Zhao, Y., Gibbons, P.B.: The cost of fault tolerance in multi-party communication complexity. JACM **61**(3), 19:1–19:64 (2014)
11. Cornejo, A., Gilbert, S., Newport, C.: Aggregation in dynamic networks. In: PODC (2012)
12. Coulouma, E., Godard, E., Peters, J.: A characterization of oblivious message adversaries for which consensus is solvable. Theor. Comput. Sci. **584**, 80–90 (2015)
13. Das Sarma, A., Holzer, S., Kor, L., Korman, A., Nanongkai, D., Pandurangan, G., Peleg, D., Wattenhofer, R.: Distributed verification and hardness of distributed approximation. In: STOC (2011)
14. Dutta, C., Pandurangan, G., Rajaraman, R., Sun, Z., Viola, E.: On the complexity of information spreading in dynamic networks. In: SODA (2013)
15. Ghaffari, M., Lynch, N., Newport, C.: The cost of radio network broadcast for different models of unreliable links. In: PODC (2013)
16. Hoeffding, W.: Probability inequalities for sums of bounded random variables. J. Am. Stat. Assoc. **58**(301), 13–30 (1963)
17. Ingram, R., Shields, P., Walter, J.: An asynchronous leader election algorithm for dynamic networks. In: IPDPS (2009)
18. Jahja, I., Yu, H., Zhao, Y.: Some lower bounds in dynamic networks with oblivious adversaries. In: DISC (2017)
19. Konig, M., Wattenhofer, R.: On local fixing. In: OPODIS (2013)
20. Kuhn, F., Lynch, N., Newport, C., Oshman, R., Richa, A.: Broadcasting in unreliable radio networks. In: PODC (2010)
21. Kuhn, F., Lynch, N., Oshman, R.: Distributed computation in dynamic networks. In: STOC (2010)
22. Kuhn, F., Moses, Y., Oshman, R.: Coordinated consensus in dynamic networks. In: PODC (2011)
23. Kuhn, F., Oshman, R.: The complexity of data aggregation in directed networks. In: DISC (2011)
24. Kuhn, F., Oshman, R.: Dynamic networks: models and algorithms. SIGACT News **42**(1), 82–96 (2011)
25. Liese, F., Vajda, I.: Convex Statistical Distances. Teubner, Leipzig (1987)
26. Schmid, U., Weiss, B., Keidar, I.: Impossibility results and lower bounds for consensus under link failures. SIAM J. Comput. **38**(5), 1912–1951 (2009)
27. Tsybakov, A.: Introduction to Nonparametric Estimation. Springer, New York (2009)
28. Weinstein, O.: Information complexity and the quest for interactive compression. SIGACT News **46**(2), 41–64 (2015)
29. Yu, H., Zhao, Y., Jahja, I.: The cost of unknown diameter in dynamic networks. In: SPAA (2016)