Electronic Colloquium on Computational Complexity, Report No. 27 (2021)

Almost Optimal Super-Constant-Pass Streaming Lower Bounds for Reachability

Lijie Chen[*] Gillat Kol[†] Dmitry Paramonov[‡]
Raghuvansh R. Saxena[§] Zhao Song[¶] Huacheng Yu[||]

Abstract

We give an *almost quadratic* $n^{2-o(1)}$ lower bound on the space consumption of any $o(\sqrt{\log n})$*-pass* streaming algorithm solving the (directed) *s-t reachability* problem. This means that any such algorithm must essentially store the entire graph. As corollaries, we obtain almost quadratic space lower bounds for additional fundamental problems, including maximum matching, shortest path, matrix rank, and linear programming.

Our main technical contribution is the definition and construction of *set hiding graphs*, that may be of independent interest: we give a general way of encoding a set $S \subseteq [k]$ as a directed graph with $n = k^{1+o(1)}$ vertices, such that deciding whether $i \in S$ boils down to deciding if t_i is reachable from s_i, for a specific pair of vertices (s_i, t_i) in the graph. Furthermore, we prove that our graph "hides" S, in the sense that no low-space streaming algorithm with a small number of passes can learn (almost) anything about S.

[*]MIT. lijieche@mit.edu
[†]Princeton University. gillat.kol@gmail.com
[‡]Princeton University. dp20@princeton.edu
[§]Princeton University. rrsaxena@princeton.edu
[¶]Institute for Advanced Study. zhaos@ias.edu
[||]Princeton University. yuhch123@gmail.com

ISSN 1433-8092

Contents

1 Introduction

Graph streaming algorithms are designed to process massive graphs and have been studied extensively over the last two decades. This study is timely as huge graphs naturally arise in many modern applications, particularly in those with structured data representing the relationships between a set of entities (*e.g.*, friendships in a social network). A graph streaming algorithm is typically presented with a sequence of graph edges in an arbitrary order and it can read them one-by-one in the order in which they appear in the sequence. We want the algorithm to only make one or few passes through the edge sequence and use limited memory, ideally much smaller than the size of the graph.

Much of the streaming literature was devoted to the study of *one-pass* algorithms, and for many basic graph problems $\Omega(n^2)$ lower bounds were shown, where n is the number of vertices. This implies that the trivial algorithm that stores the entire graph and then uses an offline algorithm to compute the output is essentially optimal. Such quadratic lower bounds were shown for maximum matching and minimum vertex cover [FKM$^+$04, GKK12], s-t reachability and topological sorting [CGMV20, FKM$^+$04, HRR98], shortest path and diameter [FKM$^+$04, FKM$^+$09], minimum or maximum cut [Zel11], maximal independent set [ACK19b, CDK19], dominating set [AKL16, ER14], and many others.

Recently, the *multi-pass* streaming setting received quite a bit of attention. For some graph problems, it was shown that going from a single pass to even a few passes can reduce the memory consumption of a streaming algorithm dramatically. For example, *semi-streaming* algorithms (which are algorithms that only use $\widetilde{O}(n)$ space and are often considered "tractable") with few passes were designed for various graph problems previously shown to admit quadratic lower bounds for single pass streaming. These include a two-pass algorithm for minimum cut in undirected graphs [RSW18], an $O(1)$-pass algorithm for approximate matching [GKMS19, GKK12, Kap13, McG05], an $O(\log\log n)$-pass algorithm for maximal independent set [ACK19b, CDK19, GGK$^+$18], and $O(\log n)$-pass algorithms for approximate dominating set [AKL16, CW16, HPIMV16] and weighted minimum cut [MN20].

1.1 Our Results

1.1.1 Lower Bound for s-t Reachability

Our main result is a *near-quadratic* lower bound on the space complexity of any streaming algorithm that solves *s-t reachability* (a.k.a, *directed connectivity*) and uses $o(\sqrt{\log n})$ passes:

Theorem 1.1 (Reachability). *Any randomized $o(\sqrt{\log n})$-pass streaming algorithm that, given an n-vertex directed graph $G = (V, E)$ with two designated vertices $s, t \in V$, can determine whether there is a directed path from s to t in G requires $n^{2-o(1)}$ space.*

The s-t reachability problem is amongst the most basic graph problems and was also one of the first to be studied in the context of streaming [HRR98]. Prior to our work, an almost-quadratic lower bound was only known for *two*-pass streaming algorithms by the very recent breakthrough of [AR20]. Prior to that, a quadratic lower bound was shown for *one*-pass streaming [HRR98, FKM+09]. For p-pass streaming algorithms with $p \geq 3$, the best space lower bound was $\Omega(n^{1+1/(2p+2)})$ [GO16]. We mention that the hard instance constructed and analyzed by [AR20] is easy (admits a semi-streaming algorithm), even with only three passes. Additionally, the hard instance used by [GO16] to prove their lower bound against p-pass streaming algorithms can be solved in $n^{1+1/\Omega(p)}$ space with a single pass and with $\widetilde{O}(n)$ space with $p+1$ passes. Thus, Theorem 1.1 cannot be shown using the hard instances constructed by previous papers, and we indeed design a very different instance.

Using a slightly different instance, the techniques used to prove Theorem 1.1 also give a non-trivial lower bound for more than $o(\sqrt{\log n})$ passes. Specifically, we obtain a lower bound of $n^{1+1/O(\log\log n)}$ on the space used by any streaming algorithm with $o(\log n/(\log\log n)^2)$ passes (see Remark 5.2). For p satisfying $p = \omega(\log\log n)$ and $p = o(\log n/(\log\log n)^2)$, this improves over the $\Omega(n^{1+1/(2p+2)})$ lower bound of [GO16]. Still, proving super-linear $n^{1+\varepsilon}$ space lower bounds for n^{ε}-pass streaming algorithms solving s-t reachability with $\varepsilon > 0$ is a great problem that we leave open. (Note that with $O(n)$-passes, semi-streaming is possible by implementing a BFS search).

Since the s-t reachability problem is a special case of the s-t *minimum cut* problem in directed graphs, the lower bound in Theorem 1.1 can also be applied to minimum cut. We note that space efficient algorithms are known for the *undirected* versions of both these problems: s-t connectivity (the undirected version of s-t reachability) has a one-pass semi-streaming algorithm (*e.g.*, by maintaining a spanning forest [FKM+04]) and there is also a two-pass streaming algorithm for s-t minimum cut in undirected graphs that only requires $O(n^{5/3})$ space ([RSW18], see also [ACK19a]).

Technique: Set Hiding. We derive Theorem 1.1 as a special case of a more general framework: given a set $S \subseteq [k]$, we are able to construct a random graph G_S that "*hides*" S, in the sense that for any two different sets S and S', no small-space streaming algorithm with a small number of passes can distinguish between G_S and $G_{S'}$ with any reasonable advantage. The graph G_S we construct has only $n = k^{1+o(1)}$ vertices, out of which k are "designated source vertices" $U = \{u_1, \cdots, u_k\}$ and k are "designated sink vertices" $V = \{v_1, \cdots, v_k\}$. There is a directed path from the source u_i to the sink v_i in G_S if and only if $i \in S$. See Section 2 for a detailed sketch of this construction.

Theorem 1.1 now follows by the following argument: let $s := u_1$ and $t := v_1$ and observe that $G_{[k]}$ has a directed path from s to t, while $G_\emptyset$ does not. This suggests that any algorithm for s-t reachability can also distinguish between $G_{[k]}$ and $G_\emptyset$, violating the hiding property, which is impossible for a small-space algorithm with a small number of passes. In fact, this argument proves a stronger statement: the s-t reachability problem remains hard even under the promise that in the n-vertex input graph, either there are no paths from s to t or there are at least $k = n^{1-o(1)}$ such paths, that are vertex disjoint[1]. This stronger statement allows

[1]To get this, start with the graph $G_{[k]}$ and add two vertices, a global source s and a global sink t. Add

us to obtain lower bounds for approximate versions of related graph problems (with modest, sub-constant approximation factors), as detailed below.

1.1.2 Lower Bounds for Additional Streaming Problems

Matching, shortest path, and rank. As in the case of the two-pass lower bound for s-t reachability proved by [AR20], Theorem 1.1 also implies multi-pass lower bounds for the *shortest path length*, *maximum bipartite matching size*, and *matrix rank* problems. This can be shown by (by now standard) reductions: s-t reachability $\preceq$ shortest path, and s-t reachability $\preceq$ maximum matching $\preceq$ matrix rank.

Theorem 1.2 (Shortest path)**.** *Any randomized $o(\sqrt{\log n})$-pass streaming algorithm that, given an n-vertex* undirected *graph $G = (V, E)$ and two designated vertices $s, t \in V$, can output the length of the shortest path connecting s and t in G requires $n^{2-o(1)}$ space.*

Theorem 1.3 (Matching)**.** *Any randomized $o(\sqrt{\log n})$-pass streaming algorithm that, given an n-vertex* undirected *bipartite graph $G = (L \sqcup R, E)$ can determine whether G has a perfect matching requires $n^{2-o(1)}$ space.*

Theorem 1.4 (Matrix rank)**.** *Any randomized $o(\sqrt{\log n})$-pass streaming algorithm that, given the rows of a matrix $M \in \mathbb{F}_q^{n\times n}$ where $q = \omega(n)$, can determine whether the matrix has full rank requires $n^{2-o(1)}$ space.*

When it comes to lower bounds, the state of affairs for (exact) shortest path, maximum matching, and matrix rank is similar to that of s-t reachability: $\Omega(n^2)$ for one-pass streaming [FKM+04, CCHM14], $\Omega(n^{2-o(1)})$ for two passes [AR20], and $\Omega(n^{1+1/(2p+2)})$ for any $p \geq 3$ [GO16]. On the upper bound front, semi-streaming algorithms with $O(\sqrt{|E|})$ passes are known for (weighted) maximum matching [LSZ20], and with $O(\sqrt{n})$ passes for shortest path [CFCHT20]. Understanding the pass-space trade-offs for these problems is a great goal.

Lower bounds for approximation algorithms. Our proofs of the above theorems also give some non-trivial results in the approximation setting. Specifically, for constant p, our almost quadratic lower bounds continue to hold even for p-pass algorithms that only give a $(1+\omega(\log n)^{-2p})$-approximation to the length of the shortest s-t path (see Theorem 5.1), or a $(1+2^{-\Omega_p(\sqrt{\log n})})$-approximation to the size of the maximum matching or to the rank of a given matrix (see Theorem 5.3 and Corollary 5.6). These lower bounds for approximate maximum matching and approximate matrix rank are possible because our lower bound for s-t reachability holds even when there are many (vertex) disjoint paths from s to t (see Section 1.1.1). In the reduction from s-t reachability to maximum matching, the number of such paths translates into the difference in the size of the maximum matchings that the streaming algorithm is unable to distinguish between.

directed edges from s to every u_i and from every v_i to t.

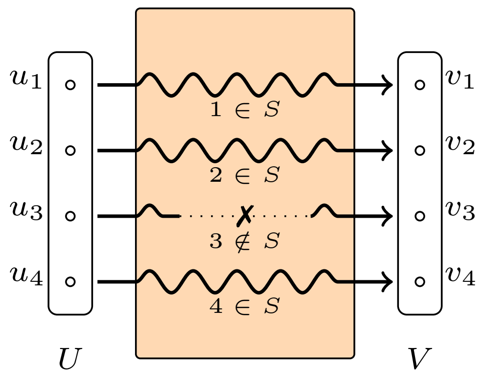


Figure 1: A graph that encodes the set $\{1, 2, 4\}$ using the reachability from U to V. Vertex u_i cannot reach v_j for $i \neq j$.

Since $O_\varepsilon(1)$-pass semi-streaming algorithms for $(1+\varepsilon)$-approximations to the size of the maximum matching and single-source shortest path are known [McG05, BKKL17], our above lower bounds for these problems cannot be strengthened to deal with constant ε. A polynomial (but sub-linear) lower bound of $n^{1-\varepsilon^{\Omega(1/p)}}$ on the space complexity of p-pass streaming algorithms that obtain a $(1+\varepsilon)$-approximation of maximum matching and of matrix rank was very recently proved by [AKSY20].

Lower bounds for additional problems. Via other known reductions, the lower bound in Theorem 1.1 can be shown to imply lower bounds for additional streaming problems, such as estimating the number of *reachable vertices* from a given source [HRR98] and approximating the *minimum feedback arc set* [CGMV20].

We also consider the *linear programming feasibility* (LP feasibility) problem, where given a set of n linear constraints (inequalities) over d variables, one needs to decide if all constraints can be satisfied simultaneously. We prove that Theorem 1.1 implies a similar lower bound for the LP feasibility problem with $d \approx n$, see Theorem 5.7 (for the low dimension $d \ll n$ regime, see [AKZ19, CC07]). To this end, we devise a reduction from s-t reachability to LP feasibility that exploits the fact that our hard s-t reachability instance is a layered graph[2].

2 Technical Overview

Our proof proceeds by designing a carefully structured hard instance for s-t reachability. The key component in our lower bound proof is a construction that *hides* a set in a random (directed) graph from streaming algorithms. Specifically, let $S \subseteq [n]$ be a set, and U, V be two sets of n vertices. We will construct a random graph, possibly adding more vertices, such that u_i (the i-th vertex in U) cannot reach v_j (the j-th vertex in V) for any $i \neq j$;

[2]While there are known reductions from s-t reachability to LP feasibility, to the best of our knowledge, our reduction is the only one that is both deterministic and generates $\Theta(n)$ dense constraints with super small coefficients ($\{0, 1, -1\}$ coefficients), as opposed to polynomially large ones.

and u_i can reach v_i *if and only if* $i \in S$ (see Figure 1). That is, the graph encodes the set S using the reachability from U to V. The most important feature of this random graph construction is that (with a proper ordering of its edges in a stream) any p-pass (for some small p) low-space streaming algorithm $\mathcal{A}$ cannot "learn anything" about S, in the sense that for any S_1 and S_2, $\mathcal{A}$ cannot distinguish between the random graphs generated based on S_1 or S_2 except with probability at most $1/n$. We call such a random graph a Set-Hiding graph.[3]

Assuming such a graph construction, the s-t reachability lower bound follows easily. To see this, we set $S_1 := \emptyset$ and $S_2 := \{1\}$, let the source s be u_1 and the sink t be v_1. Then in a Set-Hiding graph that hides S_1, s cannot reach t; and in a Set-Hiding graph that hides S_2, s can reach t. But any p-pass low-space streaming algorithm cannot distinguish between the two cases. In particular, it is impossible for such an algorithm to solve s-t reachability. In the following, we will focus on the construction of such Set-Hiding graphs.

2.1 Set-Hiding Graphs Against One-Pass Algorithms

Let us for now set the goal to constructing graphs that hide a set S from any low-space *one-pass* streaming algorithm, as a demonstration of the idea.

2.1.1 A New Communication Problem

The problem. It turns out that the indistinguishablility stems from the hardness of the following one-way communication problem:

- Alice gets nK sets $(T_j^{(k)})_{(j,k)\in[n]\times[K]}$ (think of $K = \log n$), which are subsets of $[n]$;
- Bob gets K indices $j_1, \ldots, j_K \in [n]$ and K permutations $\pi_1, \ldots, \pi_K$ on $[n]$;
- Alice sends a single message to Bob, whose goal is to learn the set

$$S := \bigoplus_{k=1}^{K} \pi_k(T_{j_k}^{(k)}),$$

  where $\oplus$ of sets is defined as the coordinate-wise XOR of their indicator vectors, and $\pi(T) := \{\pi(a) : a \in T\}$.

In other words, Alice gets K collections of sets, each collection consists of n sets, and each set is over $[n]$. Then Bob picks one set from each collection, permutes the sets according to his input, and he wishes to know the $\oplus$ of the K permuted sets.

Lower bound. We prove that for *any* two sets S_1 and S_2, if Alice's message has only $n^{1.99}$ bits (note that her input has n^2K bits), Bob is not able to distinguish between $S = S_1$ and

[3] In the formal proof, the collection of 2^n such random graphs, one for each subset of $[n]$, is called a Set-Hiding generator, and a single (deterministic) graph encoding a set is called a Set-Encoding graph. We will not differentiate between the two when discussing the intuition in this section.

$S = S_2$, except with probability exponentially small in K.[4] Note that we prove a much stronger form of lower bound than just a lower bound on the error probability of computing S, such an indistinguishability lower bound is crucial to obtain the Set-Hiding property of our graph construction.

The communication lower bound proof uses an *XOR lemma* for the INDEX problem, which we prove in this paper. Suppose the players only focus on deciding whether $1 \in S$, then Alice's input can be viewed as K arrays of length n^2, where the k-th array is the concatenation of the n indicator vectors $\mathbb{1}(T_j^{(k)})$ for $j \in [n]$, and Bob's input chooses one entry from each array. The bit indicating $1 \in S$ is precisely the XOR of the K chosen bits, i.e., the XOR of the $\pi_k^{-1}(1)$-th bit in $T_{j_k}^{(k)}$ for $k \in [K]$ (observe that both the index j_k and the random permutation π_k in the definition of the communication problem are needed to ensure that Alice doesn't know what entry is chosen by Bob in array k).

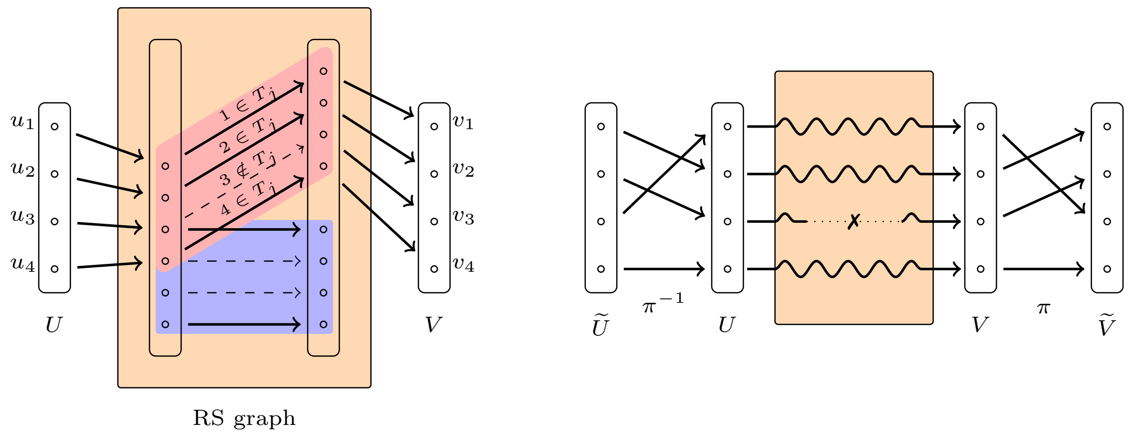
The standard INDEX lower bound shows that for one array, if the communication is less than $n^{1.99}$, from Bob's view the chosen bit is still close to uniform, with bias at most $n^{-\Omega(1)}$. If the players handle all K arrays independently, then the K chosen bits are independent from Bob's view. Therefore, their XOR has bias at most $n^{-\Omega(K)}$, by the standard result on the XOR of independently random bits. In general, an XOR lemma states that this bias bound holds even for generic protocols. We prove such an XOR lemma for INDEX by showing a discrepancy bound (a similar discrepancy bound was (implicitly) proved in [GPW17, GKMP20] using a different argument). Finally, we apply this XOR lemma and a hybrid argument to prove the indistinguishability of any two sets S_1 and S_2. See Section 6 for the XOR lemma for INDEX, and Section 7 for the communication lower bound.

2.1.2 Constructing the Set-Hiding Graph

To construct the Set-Hiding graph that hides a set S, we first generate $(T_j^{(k)})_{(j,k)}, (j_k)_k, (\pi_k)_k$ according to the hard input distribution for the communication problem, *conditioned on* the final set being S, i.e., $S = \bigoplus_{k=1}^K \pi_k(T_{j_k}^{(k)})$. Then, we will construct a graph that mimics the computation of $\bigoplus_{k=1}^K \pi_k(T_{j_k}^{(k)})$, and use the hardness of this communication problem to argue that low-space streaming algorithms cannot learn anything about S.

Representing index selection – graphs for $(T_1^{(k)}, \dots, T_n^{(k)})$ and $T_{j_k}^{(k)}$. To this end, we do this computation bottom-up, and let us first see how to "compute" the set $T_{j_k}^{(k)}$ for each k, i.e., *select* the j_k-th set from the collection $(T_1^{(k)}, \dots, T_n^{(k)})$. This is done using a similar

[4]Careful readers may have noticed that the statement as written here is technically false, as Alice could send the parity of the size for each set, taking $nK \ll n^{1.99}$ bits. In this case, Bob learns the parity of S, falsifying the statement for S_1, S_2 with different parities. In the actual proof, Alice's sets as well as Bob's permutations will be over $[4n]$, and the set S is defined to be $\bigoplus_{k=1}^K \pi_k(T_{j_k}^{(k)})$ *restricted to the first n elements* $[n]$. It resolves the above parity issue, and the indistinguishability holds in this case. The arguments below follow as well.



(a) (b)

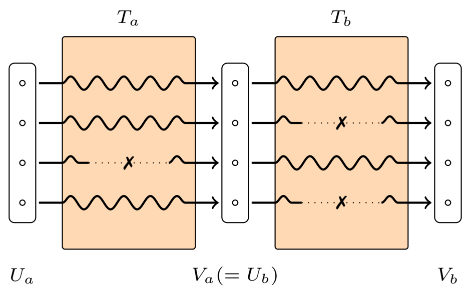
Figure 2: In (a), the red and blue boxes correspond to two *induced* matchings in the RS graph, thin dashed edges exist in the original RS graph, but are removed according to T_j.

construction to [AR20], which uses the *Ruzsa-Szemerédi graphs* (RS graphs). The version we use is a bipartite graph on m vertices, whose edges form a disjoint union of t *induced* matchings of size r (i.e., r by r bipartite subgraphs consisting of only r matching edges). Such graphs were shown to exist for $r, t = m \cdot 2^{-\Theta(\sqrt{\log m})}$ [RS78].

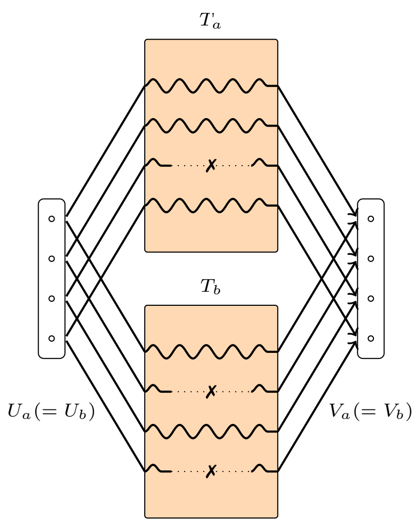
For each k, we first fix such an RS graph with $r, t \geq n$. Then, we associate the j-th matching with set $T_j^{(k)}$, and keep the i-th edge in the matching if and only if $i \in T_j^{(k)}$. The RS graph encodes the collection $(T_1^{(k)}, \ldots, T_n^{(k)})$. Intuitively, we work with RS graphs as they allow us to "pack" many sets into a small graph. We select the j_k-th set by connecting two vertex sets U and V to the corresponding matching (see Figure 2a). In this way, $u_i \in U$ can reach $v_i \in V$ if and only if $i \in T_{j_k}^{(k)}$, i.e., the reachability from U to V *encodes* the selected set $T_{j_k}^{(k)}$ (in the same way as $\mathsf{Set\text{-}Hiding}$ graphs would encode S).

Representing permutations: graph for $\pi_k(T_{j_k}^{(k)})$. Next, we implement the permutation by adding two more layers $\widetilde{U}$ and $\widetilde{V}$, and putting a matching corresponding to π_k^{-1} from $\widetilde{U}$ to U, and a matching corresponding to π_k from V to $\widetilde{V}$. Then the reachability from $\widetilde{U}$ to $\widetilde{V}$ encodes $\pi_k(T_{j_k}^{(k)})$ (see Figure 2b).

Representing XOR: graph for $\bigoplus_{k=1}^{K} \pi_k(T_{j_k}^{(k)})$. The last step is to mimic the computation of $\oplus$ of K sets. We have constructed K graphs such that in k-th graph, the reachability from U_k to V_k encodes $\pi_k(T_{j_k}^{(k)})$. We wish to combine them into a single graph, containing U and V as subsets of vertices, such that the reachability from U to V encodes $\oplus_{k=1}^{K} \pi_k(T_{j_k}^{(k)})$. The main idea is to use the fact that there exists an $\{\wedge, \vee, \neg\}$-*formula* of size $O(K^2)$ that computes the XOR of K-bit. For $x_1, \ldots, x_K \in \{\text{True}, \text{False}\}$, we can write $x_1 \oplus x_2 \oplus \cdots \oplus x_K$



(a)



(b)

Figure 3: (a) shows a graph computing $T_a \wedge T_b$, and (b) shows a graph computing $T_a \vee T_b$.

as a small Boolean formula F which only uses AND, OR and NOT gates. Moreover, we can assume that the NOTs are only applied on the x_i. F will be applied to the sets $\pi_k(T_{j_k}^{(k)})$ *coordinate-wisely*, computing $\oplus_{k=1}^{K}\pi_k(T_{j_k}^{(k)})$. We are going to construct the graph recursively according to F.

For an AND gate in F, suppose its two operands are T_a and T_b, and we have constructed a graph containing U_a and V_a as subsets of vertices that encodes T_a using the reachability from U_a to V_a, and a graph containing U_b and V_b that encodes T_b using the reachability from U_b to V_b. Then the coordinate-wise AND of T_a and T_b (equivalently, the intersection) can be computed by merging V_a and U_b into one set (see Figure 3a). For an OR gate in F with two operands T_a and T_b, their coordinate-wise OR (equivalently, the union) can be computed by merging U_a and U_b into one set, and merging V_a and V_b into one set (see Figure 3b). Eventually, we either reach an input variable corresponding to a graph that encodes one $\pi_k(T_{j_k}^{(k)})$, which we have already constructed, or reach the *negation* of an input variable, which corresponds to the *complement* of one $\pi_k(T_{j_k}^{(k)})$. It suffices to also construct a graph that encodes $[n] \setminus \pi_k(T_{j_k}^{(k)})$ for each k. Note that $[n] \setminus \pi_k(T_{j_k}^{(k)}) = \pi_k([n] \setminus T_{j_k}^{(k)})$. Therefore, this can be done by applying the construction in the last paragraph on the complement of input sets $([n] \setminus T_j^{(k)})_{(j,k)}$ (and with the same indices j_k and permutations π_k).

The order of edges in the stream. The above construction generates a graph that encodes the set $S = \bigoplus_{k=1}^{K} \pi_k(T_{j_k}^{(k)})$, which we wanted to hide. To determine the order of its edges in the stream, observe that the edges in all RS (sub)graphs only depend on the sets $(T_j^{(k)})_{(j,k)}$, and the rest of the graph only depends on the indices $(j_k)_k$ and permutations $(\pi_k)_k$. Hence, in the stream, we will first give all edges in the RS graphs, then all remaining edges. By the standard reduction from one-way communication to streaming algorithms and the hardness of the communication problem, we prove that S is hidden from any $n^{1.99}$-space one-pass streaming algorithm.

2.2 Generalizing to p Passes

Hiding the selected sets. The lower bound for the one-way communication problem uses the fact that Alice does not know the indices and permutations. Equivalently, the streaming algorithm does not know the parts encoding $(j_k)_k$ and $(\pi_k)_k$ when it sees the RS graphs, which encode $(T_j^{(k)})_{(j,k)}$. However, this is not the case if the algorithm can read the stream even just twice, as it can remember the indices and permutations in the first pass so that the second time it sees the RS graphs, it already knows which sets are selected. To generalize our hard instance to p passes, the main idea is to also *hide* the indices and permutations, from $(p-1)$-pass streaming algorithms.

More specifically, we wish to construct subgraphs (gadgets) that serve the same purposes as the parts encoding the indices and permutations (in terms of reachability), but additionally, for any $(j_k)_k, (\pi_k)_k$ and $(j'_k)_k, (\pi'_k)_k$, any low-space $(p-1)$-pass algorithm should not be able to distinguish between the subgraphs constructed based on them. Suppose we have such gadgets, then we may apply the one-way communication lower bound *to the p-th pass* (after replacing the edges from U to the RS graph and from the RS graph to V in Figure 2a by such gadgets, and replacing the edges from $\widetilde{U}$ and U and the edges from V to $\widetilde{V}$ in Figure 2b). This is because when the streaming algorithm sees $(T_j^{(k)})_{(j,k)}$ for the p-th time, it has only scanned the parts encoding the indices and permutations $p-1$ times (recall that $(T_j^{(k)})_{(j,k)}$ appears before all indices and permutations in the stream), and is not able to learn anything about them. Therefore, the one-way communication lower bound still holds.

Perm-Hiding graphs. To construct such subgraphs, we construct a gadget that allows us to hide each permutation or index separately. To be more precise, given a permutation π on $[n]$, we want to construct a (random) graph containing X and Y as subsets of vertices, such that for each $i \in [n]$, the only vertex in Y that u_i can reach is $v_{\pi(i)}$ (the "indices" are special cases of the "permutations", and they can also be hidden using such gadgets, see Section 9.3). Moreover, for any π_1, π_2, any $(p-1)$-pass low-space streaming algorithm cannot distinguish between the graphs generated from π_1 and π_2. We call such random graphs the Perm-Hiding graphs.

Hiding structured permutations via Set-Hiding graphs. We first show that (assuming n is even) if π is structured such that for each i, either $\pi(2i) = 2i$ and $\pi(2i+1) = 2i+1$, or $\pi(2i) = 2i+1$ and $\pi(2i+1) = 2i$ (i.e., for each i, π either swaps $2i$ and $2i+1$, or maps both to themselves), then we can construct a Perm-Hiding graph for π using the Set-Hiding graphs against $(p-1)$-pass streaming algorithms. To see this, we add two extra layers $\widetilde{X}, \widetilde{Y}$ of sizes

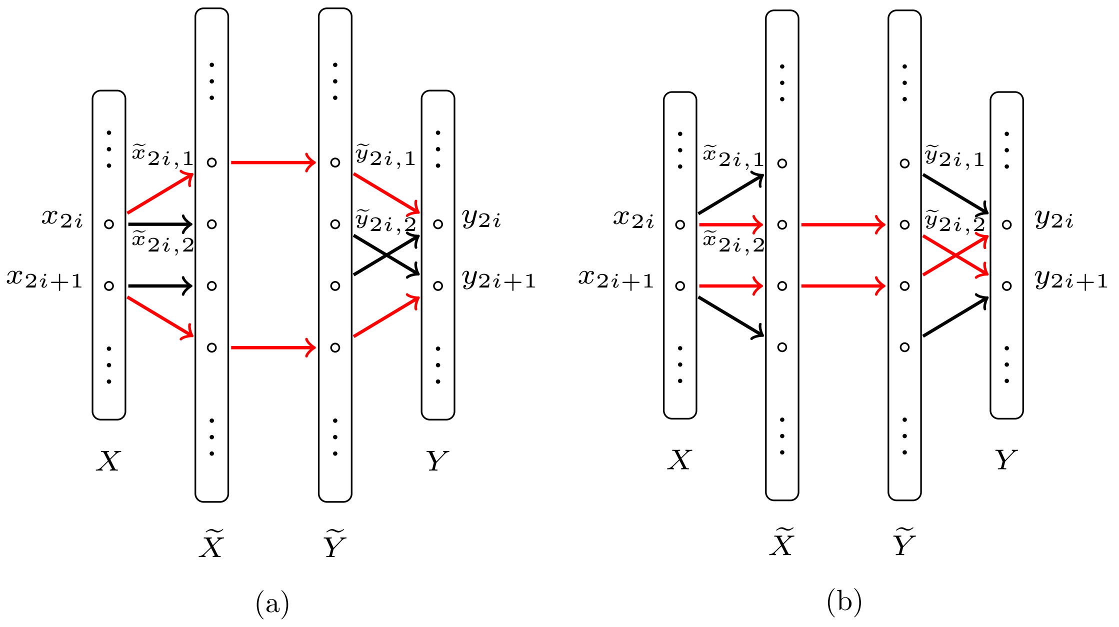


Figure 4: (a) shows the graph that does not swap $2i$ and $2i+1$, (b) shows the graph that swaps $2i$ and $2i+1$. The edges from X to $\widetilde{X}$ and the edges from $\widetilde{Y}$ to Y are fixed.

$2n$ between X and Y. Denote the vertices in $\widetilde{X}$ and $\widetilde{Y}$ by $\widetilde{x}_{j,1}, \widetilde{x}_{j,2}$ and $\widetilde{v}_{j,1}, \widetilde{v}_{j,2}$ respectively for $j \in [n]$. For all i, we add the following edges from X to $\widetilde{X}$ and from $\widetilde{Y}$ to Y:

- from x_{2i} to $\widetilde{x}_{2i,1}, \widetilde{x}_{2i,2}$, from x_{2i+1} to $\widetilde{x}_{2i+1,1}, \widetilde{x}_{2i+1,2}$, and
- from $\widetilde{y}_{2i,1}, \widetilde{y}_{2i+1,1}$ to y_{2i}, from $\widetilde{y}_{2i,2}, \widetilde{y}_{2i+1,2}$ to y_{2i+1}.

Now if we add an edge from $\widetilde{x}_{2i,1}$ to $\widetilde{y}_{2i,1}$ and an edge from $\widetilde{x}_{2i+1,2}$ to $\widetilde{y}_{2i+1,2}$, then x_{2i} reaches y_{2i} and x_{2i+1} reaches y_{2i+1} (see Figure 4a); if we add an edge from $\widetilde{x}_{2i,2}$ to $\widetilde{y}_{2i,2}$ and an edge from $\widetilde{x}_{2i+1,1}$ to $\widetilde{y}_{2i+1,1}$, then x_{2i} reaches y_{2i+1} and x_{2i+1} reaches y_{2i} (see Figure 4b). In the other words, such a permutation can always be implemented by placing a set of *parallel* edges from $\widetilde{X}$ to $\widetilde{Y}$. Therefore, to hide π, it suffices to put a Set-Hiding graph between $\widetilde{X}$ and $\widetilde{Y}$ to hide the corresponding set over $[2n]$. Since by the guarantee of Set-Hiding graphs, no low-space algorithm can distinguish between any two sets, we prove that any such two permutations cannot be distinguished.

Similarly, if there is a set of *fixed* $\leq n/2$ *disjoint* pairs of coordinates such that π may only swap two coordinates in a pair, then the same argument from the last paragraph shows that such permutations can be hidden from $(p-1)$-pass streaming algorithms as well, assuming Set-Hiding graphs.

Hiding general permutations. Finally, we use the fact that there exists $d = O(\log n)$ *fixed* sets of $\leq n/2$ disjoint pairs,[5] such that *every* permutation π can be decomposed into $\pi = \pi_d \circ \cdots \circ \pi_2 \circ \pi_1$, where π_i may only swap (a subset of) the pairs in the i-th set (e.g., this is a corollary of the existence of $O(\log n)$-depth sorting networks [AKS83]). The final

[5]It is important that the sets do not depend on π.

Perm-Hiding graph consists of d blocks concatenated with identity matchings, where the i-th block applies the construction from the last paragraph to swap pairs in the i-th set. For each π_i, we hide a set in the block using Set-Hiding. By a standard hybrid argument, we conclude that no two permutations can be distinguished.

Putting it together. Overall, the p-pass Set-Hiding graphs use the structure from Section 2.1, together with $(p-1)$-pass Perm-Hiding graphs. The $(p-1)$-pass Perm-Hiding graphs, in turn, use $(p-1)$-pass Set-Hiding graphs, which are constructed *recursively*. One may verify that the size of the graph blows up by a factor of $2^{\Theta(\sqrt{\log n})}$ in each level of recursion, due to the parameters in RS graphs. Hence, when $p = o(\sqrt{\log n})$, the final graph size N is at most $n^{1+o(1)}$, implying the space lower bound of $N^{1.99}$. See Section 9 for the formal construction of Set-Hiding graphs, Section 10 for the construction of Perm-Hiding graphs, and Section 4 for the recursive construction that combines them.

Acknowledgments

The authors would like to thank Sepehr Assadi for useful discussions.

Lijie Chen is supported by an IBM Fellowship. Zhao Song is supported in part by Schmidt Foundation, Simons Foundation, NSF, DARPA/SRC, Google and Amazon AWS.

3 Preliminary

3.1 Notation

We often use bold font letters (*e.g.*, $\boldsymbol{X}$) to denote random variables, and calligraphic font letters (*e.g.*, $\mathcal{X}$) to denote distributions. For two random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$, and for $Y \in \operatorname{supp}(\boldsymbol{Y})$, we use $(X|\boldsymbol{Y} = Y)$ to denote $\boldsymbol{X}$ conditioned on $\boldsymbol{Y} = Y$. For two lists a and b, we use $a \circ b$ to denote their concatenation.

For two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ on set $\mathcal{X}$ and $\mathcal{Y}$ respectively, we use $\mathcal{D}_1 \otimes \mathcal{D}_2$ to denote their product distribution over $\mathcal{X} \times \mathcal{Y}$, and $\|\mathcal{D}_1 - \mathcal{D}_2\|_{\mathrm{TV}}$ to denote the total variation distance between them.

Let $n \in \mathbb{N}$. We use $[n]$ to denote the set $\{1, \ldots, n\}$. For two sets $A, B \subseteq [n]$, we use $A \wedge B$ and $A \vee B$ to denote the intersection and the union of A and B, respectively. We also use $\neg_n A$ to denote the set $[n] \setminus A$, and $A \oplus B$ to denote the set of elements appearing in exactly one of A and B (*i.e.*, the symmetric difference of the sets A and B). Note that

$$A \oplus B = (A \wedge \neg_n B) \vee (\neg_n A \wedge B).$$

When it is clear from the context, we drop the subscript in $\neg_n$ for simplicity.

We also use $\mathsf{Perm}([n])$ to denote the set of permutations on $[n]$. For a predicate P, we use $\mathbb{1}(P)$ to denote the corresponding Boolean value of P, that is, $\mathbb{1}(P) = 1$ if P is true, and 0 otherwise.

3.2 Layered Graphs and Layer-Arrival Model

In this paper we will mostly consider directed layered graphs whose edges are always from one layer to its succeeding layer. We will also associate an **edge-layer ordering** to the layered graph G, which will be very convenient when we are working with graph streaming algorithms.

Directed Layered Graphs. Formally, a **directed layered graph** G is a triple $(\vec{V}, \vec{E}, \vec{\ell})$, such that:

- $\vec{V} = (V_i)_{i=1}^{k}$ is the collection of G's layers, where k is the number of layers in G;
- $\vec{\ell} = (\ell_i)_{i=1}^{k-1}$ and $\vec{E} = (E_i)_{i=1}^{k-1}$ is a list of disjoint sets of edges on the vertex set $V = \bigcup_{i=1}^{k} V_i$. For each $i \in [k-1]$, E_i is the set of all the edges in G between V_{ℓ_i} and V_{ℓ_i+1}. All the indices ℓ_i are distinct integers in $[k-1]$.

For each $i \in [k-1]$, we call the set of edges between V_i and V_{i+1} the i-th **edge-layer** of G. That is, $\vec{\ell}$ specifies an ordering of edge-layers of G, we will call it the edge-layer ordering of G. We remark that unless some edge-layers are empty, the edge list vector $\vec{E}$ always uniquely determines the ordering $\vec{\ell}$. In most cases we will just specify the edge list vector $\vec{E}$ and the $\vec{\ell}$ will be determined from the context.

We will use $E(G)$, $\vec{V}(G)$, $k(G)$ and $\vec{\ell}(G)$ to denote the set of edges, the list of layers of G, the number of layers in G and the edge-layer ordering of G, respectively. For $i \in [k]$, we use $V_i(G)$ to denote the vertex set of the i-th layer of G. We also use $V(G)$ to denote $\bigcup_{i=1}^{k} V_i(G)$.

We say a layered graph G is an $(N_G, k_G, \vec{\ell}_G)$ graph, if G has N_G vertices, k_G layers and its edge-layer ordering is $\vec{\ell}_G$.

For a layered graph G, we use $\mathsf{First}(G)$ to denote $V_1(G)$ and $\mathsf{Last}(G)$ to denote $V_{k(G)}(G)$ for convenience. For each layer, we index all the vertices by consecutive integers starting from 1. For a set S of vertices from a single layer of G (that is, $S \subseteq V_i$ for some $i \in [k]$), we use $S_{[i]}$ to denote the vertex with the i-th smallest index in S.

We note that a directed bipartite graph (all edges go from the left side to the right side) is a directed layered graph with two layers (for which the list $\vec{E}$ only contains a single set of all edges in the graph, and $\vec{\ell} = (1)$). Unless explicitly stated otherwise, we will always use layered graphs or bipartite graphs to refer to their directed versions. (The only place we study undirected graphs is in Section 5.1 and Section 5.2.)

Concatenation of two layered graphs. For two layered graphs G_1 and G_2 such that $|\mathsf{Last}(G_1)| = |\mathsf{First}(G_2)|$, we use $H = G_1 \odot G_2$ to denote their concatenation by identifying $\mathsf{Last}(G_1)$ and $\mathsf{First}(G_2)$. That is, for each $i \in [|\mathsf{Last}(G_1)|]$, we identify the vertex $\mathsf{Last}(G_1)_{[i]}$ and $\mathsf{First}(G_2)_{[i]}$. We also set $\vec{E}(H) = \vec{E}(G_1) \circ \vec{E}(G_2)$ to specify the edge-layer ordering of H.

The layer-arrival model. Our lower bounds actually hold for the *layer-arrival* setting, which is stronger than the usually studied edge-arrival or vertex-arrival models. In the following we formally define this model.

Definition 3.1 (Layer-arrival model). *Given a layered graph $G = (\vec{V}, \vec{E} = (E_i)_{i=1}^{k-1}, \vec{\ell})$ of k layers, a randomized p-pass streaming algorithm A with space s in the layer-arrival setting works as follows:*

- *The algorithm makes p-pass over the graph, each pass has $(k-1)$ phases. Hence, there are $(k-1) \cdot p$ phases in total. The algorithm starts with memory state $w_0 = 0^s$. Additionally, at the beginning A can draw an unbounded number of random bits, from a fixed distribution $\mathcal{D}_{\mathsf{rand}}$. These random bits are read-only and A can always access them freely.*[6]
- *For $i \in [p]$ and $j \in [k-1]$, let $t = (i-1) \cdot (k-1) + j$. In the t-th phase, A can use unlimited computational resource to compute another state w_t of s bits, given the previous state w_{t-1} together with the edge set E_j. (Note that w_t is indeed a random variable depending on w_{t-1} and E_j, since A is randomized.)*
- *Finally, A's output only depends on the last state $w_{p(k-1)}$ and its random bits.*

In other words, the streaming algorithm is allowed to access the graph layer by layer, and can use unlimited computational resources to process each layer. The only constraint is that it can restore at most s bits of information after processing one layer.

Clearly, lower bounds for graph streaming algorithms in the layer-arrival model immediately imply the same lower bounds for graph streaming algorithms in the edge-arrival model or vertex-arrival model.

3.3 Ruzsa-Szemerédi Graphs

A bipartite graph $G^{\mathsf{RS}} = (L \sqcup R, E)$ is a called an (r,t)-Ruzsa-Szemerédi graph (RS graph for short) if its edge set E can be partitioned into t *induced matchings* $M_1, \ldots, M_t$, each of size r.

We will use the original construction of RS graphs due to Ruzsa and Szemerédi [RS78] based on the existence of large sets of integers with no 3-term arithmetic progression, proven by Behrend [Beh46].

[6]That is, randomness is free for A and are not charged in the space complexity of A. This is very important for the hybrid argument used in this paper, see Section 3.4.

Proposition 3.2 ([RS78]). *There is an absolute constant $c^{\mathsf{RS}} \geq 1$ such that, for all sufficiently large n, there is an integer $N \leq n^{1+c^{\mathsf{RS}}/\sqrt{\log n}}$ such that there are (n,n)-RS graphs with N vertices on each side of the bipartition.*

For convenience, we define $N^{\mathsf{RS}}(n) = n^{1+c^{\mathsf{RS}}/\sqrt{\log n}}$. We also need the following construction of RS graphs with different parameters by [FLN+02].

Proposition 3.3 ([FLN+02]). *There is an absolute constant $c_2^{\mathsf{RS}} > 0$ such that, for all sufficiently large n, there are $(n, n^{c_2^{\mathsf{RS}}/\log\log n})$-RS graphs with $4n$ vertices on each side of the bipartition.*

3.4 Indistinguishability and The Hybrid Argument

We say two distributions on layered graphs $\mathcal{D}_1$ and $\mathcal{D}_2$ are ε-indistinguishable for p-pass streaming algorithms with space s in the layer-arrival model, if for every p-pass streaming algorithm A with space s in the layer-arrival model, it holds that

$$\|A(\mathcal{D}_1) - A(\mathcal{D}_2)\|_{\mathrm{TV}} \leq \varepsilon,$$

where for each $i \in [2]$, $A(\mathcal{D}_i)$ is the output distribution of A given an input graph drawn from $\mathcal{D}_i$.

Note that the above notation of indistinguishability also generalizes to streaming algorithms in the edge-arrival model or vertex-arrival model. But throughout this paper we will mostly study indistinguishability with respect to multi-pass streaming algorithms in the layer-arrival model. Hence, we will just omit the model name whenever it is clear from the context.

Given t layered graphs $G_1, \ldots, G_t$. They can be treated as a single input to a p-pass streaming algorithm A as follows: there are p passes, in each pass A process (the edges of) $G_1, \ldots, G_t$ in order. We use $(G_1, \ldots, G_k)_{\mathsf{seq}}$ to denote this new input to A.

The following lemma shows that the standard hybrid argument also applies to the setting of multi-pass graph streaming algorithms. The hybrid argument will be used throughout our proofs, and we give a proof here for completeness.

Lemma 3.4 (Hybrid argument for multi-pass streaming algorithms). *Let k be a positive integer. Let $\varepsilon \in \mathbb{R}_{\geq 0}^k$ denote k parameters. Let $(\mathcal{D}_1, \mathcal{D}'_1), (\mathcal{D}_2, \mathcal{D}'_2), \ldots, (\mathcal{D}_k, \mathcal{D}'_k)$ be k pairs of distributions over layered graphs. Suppose for each $i \in [k]$, $\mathcal{D}_i$ and $\mathcal{D}'_i$ are ε_i-indistinguishable for p-pass streaming algorithms with space s, then $(\mathcal{D}_1, \ldots, \mathcal{D}_k)_{\mathsf{seq}}$ and $(\mathcal{D}'_1, \ldots, \mathcal{D}'_k)_{\mathsf{seq}}$ are $\|\epsilon\|_1$-indistinguishable for p-pass streaming algorithms with space s.*[7]

Proof. Our proof is based on a standard hybrid argument. For each $j \in \{0, 1, \ldots, k\}$, let

$$\mathcal{H}_j = (\mathcal{D}_1, \ldots, \mathcal{D}_j, \mathcal{D}'_{j+1}, \ldots, \mathcal{D}'_k)_{\mathsf{seq}}.$$

[7] $(\mathcal{D}_1, \ldots, \mathcal{D}_k)_{\mathsf{seq}}$ denotes the distribution obtained by for each $i \in [k]$, independently drawing $D_i \leftarrow \mathcal{D}_i$, and outputting $(D_1, \ldots, D_k)_{\mathsf{seq}}$.

Observe that $\mathcal{H}_0 = (\mathcal{D}'_1, \ldots, \mathcal{D}'_k)_{\mathsf{seq}}$ and $\mathcal{H}_k = (\mathcal{D}_1, \ldots, \mathcal{D}_k)_{\mathsf{seq}}$. Let A be a p-pass streaming algorithms with space s.

We claim that for each $j \in [k]$,

$$\|A(\mathcal{H}_j) - A(\mathcal{H}_{j-1})\|_{\mathrm{TV}} \le \varepsilon_j.$$

Assuming the claim above holds, the lemma follows from the triangle inequality.

To prove the claim above, we show how to construct another streaming algorithm B with the same pass and space complexity as A, such that $\|A(\mathcal{H}_j) - A(\mathcal{H}_{j-1})\|_{\mathrm{TV}} = \|B(\mathcal{D}_j) - B(\mathcal{D}'_j)\|_{\mathrm{TV}}$. Given an input graph G, B first draws graphs $G_1 \sim \mathcal{D}_1, \ldots, G_{j-1} \sim \mathcal{D}_{j-1}$, and then draws graphs $G_{j+1} \sim \mathcal{D}'_{j+1}, \ldots, G_k \sim \mathcal{D}'_k$. B then simulates A on the input $(G_1, \ldots, G_{j-1}, G, G_{j+1}, \ldots, G_k)$.

Recall that our definition of randomized streaming algorithms (see Definition 3.1) allows unbounded randomness from any fixed distribution (which are independent from the input distribution), and the random bits are not counted in space usage. The $k-1$ sampled graphs of B are then regarded as B's randomness. Hence, B has the same pass and space complexity of A. Moreover, one can see that $B(\mathcal{D}_j)$ distributes as $A(\mathcal{H}_j)$ and $B(\mathcal{D}'_j)$ distributes as $A(\mathcal{H}_{j-1})$. Since $\mathcal{D}_j$ and $\mathcal{D}'_j$ are ε_j-indistinguishable for p-pass streaming algorithms with space s, we have

$$\|A(\mathcal{H}_j) - A(\mathcal{H}_{j-1})\|_{\mathrm{TV}} = \|B(\mathcal{D}_j) - B(\mathcal{D}'_j)\|_{\mathrm{TV}} \le \varepsilon_j,$$

which completes the proof of the claim. $\square$

3.5 Set-Encoding/Perm-Encoding Graphs and Set-Hiding/Perm-Hiding Generators

3.5.1 Set-Encoding Graphs and Perm-Encoding Graphs

The following two special layered graphs will be studied throughout the paper.

Definition 3.5 (Set-Encoding graphs and Perm-Encoding graphs)**.**

1. *(*Set-Encoding *Graphs) For a set $S \subseteq [n]$, we say a layered graph G with first and last layer each having exactly n vertices is a* $\mathsf{Set\text{-}Enc}_n(S)$ *graph (i.e., a* Set-Encoding *graph for the set S). If for each $(i, j) \in [n] \times [n]$,* $\mathsf{First}(G)_{[i]}$ *can reach* $\mathsf{Last}(G)_{[j]}$ *if and only if $i = j$ and $i \in S$.*

2. *(*Perm-Encoding *Graphs) For a permutation $\pi\colon [n] \to [n]$, we say a layered graph G with first and last layer each having exactly n vertices is a* $\mathsf{Perm\text{-}Enc}_n(\pi)$ *graph (i.e., a* Perm-Encoding *graph for the permutation π). If for each $(i, j) \in [n] \times [n]$,* $\mathsf{First}(G)_{[i]}$ *can reach* $\mathsf{Last}(G)_{[j]}$ *if and only if $\pi(i) = j$.*

3.5.2 Set-Hiding Generators and Perm-Hiding Generators

Note that a single Set-Encoding graph (resp. Perm-Encoding graph) just encodes a set, and does not hide it. Now we formally define Set-Hiding generators and Perm-Hiding generators, which generate *distributions* over Set-Enc/Perm-Enc graphs that hides the encoded set/permutation from multi-pass streaming algorithms.

Definition 3.6 (ε-secure Set-Hiding generators)**.** *Let $n \in \mathbb{N}$, and let $\mathcal{G}$ be a function from subsets of $[n]$ to distributions over layered graphs. We say $\mathcal{G}$ is ε-*Set-Hiding *for subsets of $[n]$ against p-pass algorithms with space s, if the following statements hold:*

1. *For every $S \subseteq [n]$, $\mathcal{G}(S)$ is a distribution over* $\mathsf{Set\text{-}Enc}_n(S)$ *graphs.*
2. *For every two sets $S, T \subseteq [n]$, the distributions $\mathcal{G}(S)$ and $\mathcal{G}(T)$ are ε-indistinguishable for p-pass streaming algorithms with space s.*

Definition 3.7 (ε-secure Perm-Hiding generators)**.** *Let $n \in \mathbb{N}$, and let $\mathcal{G}$ be a function from* $\mathsf{Perm}([n])$ *to distributions over layered graphs. We say $\mathcal{G}$ is ε-*Perm-Hiding *for permutations in* $\mathsf{Perm}([n])$ *against p-pass algorithms with space s, if the following statements hold:*

1. *For every $\pi \in \mathsf{Perm}([n])$, $\mathcal{G}(\pi)$ is a distribution over* $\mathsf{Perm\text{-}Enc}_n(\pi)$ *graphs.*
2. *For every two permutations $\pi_1, \pi_2 \in \mathsf{Perm}([n])$, the distributions $\mathcal{G}(\pi_1)$ and $\mathcal{G}(\pi_2)$ are ε-indistinguishable for p-pass streaming algorithms with space s.*

For a generator $\mathcal{G}$ as in Definition 3.6 and Definition 3.7, we say $\mathcal{G}$ always outputs $(N_{\mathcal{G}}, k_{\mathcal{G}}, \vec{\ell}_{\mathcal{G}})$ graphs, if for all possible inputs x, the distribution $\mathcal{G}(x)$ is supported on $N_{\mathcal{G}}$-vertex layered graphs with $k_{\mathcal{G}}$ layers and edge-layer ordering $\vec{\ell}_{\mathcal{G}}$.

Remark 3.8. *The property that $\mathcal{G}$ always outputs $(N_{\mathcal{G}}, k_{\mathcal{G}}, \vec{\ell}_{\mathcal{G}})$ graphs for some triple $(N_{\mathcal{G}}, k_{\mathcal{G}}, \vec{\ell}_{\mathcal{G}})$ is pretty strong since it forces $\mathcal{G}$ to always output graphs with the same number of vertices, the same number of layers and the same edge-layer ordering. We remark here that all our constructions in this paper have this property.*

For simplicity, we will often use $\mathcal{G}_n^{\mathsf{SH}}$ (resp. $\mathcal{G}_n^{\mathsf{PH}}$) to denote an ε-Set-Hiding (resp. ε-Perm-Hiding) generator $\mathcal{G}$ for subsets of $[n]$ (resp. permutations in $\mathsf{Perm}([n])$). We may also write $\mathcal{G}_{n,p}^{\mathsf{SH}}$ (resp. $\mathcal{G}_{n,p}^{\mathsf{PH}}$) to indicate that the generator is against p-pass streaming algorithms.

4 Construction of Set-Hiding Generators

In this section, we will give a construction of the Set-Hiding generators, which summarizes some key technical lemmas that will be proved in the later sections.

Now we are ready to state the main theorem of this section.

Theorem 4.1 (Main Theorem)**.** *There is a constant $c > 1$ and an integer $N_0 \in \mathbb{N}$ such that for every $p \in \mathbb{N}$ and every integer n satisfying $n \geq N_0$ and $p \leq c^{-1} \cdot \sqrt{\log n}$, there is a generator $\mathcal{G}^{\mathsf{SH}}_{n,p}$ such that: (1) it always outputs $(N_{\mathcal{G}^{\mathsf{SH}}}, k_{\mathcal{G}^{\mathsf{SH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{SH}}})$ graphs, where $N_{\mathcal{G}^{\mathsf{SH}}} \leq c \cdot n^{1+cp/\sqrt{\log n}}$ and $k_{\mathcal{G}^{\mathsf{SH}}} \leq c \cdot (c \log n)^{2p}$; (2) it is (n^{-1})-*Set-Hiding *against p-pass streaming algorithms with space n^2.*

We remark that Theorem 4.1 is all we need to prove the lower bounds for streaming algorithms in Section 5. The rest of this section is a proof of Theorem 4.1, with key technical lemmas proved in later sections.

Overview of the construction. Our construction works recursively. The base case will be generators against 0-pass streaming algorithms. Clearly, trivial constructions suffice for this base case since 0-pass streaming algorithms cannot read the input at all.

Next, for the case against p-pass streaming algorithms, Lemma 4.2 shows how to construct Set-Hiding generators against p-pass streaming algorithms from Perm-Hiding generators for $(p-1)$-pass streaming algorithms, and Lemma 4.3 shows how to construct Perm-Hiding generators against p-pass streaming algorithms from Set-Hiding generators for p-pass streaming algorithms. The formal proofs of Lemma 4.2 and Lemma 4.3 can be found in Section 9 and Section 10, respectively.

Lemma 4.2 (From Perm-Hiding generators to Set-Hiding generators)**.** *Let n be a sufficiently large integer. Let $p, s \in \mathbb{N}$ such that $2p \cdot s \leq \frac{1}{20} n^2 \log n$, and let $N = N^{\mathsf{RS}}(4n)$. Let $\varepsilon \in [0, 1)$ such that $\varepsilon \geq 1/n^{10}$. Suppose there is a generator $\mathcal{G}^{\mathsf{PH}}_{N,p-1}$ which always outputs $(N_{\mathcal{G}^{\mathsf{PH}}}, k_{\mathcal{G}^{\mathsf{PH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{PH}}})$ graphs and is $(\varepsilon/\log^2 n)$-*Perm-Hiding *against $(p-1)$-pass streaming algorithms with space $2p \cdot s$. Then there is a generator $\mathcal{G}^{\mathsf{SH}}_{n,p}$ such that: (1) it always outputs $(N_{\mathcal{G}^{\mathsf{SH}}}, k_{\mathcal{G}^{\mathsf{SH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{SH}}})$ graphs, where $N_{\mathcal{G}^{\mathsf{SH}}} = O(N_{\mathcal{G}^{\mathsf{PH}}} \cdot \log^2 n)$ and $k_{\mathcal{G}^{\mathsf{SH}}} = O(k_{\mathcal{G}^{\mathsf{PH}}} \cdot \log n)$; (2) it is ε-*Set-Hiding *against p-pass streaming algorithms with space s.*

Lemma 4.3 (From Set-Hiding generators to Perm-Hiding generators)**.** *Let n be a sufficiently large integer. Let $s \in \mathbb{N}$ and let $\varepsilon \in [0, 1)$. Suppose there is a generator $\mathcal{G}^{\mathsf{SH}}_{3n,p}$ which always outputs $(N_{\mathcal{G}^{\mathsf{SH}}}, k_{\mathcal{G}^{\mathsf{SH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{SH}}})$ graphs and is $(\varepsilon/\log^2 n)$-*Set-Hiding *against p-pass streaming algorithms with space s. Then there is a generator such that: (1) it always outputs $(N_{\mathcal{G}^{\mathsf{PH}}}, k_{\mathcal{G}^{\mathsf{PH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{PH}}})$ graphs where $N_{\mathcal{G}^{\mathsf{PH}}} = O(N_{\mathcal{G}^{\mathsf{SH}}} \cdot \log n)$ and $k_{\mathcal{G}^{\mathsf{PH}}} = O(k_{\mathcal{G}^{\mathsf{SH}}} \cdot \log n)$; (2) it is ε-*Perm-Hiding *against p-pass streaming algorithms with space s.*

Finally, Theorem 4.1 follows by applying Lemma 4.2 and Lemma 4.3 repeatedly.

Proof of Theorem 4.1. Let N_0 be a sufficiently large constant to be specified later. We will set N_0 so that Lemma 4.2 and Lemma 4.3 holds for all integers $n \geq N_0$. Let $c \geq 2$ be a sufficiently large constant to be specified later.

We will prove the theorem by induction on p. The theorem trivially holds when $p = 0$: since 0-pass streaming algorithm cannot read anything from the input, given an input subset S, one can simply output a bipartite graph of size (n, n) such that the i-th vertex on the left side is connected to the i-th vertex on the right side if and only if $i \in S$. Clearly, this output is a $\mathsf{Set\text{-}Enc}_n(S)$ graph.

Now, suppose the theorem holds for $p-1$, we show it holds for p as well. We fix an $n \ge N_0$ such that $p \le c^{-1} \cdot \sqrt{\log n}$, and we will show how to construct the desired generator $\mathcal{G}^{\mathsf{SH}}_{n,p}$. Let $n_2 = N^{\mathsf{RS}}(4n)$ and $n_1 = 3n_2$. We proceed as follows:

1. Since $n_1 \ge n \ge N_0$, it follows that $(p-1) \le c^{-1} \cdot \sqrt{\log n_1}$. Hence, by the induction hypothesis, there is a generator $\mathcal{G}^{\mathsf{SH}}_{n_1,p-1}$ such that: (1) it always outputs $(N_{(1)}, k_{(1)}, \vec{\ell}_{(1)})$ graphs, where $N_{(1)} \le c \cdot n_1^{1+c(p-1)/\sqrt{\log n_1}}$ and $k_{(1)} \le c \cdot (c \log n_1)^{2(p-1)}$; (2) it is $\varepsilon_{(1)}$-$\mathsf{Set\text{-}Hiding}$ against $(p-1)$-pass streaming algorithms with space n_1^2, where $\varepsilon_{(1)} = 1/n_1$.

2. Since $n_2 \ge N_0$ and $n_1 = 3n_2$, combining Lemma 4.3 with the generator $\mathcal{G}^{\mathsf{SH}}_{n_1,p-1}$, there is a generator $\mathcal{G}^{\mathsf{PH}}_{n_2,p-1}$ such that: (1) it always outputs $(N_{(2)}, k_{(2)}, \vec{\ell}_{(2)})$ graphs where $N_{(2)} \le O(N_{(1)} \cdot \log n_2)$ and $k_{(2)} \le O(k_{(1)} \cdot \log n_2)$; (2) it is $\varepsilon_{(2)}$-$\mathsf{Perm\text{-}Hiding}$ against $(p-1)$-pass streaming algorithms with space n_1^2, where $\varepsilon_{(2)} = \varepsilon_{(1)} \cdot (\log n_2)^2$.

3. Since $n \ge N_0$ and $n_2 = N^{\mathsf{RS}}(4n)$, combining Lemma 4.2 with the generator $\mathcal{G}^{\mathsf{PH}}_{n_2,p-1}$ and the fact that $2p \cdot n^2 \le \frac{1}{20} n^2 \log n$ and $n_1^2 \ge 2p \cdot n^2$, there is a generator $\mathcal{G}^{\mathsf{SH}}_{n,p}$ such that: (1) it always outputs $(N_{(3)}, k_{(3)}, \vec{\ell}_{(3)})$ graphs, where $N_{(3)} \le O(N_{(2)} \cdot \log^2 n)$ and $k_{(3)} \le O(k_{(2)} \cdot \log n)$; (2) it is $\varepsilon_{(3)}$-$\mathsf{Set\text{-}Hiding}$ against p-pass streaming algorithms with space n^2, where $\varepsilon_{(3)} = \varepsilon_{(2)} \cdot (\log n)^2$.

Now we verify that the last generator $\mathcal{G}^{\mathsf{SH}}_{n,p}$ satisfies our requirements. Noting that $\log n_2 = O(\log n)$, it follows that

$$N_{(3)} \le O\left(c \cdot \log^3 n \cdot n_1^{1+c(p-1)/\sqrt{\log n_1}}\right). \tag{1}$$

Setting N_0 to be sufficiently large, we have $n_1 = 3 \cdot N^{\mathsf{RS}}(4n) \le n^{1+2c^{\mathsf{RS}}/\sqrt{\log n}}$, and hence

$$\log n_1 \le \log n + 2c^{\mathsf{RS}} \cdot \sqrt{\log n}. \tag{2}$$

Taking log of both sides of Equation 1, it follows that

$$\log N_{(3)} \le O(1) + 3 \log\log n + \log n_1 + c(p-1) \cdot \sqrt{\log n_1}. \tag{3}$$

Setting N_0 to be sufficiently large and noting that $\sqrt{x + 2c^{\mathsf{RS}}\sqrt{x}} \le \sqrt{x} + 2c^{\mathsf{RS}}$ for any $x > 0$, it follows from Equation 2 that

$$\sqrt{\log n_1} \le \sqrt{\log n} + 2c^{\mathsf{RS}}. \tag{4}$$

Plugging Equation 2 and Equation 4 in Equation 3 and setting c to be sufficiently large, we have

$$\begin{aligned}
\log N_{(3)} &\le O(1) + 3\log\log n + \log n + 2c^{\mathsf{RS}} \cdot \sqrt{\log n} + c(p-1)\cdot(\sqrt{\log n} + 2c^{\mathsf{RS}}) \\
&\le \log n + (3c^{\mathsf{RS}} + c(p-1)) \cdot \sqrt{\log n} + c(p-1)\cdot 2c^{\mathsf{RS}} \\
&\le \log n + (5c^{\mathsf{RS}} + c(p-1)) \cdot \sqrt{\log n} && (cp \le \sqrt{\log n}) \\
&\le \log n + cp \cdot \sqrt{\log n}. && (c \text{ is sufficiently large})
\end{aligned}$$

Noting that $\log n_2 = O(\log n)$ and setting c to be sufficiently large, it follows that

$$\begin{aligned}
k_{(3)} \le O(\log^2 n \cdot k_{(1)}) &\le (c/10\log^2 n)\cdot c\cdot(c\log n_1)^{2(p-1)} \\
&\le (c/10\log^2 n)\cdot c\cdot(c\log n)^{2(p-1)} \cdot \left(1 + \frac{2c^{\mathsf{RS}}}{\sqrt{\log n}}\right)^{2(p-1)} && (\text{Equation 2}) \\
&\le c\cdot(c\log n)^{2p}. && (p \le c^{-1}\sqrt{\log n} \text{ and } c \text{ is sufficiently large})
\end{aligned}$$

Finally, since $n_1 = 4N^{\mathsf{RS}}(4n) \ge n^{1+\Omega(1/\sqrt{\log n})}$, setting N_0 to be sufficiently large, we also have $\varepsilon_{(3)} = 1/n_1 \cdot (\log n_2)^2 \cdot (\log n)^2 \le 1/n$. This completes the proof. □

Finally, we remark that if we use Remark 9.10 in place of Lemma 4.2 and proceed similarly as in the proof of Theorem 4.1, we have the following different construction of Set-Hiding generators instead.

Remark 4.4. *There is an absolute constant $c \in (0,1)$ such that, for every $p(n) = o(\log n/\log\log n)$, for every sufficiently large integer n, there is a generator $\mathcal{G}^{\mathsf{SH}}_{n,p(n)}$ such that: (1) it always outputs $(N_{\mathcal{G}^{\mathsf{SH}}}, k_{\mathcal{G}^{\mathsf{SH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{SH}}})$ graphs, where $N_{\mathcal{G}^{\mathsf{SH}}} \le n \cdot (\log n)^{O(p(n))}$; (2) it is (n^{-1})-*Set-Hiding *against p-pass streaming algorithms with space $n^{1+c/\log\log n}$.*

5 Lower Bounds for Multi-Pass Streaming Algorithms

In this section, we show that Theorem 4.1 implies our lower bounds for multi-pass streaming algorithms.

- In Section 5.1, we prove the lower bounds for s-t reachability and s-t undirected shortest-path.
- In Section 5.2, we prove our lower bounds for (approximate) bipartite perfect matching.
- In Section 5.3, we prove our lower bounds for estimating the rank of a matrix.
- In Section 5.4, we prove our lower bounds for linear programing in the row-streaming model.

5.1 s-t Reachability and s-t Undirected Shortest-Path

As already discussed in Section 2, Theorem 4.1 directly implies the following lower bounds for s-t reachability and s-t undirected shortest-path against multi-pass streaming algorithms.

Theorem 5.1 (Detailed version of Theorem 1.1 and Theorem 1.2)**.** *The following statements hold.*

1. *Given an n-vertex directed graph $G = (V, E)$ with two designated vertices $s, t \in V$, no randomized $o(\sqrt{\log n})$-pass streaming algorithm with space $n^{2-\varepsilon}$ for some constant $\varepsilon > 0$ can determine whether s can reach t in G with probability at least $2/3$.*

2. *Given an n-vertex* undirected *graph $G = (V, E)$ and two designated vertices $s, t \in V$, no randomized $o(\sqrt{\log n})$-pass streaming algorithm with space $n^{2-\varepsilon}$ for some constant $\varepsilon > 0$ can output the length of the shortest s-t path in G.*

   Moreover, for $p(n)$-pass streaming algorithms where $p(n) = o(\sqrt{\log n})$, the lower bound above for s-t undirected shortest-path still holds if the algorithm is only required to compute an $(1+\omega(\log n)^{-2p(n^2)})$-approximation to the length of the shortest path between s and t.

Proof. We first prove the theorem for s-t reachability, and then show how to adapt the proof for s-t undirected shortest-path.

Lower bounds for s-t reachability. Suppose for the sake of contradiction that there is $p(n) \le o(\sqrt{\log n})$ and a constant $\varepsilon > 0$ such that there is a $p(n)$-pass streaming algorithm A_{stReach} with $n^{2-\varepsilon}$ space, which solves s-t reachability for n-vertex graphs with probability at least $2/3$. We further assume that A_{stReach} outputs 1 if it determines that s can reach t, and 0 otherwise.

By Theorem 4.1 and noting that $p(n^2) \le o(\sqrt{\log n})$, there is $m(n) = n^{1+o(1)}$ such that for every sufficiently large n, there is a generator $\mathcal{G}^{\mathsf{SH}}_{n,p(n^2)}$ which always outputs $(m(n), k_{\mathcal{G}^{\mathsf{SH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{SH}}})$ graphs and is $(1/10)$-$\mathsf{Set\text{-}Hiding}$ for subsets of $[n]$ against $p(n^2)$-pass streaming algorithms with space n^2. For a layered graph G in the support of $\mathcal{G}^{\mathsf{SH}}_{n,p}(\emptyset)$ or $\mathcal{G}^{\mathsf{SH}}_{n,p}(\{1\})$, we set $s = \mathsf{First}(G)_{[1]}$ and $t = \mathsf{Last}(G)_{[1]}$ (s and t do not depend on the choice of graph G).

Since A_{stReach} solves s-t reachability with probability at least $2/3$, it follows that

$$\Pr_{G \leftarrow \mathcal{G}^{\mathsf{SH}}_{n,p}(\{1\})}[A_{\mathsf{stReach}}(G) = 1] \ge 2/3 \quad \text{and} \quad \Pr_{G \leftarrow \mathcal{G}^{\mathsf{SH}}_{n,p}(\emptyset)}[A_{\mathsf{stReach}}(G) = 0] \ge 2/3.$$

The above means that $\|A_{\mathsf{stReach}}(\mathcal{G}^{\mathsf{SH}}_{n,p}(\emptyset)) - A_{\mathsf{stReach}}(\mathcal{G}^{\mathsf{SH}}_{n,p}(\{1\}))\|_{\mathrm{TV}} \ge 1/3$. This contradicts the fact that $\mathcal{G}^{\mathsf{SH}}_{n,p(n^2)}$ is $(1/10)$-$\mathsf{Set\text{-}Hiding}$ against $p(n^2)$-pass algorithms with space n^2, since A_{stReach} takes $p(m) \le p(n^2)$ passes and $m^{2-\varepsilon} \le n^2$ space.

Lower bounds for s-t undirected shortest-path. We will use the same reduction in [AR20, Theorem 6]. Again suppose for the sake of contradiction that there is a $p(n)$-pass streaming algorithm A_{stUpath} with $n^{2-\varepsilon}$ space, which solves s-t undirected shortest-path for n-vertex graphs with probability at least $2/3$.

Recall that $\mathcal{G}^{\mathsf{SH}}_{n,p(n^2)}$ always outputs graphs with exactly $k_{\mathcal{G}^{\mathsf{SH}}}$ layers. Let $\overline{G}$ be the undirected version of the layered graph G in the support of $\mathcal{G}^{\mathsf{SH}}_{n,p}(\emptyset)$ or $\mathcal{G}^{\mathsf{SH}}_{n,p}(\{1\})$ (that is, $\overline{G}$ is obtained by removing the directions on all edges of G), we claim that s can reach t in G if and only if the shortest path between s and t in $\overline{G}$ has length exactly $k_{\mathcal{G}^{\mathsf{SH}}} - 1$.

To see the claim above, note that (1) the shortest path between s and t in $\overline{G}$ has length at least $k_{\mathcal{G}^{\mathsf{SH}}} - 1$, since there are $k_{\mathcal{G}^{\mathsf{SH}}}$ layers in $\overline{G}$; (2) if s can reach t in G, then the same path gives us a $(k_{\mathcal{G}^{\mathsf{SH}}} - 1)$-length path from s to t in $\overline{G}$, and vice versa. Therefore, A_{stUpath} can be similarly used to distinguish the distributions $\mathcal{G}^{\mathsf{SH}}_{n,p}(\emptyset)$ and $\mathcal{G}^{\mathsf{SH}}_{n,p}(\{1\})$. Applying a similar argument as in the case of s-t reachability gives us the desired lower bound for s-t undirected shortest-path.

Finally, A_{stUpath} is in fact only required to distinguish between (1) the shortest path between s and t in $\overline{G}$ has length exactly $k_{\mathcal{G}^{\mathsf{SH}}} - 1$ and (2) the shortest path between s and t in $\overline{G}$ has length at least $k_{\mathcal{G}^{\mathsf{SH}}}$. By Theorem 4.1 it holds that $k_{\mathcal{G}^{\mathsf{SH}}} \le O(\log n)^{2p(n^2)}$. Hence, it suffices for A_{stUpath} to compute a $(1 + (k_{\mathcal{G}^{\mathsf{SH}}})^{-1})$ approximation to the shortest path between s and t in G, and the theorem is proved by noting $\omega(\log m(n))^{2p(n^2)} \ge k_{\mathcal{G}^{\mathsf{SH}}}$ (recall that G has $m(n)$ vertices). □

Remark 5.2. *If we apply Remark 4.4 instead of Theorem 4.1 in the proof of Theorem 5.1, then it follows that s-t reachability or s-t undirected shortest-path cannot be solved by $o(\log n/(\log\log n)^2)$-pass streaming algorithms with $n^{1+o(1/\log\log n)}$ space.*

Proof sketch. We will just sketch the proof for s-t reachability here. The proof for s-t undirected shortest-path is identical. Suppose for the sake of contradiction that there is $p(n) \le o(\log n/(\log\log n)^2)$ and $s(n) = n^{1+o(1/\log\log n)}$ such that there is a $p(n)$-pass streaming algorithm A_{stReach} with $s(n)$ space, which solves s-t reachability with probability at least $2/3$.

Let $c \in (0,1)$ be the absolute constant in Remark 4.4. By Remark 4.4 and noting $p(n^2) \le o(\log n/\log\log n)$, there is $m(n) = n^{1+o(1/\log\log n)}$ such that for every sufficiently large n, there is a generator $\mathcal{G}^{\mathsf{SH}}_{n,p(n^2)}$ which always outputs $m(n)$-vertex graphs and is $(1/10)$-$\mathsf{Set\text{-}Hiding}$ against $p(n^2)$-pass streaming algorithms with space $n^{1+c/\log\log n}$. Noting that $p(m(n)) \le p(n^2)$ and $s(m(n)) \le n^{1+c/\log\log n}$ and applying the same argument as in Theorem 5.1, we can use A_{stReach} to break the generator $\mathcal{G}^{\mathsf{SH}}_{n,p(n^2)}$, which finishes the proof. □

5.2 Bipartite Perfect Matching

Next we turn to prove our lower bounds for bipartite perfect matching against multi-pass streaming algorithms. In this subsection we will focus on the edge-arrival setting for streaming algorithms, since we will consider bipartite graphs, which only has a single edge-layer.

Theorem 5.3 (Detailed version of Theorem 1.3)**.** *No $o(\sqrt{\log n})$-pass streaming algorithm with $n^{2-\varepsilon}$ space for some $\varepsilon > 0$ can determine whether a bipartite graph $G = (L \sqcup R, E)$ with $|L| = |R| = n$ has a perfect matching with probability at least $2/3$.*

Moreover, for $p(n)$-pass streaming algorithms where $p(n) = o(\sqrt{\log n})$, the lower bound above still holds if the algorithm is only required to distinguish with probability at least $2/3$ between (1) G has a perfect matching of size n and (2) G has no matching of size at least $n \cdot (1 - \delta(n))$, for some $\delta(n) = 2^{-cp(n^2)/\sqrt{\log n}}$, where $c > 1$ is an absolute constant.

Proof. We will adapt a folklore reduction from reachability to perfect matching, which is also used in [AR20, Theorem 5].

Suppose for the sake of contradiction that there is $p(n) \le o(\sqrt{\log n})$ and a constant $\varepsilon > 0$ such that there is a $p(n)$-pass streaming algorithm A_{matching} with $n^{2-\varepsilon}$ space, which determines whether a bipartite graph has a perfect matching or not with probability at least $2/3$. By Theorem 4.1 and noting that $p(n^2) \le o(\sqrt{\log n})$, there is $m(n) = n^{1+o(1)}$ and such that for every sufficiently large n, there is a generator $\mathcal{G}^{\mathsf{SH}}_{n,p(n^2)}$ which always outputs $(m(n), k_{\mathcal{G}^{\mathsf{SH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{SH}}})$ graphs and is $(1/10)$-$\mathsf{Set\text{-}Hiding}$ for subsets of $[n]$ against $p(n^2)$-pass streaming algorithms with space n^2.

For a layered graph G in the support of $\mathcal{G}^{\mathsf{SH}}_{n,p(n^2)}(\emptyset)$ or $\mathcal{G}^{\mathsf{SH}}_{n,p(n^2)}([n])$, let $V_{\mathsf{mid}} = \bigcup_{i=2}^{k_{\mathcal{G}^{\mathsf{SH}}}-1} V_i(G)$. That is, V_{mid} is the set of vertices in the middle layers of G. We will construct a bipartite graph $H = (L \sqcup R, E_H)$[8] as follows:

Bipartite Perfect Matching from Set-Hiding Generators

1. For every vertex $v \in V_{\mathsf{mid}}$, we add a vertex v^ℓ to L and a vertex v^r to R. For every vertex $s \in \mathsf{First}(G)$, we add a vertex s^ℓ to L. For every vertex $t \in \mathsf{Last}(G)$, we add a vertex t^r to R.

2. Next we enumerate all the (directed) edges (u, v) in G according to their ordering in $\vec{E}(G)$, with ties broken according the lexicographically order[a]: for each edge $(u, v) \in E(G)$, we add an edge (u^ℓ, v^r) to E_H. (Note that vertices in $\mathsf{First}(G)$ has no incoming edges, and vertices in $\mathsf{Last}(G)$ has no outgoing ones.) For every vertex $v \in V_{\mathsf{mid}}$, we also add an edge (v^ℓ, v^r) to E_H.

[a]That is, we first enumerate edges in $E_1(G)$ in lexicographical order, then edges in $E_2(G)$ and so on.

From the construction above, one can verify easily that $|L| = |R| = |V_{\mathsf{mid}}| + n = m(n) - n$. Let $n_H = |L|$. The following claim is crucial for the proof.

Claim 5.4.

[8]E_H here is a list of edges, which specify the order that the streaming algorithms read the graph.

1. *If G is a $\mathsf{Set\text{-}Enc}_n([n])$ graph, then H has a perfect matching of size n_H.*

2. *if G is a $\mathsf{Set\text{-}Enc}_n(\emptyset)$ graph, then all matchings in G have at most $|V_{\mathsf{mid}}| = n_H - n$ edges.*

To see Item (1) of Claim 5.4, consider the matching $M = \{(v^\ell, v^r) : v \in V_{\mathsf{mid}}\}$. Note that $|M| = |V_{\mathsf{mid}}| = n_H - n$, and the only unmatched vertices are s^ℓ and t^r for $s \in \mathsf{First}(G)$ and $t \in \mathsf{Last}(G)$. For every $s \in \mathsf{First}(G)$ and $t \in \mathsf{Last}(G)$, any *augmenting path* of this matching M in H between s^ℓ and t^r corresponds to a directed path from s to t in G. If G is a $\mathsf{Set\text{-}Enc}_n([n])$ graph, we can find $|\mathsf{First}(G)|$ disjoint augmenting paths, in which the i-th path is from $\mathsf{First}(G)_{[i]}$ to $\mathsf{Last}(G)_{[i]}$.[9] This means that H has a matching of size $|M| + n = n_H$, which is a perfect matching.

For Item (2) of Claim 5.4, if G is a $\mathsf{Set\text{-}Enc}_n(\emptyset)$ graph, then no augmenting path between the unmatched vertices s^ℓ and t^r can be found, since for every $(i,j) \in [n] \times [n]$, $\mathsf{First}(G)_{[i]}$ cannot reach $\mathsf{First}(G)_{[j]}$. Hence, M is a maximum matching of H.

Note that H can be generated "on the fly" in the streaming setting. Hence, since A_{matching} takes $p(|L|) \le p(n^2)$ passes and $(|L|)^{2-\varepsilon} \le m(n)^{2-\varepsilon} \le n^2$ space. By Claim 5.4, A_{matching} can be used to distinguish between the distributions $\mathcal{G}^{\mathsf{SH}}_{n,p(n^2)}(\emptyset)$ and $\mathcal{G}^{\mathsf{SH}}_{n,p(n^2)}([n])$, contradicts the security of the generator. This proves the first part of the theorem.

Finally, note that A_{matching} is indeed only required to distinguish between (1) H has perfect matching of size $|L| = m(n) - n$ and (2) H has no matching of size greater than $|V_{\mathsf{mid}}| = m(n) - 2n$. By Theorem 4.1, we have $m(n) \le c \cdot n^{1+cp(n^2)/\sqrt{\log n}}$ for some constant $c > 1$, the second part of the theorem then follows.

□

Remark 5.5. *Consider the edge list E_H constructed in the proof of Theorem 5.3, for every left vertex $u^\ell \in L$, its adjacent edges (u^ℓ, v^r) are listed consecutively in E_H. (Except for the edge (u^ℓ, v^ℓ), which is an auxiliary edge that does not depend on the given graph G.)*

5.3 Matrix Rank

Estimating the rank of an $n \times n$ matrix is an important problem in the streaming setting. There has been several results studying this problem, and we refer the readers to [BS15, LW16, AKL17, AKSY20] for details. In the following we present a lower bound for the rank estimation problem.

The following corollary follows from Theorem 5.3 and the well-known reduction from computing the size of maximum matching for bipartite graphs to computing the rank of matrices (see, *e.g.*, [MR95, Page 167] and [LP09]), we will consider the row streaming model

[9] By the definition of a $\mathsf{Set\text{-}Enc}_n([n])$ graph, for each $i \in [n]$, there exists a directed path P_i from $\mathsf{First}(G)_{[i]}$ to $\mathsf{Last}(G)_{[i]}$ in G. We further observe that these n paths are vertex-disjoint. Since otherwise, if P_i shares a vertex v with P_j for $i \neq j$, then it means that $\mathsf{First}([G])_{[i]}$ can first reach v and then reach $\mathsf{Last}(G)_{[j]}$, which contradicts the definition of $\mathsf{Set\text{-}Enc}_n([n])$ graphs.

in which the streaming algorithms get the rows of the matrix one by one in some arbitrary order.

Corollary 5.6 (Detailed version of Theorem 1.4)**.** *No $o(\sqrt{\log n})$-pass streaming algorithm with $n^{2-\varepsilon}$ space for some $\varepsilon > 0$ can determine whether a given matrix $M \in \mathbb{F}_q^{n\times n}$ for some prime power $q = \omega(n)$ has full rank with probability at least $2/3$.*

Moreover, for $p(n)$-pass streaming algorithms where $p(n) = o(\sqrt{\log n})$, the lower bound above still holds if the algorithm is only required to distinguish with probability at least $2/3$ between (1) $\mathsf{rank}(M) = n$ and (2) $\mathsf{rank}(M) \le n \cdot (1 - \delta(n))$, for some $\delta(n) = 2^{-cp(n^2)/\sqrt{\log n}}$, where $c > 1$ is an absolute constant.

Proof. For a bipartite graph $G = (L \sqcup R, E)$ where $L = \{u_1, \dots, u_n\}$ and $R = \{v_1, \dots, v_n\}$, we consider the *Edmonds matrix* M defined over the variables $\vec{x} = (x_{i,j})_{(i,j)\in[n]\times[n]}$:

$$M(i,j) := \begin{cases} x_{i,j} & \text{if } (u_i, v_j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Let $q = \omega(n)$ be a prime power. We know that $\mathsf{rank}(M)$ (rank of M over the polynomial ring $Z[\vec{x}] = Z[x_{1,1}, \dots, x_{n,n}]$) equals the size of the maximum matching in H. For a vector $\vec{r} = (r_{i,j})_{(i,j)\in[n]\times[n]} \in \mathbb{F}_q^{n\times n}$, we use $M(\vec{r})$ to denote the matrix over $\mathbb{F}_q$ obtained by substituting $x_{i,j}$ by $r_{i,j}$ for each $(i,j) \in [n] \times [n]$. Applying the Schwartz-Zippel lemma, it holds that if all the $r_{i,j}$ are i.i.d. uniform distributed over $\mathbb{F}_q$, then $\Pr_{\vec{r}}[\mathsf{rank}(M(\vec{r})) = \mathsf{rank}(M)] \ge 1 - o(1)$.

Therefore, computing the size of the maximum matching in G can be reduced to computing the rank of $M(\vec{r})$ over $\mathbb{F}_q$. The proof is then finished by combing Theorem 5.3 and Remark 5.5. $\square$

5.4 Linear Programming

Linear programming (LP) is a fundamental problem in optimization. The fastest LP solver for general dense matrix is due to [JSWZ20]. It takes n^ω time with $O(n^2)$ space, where $\omega \approx 2.37286$ is the exponent of current matrix multiplication [AW21]. For the situation where LP has roughly n constraints/variables, it is not known how to extend the classical result [JSWZ20] into streaming setting with $o(n)$ passes and $o(n^2)$ space. For matrix related problems such as linear regression and low-rank approximation [CW09, CW13, NN13, BWZ16, SWZ17], there are two natural streaming models: the row model and the entry model. In the following we will focus on the row model, which is discussed below.

The row streaming model for LP. Our lower bounds holds for the feasibility of LP in the row streaming model: in which the streaming algorithms get the constraints of the LP one by one in some arbitrary order, and is required to decide whether all the constraints can be simultaneously satisfied. Note that algorithms in the entry streaming model for LP also work in the row streaming model, hence our lower bounds hold for the entry streaming model as well.

Theorem 5.7. *No $o(\sqrt{\log n})$-pass algorithm with $n^{2-\varepsilon}$ space for some $\varepsilon > 0$ in the row streaming model can determine if a linear program of n variables and n constraints with coefficients in $\{0, 1\}$ is feasible or not.*

Proof. We will show a reduction from s-t reachability over layered graphs to the feasibility of an LP instance. Consider a layered graph G with n vertices and let s and t be two vertices in G. We construct the following linear program P with n variables:

LP from s-t Reachability

1. **Variables:** For each vertex v of G, we add a variable x_v to P. Note that as in the standard formulation of LP, all x_v are non-negative.
2. **Constraints:** For each vertex v of G such that $v \neq s$, we add a constraint
$$x_v \geq \sum_{u:\ \text{edge } (u,v) \in E(G)} x_u$$
to P.

   We also add two constraints $x_t \leq 0$ and $x_s \geq 1$ to P.
3. **Constraints Ordering:** The constraints $x_t \leq 0$ and $x_s \geq 1$ come first. Note that each of the other constraints correspond to one vertex v and all its incoming edges, which are all in the same edge-layer of G. We say that this edge-layer is the corresponding edge-layer of that constraint. Then we list all the other constraints in the ordering of their corresponding edge-layers in $\vec{\ell}(G)$ (with ties broken arbitrarily).

Clearly, one can see that if s can reach t in G, then $x_s \geq 1$ implies $x_t \geq 1$ as well, and the LP instance P is not feasible. On the other hand, if s cannot reach t in G, then one can construct an assignment to all variables x so that for every vertex v which is not reachable from s, x_v is set to 0. Hence, $x_t = 0$ as well and P is then feasible.

Finally, since the LP instance can be generated "on the fly", an algorithm deciding whether P is feasible in the row streaming model also implies a streaming algorithm deciding whether s can reach t in G in the layer-arrival model. The proof is then completed by applying Theorem 5.1. □

6 The Communication Lower Bound

In this section, we define the $\mathsf{Ind}^{\oplus}$ problem. We directly define a distributional version of the $\mathsf{Ind}^{\oplus}$ problem parametrized by integers $n, t, K > 0$. Our distribution is tailored to make the reductions in the subsequent sections easier.

Hard Input Distribution $\mathcal{D}^{\mathsf{Ind}^{\oplus}}_{n,t,K}$ for the (n, t, K)-$\mathsf{Ind}^{\oplus}$ problem

- **Construction:**
  - Draw K indices $\vec{i} = (i_k)_{k\in[K]}$ from the set $[2n]$ uniformly at random.
  - Draw K indices $\vec{j} = (j_k)_{k\in[K]}$ from the set $[t]$ uniformly at random.
  - Draw $t \cdot K$ vectors $\vec{X} = (X_{k,j})_{k\in[K],j\in[t]}$ uniformly at random from $\{0,1\}^{2n}$ conditioned on $\|X_{k,j}\|_1 = n$ for all $k \in [K]$ and $j \in [t]$.
- **Alice's Input:** The vectors $\vec{X}$.
- **Bob's Input:** The indices $\vec{j}$ and $\vec{i}$.
- **Problem:** Output $\prod_{k=1}^{K}(-1)^{X_{k,j_k,i_k}}$.

The $\mathsf{Ind}^{\oplus}$ is hard for one-way communication protocols from Alice to Bob. Formally, we prove the following theorem.

Theorem 6.1 (Communication complexity of $\mathsf{Ind}^{\oplus}$)**.** *Let $n, t, K > 1000$ be integers. Define $\delta = \frac{100}{K}$ and C to be the largest even integer at most $\frac{1}{10} \cdot (2nt)^{1-5\delta}$ and assume that $tK \leq 10C$.*

For all one-way randomized protocols Π from Alice to Bob for the (n, t, K)-$\mathsf{Ind}^{\oplus}$ problem satisfying $\mathsf{CC}(\Pi) \leq C \cdot \log n$, we have:[10]

$$\Pr_{(\vec{X},\vec{j},\vec{i})\sim\mathcal{D}^{\mathsf{Ind}^{\oplus}}_{n,t,K}}\left(\Pi(\vec{X},\vec{j},\vec{i}) = \prod_{k=1}^{K}(-1)^{X_{k,j_k,i_k}}\right) \leq \frac{1}{2} + \frac{1}{(2nt)^{20}}.$$

The proof Theorem 6.1 spans the rest of this section. Fix n, t, K, δ, C as in the theorem statement. The main idea in the proof is to upper bound the discrepancy of the 'communication matrix' $\mathcal{M}$ of the $\mathsf{Ind}^{\oplus}$ problem, *i.e.*, the matrix whose rows are indexed by Alice's inputs $\vec{X}$ and columns are indexed by Bob's inputs $(\vec{j},\vec{i})$ and the value at row $\vec{X}$ and columns $(\vec{j},\vec{i})$ is the just the "correct" output when Alice's input is $\vec{X}$ and Bob's input is $(\vec{j},\vec{i})$. For convenience sake, we shall actually consider $\mathcal{M}$ to be a matrix with rows indexed by the set $\mathcal{Z} = \left((\{0,1\}^{2n})^t\right)^K$, even though some elements in $\mathcal{Z}$ can never be an input for Alice in our hard distribution.

[10]The notation $\Pi(\vec{X},\vec{j},\vec{i})$ denotes the distribution of the output of the protocol Π when run with the inputs $(\vec{X},\vec{j},\vec{i})$.

One standard way of upper bounding the discrepancy of $\mathcal{M}$ is to upper bound the spectral norm of $\mathcal{M}^\top$. This may seem promising at first, as the spectral norm of the matrix $\mathcal{M}^\top$ can be computed exactly, and equals $\sqrt{|\mathcal{Z}|}$. Indeed, as the entries of $\mathcal{M}$ lie in $\{-1, 1\}$, each row of $\mathcal{M}^\top$ has entries in $\{-1, 1\}$, implying that the ℓ_2-norm of each row is $\sqrt{|\mathcal{Z}|}$. Using the observation that the rows of $\mathcal{M}^\top$ are orthogonal, we can conclude that the spectral norm of $\mathcal{M}^\top$ is $\mathcal{M}$ as well. However, this computation of the spectral norm of $\mathcal{M}^\top$ gives a rather weak bound on the discrepancy of $\mathcal{M}$, and we need to take a different approach.

Our approach uses the observation that in order to get a bound on the discrepancy, we only need to bound $\left\|\mathcal{M}^\top v\right\|_2$ for vectors v that (in particular) represent a probability distribution, *i.e.*, vectors with unit ℓ_1-norm and non-negative entries. In general, getting such a bound is much weaker than getting a bound on the spectral norm which is equivalent to bounding $\left\|\mathcal{M}^\top v\right\|_2$ for all vectors v. However, for Hadamard (or more generally, orthogonal) matrices the two approaches are the same. Thus, if the matrix $\mathcal{M}^\top$ were Hadamard, our approach will not yield bounds better than those obtained from the spectral norm.

Even though the matrix $\mathcal{M}^\top$ is not Hadamard, it does come close, as it is a matrix with entries in $\{-1, 1\}$ and orthogonal rows, and the only reason it is not a Hadamard matrix is that it is not square. In other words, the matrix $\mathcal{M}^\top$ can be equivalently described as the Hadamard matrix with some of the rows removed. This means that any advantage that we get from the vector v representing a probability distribution needs to come from the fact that the matrix $\mathcal{M}^\top$ does not have all the rows of the Hadamard matrix.

To quantify this advantage, we add the 'missing' rows to the matrix $\mathcal{M}^\top$ and measure the increase in $\left\|\mathcal{M}^\top v\right\|_2$. More precisely, we work with the matrix $\mathcal{N}^\top$, defined to be the matrix $\mathcal{M}^\top$ with its columns tensored C times. Tensoring the columns in this way makes sure that $\mathcal{N}^\top$ has more rows than $\mathcal{M}^\top$, and moreover, each "extra" row is either identical to one of the previous rows, or corresponds to a fresh row of the Hadamard matrix. When a new row is identical to one of the previous rows, it ends up hurting the bound that we get, but when it is fresh, it makes the matrix $\mathcal{N}^\top$ look more like a Hadamard matrix for which the spectral norm based analysis is tight. Our bound on C in Theorem 6.1 ensures that the gains are more than the losses, and get a discrepancy bound strong enough to prove Theorem 6.1.

In Section 6.2 below, we upper bound the spectral norm of $\mathcal{N}^\top$, and in the following Section 6.3, we wrap up the proof of Theorem 6.1. We note that, in the actual proof, we do not derive a bound for $\left\|\mathcal{M}^\top v\right\|_2$ in terms of $\left\|\mathcal{N}^\top v\right\|_2$ for vectors v that represent probability distributions. Instead, we show Lemma 6.5 that bounds the discrepancy directly using our bound on the spectral norm of $\mathcal{N}^\top$. Lemma 6.5 relies on $\mathcal{M}$ and $\mathcal{N}$ being nicely related, or more specifically, $\mathcal{N}^\top$ being $\mathcal{M}^\top$ with its columns tensored C times.

6.1 Notation

We use $\mathcal{X}$ to denote the set of all possible inputs for Alice and $\mathcal{Y}$ to denote the set of all possible inputs for Bob. Observe that $|\mathcal{Y}| = (2nt)^K$. As above, $\mathcal{Z} \supset \mathcal{X}$ will denote the set $\mathcal{Z} = \left((\{0,1\}^{2n})^t\right)^K$. For convenience, we shall treat Bob's input as a single vector $I = (\vec{j}, \vec{i}) \in \mathcal{Y}$ and we shall often write X_{k,I_k} instead of X_{k,j_k,i_k} and $(\vec{X}, I)$ instead of $(\vec{X}, \vec{j}, \vec{i})$. and We shall also use $\vec{I} = (I_c)_{c\in[C]} = (\vec{j}_c, \vec{i}_c)_{c\in[C]}$ to denote an element in $\mathcal{Y}^C$. This will correspond to Bob's input being tensored C times.

As above, we shall use $\mathcal{M}$ to denote the communication matrix of the $\mathsf{Ind}^{\oplus}$ problem, and $\mathcal{N}$ to denote the matrix $\mathcal{M}$ with its rows tensored C times. Thus, $\mathcal{N}^\top$ is $\mathcal{M}^\top$ with its columns tensored C times. Formally, we have for all $\vec{X} \in \mathcal{Z}$, $I \in \mathcal{Y}$, and $\vec{I} \in \mathcal{Y}^C$ that

$$\mathcal{M}_{\vec{X},I} = \prod_{k=1}^{K} (-1)^{X_{k,I_k}} \qquad \text{and} \qquad \mathcal{N}_{\vec{X},\vec{I}} = \prod_{c=1}^{C}\prod_{k=1}^{K} (-1)^{X_{k,I_{c,k}}}. \tag{5}$$

6.2 Spectral Norm of $\mathcal{N}^\top$

The goal of this section is to show the following bound:

Lemma 6.2 (Spectral norm of $\mathcal{N}^\top$). *For all $|\mathcal{Z}|$-dimensional vectors v, we have:*

$$\left\|\mathcal{N}^\top v\right\|_2 \le (20ntC)^{\frac{CK}{4}} \cdot \sqrt{|\mathcal{Z}|} \cdot \|v\|_2.$$

Recall that the rows of $\mathcal{N}^\top$ are such that each pair of rows is either identical or orthogonal. The following definition captures when they are identical. For $\vec{I} \in \mathcal{Y}^C$ let $\mathcal{N}_{\vec{I}}$ denote column $\vec{I}$ of $\mathcal{N}$ (equivalently, row $\vec{I}$ of $\mathcal{N}^\top$). For all $\vec{I}, \vec{I'} \in \mathcal{Y}^C$ and $k \in [K]$, define the $2C$-length sequence $\mathsf{Seq}_{\vec{I},\vec{I'}}(k)$ to be:

$$\mathsf{Seq}_{\vec{I},\vec{I'}}(k) = I_{1,k}, \cdots, I_{C,k}, I'_{1,k}, \cdots, I'_{C,k}. \tag{6}$$

We say that the sequence $\mathsf{Seq}_{\vec{I},\vec{I'}}(k)$ is 'balanced' if no element in the sequence is repeated an odd number of times, and say it is 'unbalanced' otherwise.

Lemma 6.3 (Rows of $\mathcal{N}^\top$). *For all $\vec{I}, \vec{I'} \in \mathcal{Y}^C$, we have $\mathcal{N}_{\vec{I}} = \mathcal{N}_{\vec{I'}}$ if and only if $\mathsf{Seq}_{\vec{I},\vec{I'}}(k)$ is balanced for all $k \in [K]$. Moreover, we have:*

$$\mathcal{N}_{\vec{I}}^\top \mathcal{N}_{\vec{I'}} = \begin{cases} |\mathcal{Z}|, & \text{if } \mathcal{N}_{\vec{I}} = \mathcal{N}_{\vec{I'}} \\ 0, & \text{if } \mathcal{N}_{\vec{I}} \neq \mathcal{N}_{\vec{I'}} \end{cases}$$

Proof. Suppose first that $\mathsf{Seq}_{\vec{I},\vec{I'}}(k)$ is balanced for all $k \in [K]$. For all $\vec{X} \in \mathcal{Z}$, we have:

$$\begin{aligned} \mathcal{N}_{\vec{X},\vec{I}} \cdot \mathcal{N}_{\vec{X},\vec{I'}} &= \prod_{c=1}^{C}\prod_{k=1}^{K} (-1)^{X_{k,I_{c,k}}} \cdot \prod_{c=1}^{C}\prod_{k=1}^{K} (-1)^{X_{k,I'_{c,k}}} && \text{(Equation 5)} \\ &= \prod_{k=1}^{K} (-1)^{\sum_{c=1}^{C} X_{k,I_{c,k}} + X_{k,I'_{c,k}}} \end{aligned}$$

$$= 1. \qquad \text{(Equation 6 and } \mathsf{Seq}_{\vec{I},\vec{I}'}(k) \text{ is balanced for all } k \in [K])$$

As the entries of $\mathcal{N}$ are from $\{-1, 1\}$, we can conclude that $\mathcal{N}_{\vec{X},\vec{I}} = \mathcal{N}_{\vec{X},\vec{I}'}$ for all $\vec{X} \in \mathcal{Z}$ implying that $\mathcal{N}_{\vec{I}} = \mathcal{N}_{\vec{I}'}$. For the "moreover" part, note that $\mathcal{N}_{\vec{I}} = \mathcal{N}_{\vec{I}'}$ together with the fact that the entries of $\mathcal{N}$ are from $\{-1, 1\}$ implies that $\mathcal{N}_{\vec{I}}^{\top}\mathcal{N}_{\vec{I}'} = |\mathcal{Z}|$, as desired.

Next, assume that there exists $k^\star \in [K]$ such that $\mathsf{Seq}_{\vec{I},\vec{I}'}(k^\star)$ is unbalanced. It is sufficient to show that $\mathcal{N}_{\vec{I}}^{\top}\mathcal{N}_{\vec{I}'} = 0$ as it implies $\mathcal{N}_{\vec{I}} \neq \mathcal{N}_{\vec{I}'}$. We have:

$$\begin{aligned}
\mathcal{N}_{\vec{I}}^{\top}\mathcal{N}_{\vec{I}'} &= \sum_{\vec{X}\in\mathcal{Z}} \mathcal{N}_{\vec{X},\vec{I}} \cdot \mathcal{N}_{\vec{X},\vec{I}'} \\
&= \sum_{\vec{X}\in\mathcal{Z}} \prod_{c=1}^{C}\prod_{k=1}^{K}(-1)^{X_{k,I_{c,k}}} \cdot \prod_{c=1}^{C}\prod_{k=1}^{K}(-1)^{X_{k,I'_{c,k}}} && \text{(Equation 5)} \\
&= \sum_{\vec{X}\in\mathcal{Z}} \prod_{k=1}^{K}\left(\prod_{c=1}^{C}(-1)^{X_{k,I_{c,k}}}\right) \cdot \left(\prod_{c=1}^{C}(-1)^{X_{k,I'_{c,k}}}\right).
\end{aligned}$$

To continue, we recall that $\mathcal{Z} = \left((\{0,1\}^{2n})^t\right)^K$.

$$\mathcal{N}_{\vec{I}}^{\top}\mathcal{N}_{\vec{I}'} = \prod_{k=1}^{K} \sum_{X_k\in(\{0,1\}^{2n})^t} \left(\prod_{c=1}^{C}(-1)^{X_{k,I_{c,k}}}\right) \cdot \left(\prod_{c=1}^{C}(-1)^{X_{k,I'_{c,k}}}\right).$$

We conclude that $\mathcal{N}_{\vec{I}}^{\top}\mathcal{N}_{\vec{I}'} = 0$ follows if we show that the factor corresponding to $k^\star$ above is 0. We do this using the fact that $\mathsf{Seq}_{\vec{I},\vec{I}'}(k^\star)$ is unbalanced. Thus, there exists an element $I^\star$ that is repeated an odd number of times in $\mathsf{Seq}_{\vec{I},\vec{I}'}(k^\star)$. For $X \in (\{0,1\}^{2n})^t$, let $X_{-I^\star}$ denote the vector X with the coordinate $I^\star$ removed. We have:

$$\begin{aligned}
&\sum_{X\in(\{0,1\}^{2n})^t} \left(\prod_{c=1}^{C}(-1)^{X_{I_{c,k^\star}}}\right) \cdot \left(\prod_{c=1}^{C}(-1)^{X_{I'_{c,k^\star}}}\right) \\
&= \sum_{X_{I^\star}\in\{0,1\}} \sum_{X_{-I^\star}\in\{0,1\}^{2nt-1}} \left(\prod_{c=1}^{C}(-1)^{X_{I_{c,k^\star}}}\right) \cdot \left(\prod_{c=1}^{C}(-1)^{X_{I'_{c,k^\star}}}\right) \\
&= \left(\sum_{X_{I^\star}\in\{0,1\}} (-1)^{X_{I^\star}}\right) \times \left(\sum_{X_{-I^\star}\in\{0,1\}^{2nt-1}} \left(\prod_{\substack{c=1 \\ I_{c,k^\star}\neq I^\star}}^{C} (-1)^{X_{I_{c,k^\star}}}\right) \cdot \left(\prod_{\substack{c=1 \\ I'_{c,k^\star}\neq I^\star}}^{C} (-1)^{X_{I'_{c,k^\star}}}\right)\right) \\
&\qquad \text{(Equation 6 and } I^\star \text{ that is repeated an odd number of times in } \mathsf{Seq}_{\vec{I},\vec{I}'}(k^\star)) \\
&= 0.
\end{aligned}$$

□

If the rows of $\mathcal{N}^\top$ were mutually orthogonal, then $\mathcal{N}^\top\mathcal{N}$ would be a scaled identity matrix implying that $\|\mathcal{N}v\|_2 = \sqrt{|\mathcal{Z}|}\cdot\|v\|_2$ for all $\left|\mathcal{Y}^C\right|$-dimensional vectors v. However, some rows of $\mathcal{N}^\top$ are identical, and therefore, we have the following weaker lemma

Lemma 6.4 (Repeated rows in $\mathcal{N}^\top$)**.** *For all $\left|\mathcal{Y}^C\right|$-dimensional vectors v satisfying for all $\vec{I},\vec{I}' \in \mathcal{Y}^C$ that $\mathcal{N}_{\vec{I}} = \mathcal{N}_{\vec{I}'} \implies v_{\vec{I}} = v_{\vec{I}'}$, we have:*

$$\|\mathcal{N}v\|_2 \le (20ntC)^{\frac{CK}{4}} \cdot \sqrt{|\mathcal{Z}|}\cdot\|v\|_2.$$

Proof. By Lemma 6.3, we have:

$$\|\mathcal{N}v\|_2^2 = v^\top\mathcal{N}^\top\mathcal{N}v = \sum_{\vec{I}\in\mathcal{Y}^C}\sum_{\vec{I}'\in\mathcal{Y}^C} v_{\vec{I}}\cdot v_{\vec{I}'}\cdot\left(\mathcal{N}_{\vec{I}}^\top\mathcal{N}_{\vec{I}'}\right) = \sum_{\vec{I}\in\mathcal{Y}^C}\sum_{\substack{\vec{I}'\in\mathcal{Y}^C\\ \mathcal{N}_{\vec{I}}=\mathcal{N}_{\vec{I}'}}} v_{\vec{I}}\cdot v_{\vec{I}'}\cdot|\mathcal{Z}|.$$

Under the conditions of the lemma, this gives:

$$\begin{aligned}\|\mathcal{N}v\|_2 &\le \sqrt{\left(\max_{\vec{I}\in\mathcal{Y}^C}\left|\{\vec{I}'\in\mathcal{Y}^C \mid \mathcal{N}_{\vec{I}}=\mathcal{N}_{\vec{I}'}\}\right|\right)\cdot|\mathcal{Z}|\cdot\left(\sum_{\vec{I}\in\mathcal{Y}^C} v_{\vec{I}}\cdot v_{\vec{I}}\right)}\\ &\le \|v\|_2\cdot\sqrt{\left(\max_{\vec{I}\in\mathcal{Y}^C}\left|\{\vec{I}'\in\mathcal{Y}^C \mid \mathcal{N}_{\vec{I}}=\mathcal{N}_{\vec{I}'}\}\right|\right)\cdot|\mathcal{Z}|}.\end{aligned}$$

The lemma thus, follows once we show that for any $\vec{I}\in\mathcal{Y}^C$ at most $(20ntC)^{\frac{CK}{2}}$ values of $\vec{I}'\in\mathcal{Y}^C$ satisfy $\mathcal{N}_{\vec{I}} = \mathcal{N}_{\vec{I}'}$. Using Lemma 6.3, this is equivalent to showing that at most $(20ntC)^{\frac{CK}{2}}$ values of $\vec{I}'\in\mathcal{Y}^C$ satisfy the property that $\mathsf{Seq}_{\vec{I},\vec{I}'}(k)$ is balanced for all $k\in[K]$.

Using the fact that $\mathsf{Seq}_{\vec{I},\vec{I}'}(k)$ is determined by the values $I_{1,k},\cdots,I_{C,k},I'_{1,k},\cdots,I'_{C,k}$ (see Equation 6), we get that it is sufficient to show that for any k and any values of $I_{1,k},\cdots,I_{C,k}$, at most $(20ntC)^{\frac{C}{2}}$ values of $I'_{1,k},\cdots,I'_{C,k}$ make the sequence $\mathsf{Seq}_{\vec{I},\vec{I}'}(k)$ balanced. To this end, fix k and values $I_{1,k},\cdots,I_{C,k}$. We (over) count the number of values of $I'_{1,k},\cdots,I'_{C,k}$ that make the sequence $\mathsf{Seq}_{\vec{I},\vec{I}'}(k)$ balanced as follows:

- Count the number of different multi-sets $I'_{1,k},\cdots,I'_{C,k}$. As $I'_{c,k}$ for all $c\in[C]$, is an element of $[t]\times[2n]$, a multi-set is equivalently defined by $2nt$ non-negative integers $\{z_i\}_{i\in[t]\times[2n]}$ such that $\sum_{i\in[t]\times[2n]} z_i = C$. Moreover, the condition that $\mathsf{Seq}_{\vec{I},\vec{I}'}(k)$ is balanced translates to z_i being odd for those i that appear an odd number of times in $I_{1,k},\cdots,I_{C,k}$ and even for those i that appear an even number of times in $I_{1,k},\cdots,I_{C,k}$. Let there be D values of i of the former type and assume without loss of generality that these are the D lexicographically smallest values. This means that we want to count the number of non-negative integer solutions of $\sum_{i\in[t]\times[2n]} z_i = C$ such that D lexicographically smallest variables are odd, and the rest are even.

  This is equivalent to counting the non-negative integer solutions of $\sum_{i\in[t]\times[2n]} z_i = \frac{C-D}{2}$, and by a standard argument[11], equal to

$$\binom{2nt + \frac{C-D}{2} - 1}{\frac{C-D}{2}} \leq \binom{2nt + \frac{C}{2}}{\frac{C}{2}} \leq \left(\frac{20nt}{C}\right)^{\frac{C}{2}}.$$

- Permute the multi-set in at most $C! \leq C^C$ ways.

Overall, we get that:

$$\max_{\vec{I} \in \mathcal{Y}^C} \left| \{ \vec{I'} \in \mathcal{Y}^C \mid \mathcal{N}_{\vec{I}} = \mathcal{N}_{\vec{I'}} \} \right| \leq \left(\frac{20nt}{C}\right)^{\frac{C}{2}} \cdot C^C \leq (20ntC)^{\frac{C}{2}},$$

finishing the proof. □

We finish this section by showing Lemma 6.2

Proof of Lemma 6.2. Fix v. We have:

$$\begin{aligned} \left\| \mathcal{N}^\top v \right\|_2 &= \frac{v^\top \mathcal{N} \mathcal{N}^\top v}{\left\| \mathcal{N}^\top v \right\|_2} \\ &\leq \frac{\|v\|_2 \left\| \mathcal{N} \mathcal{N}^\top v \right\|_2}{\left\| \mathcal{N}^\top v \right\|_2} && \text{(Cauchy-Schwarz inequality)} \\ &\leq (20ntC)^{\frac{CK}{4}} \cdot \sqrt{|\mathcal{Z}|} \cdot \|v\|_2. && \text{(Lemma 6.4)} \end{aligned}$$

□

6.3 Proof of Theorem 6.1

Proof of Theorem 6.1. As a randomized protocol Π is simply a distribution over deterministic protocols, it is sufficient to prove the theorem for all deterministic protocols Π from Alice to Bob. Fix such a protocol Π and assume without loss of generality that $\mathsf{CC}(\Pi) = C \cdot \log n$. We also assume, without loss of generality that the messages in Π are from the set $\left[2^{\mathsf{CC}(\Pi)}\right]$. For $m \in \left[2^{\mathsf{CC}(\Pi)}\right]$, define the set $\mathcal{X}_m \subseteq \mathcal{X}$ to be the subsets of inputs for which Alice sends the message m. In particular, we have that the sets $\mathcal{X}_1, \cdots, \mathcal{X}_{2^{\mathsf{CC}(\Pi)}}$ form a partition of the set $\mathcal{X}$. Also, for $m \in \left[2^{\mathsf{CC}(\Pi)}\right]$ and $z \in \{-1, 1\}$, let $\mathcal{Y}_{m,z} \subseteq \mathcal{Y}$ be the set of inputs for Bob for which he outputs z on receiving message m from Alice. Note that,

[11]We provide the argument here for completeness. The claim is that, for $n, r > 0$, the number of non-negative integer solutions of $\sum_{i \in [n]} z_i = r$ is equal to $\binom{n+r-1}{r}$. Indeed, writing every solution in unary with the different variables separated by zeros gives a binary string of length $n+r-1$ with $n-1$ zeros, and every such string corresponds to a solution. Thus, the number of solutions is equal to the number of strings, which is equal to $\binom{n+r-1}{r}$.

for all $m \in \left[2^{\mathsf{CC}(\Pi)}\right]$, the sets $\mathcal{Y}_{m,-1}$ and $\mathcal{Y}_{m,1}$ from a partition of $\mathcal{Y}$. We derive:

$$\begin{aligned}
&\Pr_{(\vec{X},I)\sim\mathcal{D}^{\mathsf{Ind}\oplus}_{n,t,K}}\left(\Pi(\vec{X},I) = \prod_{k=1}^{K}(-1)^{X_{k,I_k}}\right) \\
&\quad = \frac{1}{|\mathcal{X}|\cdot|\mathcal{Y}|}\cdot\sum_{\vec{X}\in\mathcal{X}}\sum_{I\in\mathcal{Y}}\mathbb{1}\left(\Pi(\vec{X},I) = \mathcal{M}_{\vec{X},I}\right) && \text{(Equation 5)} \\
&\quad = \frac{1}{|\mathcal{X}|\cdot|\mathcal{Y}|}\cdot\sum_{m\in\left[2^{\mathsf{CC}(\Pi)}\right]}\sum_{z\in\{-1,1\}}\sum_{\vec{X}\in\mathcal{X}_m}\sum_{I\in\mathcal{Y}_{m,z}}\mathbb{1}\left(\Pi(\vec{X},I) = \mathcal{M}_{\vec{X},I}\right).
\end{aligned}$$

Next, note by the definition of $\mathcal{X}_m$ and $\mathcal{Y}_{m,z}$ that $\Pi(\vec{X},I) = z$ for all $\vec{X} \in \mathcal{X}_m, I \in \mathcal{Y}_{m,z}$. This gives:

$$\Pr_{(\vec{X},I)\sim\mathcal{D}^{\mathsf{Ind}\oplus}_{n,t,K}}\left(\Pi(\vec{X},I) = \prod_{k=1}^{K}(-1)^{X_{k,I_k}}\right) = \frac{1}{|\mathcal{X}|\cdot|\mathcal{Y}|}\cdot\sum_{m\in\left[2^{\mathsf{CC}(\Pi)}\right]}\sum_{z\in\{-1,1\}}\sum_{\vec{X}\in\mathcal{X}_m}\sum_{I\in\mathcal{Y}_{m,z}}\mathbb{1}\left(z = \mathcal{M}_{\vec{X},I}\right).$$

As both z and $\mathcal{M}_{\vec{X},I}$ take values in $\{-1,1\}$, we have that $\mathbb{1}\left(z = \mathcal{M}_{\vec{X},I}\right) = \frac{1}{2}\cdot\left(1 + z\cdot\mathcal{M}_{\vec{X},I}\right)$. Plugging in, we get:

$$\begin{aligned}
&\Pr_{(\vec{X},I)\sim\mathcal{D}^{\mathsf{Ind}\oplus}_{n,t,K}}\left(\Pi(\vec{X},I) = \prod_{k=1}^{K}(-1)^{X_{k,I_k}}\right) \\
&\quad = \frac{1}{|\mathcal{X}|\cdot|\mathcal{Y}|}\cdot\sum_{m\in\left[2^{\mathsf{CC}(\Pi)}\right]}\sum_{z\in\{-1,1\}}\sum_{\vec{X}\in\mathcal{X}_m}\sum_{I\in\mathcal{Y}_{m,z}}\frac{1}{2}\cdot\left(1 + z\cdot\mathcal{M}_{\vec{X},I}\right) \\
&\quad = \frac{1}{2} + \frac{1}{2\cdot|\mathcal{X}|\cdot|\mathcal{Y}|}\cdot\sum_{m\in\left[2^{\mathsf{CC}(\Pi)}\right]}\sum_{z\in\{-1,1\}} z\cdot\left(\sum_{\vec{X}\in\mathcal{X}_m}\sum_{I\in\mathcal{Y}_{m,z}}\mathcal{M}_{\vec{X},I}\right) \\
&\quad \leq \frac{1}{2} + \frac{1}{2\cdot|\mathcal{X}|\cdot|\mathcal{Y}|}\cdot\sum_{m\in\left[2^{\mathsf{CC}(\Pi)}\right]}\sum_{z\in\{-1,1\}}\left|\mathbb{1}(\mathcal{X}_m)^{\top}\mathcal{M}\mathbb{1}(\mathcal{Y}_{m,z})\right|, && \text{(As } z\in\{-1,1\}\text{)}
\end{aligned}$$

where $\mathbb{1}(\mathcal{X}_m)$ denotes the $|\mathcal{Z}|$-dimensional indicator vector for the set $\mathcal{X}_m$ and $\mathbb{1}(\mathcal{Y}_{m,z})$ denotes the $|\mathcal{Y}|$-dimensional indicator for the set $\mathcal{Y}_{m,z}$.

Next, call a pair $(m,z) \in \left[2^{\mathsf{CC}(\Pi)}\right]\times\{-1,1\}$ "Alice-atypical" if $|\mathcal{X}_m| \leq |\mathcal{X}|\cdot 2^{-2\cdot\mathsf{CC}(\Pi)}$ and "Bob-atypical" if $|\mathcal{Y}_{m,z}| \leq |\mathcal{Y}|^{1-\delta}$. Note that a pair (m,z) may be both Alice-atypical and Bob-atypical. We call a pair that is neither Alice-atypical nor Bob-atypical a "typical" pair. The following holds for all typical pairs.

Lemma 6.5 (Property of typical pairs)**.** *For all typical pairs (m,z), we have:*

$$\left|\mathbb{1}(\mathcal{X}_m)^{\top}\mathcal{M}\mathbb{1}(\mathcal{Y}_{m,z})\right| \leq (2nt)^{-50}\cdot 2^{\frac{\mathsf{CC}(\Pi)}{C}}\cdot|\mathcal{X}_m|\cdot|\mathcal{Y}_{m,z}|.$$

We show Lemma 6.5 later but assuming it for now, we have:

$$\begin{aligned}
&\Pr_{(\vec{X},I)\sim\mathcal{D}^{\mathsf{Ind}\oplus}_{n,t,K}}\left(\Pi(\vec{X},I)=\prod_{k=1}^{K}(-1)^{X_{k,I_k}}\right)\\
&\quad\le \frac{1}{2}+\frac{1}{2\cdot|\mathcal{X}|\cdot|\mathcal{Y}|}\cdot\sum_{\substack{(m,z)\in\left[2^{\mathsf{CC}(\Pi)}\right]\times\{-1,1\}\\ (m,z)\text{ typical}}}\left|\mathbb{1}(\mathcal{X}_m)^{\top}\mathcal{M}\mathbb{1}(\mathcal{Y}_{m,z})\right|\\
&\qquad+\frac{1}{2\cdot|\mathcal{X}|\cdot|\mathcal{Y}|}\cdot\sum_{\substack{(m,z)\in\left[2^{\mathsf{CC}(\Pi)}\right]\times\{-1,1\}\\ (m,z)\text{ Alice-atypical}}}\left|\mathbb{1}(\mathcal{X}_m)^{\top}\mathcal{M}\mathbb{1}(\mathcal{Y}_{m,z})\right|\\
&\qquad+\frac{1}{2\cdot|\mathcal{X}|\cdot|\mathcal{Y}|}\cdot\sum_{\substack{(m,z)\in\left[2^{\mathsf{CC}(\Pi)}\right]\times\{-1,1\}\\ (m,z)\text{ Bob-atypical}}}\left|\mathbb{1}(\mathcal{X}_m)^{\top}\mathcal{M}\mathbb{1}(\mathcal{Y}_{m,z})\right|.
\end{aligned}$$

Using Lemma 6.5 on the first term and the trivial bound of $|\mathcal{X}_m|\cdot|\mathcal{Y}_{m,z}|$ on the other terms, we get:

$$\begin{aligned}
&\Pr_{(\vec{X},I)\sim\mathcal{D}^{\mathsf{Ind}\oplus}_{n,t,K}}\left(\Pi(\vec{X},I)=\prod_{k=1}^{K}(-1)^{X_{k,I_k}}\right)\\
&\le \frac{1}{2}+\frac{1}{2\cdot|\mathcal{X}|\cdot|\mathcal{Y}|}\cdot\sum_{\substack{(m,z)\in\left[2^{\mathsf{CC}(\Pi)}\right]\times\{-1,1\}\\ (m,z)\text{ typical}}}(2nt)^{-50}\cdot 2^{\frac{\mathsf{CC}(\Pi)}{C}}\cdot|\mathcal{X}_m|\cdot|\mathcal{Y}_{m,z}|\\
&\qquad+\frac{1}{2\cdot|\mathcal{X}|\cdot|\mathcal{Y}|}\left(\sum_{\substack{(m,z)\in\left[2^{\mathsf{CC}(\Pi)}\right]\times\{-1,1\}\\ (m,z)\text{ Alice-atypical}}}|\mathcal{X}_m|\cdot|\mathcal{Y}_{m,z}|+\sum_{\substack{(m,z)\in\left[2^{\mathsf{CC}(\Pi)}\right]\times\{-1,1\}\\ (m,z)\text{ Bob-atypical}}}|\mathcal{X}_m|\cdot|\mathcal{Y}_{m,z}|\right)\\
&\le \frac{1}{2}+\frac{1}{2\cdot|\mathcal{X}|\cdot|\mathcal{Y}|}\cdot\sum_{\substack{(m,z)\in\left[2^{\mathsf{CC}(\Pi)}\right]\times\{-1,1\}\\ (m,z)\text{ typical}}}(2nt)^{-50}\cdot 2^{\frac{\mathsf{CC}(\Pi)}{C}}\cdot|\mathcal{X}_m|\cdot|\mathcal{Y}_{m,z}|\\
&\qquad+\frac{1}{2\cdot|\mathcal{X}|\cdot|\mathcal{Y}|}\left(\sum_{\substack{(m,z)\in\left[2^{\mathsf{CC}(\Pi)}\right]\times\{-1,1\}\\ (m,z)\text{ Alice-atypical}}}\frac{|\mathcal{X}|\cdot|\mathcal{Y}_{m,z}|}{2^{2\cdot\mathsf{CC}(\Pi)}}+\sum_{\substack{(m,z)\in\left[2^{\mathsf{CC}(\Pi)}\right]\times\{-1,1\}\\ (m,z)\text{ Bob-atypical}}}|\mathcal{X}_m|\cdot|\mathcal{Y}|^{1-\delta}\right)\\
&\hspace{20em}\text{(Definition of Alice-atypical and Bob-atypical)}\\
&\le \frac{1}{2}+\frac{1}{2^{\mathsf{CC}(\Pi)}}+\frac{1}{|\mathcal{Y}|^{\delta}}+\frac{1}{2\cdot|\mathcal{X}|\cdot|\mathcal{Y}|}\cdot\sum_{(m,z)\in\left[2^{\mathsf{CC}(\Pi)}\right]\times\{-1,1\}}(2nt)^{-50}\cdot 2^{\frac{\mathsf{CC}(\Pi)}{C}}\cdot|\mathcal{X}_m|\cdot|\mathcal{Y}_{m,z}|\\
&\le \frac{1}{2}+\frac{1}{2^{\mathsf{CC}(\Pi)}}+\frac{1}{|\mathcal{Y}|^{\delta}}+(2nt)^{-50}\cdot 2^{\frac{\mathsf{CC}(\Pi)}{C}}
\end{aligned}$$

$$\leq \frac{1}{2} + \frac{1}{(2nt)^{20}},$$

as $\mathsf{CC}(\Pi) \leq C \cdot \log n$ and $C \geq \frac{1}{20} \cdot (2nt)^{1-5\delta}$ and $|\mathcal{Y}| = (2nt)^K$ and $\delta = \frac{100}{K}$ finishing the proof. $\square$

We now show Lemma 6.5.

Proof of Lemma 6.5. In this proof, we use $\mathbb{1}(\mathcal{Y}^C_{m,z})$ denotes the $|\mathcal{Y}^C|$-dimensional indicator for the set $\mathcal{Y}^C_{m,z}$. As $C > 0$, we have that the function x^C is convex. This gives:

$$\begin{aligned}\left(\frac{1}{|\mathcal{X}_m|} \cdot \mathbb{1}(\mathcal{X}_m)^\top \mathcal{M}\mathbb{1}(\mathcal{Y}_{m,z})\right)^C &\leq \frac{1}{|\mathcal{X}_m|} \cdot \mathbb{1}(\mathcal{X}_m)^\top \mathcal{N}\mathbb{1}(\mathcal{Y}^C_{m,z}) && \text{(Jensen's inequality)}\\ &\leq \frac{1}{|\mathcal{X}_m|} \cdot \left\|\mathbb{1}(\mathcal{Y}^C_{m,z})\right\|_2 \left\|\mathcal{N}^\top \mathbb{1}(\mathcal{X}_m)\right\|_2 \\ & && \text{(Cauchy-Schwarz inequality)}\\ &\leq \frac{1}{|\mathcal{X}_m|} \cdot \left\|\mathbb{1}(\mathcal{Y}^C_{m,z})\right\|_2 \cdot (20ntC)^{\frac{CK}{4}} \cdot \sqrt{|\mathcal{Z}|} \cdot \|\mathbb{1}(\mathcal{X}_m)\|_2. \\ & && \text{(Lemma 6.2)}\end{aligned}$$

Next, we use the fact that (m, z) is typical to get $|\mathcal{X}_m| \geq |\mathcal{X}| \cdot 2^{-2\cdot\mathsf{CC}(\Pi)} \geq |\mathcal{Z}| \cdot 2^{-2\cdot\mathsf{CC}(\Pi)} \cdot (3n)^{-tK}$ and $|\mathcal{Y}_{m,z}| \geq |\mathcal{Y}|^{1-\delta} \geq (20ntC)^{\frac{K}{2}} \cdot (2nt)^{\frac{3\delta K}{2}}$. Plugging in:

$$\begin{aligned}\left(\frac{1}{|\mathcal{X}_m|} \cdot \mathbb{1}(\mathcal{X}_m)^\top \mathcal{M}\mathbb{1}(\mathcal{Y}_{m,z})\right)^C &\leq (3n)^{\frac{tK}{2}} \cdot 2^{\mathsf{CC}(\Pi)} \cdot |\mathcal{Y}_{m,z}|^C \cdot (2nt)^{-\frac{3\delta CK}{4}}\\ &\leq (2nt)^{-50C} \cdot 2^{\mathsf{CC}(\Pi)} \cdot |\mathcal{Y}_{m,z}|^C. \quad (\text{As } tK \leq 10C \text{ and } \delta = \tfrac{100}{K})\end{aligned}$$

Rearranging gives the result. $\square$

7 Indistinguishability of Set-Hiding-Game

We will consider the following one-way communication problem $\mathsf{Set\text{-}Hiding\text{-}Game}_{n,t,K}$ defined as follows.

Definition 7.1. *For integers $n, t, K > 0$. $\mathsf{Set\text{-}Hiding\text{-}Game}_{n,t,K}$ is defined as the following one-way communication game: Alices gets $t \cdot K$ sets $\vec{T} = (T_{k,j})_{(k\in[K],j\in[t])}$, each being a subset of $[4n]$, and Bob gets K indices $\vec{j} = (j_k)_{k\in[K]}$ and K permutations $\vec{\pi} = (\pi_k)_{k\in[K]}$, each being a permutation on $[4n]$.*

Alice sends a message to Bob, and the goal of the game is to output

$$\left(\bigoplus_{k=1}^{K} \pi_k(T_{k,j_k})\right) \cap [n].$$

Hard Input Distribution $\mathcal{D}^{\mathsf{hard}}_{n,t,K}(T)$ to Set-Hiding-Game

- **Parameters:** A set $T \subseteq [n]$.
- **Construction:**
  - Sample K indices $\vec{j} = (j_k)_{k\in[K]}$ uniformly at random from $[t]$.
  - Sample K permutations $\vec{\pi} = (\pi_k)_{k\in[K]}$ on the set $[4n]$ uniformly at random.
  - Sample $n \cdot K$ bits $\vec{b} = (b_{k,i})_{k\in[K],i\in[n]}$ from $\{0,1\}$ uniformly at random conditioned on $\bigoplus_{k=1}^{K} b_{k,i} = \mathbb{1}(i \in T)$ for all $i \in [n]$.
  - Sample $t \cdot K$ sets $\vec{T} = (T_{k,j})_{k\in[K],j\in[t]}$, where each $T_{k,j} \subseteq [4n]$, as follows:
    * For $k \in [K], j \in [t] \setminus \{j_k\}$, sample the set $T_{k,j}$ uniformly conditioned on $|T_{k,j}| = 2n$.
    * For $k \in [K]$, sample the set T_{k,j_k} uniformly at random conditioned on:
      · $|T_{k,j_k}| = 2n$.
      · For all $i \in [n]$, we have $\pi_k^{-1}(i) \in T_{k,j_k}$ if and only if $b_{k,i} = 1$.
- **Alice's Input:** The $t \cdot K$ sets $\vec{T}$.
- **Bob's Input:** The indices $\vec{j}$ and the permutations $\vec{\pi}$.

Observation 7.2 (Property of Set-Hiding-Game). *For every triple $(\vec{T}, \vec{j}, \vec{\pi})$ in the support of $\mathcal{D}^{\mathsf{hard}}_{n,t,K}(T)$, it holds that*

$$\left(\bigoplus_{k=1}^{K} \pi_k(T_{k,j_k})\right) \cap [n] = T.$$

Lemma 7.4 below captures the indistinguishability of the Set-Hiding-Game. The crux of this lemma is the communication lower bound shown in Theorem 6.1. We shall use it in the following form:

Corollary 7.3 (Lower bound of Communication complexity of $\mathsf{Ind}^{\oplus}$). *Let $n, t, K > 1000$ be integers. Define $\delta = \frac{100}{K}$ and $C = \frac{1}{20} \cdot (2nt)^{1-5\delta}$ and assume that $tK \le 10C$. For all one-way randomized protocols Π from Alice to Bob for the (n,t,K)-$\mathsf{Ind}^{\oplus}$ problem satisfying $\mathsf{CC}(\Pi) \le C \cdot \log n$, we have:*

$$\left\| \Pi\left(\mathcal{D}^{\mathsf{Ind}^{\oplus}}_{n,t,K} \mid \prod_{k=1}^{K} (-1)^{X_{k,j_k,i_k}} = 1\right) - \Pi\left(\mathcal{D}^{\mathsf{Ind}^{\oplus}}_{n,t,K} \mid \prod_{k=1}^{K} (-1)^{X_{k,j_k,i_k}} = -1\right) \right\|_{\mathrm{TV}} \le \frac{2}{(2nt)^{20}}.$$

Lemma 7.4 (Indistinguishability of Set-Hiding-Game). *Let $n, t, K > 1000$ be integers and $S, T \subseteq [n]$ be sets. Define $\delta = \frac{100}{K}$ and $C = \frac{1}{20} \cdot (2nt)^{1-5\delta}$ and assume that $tK \le 10C$. For all one-way randomized protocols Π from Alice to Bob for the Set-Hiding-Game satisfying*

$\mathsf{CC}(\Pi) \leq C \cdot \log n$, *we have:*

$$\left\|\Pi\big(\mathcal{D}^{\mathsf{hard}}_{n,t,K}(S)\big) - \Pi\big(\mathcal{D}^{\mathsf{hard}}_{n,t,K}(T)\big)\right\|_{\mathrm{TV}} \leq \frac{1}{(2nt)^{15}}.$$

Proof. Fix a protocol Π and define the sets $Z_0 = S$ and $Z_i = (T \cap [i]) \cup (S \setminus [i])$ for all $i \in [n]$. In order to show the lemma, we show that for all $i \in [n]$, we have:

$$\left\|\Pi\big(\mathcal{D}^{\mathsf{hard}}_{n,t,K}(Z_{i-1})\big) - \Pi\big(\mathcal{D}^{\mathsf{hard}}_{n,t,K}(Z_i)\big)\right\|_{\mathrm{TV}} \leq \frac{2}{(2nt)^{20}}. \tag{7}$$

The lemma then follows by adding Equation 7 for all $i \in [n]$ and using the triangle inequality. Henceforth, we focus on showing Equation 7. Fix an $i^\star \in [n]$ and observe that Equation 7 is trivial if $Z_{i^\star-1} = Z_{i^\star}$. We therefore assume otherwise. For convenience, define $Z^\star_1 = Z_{i^\star} \setminus \{i^\star\}$ and observe that $Z_{i^\star-1} = Z^\star_1 \cup (S \cap \{i^\star\})$. and $Z_{i^\star} = Z^\star_1 \cup (T \cap \{i^\star\})$. As we assume that $Z_{i^\star-1} \neq Z_{i^\star}$, we must have that one of the sets $Z_{i^\star-1}$ and $Z_{i^\star}$ must be $Z^\star_1$ and the other one must be $Z^\star_1 \cup \{i^\star\}$. It shall be convenient to denote $Z^\star_{-1} = Z^\star_1 \cup \{i^\star\}$. Rewriting Equation 7 in this notation, we get that we need to show:

$$\left\|\Pi\big(\mathcal{D}^{\mathsf{hard}}_{n,t,K}(Z^\star_1)\big) - \Pi\big(\mathcal{D}^{\mathsf{hard}}_{n,t,K}(Z^\star_{-1})\big)\right\|_{\mathrm{TV}} \leq \frac{2}{(2nt)^{20}}. \tag{8}$$

We shall show Equation 8 with the help of the protocol Ψ for the $(n+1,t,K)$-$\mathsf{Ind}^{\oplus}$ problem defined below:

By definition, Ψ is a protocol for the $(n+1,t,K)$-$\mathsf{Ind}^{\oplus}$ problem satisfying $\mathsf{CC}(\Psi) \leq C \cdot \log n$. We shall consider the protocol Ψ when the inputs are drawn from the distribution $\mathcal{D}^{\mathsf{Ind}^{\oplus}}_{n+1,t,K}$. For $z \in \{-1,1\}$, define the event E_z over the randomness in the distribution $\mathcal{D}^{\mathsf{Ind}^{\oplus}}_{n+1,t,K}$ as:

$$E_z := \left[\prod_{k=1}^{K} (-1)^{X_{k,j_k,i_k}} = z\right]. \tag{9}$$

Next, for inputs $\vec{X}$ for Alice and $\vec{i}, \vec{j}$ for Bob, define $\Psi_{\mathsf{mid}}(\vec{X}, \vec{i}, \vec{j})$ to be the joint distribution of the tuple $(\vec{T}, \vec{\pi}, \vec{j})$ that the protocol Ψ uses as input for the protocol Π in Line 5. Moreover, for a distribution $\mathcal{D}$ over the inputs $(\vec{X}, \vec{i}, \vec{j})$, let $\Psi_{\mathsf{mid}}(\mathcal{D})$ denote the distribution of the tuple $(\vec{T}, \vec{\pi}, \vec{j})$ that the protocol Ψ uses as input for the protocol Π in Line 5 when the inputs $(\vec{X}, \vec{i}, \vec{j})$ are sampled from $\mathcal{D}$. The following is the main property satisfied by Ψ.

Lemma 7.5 (Property of Ψ). *For all $z \in \{-1,1\}$, it holds that:*

$$\Psi_{\mathsf{mid}}\big(\mathcal{D}^{\mathsf{Ind}^{\oplus}}_{n+1,t,K} \mid E_z\big) = \mathcal{D}^{\mathsf{hard}}_{n,t,K}(Z^\star_z).$$

We prove Lemma 7.5 later but use it now to finish the proof of Lemma 7.4. We have by Lemma 7.5 that:

Algorithm 1 Protocol Ψ.

```
Input: Alice's input is vectors $\vec{X} = (X_{k,j})_{k\in[K],j\in[t]}$ satisfying $X_{k,j} \in \{0,1\}^{2(n+1)}$ and
    $\|X_{k,j}\|_1 = n+1$ for all $k \in [K]$ and $j \in [t]$. Bob's input is vectors $\vec{i} \in [2(n+1)]^K, \vec{j} \in [t]^K$.
1: For $k \in [K], j \in [t]$, using public randomness, sample a permutation $\sigma_{k,j}$ on $[4n]$.
2: For $k \in [K]$, Bob, using his private randomness, samples a permutation $\pi_k$ on $[4n]$
   uniformly at random conditioned on:
     • $\pi_k(\sigma_{k,j_k}(i_k)) = i^\star$.
     • For all $a < i^\star \in [n]$, $\pi_k(\sigma_{k,j_k}(2(n+1)+a)) = a$.
     • For all $a > i^\star \in [n]$, $\pi_k(\sigma_{k,j_k}(2(n+1)+a-1)) = a$.
3: For $k \in [K], a \in [n] \setminus \{i^\star\}$, using public randomness, sample a bit $b_{k,a}$ from $\{0,1\}$
   uniformly at random conditioned on $\bigoplus_{k=1}^K b_{k,a} = \mathbb{1}(a \in Z_1^\star)$ for all $a \in [n] \setminus \{i^\star\}$.
4: For $k \in [K], j \in [t]$, Alice constructs the set $T_{k,j} \subseteq [4n]$ privately as follows:
     • For $i \in [2(n+1)]$, we have $\mathbb{1}(\sigma_{k,j}(i) \in T_{k,j}) = X_{k,j,i}$.
     • For all $a < i^\star \in [n], b \in \{0,1\}$, we have
           $\mathbb{1}(\sigma_{k,j}(2(n+1)+a+(n-1)(1-b)) \in T_{k,j}) = \mathbb{1}(b = b_{k,a})$.
     • For all $a > i^\star \in [n], b \in \{0,1\}$, we have
           $\mathbb{1}(\sigma_{k,j}(2(n+1)+a+(n-1)(1-b)-1) \in T_{k,j}) = \mathbb{1}(b = b_{k,a})$.
5: Alice and Bob run the protocol $\Pi$ with inputs $\vec{T} = (T_{k,j})_{k\in[K],j\in[t]}$ and $\vec{\pi} = (\pi_k)_{k\in[K]}, \vec{j}$
   respectively. They output 1 if $\Pi$ outputs $Z_1^\star$ and $-1$ if $\Pi$ outputs $Z_{-1}^\star$.
```

$$\left\|\Pi\big(\mathcal{D}^{\mathsf{hard}}_{n,t,K}(Z_1^\star)\big) - \Pi\big(\mathcal{D}^{\mathsf{hard}}_{n,t,K}(Z_{-1}^\star)\big)\right\|_{\mathrm{TV}} = \left\|\Pi\Big(\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_1)\Big) - \Pi\Big(\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_{-1})\Big)\right\|_{\mathrm{TV}}.$$

To continue, note that in Line 5, that the protocol Ψ simply runs the protocol Π on the inputs sampled from $\Psi_{\mathsf{mid}}(\cdot)$, and outputs z if and only if the protocol Π outputs $Z_z^\star$. This means that we can continue as:

$$\begin{aligned}\left\|\Pi\big(\mathcal{D}^{\mathsf{hard}}_{n,t,K}(Z_1^\star)\big) - \Pi\big(\mathcal{D}^{\mathsf{hard}}_{n,t,K}(Z_{-1}^\star)\big)\right\|_{\mathrm{TV}} &= \left\|\Psi\Big(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_1\Big) - \Psi\Big(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_{-1}\Big)\right\|_{\mathrm{TV}} \\ &\le \frac{2}{(2nt)^{20}}, \qquad \text{(Corollary 7.3)}\end{aligned}$$

as required to show Equation 8.

□

Proof of Lemma 7.5. Fix $z \in \{-1, 1\}$. Observe that Algorithm 1 has variables $b_{k,a}$ for all $k \in [K], a \in [n] \setminus \{i^\star\}$. We also define the variables $b_{k,i^\star} = X_{k,j_k,i_k}$ for all $k \in [K]$. These variables are not known to Alice or Bob and only used for the purpose of analysis. We consider the distribution $\mathcal{D}^{\mathsf{hard}}_{n,t,K}(Z^\star_z)$ as a distribution over tuples $(\vec{j}, \vec{\pi}, \vec{b}, \vec{T})$ and $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_z)$ as a joint distribution over all the variables in Algorithm 1 and show the stronger statement that the distribution of the tuple $(\vec{j}, \vec{\pi}, \vec{b}, \vec{T})$ is identical in both the distributions. This is done in several steps.

The marginal distribution of $\vec{j}$ is identical. In the distribution $\mathcal{D}^{\mathsf{hard}}_{n,t,K}(Z^\star_z)$, the marginal distribution of $\vec{j}$ is simply the uniform distribution over $[t]^K$. As Algorithm 1 does not affect the variables $\vec{j}$, the marginal distribution of $\vec{j}$ in the distribution $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_z)$ is the same as that in the distribution $\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_z$, which is also uniform over $[t]^K$, as desired.

Conditioned on $\vec{j}$, the marginal distribution of $\vec{\pi}$ is identical. The marginal distribution of $\vec{\pi}$ in the distribution $\mathcal{D}^{\mathsf{hard}}_{n,t,K}(Z^\star_z) \mid \vec{j}$ is such that each coordinate is independently uniform over all permutations on $[4n]$. To show that the marginal distribution of $\vec{\pi}$ in the distribution $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_z) \mid \vec{j}$ is identical, we in fact consider the stronger conditioning $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_z) \mid \vec{i}, \vec{j}$. Observe that the latter is just $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_z, \vec{i}, \vec{j})$.

Next, note from Line 1 and Line 2 that all the coordinates of $\vec{\pi}$ in the distribution $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_z, \vec{i}, \vec{j})$ are independent and it suffices to show that each one of them is a uniformly random permutation over $[4n]$. For this, fix a coordinate $k \in [K]$ and note that the marginal distribution of π_k in the distribution $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_z, \vec{i}, \vec{j})$ is simply uniform conditioned on:

- $\sigma_{k,j_k}(i_k) = \pi_k^{-1}(i^\star)$.
- For all $a < i^\star \in [n]$, $\sigma_{k,j_k}(2(n+1) + a) = \pi_k^{-1}(a)$.
- For all $a > i^\star \in [n]$, $\sigma_{k,j_k}(2(n+1) + a - 1) = \pi_k^{-1}(a)$.

As σ_{k,j_k} is a uniformly random permutation on $[4n]$, it follows that π_k is also a uniformly random permutation on $[4n]$.

Conditioned on $\vec{j}, \vec{\pi}$, the marginal distribution of $\vec{b'} = (b_{k,i})_{k\in[K], i\in[n]\setminus\{i^\star\}}$ is identical. The marginal distribution of $\vec{b'}$ in the distribution $\mathcal{D}^{\mathsf{hard}}_{n,t,K}(Z^\star_z) \mid \vec{j}, \vec{\pi}$ is uniform conditioned on $\bigoplus_{k=1}^{K} b_{k,i} = \mathbb{1}(i \in Z^\star_z)$ for all $i \in [n] \setminus \{i^\star\}$. Using Line 3 and the fact that $\mathbb{1}(i \in Z^\star_1) = \mathbb{1}(i \in Z^\star_z)$ for all $i \in [n] \setminus \{i^\star\}$, the marginal distribution of $\vec{b'}$ in the distribution $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_z) \mid \vec{j}, \vec{\pi}$ is identical

Conditioned on $\vec{j}, \vec{\pi}, \vec{b}'$, the marginal distribution of $(b_{k,i^\star})_{k \in [K]}$ is identical. The marginal distribution of $(b_{k,i^\star})_{k \in [K]}$ in the distribution $\mathcal{D}^{\mathsf{hard}}_{n,t,K}(Z^\star_z) \mid \vec{j}, \vec{\pi}, \vec{b}'$ is simply the uniform distribution conditioned on $\bigoplus_{k=1}^{K} b_{k,i^\star} = \mathbb{1}(i^\star \in Z^\star_z)$. To show that the marginal distribution of $(b_{k,i^\star})_{k \in [K]}$ in the distribution $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_z) \mid \vec{j}, \vec{\pi}, \vec{b}'$ is identical, we in fact consider the stronger conditioning $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_z) \mid \vec{i}, \vec{j}, \vec{\pi}, \vec{b}'$. Observe that the latter is just $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_z, \vec{i}, \vec{j}) \mid \vec{\pi}, \vec{b}'$.

Now using the fact that Line 1, Line 2, and Line 3 can be carried out without knowing Alice's input we conclude that the marginal distribution of $(b_{k,i^\star})_{k \in [K]}$ in the distribution $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_z, \vec{i}, \vec{j}) \mid \vec{\pi}, \vec{b}'$ is the same as in the marginal distribution of X_{k,j_k,i_k} in the distribution $\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_z, \vec{i}, \vec{j}$ which by definition is uniformly random conditioned on $\bigoplus_{k=1}^{K} X_{k,j_k,i_k} = \mathbb{1}(i^\star \in Z^\star_z)$.

Conditioned on $\vec{j}, \vec{\pi}, \vec{b}$, the marginal distribution of $\vec{T}' = (T_{k,j})_{k \in [K], j \in [t] \setminus \{j_k\}}$ is identical. The marginal distribution of $\vec{T}'$ in the distribution $\mathcal{D}^{\mathsf{hard}}_{n,t,K}(Z^\star_z) \mid \vec{j}, \vec{\pi}, \vec{b}$ is simply uniform conditioned on each coordinate being of size $2n$. To show that the marginal distribution of $\vec{T}'$ in the distribution $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_z) \mid \vec{j}, \vec{\pi}, \vec{b}$ is identical, we consider the stronger conditioning $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_z) \mid \Upsilon_1$, where:

$$\Upsilon_1 = (\Upsilon_{1,1}, \Upsilon_{1,2}) \quad \text{where} \quad \Upsilon_{1,1} = \left(\vec{j}, \vec{i}, \vec{X}\right) \quad \text{and} \quad \Upsilon_{1,2} = \left(\vec{\pi}, \vec{b}', (\sigma_{k,j_k})_{k \in [K]}\right).$$

As $\Upsilon_{1,1}$ determines E_z, the latter is the same as $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid \Upsilon_{1,1}) \mid \Upsilon_{1,2}$ [12]. Observe that under this conditioning, whether or not $\sigma_{k,j}(i) \in T_{k,j}$ is fixed (see Line 4) for all $k \in [K]$, $j \in [t] \setminus \{j_k\}$ and $i \in [4n]$, and the only randomness in $\vec{T}'$ is over the choice of $(\sigma_{k,j})_{k \in [K], j \in [t] \setminus \{j_k\}}$. As these chosen uniformly and independently, we have that the marginal distribution of each coordinate of $\vec{T}'$ in the distribution $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid \Upsilon_{1,1}) \mid \Upsilon_{1,2}$ is simply uniform (and independent) conditioned on each coordinate being of size $2n$, as desired.

Conditioned on $\vec{j}, \vec{\pi}, \vec{b}, \vec{T}'$, the marginal distribution of $(T_{k,j_k})_{k \in [K]}$ is identical. The marginal distribution of $(T_{k,j_k})_{k \in [K]}$ in the distribution $\mathcal{D}^{\mathsf{hard}}_{n,t,K}(Z^\star_z) \mid \vec{j}, \vec{\pi}, \vec{b}, \vec{T}'$ is simply uniform conditioned on:

- For all $k \in [K]$, $|T_{k,j_k}| = 2n$.
- For all $k \in [K], i \in [n]$, we have $\pi_k^{-1}(i) \in T_{k,j_k}$ if and only if $b_{k,i} = 1$.

To show that the marginal distribution of $(T_{k,j_k})_{k \in [K]}$ in the distribution $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_z) \mid \vec{j}, \vec{\pi}, \vec{b}, \vec{T}'$ is identical, we consider the stronger conditioning $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid E_z) \mid \Upsilon_2$ where:

$$\Upsilon_2 = (\Upsilon_{2,1}, \Upsilon_{2,2}) \quad \text{where} \quad \Upsilon_{2,1} = \left(\vec{j}, \vec{i}, \vec{X}\right) \quad \text{and} \quad \Upsilon_{2,2} = \left(\vec{\pi}, \vec{b}', \vec{T}', (\sigma_{k,j})_{k \in [K], j \in [t] \setminus \{j_k\}}\right).$$

[12]The distribution $\mathcal{D}^{\mathsf{Ind}^\oplus}_{n+1,t,K} \mid \Upsilon_{1,1}$ is actually just a point mass.

As $\Upsilon_{1,1}$ determines E_z, the latter is the same as $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^{\oplus}}_{n+1,t,K} \mid \Upsilon_{2,1}) \mid \Upsilon_{2,2}$.[13] Observe from Line 4 that under this conditioning whether or not $\sigma_{k,j_k}(i) \in T_{k,j_k}$ is fixed for all $i \in [4n]$ and the only randomness $(T_{k,j_k})_{k\in[K]}$ is the randomness in σ_{k,j_k}. Also, observe from Line 2 that the marginal distribution of σ_{k,j_k} in the distribution $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^{\oplus}}_{n+1,t,K} \mid \Upsilon_{2,1}) \mid \Upsilon_{2,2}$ is independent for all $k \in [K]$ and simply uniform conditioned on:

- $\sigma_{k,j_k}(i_k) = \pi_k^{-1}(i^\star)$.
- For all $a < i^\star \in [n]$, $\sigma_{k,j_k}(2(n+1)+a) = \pi_k^{-1}(a)$.
- For all $a > i^\star \in [n]$, $\sigma_{k,j_k}(2(n+1)+a-1) = \pi_k^{-1}(a)$.

This implies that the marginal distribution of $(T_{k,j_k})_{k\in[K]}$ in the distribution $\Psi_{\mathsf{mid}}(\mathcal{D}^{\mathsf{Ind}^{\oplus}}_{n+1,t,K} \mid \Upsilon_{2,1}) \mid \Upsilon_{2,2}$ is also independent for all $k \in [K]$ and uniform conditioned on

- $|T_{k,j_k}| = 2n$.
- $\mathbb{1}\left(\pi_k^{-1}(i^\star) \in T_{k,j_k}\right) = b_{k,i^\star}$.
- For all $a < i^\star \in [n]$, $\mathbb{1}\left(\pi_k^{-1}(a) \in T_{k,j_k}\right) = b_{k,a}$.
- For all $a > i^\star \in [n]$, $\mathbb{1}\left(\pi_k^{-1}(a) \in T_{k,j_k}\right) = b_{k,a}$.

This can be rewritten as uniform conditioned on

- $|T_{k,j_k}| = 2n$ and
- for all $i \in [n]$, $\mathbb{1}\left(\pi_k^{-1}(i) \in T_{k,j_k}\right) = b_{k,i}$,

as desired. $\square$

8 Operations on Set-Encoding Graphs

In this section we present several operations on Set-Encoding graphs, which will be the building blocks for the more sophisticated construction in Section 9.

This section is organized as follows: In Section 8.1, we present the AND operation and the OR operation on Set-Encoding graphs. In Section 8.2, we first formally define Set-Encoding graph pairs, and then present the XOR operation on them.

8.1 The AND/OR Operation on Set-Encoding Graphs

In the following, we specify the AND and OR operations on Set-Encoding graphs.

[13] The distribution $\mathcal{D}^{\mathsf{Ind}^{\oplus}}_{n+1,t,K} \mid \Upsilon_{2,1}$ is actually just a point mass.

Construction of Graphs $\textsf{Set-Enc-AND}(G_S, G_T)$ **and** $\textsf{Set-Enc-OR}(G_S, G_T)$

- **Input:** For two sets $S, T \subseteq [n]$. We are given a $\textsf{Set-Enc}_n(S)$ graph G_S and a $\textsf{Set-Enc}_n(T)$ graph G_T.
- **Assumption:** We assume that G_S and G_T are both $(N, k, \vec{\ell})$ graphs.
- **Construction of** $\textsf{Set-Enc-AND}(G_S, G_T)$**:** $\textsf{Set-Enc-AND}(G_S, G_T) = G_S \odot G_T$. For simplicity we will also use $G_S \wedge G_T$ to denote $\textsf{Set-Enc-AND}(G_S, G_T)$.
- **Construction of** $\textsf{Set-Enc-OR}(G_S, G_T)$**:** We build a graph H with first layer and last layer both having n vertices. We add a copy of G_S and a copy of G_T in H, between the first and the last layer of H. Now for convenience we will use G_S and G_T to denote the corresponding subgraphs in H. H is then specified as follows:

  1. **Auxiliary Edges:** We add a matching between (1) $\textsf{First}(H)$ and $\textsf{First}(G_S)$; (2) $\textsf{First}(H)$ and $\textsf{First}(G_T)$; (3) $\textsf{Last}(G_S)$ and $\textsf{Last}(H)$; (4) $\textsf{Last}(G_T)$ and $\textsf{Last}(H)$. That is, in case (1), for each $i \in [n]$, we add an edge from $\textsf{First}(H)_{[i]}$ to $\textsf{First}(G_S)_{[i]}$; we add edges similarly in the other three cases.

     Let $E_{\textsf{first}}$ be the set of the edges added to H in Case (1) and Case (2) above, and $E_{\textsf{last}}$ be the set of the edges added to H in Case (3) and Case (4) above.
  2. **Layers:** The graph H has $k+2$ layers. For each $i \in [k]$, we set $V_{i+1}(H) = V_i(G_S) \cup V_i(G_T)$.
  3. **Edge-layer ordering:** Now we set $\vec{E}(H)$ as a list of $k+1$ sets of edges such that $\vec{E}(H)_1 = E_{\textsf{first}}$, $\vec{E}(H)_2 = E_{\textsf{last}}$, and $\vec{E}(H)_{i+2} = (\vec{E}_S)_i \cup (\vec{E}_T)_i$ for each $i \in [k-1]$. $\vec{\ell}(H)$ is set accordingly.

  We output $\textsf{Set-Enc-OR}(G_S, G_T) = H$. For simplicity we will also use $G_S \vee G_T$ to denote $\textsf{Set-Enc-OR}(G_S, G_T)$.

The following two observations are in order.

Observation 8.1 (AND/OR operations on two Set-Encoding graphs)**.**

1. $G_S \wedge G_T$ is an $(N', k', \vec{\ell'})$ graph, where $N' = 2N - n$, $k' = 2k - 1$ and $\vec{\ell'}$ only depends on $\vec{\ell}$.

2. $G_S \vee G_T$ is an $(N', k', \vec{\ell'})$ graph, where $N' = 2N + 2n$, $k' = k + 2$ and $\vec{\ell'}$ only depends on $\vec{\ell}$.

8.2 The XOR Operation on Set-Encoding Graph Pairs

Next, we formally define $\textsf{Set-Enc}_n(S)$ graph pairs.

Definition 8.2 ($\mathsf{Set\text{-}Enc}_n(S)$ graph pairs)**.** *A pair of graph* $\mathsf{pair}_S = (G_S, G_{\neg S})$ *is a* $\mathsf{Set\text{-}Enc}_n(S)$ *graph pair, if the following two conditions hold:*

1. *G_S is a $\mathsf{Set\text{-}Enc}_n(S)$ graph and $G_{\neg S}$ a $\mathsf{Set\text{-}Enc}_n(\neg S)$ graph.*
2. *G_S and $G_{\neg S}$ have the same number of layers and the same edge-layer ordering ($\vec{\ell}(G_S) = \vec{\ell}(G_{\neg S})$).*

For two layered graphs G_1 and G_2 with the same number of layers and the same edge-layer ordering, we use $\begin{bmatrix} G_1 \\ G_2 \end{bmatrix}$ to denote the graph H constructed by merging G_1 and G_2 in parallel. That is: (1) H has the same number of layers and the same edge-layer ordering as G_1 (and G_2); (2) Let k be the number of layers in H. For each $i \in [k]$, $V_i(H) = V_i(G_1) \cup V_i(G_2)$; (3) For each $i \in [k-1]$, $E_i(H) = E_i(G_1) \cup E_i(G_2)$.

We say pair_S is an $(N, k, \vec{\ell})$ graph pair, if both of the graphs in pair_S are $(N, k, \vec{\ell})$ graphs. For a distribution $\mathcal{D}_{\mathsf{pair}}$ on $\mathsf{Set\text{-}Enc}_n$ graph pairs, we use $\mathsf{stack}(\mathcal{D}_{\mathsf{pair}})$ to denote the distribution obtained by drawing $(A, B) \leftarrow \mathcal{D}^{\mathsf{pair}}$, and outputting $\begin{bmatrix} A \\ B \end{bmatrix}$.

Construction of Graph Pairs $\mathsf{Set\text{-}Enc\text{-}XOR}(\mathsf{pair}_S, \mathsf{pair}_T)$

- **Input:** For two sets $S, T \subseteq [n]$. We are given a $\mathsf{Set\text{-}Enc}_n(S)$ graph pair $\mathsf{pair}_S = (G_S, G_{\neg S})$ and a $\mathsf{Set\text{-}Enc}_n(T)$ graph pair $\mathsf{pair}_T = (G_T, G_{\neg T})$.
- **Assumption:** We assume that both of pair_S and pair_T are $(N, k, \vec{\ell})$ graph pairs.
- **Output:** We set
$$G_{S \oplus T} = (G_S \wedge G_{\neg T}) \vee (G_{\neg S} \wedge G_T),$$
and
$$G_{\neg(S \oplus T)} = (G_S \wedge G_T) \vee (G_{\neg S} \wedge G_{\neg T}).$$
We output $\mathsf{Set\text{-}Enc\text{-}XOR}(\mathsf{pair}_S, \mathsf{pair}_T) = (G_{S \oplus T}, G_{\neg(S \oplus T)})$. For simplicity, we will also use $\mathsf{pair}_S \oplus \mathsf{pair}_T$ to denote $\mathsf{Set\text{-}Enc\text{-}XOR}(\mathsf{pair}_S, \mathsf{pair}_T)$.

We make the following observation, which follows immediately from Observation 8.1.

Observation 8.3 (The XOR operation on two $\mathsf{Set\text{-}Encoding}$ graph pairs)**.** *$\mathsf{pair}_S \oplus \mathsf{pair}_T$ is a $\mathsf{Set\text{-}Enc}_n(S \oplus T)$ graph pair. Moreover, $\mathsf{pair}_S \oplus \mathsf{pair}_T$ is also an $(N', k', \vec{\ell'})$ graph pair, where $N' = 4N$, $k' = 2k + 1$ and $\vec{\ell'}$ only depends on $\vec{\ell}$.*

Construction of Graph Pairs $\bigoplus_{k=1}^{K} \mathsf{pair}_i$

- **Input:** Let $K \in \mathbb{N}$ be a power of 2. (If the number of graph pairs is not a power of 2, we can always pad some dummy $\mathsf{Set\text{-}Enc}_n(\emptyset)$ pairs to make it a power of 2.

This only causes a constant blow-up on the size of the final graph.)

For K sets $S_1, \dots, S_K \subseteq [n]$, we are given a $\mathsf{Set\text{-}Enc}_n(S_i)$ graph pair $\mathsf{pair}_i = (G_{S_i}, G_{\neg S_i})$ for each $i \in [K]$.

- **Assumption:** We assume that all the K graph pairs are $(N, k, \vec{\ell})$ graph pairs.
- **Output:** If $K = 2$, we output $\mathsf{pair}_1 \oplus \mathsf{pair}_2$. Otherwise, letting $\mathsf{pair}_\mathsf{A} = \bigoplus_{i=1}^{K/2} \mathsf{pair}_i$ and $\mathsf{pair}_\mathsf{B} = \bigoplus_{i=1}^{K/2} \mathsf{pair}_{i+K/2}$, we output $\mathsf{pair}_\mathsf{A} \oplus \mathsf{pair}_\mathsf{B}$.

The following observation follows immediately from Observation 8.3.

Observation 8.4 (The XOR operation on a list of Set-Encoding graph pairs)**.** *$\bigoplus_{k=1}^{K} \mathsf{pair}_i$ is a $\mathsf{Set\text{-}Enc}_n(\bigoplus_{k=1}^{K} S_i)$ graph pair. It is also an $(N', k', \vec{\ell}')$ graph pair where $N' = K^2 \cdot N$, $k' = K \cdot (k+1) - 1$ and $\vec{\ell}'$ only depends on $\vec{\ell}$.*

For a list of n distributions $(\mathcal{D}_i^{\mathsf{pair}})_{i=1}^n$, we define $\bigoplus_{i=1}^n \mathcal{D}_i^{\mathsf{pair}}$ as the distribution obtained by drawing $\mathsf{pair}_i \leftarrow \mathcal{D}_i^{\mathsf{pair}}$ independently, and outputting $\oplus_{i=1}^n \mathsf{pair}_i$. The following lemma will be useful for analyzing our construction in Section 9.

Lemma 8.5 (The XOR operation preserves indistinguishability)**.** *Let n be a power of 2, let $\vec{\mathcal{D}}^{\mathsf{pair}} = (\mathcal{D}_i^{\mathsf{pair}})_{i=1}^n$ and $\vec{\mathcal{E}}^{\mathsf{pair}} = (\mathcal{E}_i^{\mathsf{pair}})_{i=1}^n$ be two list of distributions on $\mathsf{Set\text{-}Enc}_n$ graph pairs. We further assume that all graph pairs in the support of the distributions in $\vec{\mathcal{D}}^{\mathsf{pair}}$ and $\vec{\mathcal{E}}^{\mathsf{pair}}$ are $(N, k, \vec{\ell})$ graph pairs. Let $\varepsilon \in \mathbb{R}_{\geq 0}^n$ be n non-negative parameters. If for each $i \in [n]$, $\mathsf{stack}(\mathcal{D}_i^{\mathsf{pair}})$ and $\mathsf{stack}(\mathcal{E}_i^{\mathsf{pair}})$ are ε_i-indistinguishable for p-pass streaming algorithms with space s, then $\mathsf{stack}(\bigoplus_{i=1}^n \mathcal{D}_i^{\mathsf{pair}})$ and $\mathsf{stack}(\bigoplus_{i=1}^n \mathcal{E}_i^{\mathsf{pair}})$ are $\|\varepsilon\|_1$-indistinguishable for p-pass streaming algorithms with space s.*

Proof. Clearly, the lemma is trivial when $n = 1$. In the following we will prove the general case when n is an arbitrary power of 2 by induction.

Let n be a power of 2 which is at least 2. By the induction hypothesis, we know that the lemma holds for $n/2$. We set

$$(\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \mathcal{W}) = \left(\bigoplus_{i=1}^{n/2} \mathcal{D}_i^{\mathsf{pair}}, \bigoplus_{i=1}^{n/2} \mathcal{D}_{i+n/2}^{\mathsf{pair}}, \bigoplus_{i=1}^{n/2} \mathcal{E}_i^{\mathsf{pair}}, \bigoplus_{i=1}^{n/2} \mathcal{E}_{i+n/2}^{\mathsf{pair}} \right),$$

$\overline{\varepsilon}^{(1)} = (\varepsilon_i)_{i=1}^{n/2}$ and $\overline{\varepsilon}^{(2)} = (\varepsilon_{i+n/2})_{i=1}^{n/2}$. Since the lemma holds for $n/2$, it follows that: (1) $\mathsf{stack}(\mathcal{X})$ and $\mathsf{stack}(\mathcal{Z})$ are $\|\overline{\varepsilon}^{(1)}\|_1$-indistinguishable; (2) $\mathsf{stack}(\mathcal{Y})$ and $\mathsf{stack}(\mathcal{W})$ are $\|\overline{\varepsilon}^{(2)}\|_1$-indistinguishable.

Let $\boldsymbol{X} \leftarrow \mathcal{X}$, and we write $\boldsymbol{X} = (\boldsymbol{X}_+, \boldsymbol{X}_-)$ to denote the two components of $\boldsymbol{X}$ (note that these two components may not be independent). Similarly, we define random variables $\boldsymbol{Y}$, $\boldsymbol{Z}$ and $\boldsymbol{W}$, and their components.

By definition, the distribution $\mathsf{stack}(\bigoplus_{i=1}^{n} \mathcal{D}_i^{\mathsf{pair}})$ samples $\boldsymbol{X} \leftarrow \mathcal{X}$, $\boldsymbol{Y} \leftarrow \mathcal{Y}$ independently and outputs

$$\boldsymbol{X} \oplus \boldsymbol{Y} = \begin{bmatrix} (\boldsymbol{X}_+ \wedge \boldsymbol{Y}_-) \vee (\boldsymbol{X}_- \wedge \boldsymbol{Y}_+) \\ (\boldsymbol{X}_- \wedge \boldsymbol{Y}_-) \vee (\boldsymbol{X}_+ \wedge \boldsymbol{Y}_+) \end{bmatrix}.$$

Similarly, the distribution $\mathsf{stack}(\bigoplus_{i=1}^{n} \mathcal{E}_i^{\mathsf{pair}})$ samples $\boldsymbol{Z} \leftarrow \mathcal{Z}$, $\boldsymbol{W} \leftarrow \mathcal{W}$ independently and outputs

$$\boldsymbol{Z} \oplus \boldsymbol{W} = \begin{bmatrix} (\boldsymbol{Z}_+ \wedge \boldsymbol{W}_-) \vee (\boldsymbol{Z}_- \wedge \boldsymbol{W}_+) \\ (\boldsymbol{Z}_- \wedge \boldsymbol{W}_-) \vee (\boldsymbol{Z}_+ \wedge \boldsymbol{W}_+) \end{bmatrix}.$$

Ignoring the fixed auxiliary edges added in the Set-Enc-OR operation, one can see that a streaming algorithm A reading samples from $\mathsf{stack}(\bigoplus_{i=1}^{n} \mathcal{D}_i^{\mathsf{pair}})$ is essentially reading $(\begin{bmatrix} \boldsymbol{X}_+ \\ \boldsymbol{X}_- \end{bmatrix}, \begin{bmatrix} \boldsymbol{Y}_+ \\ \boldsymbol{Y}_- \end{bmatrix})_{\mathsf{Seq}}$ and similarly reading samples from $\mathsf{stack}(\bigoplus_{i=1}^{n} \mathcal{E}_i^{\mathsf{pair}})$ is equivalent to reading $(\begin{bmatrix} \boldsymbol{Z}_+ \\ \boldsymbol{Z}_- \end{bmatrix}, \begin{bmatrix} \boldsymbol{W}_+ \\ \boldsymbol{W}_- \end{bmatrix})_{\mathsf{Seq}}$.[14]

Noting that $\begin{bmatrix} \boldsymbol{X}_+ \\ \boldsymbol{X}_- \end{bmatrix}$ distributes as $\mathsf{stack}(\mathcal{X})$ (and the same holds for $\mathcal{Y}$, $\mathcal{Z}$ and $\mathcal{W}$), and the pairs $(\mathsf{stack}(\mathcal{X}), \mathsf{stack}(\mathcal{Z}))$ and $(\mathsf{stack}(\mathcal{Y}), \mathsf{stack}(\mathcal{W}))$ are $\|\bar{\varepsilon}^{(1)}\|_1$- and $\|\bar{\varepsilon}^{(2)}\|_1$-indistinguishable, respectively. Applying the hybrid argument (Lemma 3.4) proves the base case.

□

9 Set-Hiding Generators from Perm-Hiding Generators

In this section we present a construction of Set-Hiding generators from Perm-Hiding generators, and prove Lemma 4.2.

This section is organized as follows: In Section 9.1, we construct the container graphs using the well-known Ruzsa-Szemerédi graphs. In Section 9.2, we define the Index-Encoding graphs and show that they can be used as selectors for the container graphs. In Section 9.3, we construct Index-Hiding generators from Perm-Hiding generators. In Section 9.4, we present the construction for Set-Hiding generators from Perm-Hiding generators.

9.1 Construction of Container Graphs Using Ruzsa-Szemerédi Graphs

Notation for Ruzsa-Szemerédi (RS) graphs. Before constructing the container graphs we need, we introduce some notation for RS graphs first. Recall that $N^{\mathsf{RS}}(n) = n^{1+c^{\mathsf{RS}}/\sqrt{\log n}}$ so that for sufficiently large n, there exists an (n,n)-RS graph [RS78] with $N^{\mathsf{RS}}(n)$ vertices on both sides. More specifically, we use RS_n to denote an (n,n)-RS graph with $N^{\mathsf{RS}}(n)$

[14]See Section 3.4 for the definition of $(\begin{bmatrix} \boldsymbol{X}_+ \\ \boldsymbol{X}_- \end{bmatrix}, \begin{bmatrix} \boldsymbol{Y}_+ \\ \boldsymbol{Y}_- \end{bmatrix})_{\mathsf{Seq}}$ and $(\begin{bmatrix} \boldsymbol{Z}_+ \\ \boldsymbol{Z}_- \end{bmatrix}, \begin{bmatrix} \boldsymbol{W}_+ \\ \boldsymbol{W}_- \end{bmatrix})_{\mathsf{Seq}}$.

vertices on both sides. For each $i \in [n]$, we construct two indexing functions

$$\mathsf{RS}_n^{(i)}\text{-}\mathsf{IndexL}, \mathsf{RS}_n^{(i)}\text{-}\mathsf{IndexR} \colon [n] \to \left[N^{\mathsf{RS}}(n)\right]$$

as follows: for each $j \in [n]$, $\mathsf{RS}_n^{(i)}\text{-}\mathsf{IndexL}(j)$ (resp. $\mathsf{RS}_n^{(i)}\text{-}\mathsf{IndexR}(j)$) is set to the index of the j-th vertex in the left (resp. right) side of the i-th matching in RS_n. In other words, the i-th matching in RS_n consists of n edges $\{(\mathsf{RS}_n^{(i)}\text{-}\mathsf{IndexL}(j), \mathsf{RS}_n^{(i)}\text{-}\mathsf{IndexR}(j))\}_{j\in[n]}$.

Construction of container graphs. We will need the following construction involving RS graphs. Given n sets $T_1, \dots, T_n \subseteq [m]$ such that $n \le m$, we define the graph $G = \mathsf{Container}_m((T_i)_{i=1}^n)$, which is a bipartite graph having $N = N^{\mathsf{RS}}(m)$ vertices on each side, as follows: For each $(i,j) \in [n] \times [m]$, let $a = \mathsf{RS}_m^{(i)}\text{-}\mathsf{IndexL}(j)$ and $b = \mathsf{RS}_m^{(i)}\text{-}\mathsf{IndexR}(j)$, we add an edge between $\mathsf{First}(G)_{[a]}$ and $\mathsf{Last}(G)_{[b]}$ if and only if $j \in T_i$. Note that the edge layer ordering for container graphs is trivial as there is only one layer.

9.2 Index-Encoding Graphs

Now we define the following two types of Index-Encoding graphs, which will be used to "select" one set from a container graph $\mathsf{Container}_m((T_i)_{i=1}^n)$.

Definition 9.1 (Left-Index-Encoding graphs and Right-Index-Encoding graphs)**.** *For $n, m \in \mathbb{N}$ such that $n \le m$, and for an injective function $f \colon [n] \to [m]$:*

1. *(Left-Index-Encoding **Graphs**) We say a layered graph G is an $\mathsf{LIndex\text{-}Enc}_{n,m}(f)$ graph (i.e., a Left-Index-Encoding graph for the function f), if (1) $|\mathsf{First}(G)| = n$ and $|\mathsf{Last}(G)| = m$ and (2) for each $(i,j) \in [n] \times [m]$, $\mathsf{First}(G)_{[i]}$ can reach $\mathsf{Last}(G)_{[j]}$ if and only if $f(i) = j$.*

2. *(Right-Index-Encoding **Graphs**) We say a layered graph G is an $\mathsf{RIndex\text{-}Enc}_{m,n}(f)$ graph (i.e., a Right-Index-Encoding graph for the function f), if (1) $|\mathsf{First}(G)| = m$ and $|\mathsf{Last}(G)| = n$ and (2) for each $(j,i) \in [m] \times [n]$, $\mathsf{First}(G)_{[j]}$ can reach $\mathsf{Last}(G)_{[i]}$ if and only if $f(i) = j$.*

Index-Encoding graphs as selectors. The following lemma formally states how to use two Index-Encoding graphs to select a set out of the n sets stored in a container graph $\mathsf{Container}_m((T_i)_{i=1}^n)$.

Lemma 9.2 (Selecting a set in a container graph)**.** *Let $n, m \in \mathbb{N}$ such that $n \le m$, and let $(T_i)_{i=1}^n$ be n subsets of $[m]$. Let $i \in [n]$ and $N = N^{\mathsf{RS}}(m)$. Suppose the following hold:*

1. *G_L is an $\mathsf{LIndex\text{-}Enc}_{m,N}(\mathsf{RS}_m^{(i)}\text{-}\mathsf{IndexL})$ graph;*

2. *G_R is an $\mathsf{RIndex\text{-}Enc}_{N,m}(\mathsf{RS}_m^{(i)}\text{-}\mathsf{IndexR})$ graph;*

3. $G_M = \mathsf{Container}_m((T_i)_{i=1}^n)$.

Then $G_L \odot G_M \odot G_R$ *is a* $\mathsf{Set\text{-}Enc}_m(T_i)$ *graph.*

Proof. Let $G = G_L \odot G_M \odot G_R$. From now on we will use G_L, G_M and G_R to denote the corresponding subgraphs in G.

Since $\mathsf{First}(G) = \mathsf{First}(G_L)$ and $\mathsf{Last}(G) = \mathsf{Last}(G_R)$, each of $\mathsf{First}(G)$ and $\mathsf{Last}(G)$ has exactly m vertices. Now for each $(a, b) \in [m] \times [m]$, by Definition 9.1, $\mathsf{First}(G)_{[a]}$ can only reach $\mathsf{First}(G_M)_{[x]}$ where $x = \mathsf{RS}_m^{(i)}\text{-}\mathsf{IndexL}(a)$, and only $\mathsf{Last}(G_M)_{[y]}$ can reach $\mathsf{Last}(G)_{[b]}$ where $y = \mathsf{RS}_m^{(i)}\text{-}\mathsf{IndexR}(b)$. From the construction of container graph G_M, $\mathsf{First}(G_M)_{[x]}$ can reach $\mathsf{Last}(G_M)_{[y]}$ if and only if both $a = b$ and $a \in T_i$. Therefore, G is a $\mathsf{Set\text{-}Enc}_m(T_i)$ graph. $\square$

9.3 Index-Hiding Generators

Similar to Definition 3.6 and Definition 3.7, we can define Left-Index-Hiding generators and Right-Index-Hiding generators as follows.

Definition 9.3 (ε-secure Index-Hiding generators)**.** *Let $n, m \in \mathbb{N}$ such that $n \le m$, and let $\mathcal{G}$ be a function from injective functions from $[n] \to [m]$ to distributions over layered graphs. We say $\mathcal{G}$ is ε-*Left-Index-Hiding *(resp. ε-*Right-Index-Hiding*) for injective functions from $[n] \to [m]$ against p-pass algorithms with space s, if the following statements hold:*

1. *For every injective function $f\colon [n] \to [m]$, $\mathcal{G}(f)$ is a distribution over* $\mathsf{LIndex\text{-}Enc}_{n,m}(f)$ *(resp.* $\mathsf{RIndex\text{-}Enc}_{m,n}(f)$*) graphs.*

2. *For every two injective functions $f_1, f_2\colon [n] \to [m]$, the distributions $A(\mathcal{G}(f_1))$ and $A(\mathcal{G}(f_2))$ are ε-indistinguishable for p-pass streaming algorithms A with space s.*

Similar to $\mathcal{G}^{\mathsf{SH}}_{n,p}$ and $\mathcal{G}^{\mathsf{PH}}_{n,p}$, for simplicity, we will often use $\mathcal{G}^{\mathsf{LIH}}_{n,m}$ (resp. $\mathcal{G}^{\mathsf{RIH}}_{m,n}$) to denote an ε-Left-Index-Hiding (resp. ε-Right-Index-Hiding) generator for injective functions from $[n] \to [m]$. We may also write $\mathcal{G}^{\mathsf{LIH}}_{n,m,p}$ (resp. $\mathcal{G}^{\mathsf{RIH}}_{m,n,p}$) to indicate that the generator is against p-pass streaming algorithms.

Next, we show how to construct Index-Hiding generators from Perm-Hiding generators.

Construction of $\mathcal{G}^{\mathsf{LIH}}_{n,m,p}$ and $\mathcal{G}^{\mathsf{RIH}}_{m,n,p}$ for p-Pass Streaming Algorithms

- **Input:** An injective function $f\colon [n] \to [m]$. The security parameter ε, pass number p and space s.

- **Assumption:** There is a generator $\mathcal{G}^{\mathsf{PH}}_{m,p}$ which always outputs $(N_{\mathcal{G}^{\mathsf{PH}}}, k_{\mathcal{G}^{\mathsf{PH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{PH}}})$ graphs and is ε-Perm-Hiding against p-pass algorithms with space s.

- **Construction:**

  1. ($\mathcal{G}^{\mathsf{LIH}}_{n,m,p}$) Draw π as a uniformly random permutation on $[m]$ conditioning

on that $\pi(i) = f(i)$ for every $i \in [n]$. Let $G \leftarrow \mathcal{G}^{\mathsf{PH}}_{m,p}(\pi)$. Delete the vertices $\mathsf{First}(G)_{[n+1]}, \dots, \mathsf{First}(G)_{[m]}$ from G together with all edges adjacent to them. $\mathcal{G}^{\mathsf{LIH}}_{n,m,p}(f)$ then outputs G.

2. ($\mathcal{G}^{\mathsf{RIH}}_{m,n,p}$) Draw π as a uniformly random permutation on $[m]$ conditioning on that $\pi(f(i)) = i$ for every $i \in [n]$. Let $G \leftarrow \mathcal{G}^{\mathsf{PH}}_{m,p}(\pi)$. Delete the vertices $\mathsf{Last}(G)_{[n+1]}, \dots, \mathsf{Last}(G)_{[m]}$ from G together with all edges adjacent to them. $\mathcal{G}^{\mathsf{LIH}}_{n,m,p}(f)$ then output G.

Lemma 9.4 (From Perm-Hiding generators to Index-Hiding generators)**.** *Let $n, m, p, s \in \mathbb{N}$ such that $n \le m$. Let $\varepsilon \in [0, 1)$. Suppose there is a generator $\mathcal{G}^{\mathsf{PH}}_{m,p}$ which always outputs $(N_{\mathcal{G}^{\mathsf{PH}}}, k_{\mathcal{G}^{\mathsf{PH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{PH}}})$ graphs and is ε-Perm-Hiding against p-pass streaming algorithms with space s. Then, the generator $\mathcal{G}^{\mathsf{LIH}}_{n,m,p}$ (resp. $\mathcal{G}^{\mathsf{RIH}}_{m,n,p}$) as defined above always outputs $(N_{\mathcal{G}^{\mathsf{PH}}} - (m-n), k_{\mathcal{G}^{\mathsf{PH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{PH}}})$ graphs and is ε-Left-Index-Hiding (resp. ε-Right-Index-Hiding) against p-pass streaming algorithms with space s.*

Proof. The security of $\mathcal{G}^{\mathsf{LIH}}_{n,m,p}$ and $\mathcal{G}^{\mathsf{RIH}}_{m,n,p}$ follows directly from the assumed security of $\mathcal{G}^{\mathsf{PH}}_{m,p}$. Also, from the construction of $\mathcal{G}^{\mathsf{LIH}}_{n,m,p}$ and $\mathcal{G}^{\mathsf{RIH}}_{m,n,p}$, it is clear that we delete exactly $(m-n)$ vertices from the output graph of $\mathcal{G}^{\mathsf{PH}}_{m,p}$, hence $\mathcal{G}^{\mathsf{LIH}}_{n,m,p}$ and $\mathcal{G}^{\mathsf{RIH}}_{m,n,p}$ always output $(N_{\mathcal{G}^{\mathsf{PH}}} - (m-n), k_{\mathcal{G}^{\mathsf{PH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{PH}}})$ graphs. □

9.4 Construction of Set-Hiding Generators

Now we are ready to present our construction of Set-Hiding generators from Perm-Hiding generators.

Construction of $\mathcal{G}^{\mathsf{SH}}_{n,p}$ for p-Pass Streaming Algorithms

- **Input:** A set $S \subseteq [n]$. The security parameter ε, pass number p and space s.
- **Assumption:** Let $N = N^{\mathsf{RS}}(4n)$. There is a generator $\mathcal{G}^{\mathsf{PH}}_{N,p-1}$ which always outputs $(N_{\mathcal{G}^{\mathsf{PH}}}, k_{\mathcal{G}^{\mathsf{PH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{PH}}})$ graphs and is ε_0-Perm-Hiding against $(p-1)$-pass algorithms with space $2p \cdot s$, where $\varepsilon_0 = \varepsilon / \log^2 n$.
- **Setup:** Let $\mathcal{G}^{\mathsf{LIH}}_{4n,N,p-1}$ and $\mathcal{G}^{\mathsf{RIH}}_{N,4n,p-1}$ be the resulting ε_0-Left-Index-Hiding generator and ε_0-Right-Index-Hiding generator from Lemma 9.4, respectively.

  The generator $\mathcal{G}^{\mathsf{PH}}_{N,p-1}$ also implies a generator $\mathcal{G}^{\mathsf{PH}}_{4n,p-1}$ which is ε_0-Perm-Hiding against $(p-1)$-pass algorithms with space $2p \cdot s$.
- **Construction:**

  1. Let $K = 1500 \cdot \log n$.

2. Draw $(\vec{T}, \vec{\pi}, \vec{j}) \leftarrow \mathcal{D}^{\mathsf{hard}}_{n,n,K}(S)$ (defined in Section 7).
3. For each $k \in [K]$, we draw
   (a) $G^{(k)}_{\mathsf{pmL}} \leftarrow \mathcal{G}^{\mathsf{PH}}_{4n,p-1}(\pi_k^{-1})$; $G^{(k)}_{\mathsf{pmR}} \leftarrow \mathcal{G}^{\mathsf{PH}}_{4n,p-1}(\pi_k)$;
   (b) $G^{(k)}_{\mathsf{idxL}} \leftarrow \mathcal{G}^{\mathsf{LIH}}_{4n,N,p-1}(\mathsf{RS}^{(j_k)}_{4n}\text{-}\mathsf{IndexL})$; $G^{(k)}_{\mathsf{idxR}} \leftarrow \mathcal{G}^{\mathsf{RIH}}_{N,4n,p-1}(\mathsf{RS}^{(j_k)}_{4n}\text{-}\mathsf{IndexR})$;

   Then we set

$$X^{(k)} = G^{(k)}_{\mathsf{pmL}} \odot G^{(k)}_{\mathsf{idxL}} \odot \quad \mathsf{Container}_{4n}((T_{k,j})_{j\in[n]}) \qquad \odot G^{(k)}_{\mathsf{idxR}} \odot G^{(k)}_{\mathsf{pmR}},$$
$$Y^{(k)} = G^{(k)}_{\mathsf{pmL}} \odot G^{(k)}_{\mathsf{idxL}} \odot \quad \mathsf{Container}_{4n}((\neg_{4n} T_{k,j})_{j\in[n]}) \quad \odot G^{(k)}_{\mathsf{idxR}} \odot G^{(k)}_{\mathsf{pmR}}, \text{ and}$$
$$\mathsf{pair}_k = (X^{(k)}, Y^{(k)}).$$

4. Let $\mathsf{pair}_{\mathsf{xor}} = \oplus_{k=1}^{K} \mathsf{pair}_k$, and G_{xor} be the first graph in $\mathsf{pair}_{\mathsf{xor}}$.
5. **Edge-Layer Ordering:**
   We have to adjust the edge-layer ordering in G_{xor}. There are three possible types of edge-layers in G_{xor} (see Claim 9.5): (1) consisting entirely of the auxiliary edges added in the Set-Enc-OR operations; (2) consisting entirely of the edges in some copies of $\mathsf{Container}_{4n}((\neg T_{k,j})_{j\in[n]})$; (3) consisting entirely of the edges in some copies of $G^{(k)}_{\mathsf{pmL}}$, $G^{(k)}_{\mathsf{idxL}}$, $G^{(k)}_{\mathsf{idxR}}$ or $G^{(k)}_{\mathsf{pmR}}$.
   Let E_{aux}, E_{A} and E_{B} be the lists of edge-layers in $E(G_{\mathsf{xor}})$ of type (1), (2) and (3), respectively. Each of them is ordered according to $\vec{E}(G_{\mathsf{xor}})$. We construct another graph G_{adjust} from G_{xor} such that they are the same except for the edge-layer ordering; we set $E(G_{\mathsf{adjust}}) = E_{\mathsf{aux}} \circ E_{\mathsf{A}} \circ E_{\mathsf{B}}$, and $\vec{\ell}(G_{\mathsf{adjust}})$ is set accordingly.
6. Finally, we construct a new graph G_{final} by deleting the vertices $\mathsf{First}(G_{\mathsf{adjust}})_{[i]}$ and $\mathsf{Last}(G_{\mathsf{adjust}})_{[i]}$ from G_{adjust} for every $i \in \{n+1, \ldots, 4n\}$.

- **Output:** Finally, $\mathcal{G}^{\mathsf{SH}}_{n,p}$ outputs G_{final}.

Reminder of Lemma 4.2. *Let n be a sufficiently large integer. Let $p, s \in \mathbb{N}$ such that $2p \cdot s \le \frac{1}{20} n^2 \log n$, and let $N = N^{\mathsf{RS}}(4n)$. Let $\varepsilon \in [0,1)$ such that $\varepsilon \ge 1/n^{10}$. Suppose there is a generator $\mathcal{G}^{\mathsf{PH}}_{N,p-1}$ which always outputs $(N_{\mathcal{G}^{\mathsf{PH}}}, k_{\mathcal{G}^{\mathsf{PH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{PH}}})$ graphs and is $(\varepsilon/\log^2 n)$-*Perm-Hiding *against $(p-1)$-pass streaming algorithms with space $2p \cdot s$. Then there is a generator $\mathcal{G}^{\mathsf{SH}}_{n,p}$ such that: (1) it always outputs $(N_{\mathcal{G}^{\mathsf{SH}}}, k_{\mathcal{G}^{\mathsf{SH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{SH}}})$ graphs, where $N_{\mathcal{G}^{\mathsf{SH}}} = O(N_{\mathcal{G}^{\mathsf{PH}}} \cdot \log^2 n)$ and $k_{\mathcal{G}^{\mathsf{SH}}} = O(k_{\mathcal{G}^{\mathsf{PH}}} \cdot \log n)$; (2) it is ε-*Set-Hiding *against p-pass streaming algorithms with space s.*

Proof. We begin with some notation. We use $\vec{T}$ to denote $(T_{k,j})_{(j,k)\in[n]\times[K]}$, $\vec{\pi}$ to denote $(\pi_i)_{i\in[K]}$, and $\vec{j}$ to denote $(j_i)_{i\in[K]}$. We also let S_{Alice} and S_{Bob} be the sets of all possible $\vec{T}$ and $(\vec{\pi}, \vec{j})$, respectively.

Note that an output graph G_{final} of $\mathcal{G}^{\mathsf{SH}}_{n,p}$ can be decomposed into three subgraphs G_{fixed}, G_{A} and G_{B} (corresponding to the edge-layers E_{aux}, E_{A} and E_{B} in the construction), such that G_{fixed} is a fixed graph, G_{A} is a deterministic function of $\vec{T}$ and G_{B} is generated from $\vec{\pi}$ and $\vec{j}$. Note that edges comes in the order of G_{fixed}, G_{A} and G_{B}.

We use $G_{\mathsf{A}}(\vec{T})$ to denote the graph G_{A} constructed from $\vec{T}$, and $\mathcal{G}_{\mathsf{B}}(\vec{\pi}, \vec{j})$ to denote the distribution of the graph G_{B} given inputs $\vec{\pi}$ and $\vec{j}$. We use $\operatorname{supp}(\mathcal{G}_{\mathsf{B}})$ to denote the union of the supports of $\mathcal{G}_{\mathsf{B}}(\vec{\pi}, \vec{j})$ for every $(\vec{\pi}, \vec{j}) \in S_{\mathsf{Bob}}$.

G_{final} is a $\mathsf{Set\text{-}Enc}_n(S)$ graph. We first claim that G_{final} is a $\mathsf{Set\text{-}Enc}_n(S)$ graph. By the definition of $\mathcal{G}^{\mathsf{PH}}_{4n,p-1}$ and Lemma 9.2, it follows that for each $k \in [K]$, pair_k is a $\mathsf{Set\text{-}Enc}_{4n}(\pi_k(T_{k,j_k}))$ graph pair. Then by Observation 8.4, $\mathsf{pair}_{\mathsf{xor}}$ is a $\mathsf{Set\text{-}Enc}_{4n}(\bigoplus_{k=1}^{K} \pi_k(T_{k,j_k}))$ graph pair. Therefore, both of G_{xor} and G_{adjust} are $\mathsf{Set\text{-}Enc}_{4n}(\bigoplus_{k=1}^{K} \pi_k(T_{k,j_k}))$ graphs.

Finally, since G_{final} is obtained by deleting $\mathsf{First}(G_{\mathsf{adjust}})_{[i]}$ and $\mathsf{Last}(G_{\mathsf{adjust}})_{[i]}$ for every $i \in \{n+1, \ldots, 4n\}$, it follows that G_{final} is a $\mathsf{Set\text{-}Enc}_n(\bigoplus_{k=1}^{K} \pi_k(T_{k,j_k}) \cap [n])$ graph. From Observation 7.2, $S = \bigoplus_{k=1}^{K} \pi_k(T_{k,j_k}) \cap [n]$ and the claim is proved.

Structure of the graph G_{xor}. Next, we need the following claim summarizing the structure properties of G_{xor} needed for us.

Claim 9.5 (Structure properties of G_{xor})**.** *G_{xor} is an $(N_{\mathsf{xor}}, k_{\mathsf{xor}}, \vec{\ell}_{\mathsf{xor}})$ graph, where $N_{\mathsf{xor}} = O(\log^2 n \cdot N_{\mathcal{G}^{\mathsf{PH}}})$ and $k_{\mathsf{xor}} = O(\log n \cdot k_{\mathcal{G}^{\mathsf{PH}}})$. Moreover, there are three possible types of edge-layers in G_{xor}: (1) consisting entirely of the auxiliary edges added in the* $\mathsf{Set\text{-}Enc\text{-}OR}$ *operations; (2) consisting entirely of the edges in some copies of* $\mathsf{Container}_{4n}((\neg T_{k,j})_{j\in[n]})$*; (3) consisting entirely of the edges in some copies of $G^{(k)}_{\mathsf{pmL}}$, $G^{(k)}_{\mathsf{idxL}}$, $G^{(k)}_{\mathsf{idxR}}$ or $G^{(k)}_{\mathsf{pmR}}$.*

In the following, we prove Claim 9.5. Note that the container graphs $\mathsf{Container}_{4n}((T_{k,j})_{j\in[n]})$ and $\mathsf{Container}_{4n}((\neg_{4n} T_{k,j})_{j\in[n]})$ each has $2N$ vertices. By the properties of $\mathcal{G}^{\mathsf{PH}}_{4n,p-1}$, $\mathcal{G}^{\mathsf{LIH}}_{4n,N,p-1}$ and $\mathcal{G}^{\mathsf{RIH}}_{N,4n,p-1}$, it follows that for each $k \in [K]$, pair_k is an $(N_{\mathsf{pair}}, k_{\mathsf{pair}}, \vec{\ell}_{\mathsf{pair}})$ graph pair, where $N_{\mathsf{pair}} = O(N_{\mathcal{G}^{\mathsf{PH}}} + N) = O(N_{\mathcal{G}^{\mathsf{PH}}})$, $k_{\mathsf{pair}} = O(k_{\mathcal{G}^{\mathsf{PH}}})$ and $\vec{\ell}_{\mathsf{pair}}$ only depends on $\vec{\ell}_{\mathcal{G}^{\mathsf{PH}}}$.

By Observation 8.4 and noting that $K = O(\log n)$, it follows that G_{xor} is an $(N_{\mathsf{xor}}, k_{\mathsf{xor}}, \vec{\ell}_{\mathsf{xor}})$ graph, where $N_{\mathsf{xor}} = O(\log^2 n \cdot N_{\mathcal{G}^{\mathsf{PH}}})$ and $k_{\mathsf{xor}} = O(\log n \cdot k_{\mathcal{G}^{\mathsf{PH}}})$. This proves the first statement of Claim 9.5. For the moreover part, note that for each $k \in [K]$, edge-layers in both graphs in pair_k are either type (2) or type (3). Since all the graph pairs pair_k have the same edge-layer ordering, one can show that $\bigoplus_{k=1}^{K} \mathsf{pair}_k$ maintains this property by induction. Also, it is not hard to see that if an edge-layer contains some auxiliary edges added in the $\mathsf{Set\text{-}Enc\text{-}OR}$ operations, then it consists entirely of auxiliary edges. This completes the proof of Claim 9.5.

It then follows directly from Claim 9.5 that $\mathcal{G}^{\mathsf{SH}}_{n,p}$ always outputs $(N_{\mathcal{G}^{\mathsf{SH}}}, k_{\mathcal{G}^{\mathsf{SH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{SH}}})$ graphs, where $N_{\mathcal{G}^{\mathsf{SH}}} = O(\log^2 n \cdot N_{\mathcal{G}^{\mathsf{PH}}})$, $k_{\mathcal{G}^{\mathsf{SH}}} = O(\log n \cdot k_{\mathcal{G}^{\mathsf{PH}}})$.

Indistinguishability. To simplify the analysis, in the rest of the proof, we will assume that $\mathcal{G}^{\mathsf{SH}}_{n,p}$ outputs G_{adjust} instead of G_{final}. This is valid since for every $T_1, T_2 \subseteq [n]$, this change will only increase the total variation distance $\|A(\mathcal{G}^{\mathsf{SH}}_{n,p}(T_1)) - A(\mathcal{G}^{\mathsf{SH}}_{n,p}(T_2))\|_{\mathrm{TV}}$.

Suppose for the sake of contradiction that $\mathcal{G}^{\mathsf{SH}}_{n,p}$ is not ε-Set-Hiding for p-pass streaming algorithms with space s. That is, there are two subsets $T_1, T_2 \subseteq [n]$ and a p-pass streaming algorithm A with space s such that

$$\|A(\mathcal{G}^{\mathsf{SH}}_{n,p}(T_1)) - A(\mathcal{G}^{\mathsf{SH}}_{n,p}(T_2))\|_{\mathrm{TV}} > \varepsilon. \tag{10}$$

We can further assume without loss of generality that A is a *deterministic algorithm.* We will show how to construct a one-way communication protocol from Alice to Bob which contradicts Lemma 7.4. Setting $t = n$ and $K = 1500 \cdot \log n$ in Lemma 7.4, we have the following claim.

Claim 9.6 (An instantiation of Lemma 7.4). *For a sufficiently large integer n. Let $T_1, T_2 \subseteq [n]$ be sets. For all one-way randomized protocols Π from Alice to Bob for the* Set-Hiding-Game *satisfying* $\mathsf{CC}(\Pi) \le \frac{1}{20} n^2 \log n$, *it holds that*

$$\left\|\Pi\left(\mathcal{D}^{\mathsf{hard}}_{n,n,K}(T_1)\right) - \Pi\left(\mathcal{D}^{\mathsf{hard}}_{n,n,K}(T_2)\right)\right\|_{\mathrm{TV}} \le \frac{1}{n^{30}}.$$

In more details, plugging in $t = n$ and $K = 1500 \cdot \log n$ in Lemma 7.4, we have $\delta = \frac{100}{K} = \frac{1}{15 \log n}$ and $C = \frac{1}{20} \cdot (2nt)^{1-5\delta} = \frac{1}{20} \cdot (2n^2)^{1-1/3 \log n} \ge \frac{1}{20} n^2$. We can also verify that $tK \le 10C$, and Claim 9.6 follows from the fact that $C \log n \ge \frac{1}{20} n^2 \log n$.

We first consider a faithful simulation of the p-pass algorithm A on the graph G_{final} by Alice and Bob, specified by Algorithm 2.

The following lemma will be crucial for our proof, and we defer its proof until we finish the proof of Lemma 4.2.

Lemma 9.7 (Resampling preserves the distributions). *There is a mapping $\mathcal{G}_{\mathsf{resamp}}$ from $\{0,1\}^{(2p-1)s} \times S_{\mathsf{Bob}}$ to distributions over $\mathrm{supp}(\mathcal{G}_{\mathsf{B}})$ such that for every distribution $\mathcal{D}$ over $S_{\mathsf{Alice}} \times S_{\mathsf{Bob}}$, letting $(\vec{\boldsymbol{T}}, \vec{\boldsymbol{\pi}}, \vec{\boldsymbol{j}}) \leftarrow \mathcal{D}$, the following two distributions are identical:*

1. *(The Original Distribution.)* $\Pi_{\mathsf{sim}}(G_{\mathsf{A}}(\vec{\boldsymbol{T}}), \mathcal{G}_{\mathsf{B}}(\vec{\boldsymbol{\pi}}, \vec{\boldsymbol{j}}))$.
2. *(The Resampled Distribution.) Draw* $\boldsymbol{M} \leftarrow \Pi^{\mathsf{trans}}_{\mathsf{sim}}(G_{\mathsf{A}}(\vec{\boldsymbol{T}}), \mathcal{G}_{\mathsf{B}}(\vec{\boldsymbol{\pi}}, \vec{\boldsymbol{j}}))$, *and output* $A_{\mathsf{Last}}(\boldsymbol{M}, \mathcal{G}_{\mathsf{resamp}}(\boldsymbol{M}, \vec{\boldsymbol{\pi}}, \vec{\boldsymbol{j}}))$.

   In above $A_{\mathsf{Last}}(x, y)$ is the output of the protocol Π_{sim} given the transcript x and Bob's graph y. In other words, $A_{\mathsf{Last}}(x, y)$ is computed by Bob simulating A using his graph y and the last state recorded in the transcript x and then outputting A's output.

The lemma above essentially says that, if Bob *has forgotten* his graph G_{B} right before the $2p$-th round in the protocol Π_{sim}, using its input $(\vec{\pi}, \vec{j})$ and the transcript M between Alice and Bob, he can still *resample* a graph G'_{B} from the distribution $\mathcal{G}_{\mathsf{resamp}}(M, \vec{\pi}, \vec{j})$ so

Algorithm 2 Protocol $\Pi_{\mathsf{sim}}(G_{\mathsf{A}}, G_{\mathsf{B}})$.

```
Input: Alice gets $G_{\mathsf{A}} = G_{\mathsf{A}}(\vec{T})$ for some $\vec{T} \in S_{\mathsf{Alice}}$, and Bob gets $G_{\mathsf{B}} \in \mathrm{supp}(\mathcal{G}_{\mathsf{B}})$.
1: Alice and Bob jointly simulate the algorithm $A$ on the graph $G_{\mathsf{fixed}} \cup G_{\mathsf{A}} \cup G_{\mathsf{B}}$ in $2p$ rounds. Alice starts with the initial state of the algorithm $A$. For each $i \in [p]$, Alice and Bob proceed as follows:
    • in the $(2i-1)$-th round, Alice simulates $A$ on $G_{\mathsf{fixed}}$ and $G_{\mathsf{A}}$ (note that the ordering of subgraphs is $G_{\mathsf{fixed}}$, $G_{\mathsf{A}}$ and $G_{\mathsf{B}}$) using the initial state of $A$ (if $i = 1$) or the most recent state received from Bob (if $i > 1$), and then sends the final state of $A$ to Bob;
    • in the $2i$-th round, Bob simulates $A$ on $G_{\mathsf{B}}$ using the most recent received state from Alice, and sends $A$'s state back to Alice (if $i < p$) or outputs $A$'s output (if $i = p$).
   The output of the protocol $\Pi_{\mathsf{sim}}$ is the final output of Bob.
Output: We use $\Pi^{\mathsf{trans}}_{\mathsf{sim}}(G_{\mathsf{A}}, G_{\mathsf{B}})$ to denote the concatenation of messages sent between Alice and Bob in the first $(2p-1)$ rounds (that is, the transcript of the protocol $\Pi_{\mathsf{sim}}$), and $\Pi_{\mathsf{sim}}(G_{\mathsf{A}}, G_{\mathsf{B}})$ to denote the output of the protocol given inputs $G_{\mathsf{A}}$ and $G_{\mathsf{B}}$. Since $A$ is deterministic, both of $\Pi^{\mathsf{trans}}_{\mathsf{sim}}(G_{\mathsf{A}}, G_{\mathsf{B}})$ and $\Pi_{\mathsf{sim}}(G_{\mathsf{A}}, G_{\mathsf{B}})$ are deterministic functions of $G_{\mathsf{A}}$ and $G_{\mathsf{B}}$.
```

that continuing the simulation of A with G'_{B} instead of G_{B} does not change the final output distribution.

In the following we fix a subset $S \subseteq [n]$ and consider a protocol Π for Set-Hiding-Game with input distribution $\mathcal{D}^{\mathsf{hard}}_{n,n,K}(S)$, specified by Algorithm 3.

We first make three observations:

1. $\boldsymbol{O}_{\mathsf{output\text{-}Bob}}$ distributes as $\Pi(\mathcal{D}^{\mathsf{hard}}_{n,n,K}(S))$[15];
2. $\boldsymbol{G}_{\mathsf{real}}$ distributes as $\mathcal{G}^{\mathsf{SH}}_{n,p}(S)$, therefore $\boldsymbol{O}_{\mathsf{output\text{-}real}}$ distributes as $A(\mathcal{G}^{\mathsf{SH}}_{n,p}(S))$ by Lemma 9.7;
3. $\mathsf{CC}(\Pi) \leq 2p \cdot s \leq \frac{1}{20} n^2 \log n$.

The following claim follows from our assumption on the generator $\mathcal{G}^{\mathsf{PH}}_{N,p-1}$, and we defer its proof until we finish the proof of Lemma 4.2.

Claim 9.8 ($\boldsymbol{M}_{\mathsf{real}}$ is close to $\boldsymbol{M}_{\mathsf{Alice}}$). *For every $(\vec{T}, \vec{\pi}, \vec{j}) \in \mathrm{supp}(\mathcal{D}^{\mathsf{hard}}_{n,n,K}(S))$, it holds that*

$$\left\| \left[\boldsymbol{M}_{\mathsf{real}} \mid (\vec{\boldsymbol{T}}, \vec{\boldsymbol{\pi}}, \vec{\boldsymbol{j}}) = (\vec{T}, \vec{\pi}, \vec{j}) \right] - \left[\boldsymbol{M}_{\mathsf{Alice}} \mid (\vec{\boldsymbol{T}}, \vec{\boldsymbol{\pi}}, \vec{\boldsymbol{j}}) = (\vec{T}, \vec{\pi}, \vec{j}) \right] \right\|_{\mathrm{TV}} < \varepsilon/3.$$

[15] $\Pi(\mathcal{D}^{\mathsf{hard}}_{n,n,K}(S))$ denotes the distribution of the output of the protocol Π running on the input drawn from the distribution $\mathcal{D}^{\mathsf{hard}}_{n,n,K}(S)$.

Algorithm 3 Protocol Π.

Input: Let $\vec{\boldsymbol{T}} = (T_{k,j})_{(j,k)\in[n]\times[K]}$, $\vec{\boldsymbol{\pi}} = (\pi_i)_{i\in[K]}$ and $\vec{\boldsymbol{j}} = (j_i)_{i\in[K]}$ be a sample drawn from $\mathcal{D}^{\mathsf{hard}}_{n,n,K}(S)$. Alice is given $\vec{\boldsymbol{T}}$; Bob is given $\vec{\boldsymbol{\pi}}$ and $\vec{\boldsymbol{j}}$. Alice then sets $\boldsymbol{G}_{\mathsf{A}} = G_{\mathsf{A}}(\vec{\boldsymbol{T}})$, and Bob draws $\boldsymbol{G}_{\mathsf{B}}$ from the distribution $\mathcal{G}_{\mathsf{B}}(\vec{\boldsymbol{\pi}}, \vec{\boldsymbol{j}})$. We use $\boldsymbol{G}_{\mathsf{real}}$ to denote the graph $G_{\mathsf{fixed}} \cup \boldsymbol{G}_{\mathsf{A}} \cup \boldsymbol{G}_{\mathsf{B}}$.

1: Alice draws $\widetilde{\boldsymbol{G}}_{\mathsf{B}}$ from the distribution $\mathcal{G}_{\mathsf{B}}(\vec{\pi}_{\mathsf{A}}, \vec{j}_{\mathsf{A}})$, such that $\vec{\pi}_{\mathsf{A}}$ consists of K identity permutations, and $\vec{j}_{\mathsf{A}}$ is the all-one vector of length K. Alice then simulates the protocol Π_{sim} on the graphs $\boldsymbol{G}_{\mathsf{A}}$ and $\widetilde{\boldsymbol{G}}_{\mathsf{B}}$, to compute the random variable $\boldsymbol{M}_{\mathsf{Alice}} = \Pi^{\mathsf{trans}}_{\mathsf{sim}}(\boldsymbol{G}_{\mathsf{A}}, \widetilde{\boldsymbol{G}}_{\mathsf{B}})$.
We also let $\boldsymbol{M}_{\mathsf{real}}$ be the random variable $\Pi^{\mathsf{trans}}_{\mathsf{sim}}(\boldsymbol{G}_{\mathsf{A}}, \boldsymbol{G}_{\mathsf{B}})$ (for analysis only).

2: Then Alice sends $\boldsymbol{M}_{\mathsf{Alice}}$ to Bob and it takes $(2p-1)\cdot s$ bits. Bob then draws a graph $\boldsymbol{G}'_{\mathsf{B}}$ from $\mathcal{G}_{\mathsf{resamp}}(\boldsymbol{M}_{\mathsf{Alice}}, \vec{\boldsymbol{\pi}}, \vec{\boldsymbol{j}})$, and outputs $A_{\mathsf{Last}}(\boldsymbol{M}_{\mathsf{Alice}}, \boldsymbol{G}'_{\mathsf{B}})$.

Output: We use $\boldsymbol{O}_{\mathsf{output\text{-}Bob}}$ to denote the output of Bob, and we use $\boldsymbol{O}_{\mathsf{output\text{-}real}}$ to denote the output of Bob if he received the message $\boldsymbol{M}_{\mathsf{real}}$ instead of $\boldsymbol{M}_{\mathsf{Alice}}$.

Note that conditioning on $(\vec{\boldsymbol{T}}, \vec{\boldsymbol{\pi}}, \vec{\boldsymbol{j}}) = (\vec{T}, \vec{\pi}, \vec{j})$ for a tuple $(\vec{T}, \vec{\pi}, \vec{j}) \in \mathrm{supp}(\mathcal{D}^{\mathsf{hard}}_{n,n,K}(S))$, $\boldsymbol{O}_{\mathsf{output\text{-}Bob}}$ and $\boldsymbol{O}_{\mathsf{output\text{-}real}}$ are obtained by applying the same randomized procedure to $\boldsymbol{M}_{\mathsf{real}}$ and $\boldsymbol{M}_{\mathsf{Alice}}$, respectively.[16] By Claim 9.8, it follows that $\|\boldsymbol{O}_{\mathsf{output\text{-}Bob}} - \boldsymbol{O}_{\mathsf{output\text{-}real}}\|_{\mathrm{TV}} < \varepsilon/3$.

Recall that $\boldsymbol{O}_{\mathsf{output\text{-}Bob}}$ distributes as $\Pi(\mathcal{D}^{\mathsf{hard}}_{n,n,K}(S))$ and $\boldsymbol{O}_{\mathsf{output\text{-}real}}$ distributes as $A(\mathcal{G}^{\mathsf{SH}}_{n,p}(S))$, it follows that

$$\|\Pi(\mathcal{D}^{\mathsf{hard}}_{n,n,K}(S)) - A(\mathcal{G}^{\mathsf{SH}}_{n,p}(S))\|_{\mathrm{TV}} < \varepsilon/3$$

for every $S \subseteq [n]$.

Combing with Equation 10 and noting that $\varepsilon \ge 1/n^{10}$ and $n \ge 3$, it follows that

$$\|\Pi(\mathcal{D}^{\mathsf{hard}}_{n,n,K}(T_1)) - \Pi(\mathcal{D}^{\mathsf{hard}}_{n,n,K}(T_2))\|_{\mathrm{TV}} > \varepsilon/3 > \frac{1}{n^{30}},$$

contradicting Claim 9.6 since $\mathsf{CC}(\Pi) \le \frac{1}{20} n^2 \log n$. $\square$

Now, we first prove Lemma 9.7.

Proof of Lemma 9.7. We begin by introducing some notation.

Notation. For every possible $\vec{T}, \vec{\pi}, \vec{j} \in S_{\mathsf{Alice}} \times S_{\mathsf{Bob}}$, we use $\mathcal{D}_{\mathsf{MB}}(\vec{T}, \vec{\pi}, \vec{j})$ to denote the following induced joint distribution on $\{0,1\}^{(2p-1)s} \times S_{\mathsf{Bob}}$: we draw $G_{\mathsf{B}} \leftarrow \mathcal{G}_{\mathsf{B}}(\vec{\pi}, \vec{j})$, set $M = \Pi^{\mathsf{trans}}_{\mathsf{sim}}(G_{\mathsf{A}}(\vec{T}), G_{\mathsf{B}})$, and output (M, G_{B}).

[16] In more details, conditioning on $(\vec{\boldsymbol{T}}, \vec{\boldsymbol{\pi}}, \vec{\boldsymbol{j}}) = (\vec{T}, \vec{\pi}, \vec{j})$, for every $m \in \{0,1\}^{(2p-1)\cdot s}$, we can define $\mathcal{Z}(m)$ be the distribution obtained by drawing $G_{\mathsf{B}} \leftarrow \mathcal{G}_{\mathsf{resamp}}(m, \vec{\pi}, \vec{j})$ and outputting $A_{\mathsf{Last}}(m, G_{\mathsf{B}})$. Then $\boldsymbol{M}_{\mathsf{Alice}}$ and $\boldsymbol{M}_{\mathsf{real}}$ distributes as $\mathcal{Z}(\boldsymbol{M}_{\mathsf{Alice}})$ and $\mathcal{Z}(\boldsymbol{M}_{\mathsf{real}})$, respectively.

For a distribution $\mathcal{D}$ on $S_{\mathsf{Alice}} \times S_{\mathsf{Bob}}$, let $\mathcal{D}_{\mathsf{origin}}(\mathcal{D})$ and $\mathcal{D}_{\mathsf{resamp}}(\mathcal{D})$ be the original distribution and resampled distribution from the lemma statement, respectively. Using the above notation $\mathcal{D}_{\mathsf{MB}}$, we further define the following two distributions

$$\mathcal{H}_{\mathsf{origin}}(\mathcal{D}) = \mathop{\mathbb{E}}_{(\vec{T},\vec{\pi},\vec{j}) \leftarrow \mathcal{D}} [\mathcal{D}_{\mathsf{MB}}(\vec{T},\vec{\pi},\vec{j})]$$

and

$$\mathcal{H}_{\mathsf{resamp}}(\mathcal{D}) = \mathop{\mathbb{E}}_{(\vec{T},\vec{\pi},\vec{j}) \leftarrow \mathcal{D}} \; \mathop{\mathbb{E}}_{(M,G_{\mathsf{B}}) \leftarrow \mathcal{D}_{\mathsf{MB}}(\vec{T},\vec{\pi},\vec{j})} [(M, \mathcal{G}_{\mathsf{resamp}}(M,\pi,\vec{j})],$$

where $(M, \mathcal{G}_{\mathsf{resamp}}(M,\pi,\vec{j}))$ denotes the distribution obtained by drawing a sample $G_{\mathsf{B}} \leftarrow \mathcal{G}_{\mathsf{resamp}}(M,\pi,\vec{j})$ and then output (M, G_{B}). ($\mathcal{G}_{\mathsf{resamp}}(M,\pi,\vec{j})$ will be defined shortly.)

Observe that $\mathcal{D}_{\mathsf{origin}}(\mathcal{D})$ (resp. $\mathcal{D}_{\mathsf{resamp}}(\mathcal{D})$) can be obtained by drawing a sample (M, G_{B}) from $\mathcal{H}_{\mathsf{origin}}(\mathcal{D})$ (resp. $\mathcal{H}_{\mathsf{resamp}}(\mathcal{D})$) and then output $A_{\mathsf{Last}}(M, G_{\mathsf{B}})$. Hence, to establish the theorem, it suffices to construct the function $\mathcal{G}_{\mathsf{resamp}}$ so that $\mathcal{H}_{\mathsf{origin}}(\mathcal{D})$ is identical to $\mathcal{H}_{\mathsf{resamp}}(\mathcal{D})$, for every distribution $\mathcal{D}$ on $S_{\mathsf{Alice}} \times S_{\mathsf{Bob}}$.

Construction of $\mathcal{G}_{\mathsf{resamp}}$. Now we specify our construction of $\mathcal{G}_{\mathsf{resamp}}$. For $M \in \{0,1\}^{(2p-1)s}$, by the rectangular property of communication protocols, there exists two subsets $U_M \subseteq \{G_{\mathsf{A}}(\vec{T}) \mid \vec{T} \in S_{\mathsf{Alice}}\}$ and $V_M \subseteq \operatorname{supp}(\mathcal{G}_{\mathsf{B}})$, such that $\Pi^{\mathsf{trans}}_{\mathsf{sim}}(G_{\mathsf{A}}(\vec{T}), G_{\mathsf{B}}) = M$ if and only if $G_{\mathsf{A}}(\vec{T}) \in U_M$ and $G_{\mathsf{B}} \in V_M$.

We now set $\mathcal{G}_{\mathsf{resamp}}(M,\vec{\pi},\vec{j})$ to be the conditional distribution on the set V_M induced by $\mathcal{G}_{\mathsf{B}}(\vec{\pi},\vec{j})$.

The key property of $\mathcal{G}_{\mathsf{resamp}}$. For every possible $(\vec{T}, M, \vec{\pi}, \vec{j})$, we consider the distribution $\mathcal{G}_{\mathsf{B}}(\vec{T}, M, \vec{\pi}, \vec{j})$ obtained by outputting $G_{\mathsf{B}} \leftarrow \mathcal{G}_{\mathsf{B}}(\vec{\pi},\vec{j})$ conditioning on the event that $\Pi^{\mathsf{trans}}_{\mathsf{sim}}(G_{\mathsf{A}}(\vec{T}), G_{\mathsf{B}}) = M$. Note that $\mathcal{G}_{\mathsf{B}}(\vec{T}, M, \vec{\pi}, \vec{j})$ can be undefined if $\Pi^{\mathsf{trans}}_{\mathsf{sim}}(G_{\mathsf{A}}(\vec{T}), G_{\mathsf{B}}) = M$ happens with zero probability. The following claim is crucial for us.

Claim 9.9. *$\mathcal{G}_{\mathsf{B}}(\vec{T}, M, \vec{\pi}, \vec{j})$ is either undefined, or equals to $\mathcal{G}_{\mathsf{resamp}}(M,\vec{\pi},\vec{j})$.*

Proof. Recall that $\Pi^{\mathsf{trans}}_{\mathsf{sim}}(G_{\mathsf{A}}(\vec{T}), G_{\mathsf{B}}) = M$ is equivalent to that $G_{\mathsf{A}}(\vec{T}) \in U_M$ and $G_{\mathsf{B}} \in V_M$ for two sets U_M and V_M. If $G_{\mathsf{A}}(\vec{T}) \notin U_M$, $\mathcal{G}_{\mathsf{B}}(\vec{T}, M, \vec{\pi}, \vec{j})$ is clearly undefined.

Otherwise, we have $G_{\mathsf{A}}(\vec{T}) \in U_M$. In this case we have $G_{\mathsf{B}} \in V_M$, and one can see that $\mathcal{G}_{\mathsf{B}}(\vec{T}, M, \vec{\pi}, \vec{j})$ is the induced conditional distribution of $\mathcal{G}_{\mathsf{B}}(\vec{\pi},\vec{j})$ on the set V_M, which is exactly $\mathcal{G}_{\mathsf{resamp}}(M,\vec{\pi},\vec{j})$. □

Proof of Correctness For every possible $(\vec{T},\vec{\pi},\vec{j})$, note that $\mathcal{D}_{\mathsf{MB}}(\vec{T},\vec{\pi},\vec{j})$ can alternatively be generated as follows: draw $(M, G_{\mathsf{B}}) \leftarrow \mathcal{D}_{\mathsf{MB}}(\vec{T},\vec{\pi},\vec{j})$, and then draw $G'_{\mathsf{B}} \leftarrow \mathcal{G}_{\mathsf{B}}(\vec{T}, M, \vec{\pi}, \vec{j})$, and output (M, G'_{B}).

Using the above observation, for every possible distribution $\mathcal{D}$ on $S_{\mathsf{Alice}} \times S_{\mathsf{Bob}}$, we have

$$\|\mathcal{H}_{\mathsf{origin}}(\mathcal{D}) - \mathcal{H}_{\mathsf{resamp}}(\mathcal{D})\|_{\mathrm{TV}}$$

$$\leq \mathop{\mathbb{E}}_{(\vec{T},\vec{\pi},\vec{j})\leftarrow\mathcal{D}} \left\| \mathcal{D}_{\mathsf{MB}}(\vec{T},\vec{\pi},\vec{j}) - \mathop{\mathbb{E}}_{(M,G_{\mathsf{B}})\leftarrow\mathcal{D}_{\mathsf{MB}}(\vec{T},\vec{\pi},\vec{j})} [(M, \mathcal{G}_{\mathsf{resamp}}(M,\pi,\vec{j})] \right\|_{\mathrm{TV}}$$

$$\leq \mathop{\mathbb{E}}_{(\vec{T},\vec{\pi},\vec{j})\leftarrow\mathcal{D}} \left\| \mathop{\mathbb{E}}_{(M,G_{\mathsf{B}})\leftarrow\mathcal{D}_{\mathsf{MB}}(\vec{T},\vec{\pi},\vec{j})} [(M, \mathcal{G}_{\mathsf{B}}(\vec{T},M,\pi,\vec{j})] - \mathop{\mathbb{E}}_{(M,G_{\mathsf{B}})\leftarrow\mathcal{D}_{\mathsf{MB}}(\vec{T},\vec{\pi},\vec{j})} [(M, \mathcal{G}_{\mathsf{resamp}}(M,\pi,\vec{j})] \right\|_{\mathrm{TV}}.$$

$$\leq \mathop{\mathbb{E}}_{(\vec{T},\vec{\pi},\vec{j})\leftarrow\mathcal{D}} \mathop{\mathbb{E}}_{(M,G_{\mathsf{B}})\leftarrow\mathcal{D}_{\mathsf{MB}}(\vec{T},\vec{\pi},\vec{j})} \left\| \mathcal{G}_{\mathsf{B}}(\vec{T},M,\pi,\vec{j}) - \mathcal{G}_{\mathsf{resamp}}(M,\pi,\vec{j}) \right\|_{\mathrm{TV}}.$$

Applying Claim 9.9 to the above equality concludes the proof. □

Finally, we prove Claim 9.8.

Proof of Claim 9.8. We first fix a tuple $(\vec{T},\vec{\pi},\vec{j}) \in \mathrm{supp}(\mathcal{D}^{\mathsf{hard}}_{n,n,K}(S))$, and we will condition on $(\boldsymbol{\vec{T}},\boldsymbol{\vec{\pi}},\boldsymbol{\vec{j}}) = (\vec{T},\vec{\pi},\vec{j})$.

Since the G_{fixed} part is always fixed, and the G_{A} part is fixed to $G_{\mathsf{A}}(\vec{T})$ as we are conditioning on $\boldsymbol{\vec{T}} = \vec{T}$, it means that A can be seen as an algorithm reading edge-layers in G_{B}, in the order specified by $\vec{\ell}(G_{\mathsf{B}})$. We use $B_{\vec{T}}$ to denote a new streaming algorithm on G_{B}, which simulates A on $G_{\mathsf{fixed}} \cup G_{\mathsf{A}} \cup G_{\mathsf{B}}$, and outputs $\Pi^{\mathsf{trans}}_{\mathsf{sim}}(G_{\mathsf{A}}, G_{\mathsf{B}})$.

We claim that $B_{\vec{T}}$ can be implemented by a $2p \cdot s$-space, $(p-1)$-pass algorithm. The space bound follows from the observation that $B_{\vec{T}}$ can use s space to simulate A and $(2p-1)\cdot s$ additional space to store the intermediate states appearing in $\Pi^{\mathsf{trans}}_{\mathsf{sim}}(G_{\mathsf{A}}, G_{\mathsf{B}})$. $B_{\vec{T}}$ only takes $(p-1)$ passes because to compute $\Pi^{\mathsf{trans}}_{\mathsf{sim}}(G_{\mathsf{A}}, G_{\mathsf{B}})$, it does not have to go over G_{B} in the p-th pass.

Furthermore, noting that $\widetilde{\boldsymbol{G}}_{\mathsf{B}}$ is independent of $(\boldsymbol{\vec{T}},\boldsymbol{\vec{\pi}},\boldsymbol{\vec{j}})$, the claim reduces to prove

$$\| B_{\vec{T}}(\widetilde{\boldsymbol{G}}_{\mathsf{B}}) - B_{\vec{T}}(\boldsymbol{G}_{\mathsf{B}} | (\boldsymbol{\vec{\pi}},\boldsymbol{\vec{j}}) = (\vec{\pi},\vec{j})) \|_{\mathrm{TV}} < \varepsilon/3.$$

For $(\vec{\pi},\vec{j}) \in S_{\mathsf{Bob}}$, for each $k \in [K]$, we construct the following distribution $\mathcal{P}^{(k)}_{\vec{\pi},\vec{j}}$ on graph pairs:

Construction of The Distribution $\mathcal{P}^{(k)}_{\vec{\pi},\vec{j}}$

1. Draw $G^{(k)}_{\mathsf{pmL}} \leftarrow \mathcal{G}^{\mathsf{PH}}_{4n,p-1}(\pi_k^{-1})$ and $G^{(k)}_{\mathsf{pmR}} \leftarrow \mathcal{G}^{\mathsf{PH}}_{4n,p-1}(\pi_k)$.
2. Draw $G^{(k)}_{\mathsf{idxL}} \leftarrow \mathcal{G}^{\mathsf{LIH}}_{4n,N,p-1}(\mathsf{RS}^{(j_k)}_{4n}\text{-}\mathsf{IndexL})$ and $G^{(k)}_{\mathsf{idxR}} \leftarrow \mathcal{G}^{\mathsf{RIH}}_{N,4n,p-1}(\mathsf{RS}^{(j_k)}_{4n}\text{-}\mathsf{IndexR})$.
3. We set

$$\begin{aligned} X^{(k)} &= G^{(k)}_{\mathsf{pmL}} \odot G^{(k)}_{\mathsf{idxL}} \odot \quad \mathsf{Container}_{4n}(([4n])_{j\in[n]}) \quad \odot G^{(k)}_{\mathsf{idxR}} \odot G^{(k)}_{\mathsf{pmR}}, \\ Y^{(k)} &= G^{(k)}_{\mathsf{pmL}} \odot G^{(k)}_{\mathsf{idxL}} \odot \quad \mathsf{Container}_{4n}((\emptyset)_{j\in[n]}) \quad \odot G^{(k)}_{\mathsf{idxR}} \odot G^{(k)}_{\mathsf{pmR}}. \end{aligned}$$

   In above $([4n])_{j\in[n]}$ (resp. $(\emptyset)_{j\in[n]}$) denotes a list of n sets, each being $[4n]$ (resp. empty set).[a]

4. **Output:** The graph pair $(X^{(k)}, Y^{(k)})$.

[a]We added these dummy container graphs to make sure $(X^{(k)}, Y^{(k)})$ is a graph pair.

We will show that for every two pairs $(\vec{\pi}, \vec{j})$ and $(\vec{\pi}', \vec{j}')$ from S_{Bob},

$$\left\| B_{\vec{T}}\left(\mathsf{stack}\left(\bigoplus_{k=1}^{K} \mathcal{P}^{(k)}_{\vec{\pi},\vec{j}}\right)\right) - B_{\vec{T}}\left(\mathsf{stack}\left(\bigoplus_{k=1}^{K} \mathcal{P}^{(k)}_{\vec{\pi}',\vec{j}'}\right)\right)\right\|_{TV} \le 4 \cdot K \cdot \varepsilon_0 < \varepsilon/3, \tag{11}$$

the last inequality above follows from $K = O(\log n)$ and $\varepsilon_0 = \varepsilon/\log^2 n$, in the following we establish the first inequality above.

Note that reading $\mathsf{stack}(\mathcal{P}^{(k)}_{\vec{\pi},\vec{j}})$ is essentially reading $\left(G^{(k)}_{\mathsf{pmL}}, G^{(k)}_{\mathsf{idxL}}, G^{(k)}_{\mathsf{idxR}}, G^{(k)}_{\mathsf{pmR}}\right)_{\mathsf{seq}}$. Combing the hybrid argument (Lemma 3.4) with the ε_0-indistinguishability of $\mathcal{G}^{\mathsf{PH}}_{4n,p-1}$, $\mathcal{G}^{\mathsf{LIH}}_{4n,N,p-1}$ and $\mathcal{G}^{\mathsf{RIH}}_{N,4n,p-1}$, it follows that

$$\left\| B_{\vec{T}}\left(\mathsf{stack}(\mathcal{P}^{(k)}_{\vec{\pi},\vec{j}})\right) - B_{\vec{T}}\left(\mathsf{stack}(\mathcal{P}^{(k)}_{\vec{\pi}',\vec{j}'})\right)\right\|_{TV} \le 4 \cdot \varepsilon_0. \tag{12}$$

Combining Equation 12 and Lemma 8.5 proves Equation 11.

Finally, one can see that $\widetilde{\boldsymbol{G}}_{\mathsf{B}}$ distributes as $\left(\bigoplus_{k=1}^{K} \mathcal{P}^{(k)}_{\vec{\pi}_{\mathsf{A}},\vec{j}_{\mathsf{A}}}\right)_1$ if we remove all its dummy container layers (that is, removing all container layers in the first graph from the graph pair $\bigoplus_{k=1}^{K} \mathcal{P}^{(k)}_{\vec{\pi}_{\mathsf{A}},\vec{j}_{\mathsf{A}}}$) and $\boldsymbol{G}_{\mathsf{B}} | (\boldsymbol{\vec{\pi}}, \boldsymbol{\vec{j}}) = (\vec{\pi}, \vec{j})$ distributes as $\left(\bigoplus_{k=1}^{K} \mathcal{P}^{(k)}_{\vec{\pi},\vec{j}}\right)_1$ if we also removes all its dummy container layers. (Recall that now we are assuming $\mathcal{G}^{\mathsf{SH}}_{n,p}$ outputs G_{adjust} instead of G_{final} to simplify the analysis.) The lemma then follows from Equation 11.

□

Finally, we remark that if we use the construction of Ruzsa-Szemerédi graphs in Proposition 3.3, we can prove the following variant of Lemma 4.2.

Remark 9.10 (From Perm-Hiding generators to Set-Hiding generators, Version 2)**.** *There is an absolute constant $c \in (0, 1)$ such that the following holds. Let n be a sufficiently large integer. Let $p, s \in \mathbb{N}$ such that $s \le n^{1+c/\log\log n}$, and let $N = 16n$. Let $\varepsilon \in [0, 1)$ such that $\varepsilon \ge 1/n^{10}$. Suppose there is a generator $\mathcal{G}^{\mathsf{PH}}_{N,p-1}$ which always outputs $(N_{\mathcal{G}^{\mathsf{PH}}}, k_{\mathcal{G}^{\mathsf{PH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{PH}}})$ graphs and is $(\varepsilon/\log^2 n)$-Perm-Hiding against $(p-1)$-pass streaming algorithms with space $2p\cdot s$. Then there is a generator $\mathcal{G}^{\mathsf{SH}}_{n,p}$ such that: (1) it always outputs $(N_{\mathcal{G}^{\mathsf{SH}}}, k_{\mathcal{G}^{\mathsf{SH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{SH}}})$ graphs, where $N_{\mathcal{G}^{\mathsf{SH}}} = O(N_{\mathcal{G}^{\mathsf{PH}}} \cdot \log^2 n)$ and $k_{\mathcal{G}^{\mathsf{SH}}} = O(k_{\mathcal{G}^{\mathsf{PH}}} \cdot \log n)$; (2) it is ε-Set-Hiding against p-pass streaming algorithms with space s.*

10 Perm-Hiding Generators from Set-Hiding Generators

In this section, we show how to obtain Perm-Hiding generators from Set-Hiding generators, and prove Lemma 4.3. The major idea behind our construction is using a sorting network.

There is a long list of work studying sorting networks [ON62, Bat68, AKS83, Pat90, AHRV07, Goo14]. Our results in this section rely on [AKS83].

This section is organized as follows: In Section 10.1, we state a useful lemma which is proved by the well-known AKS construction of sorting networks. In Section 10.2, we define and construct $\mathsf{Perm}(M)$-Hiding generators for permutations in $\mathsf{Perm}(M)$, where M is a matching on $[n]$. Finally, in Section 10.3 we present the construction of Perm-Hiding generators by composing many $\mathsf{Perm}(M)$-Hiding generators.

10.1 Decomposition of Permutations via Low-depth Sorting Networks

Let M be a (potentially partial) matching on a graph with vertices set $[n]$ (we will just say M is matching on $[n]$ for brevity). For convenience, we define a function $f_M\colon [n] \to [n]$ such that: (1) if vertex a is matched to vertex b in the matching M, then $f_M(a) = b$ and $f_M(b) = a$; (2) if vertex a is not matched to any other vertex in the matching M, then $f_M(a) = a$.

For a matching M on $[n]$, we define $\mathsf{Perm}(M)$ to be the set of permutations which can be implemented via swapping some matched pairs in M. That is, a permutation $\pi \in \mathsf{Perm}(M)$ if and only if for every $a \in [n]$, $\pi(a)$ equals a or $f_M(a)$. (Note that for an edge $(a,b) \in M$, if $\pi(a) = f_M(a) = b$, then since π is a permutation, the foregoing condition forces $\pi(b) = a$.)

We need the following lemma from the well-known $O(\log n)$-depth construction of sorting networks.

Lemma 10.1 ([AKS83])**.** *There exists an absolute constant $c_{\mathsf{AKS}} > 1$ such that for every integer n, there exist $d = c_{\mathsf{AKS}} \cdot \log n$ matchings $M_1, M_2, \dots, M_{d-1}, M_d$ on $[n]$ such that the following holds: For each permutation π on $[n]$, there exist d permutations $\pi_1, \pi_2, \dots, \pi_{d-1}, \pi_d$ such that $\pi_i \in \mathsf{Perm}(M_i)$ for each $i \in [d]$ and $\pi = \pi_d \circ \pi_{d-1} \circ \cdots \circ \pi_2 \circ \pi_1$.*

To utilize the lemma above in our construction, we need the following observation on Perm-$\mathsf{Encoding}$ graphs.

Observation 10.2 (Composition of Perm-$\mathsf{Encoding}$ graphs)**.** *Let $n, d \in \mathbb{N}$, and $\pi_1, \pi_2, \dots, \pi_{d-1}, \pi_d$ be d permutations on $[n]$. Let $G_1, G_2, \dots, G_{d-1}, G_d$ be d layered graphs such that for each $i \in [d]$, G_i is a Perm-$\mathsf{Enc}_n(\pi_i)$ graph. Then $G_1 \odot G_2 \odot \cdots \odot G_{d-1} \odot G_d$ is a Perm-$\mathsf{Enc}_n(\pi)$ graph for $\pi = \pi_d \circ \pi_{d-1} \circ \cdots \circ \pi_2 \circ \pi_1$.*

10.2 Perm-Hiding Generators for Permutations in $\mathsf{Perm}(M)$

From Lemma 10.1 and Observation 10.2, to construct a Perm-Hiding generator for all permutations on $[n]$, the first step is to construct a Perm-Hiding generator for all permutations in a particular set $\mathsf{Perm}(M)$, where M is a matching on $[n]$. We first formulate the formal definition of such a Perm-Hiding generator.

Definition 10.3 (ε-secure $\mathsf{Perm\text{-}Hiding}$ generators for permutations in $\mathsf{Perm}(M)$)**.** *Let n be an integer and M be a matching on $[n]$. Let $\mathcal{G}$ be a function from permutations in $\mathsf{Perm}(M)$ to distributions over layered graphs. We say $\mathcal{G}$ is ε-$\mathsf{Perm}(M)$-Hiding for permutations in $\mathsf{Perm}(M)$ against p-pass algorithms with space s, if the following statements hold:*

1. *For every $\pi \in \mathsf{Perm}(M)$, $\mathcal{G}(\pi)$ is a distribution over $\mathsf{Perm\text{-}Enc}_n(\pi)$ graphs.*

2. *For every two permutations $\pi_1, \pi_2 \in \mathsf{Perm}(M)$, the distributions $A(\mathcal{G}(\pi_1))$ and $A(\mathcal{G}(\pi_2))$ are ε-indistinguishable for p-pass streaming algorithms with space s.*

Construction of $\mathcal{G}^{\mathsf{PH}}_{M,n,p}$ for p-Pass Streaming Algorithms

- **Input:** A matching M on $[n]$ and a permutation $\pi \in \mathsf{Perm}(M)$. The security parameter ε, pass number p and space s.

- **Assumption:** There is a generator $\mathcal{G}^{\mathsf{SH}}_{3n,p}$, which always outputs $(N_{\mathcal{G}^{\mathsf{SH}}}, k_{\mathcal{G}^{\mathsf{SH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{SH}}})$ graphs and is ε-$\mathsf{Set\text{-}Hiding}$ against p-pass streaming algorithms with space s. Let $N = 2 \cdot |M| + n \le 3n$. One can easily construct another generator $\mathcal{G}^{\mathsf{SH}}_{N,p}$ from $\mathcal{G}^{\mathsf{SH}}_{3n,p}$.

- **Construction:**

  1. We first construct a set S of size N as follows: for each $a \in [n]$, we add the pair (a, a) to S; for each edge $(a, b) \in M$ (note that $a \neq b$), we add both (a, b) and (b, a) to S.

  2. Let ψ_M be a bijection between S and $[N]$ (ψ_M only depends on the input M), and let $T = \{\psi_M((i, \pi(i))) : i \in [n]\}$. Note that $T \subseteq [N]$, and we draw $H \sim \mathcal{G}^{\mathsf{SH}}_{N,p}(T)$.

  3. Now we construct a new graph G such that G contains a copy of H, together with two new layers U and V, each of size n, such that U is the first layer and V is the last layer.

  4. From now on we use H to denote the corresponding subgraph in G. For each $(x, y) \in S$, letting $\ell = \psi_M((x, y))$, we add (1) an edge in G from $U_{[x]}$ to $\mathsf{First}(H)_{[\ell]}$ and (2) an edge from $\mathsf{Last}(H)_{[\ell]}$ to $V_{[y]}$. Let E_{first} be the set of edges added in Case (1), and E_{last} be the set of edges added in Case (2).

  5. **List of edge sets and edge-layer ordering:** Let k be the number of layers in H. Note that G has $k+2$ layers. Now we set $\vec{E}(G)$ as a list of $k+1$ sets of edges, such that $\vec{E}(G)_1 = E_{\mathsf{first}}$, $\vec{E}(G)_2 = E_{\mathsf{last}}$, and $\vec{E}(G)_{i+2} = \vec{E}(H)_i$ for each $i \in [k-1]$. $\vec{\ell}(G)$ is also set accordingly.

  6. Finally, $\mathcal{G}^{\mathsf{PH}}_{M,n,p}$ outputs G.

Lemma 10.4 (From $\mathsf{Set\text{-}Hiding}$ generators to $\mathsf{Perm}(M)$-Hiding generators)**.** *Let $n, s \in \mathbb{N}$ and let $\varepsilon \in [0, 1)$. Suppose there is a generator $\mathcal{G}^{\mathsf{SH}}_{3n,p}$ which always outputs $(N_{\mathcal{G}^{\mathsf{SH}}}, k_{\mathcal{G}^{\mathsf{SH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{SH}}})$ graphs and is ε-$\mathsf{Set\text{-}Hiding}$ against p-pass streaming algorithms with space s. Then, for all matchings M on $[n]$, there is a generator $\mathcal{G}^{\mathsf{PH}}_{M,n,p}$ such that: (1) it always outputs $(N_{\mathcal{G}^{\mathsf{PH}}}, k_{\mathcal{G}^{\mathsf{PH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{PH}}})$ graphs where $N_{\mathcal{G}^{\mathsf{PH}}} \le O(N_{\mathcal{G}^{\mathsf{SH}}} + n)$ and $k_{\mathcal{G}^{\mathsf{PH}}} = k_{\mathcal{G}^{\mathsf{SH}}} + 2$; (2) it is ε-$\mathsf{Perm}(M)$-Hiding against p-pass streaming algorithms with space s.*

Proof. First, for two permutations $\pi_1, \pi_2 \in \mathsf{Perm}(M)$, the ε-indistinguishability between $\mathcal{G}^{\mathsf{PH}}_{M,n,p}(\pi_1)$ and $\mathcal{G}^{\mathsf{PH}}_{M,n,p}(\pi_2)$ follows from the assumption that $\mathcal{G}^{\mathsf{SH}}_{3n,p}$ is ε-$\mathsf{Set\text{-}Hiding}$. Also, one can directly verify that number of vertices in the output graphs of $\mathcal{G}^{\mathsf{PH}}_{M,n,p}$ is at most $N_{\mathcal{G}^{\mathsf{SH}}} + 2n$, and the construction adds exactly two layers (hence $k_{\mathcal{G}^{\mathsf{PH}}} = k_{\mathcal{G}^{\mathsf{SH}}} + 2$).

We still have to verify that for a permutation $\pi \in \mathsf{Perm}(M)$, every graph $G \in \operatorname{supp}(\mathcal{G}^{\mathsf{PH}}_{M,n,p}(\pi))$ is a $\mathsf{Perm\text{-}Enc}_n(\pi)$ graph. To establish this, we have to verify that for every $a \in [n]$, $\mathsf{First}(G)_{[a]}$ can only reach the vertex $\mathsf{Last}(G)_{[\pi(a)]}$ in $\mathsf{Last}(G)$. Let T be the corresponding set in the construction of G, and H be the middle subgraph of G corresponding to the graph generated by $\mathcal{G}^{\mathsf{SH}}_{N,p}(T)$.

From the Step (4) of the construction of G and noting that H is a $\mathsf{Set\text{-}Enc}_N(T)$ graph, for $\mathsf{First}(G)_{[a]}$ to reach a vertex $\mathsf{Last}(G)_{[b]}$ in $\mathsf{Last}(G)$, it has to pass H via two vertices $\mathsf{First}(H)_{[\ell]}$ and $\mathsf{First}(H)_{[\ell]}$ such that $\ell \in T$. Let (x, y) be the pair from S so that $\psi_M((x, y)) = \ell$, it must be the case that $(a, b) = (x, y)$. Since $\ell \in T$, we have $b = y = \pi(x) = \pi(a)$. The above discussion shows if $\mathsf{First}(G)_{[a]}$ can reach $\mathsf{Last}(G)_{[b]}$ then we have $b = \pi(a)$. For the other direction, note that if $b = \pi(a)$, then $\mathsf{First}(G)_{[a]}$ can reach $\mathsf{Last}(G)_{[b]}$ through the middle vertices $\mathsf{First}(H)_{[\ell]}$ and $\mathsf{First}(H)_{[\ell]}$, where $\ell = \psi_M((a, \pi(a))) \in T$. This completes the proof.

□

10.3 Construction of $\mathsf{Perm\text{-}Hiding}$ Generators

Construction of $\mathcal{G}^{\mathsf{PH}}_{n,p}$ for p-Pass Streaming Algorithms

- **Input:** A permutation $\pi \in \mathsf{Perm}([n])$. The security parameter ε, pass number p and space s.

- **Assumption:**

  There is a generator $\mathcal{G}^{\mathsf{SH}}_{3n,p}$, which always outputs $(N_{\mathcal{G}^{\mathsf{SH}}}, k_{\mathcal{G}^{\mathsf{SH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{SH}}})$ graphs and is ε_0-$\mathsf{Set\text{-}Hiding}$ for p-pass streaming algorithms with space s, where $\varepsilon_0 = \varepsilon / \log^2 n$.

- **Setup:**

  1. Let $d = c_{\mathsf{AKS}} \cdot \log n$, and fix d matchings $M_1, \dots, M_d$ from Lemma 10.1.
  2. For each $i \in [d]$, let $\mathcal{G}^{\mathsf{PH}}_{M_i,n,p}$ be the ε_0-$\mathsf{Perm}(M_i)$-Hiding generator guaranteed by Lemma 10.4 and the existence of $\mathcal{G}^{\mathsf{SH}}_{3n,p}$.

- **Construction:**

  1. Find d permutations $\pi_1, \ldots, \pi_d$ such that $\pi_i \in \mathsf{Perm}(M_i)$ for each $i \in [d]$ and $\pi = \pi_d \circ \cdots \circ \pi_1$. (Such d permutations exist from Lemma 10.1. If there are multiple such d-tuples we pick the lexicographically first one).
  2. The final graph is

$$\mathcal{G}^{\mathsf{PH}}_{M_1,n,p}(\pi_1) \odot \cdots \odot \mathcal{G}^{\mathsf{PH}}_{M_d,n,p}(\pi_d).$$

Reminder of Lemma 4.3. *Let n be a sufficiently large integer. Let $s \in \mathbb{N}$ and let $\varepsilon \in [0, 1)$. Suppose there is a generator $\mathcal{G}^{\mathsf{SH}}_{3n,p}$ which always outputs $(N_{\mathcal{G}^{\mathsf{SH}}}, k_{\mathcal{G}^{\mathsf{SH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{SH}}})$ graphs and is $(\varepsilon/\log^2 n)$-*Set-Hiding *against p-pass streaming algorithms with space s. Then there is a generator such that: (1) it always outputs $(N_{\mathcal{G}^{\mathsf{PH}}}, k_{\mathcal{G}^{\mathsf{PH}}}, \vec{\ell}_{\mathcal{G}^{\mathsf{PH}}})$ graphs where $N_{\mathcal{G}^{\mathsf{PH}}} = O(N_{\mathcal{G}^{\mathsf{SH}}} \cdot \log n)$ and $k_{\mathcal{G}^{\mathsf{PH}}} = O(k_{\mathcal{G}^{\mathsf{SH}}} \cdot \log n)$; (2) it is ε-*Perm-Hiding *against p-pass streaming algorithms with space s.*

Proof. Applying Lemma 10.4, one can directly verify that $N_{\mathcal{G}^{\mathsf{PH}}} = O(N_{\mathcal{G}^{\mathsf{SH}}} \cdot \log n)$ and $k_{\mathcal{G}^{\mathsf{PH}}} = O(k_{\mathcal{G}^{\mathsf{SH}}} \cdot \log n)$. Also, since for each $i \in [d]$, all outputs of $\mathcal{G}^{\mathsf{PH}}_{M_i,n,p}(\pi_i)$ is a $\mathsf{Perm\text{-}Enc}_n(\pi_i)$ graph, $\mathcal{G}^{\mathsf{PH}}_{M_1,n,p}(\pi_1) \odot \cdots \odot \mathcal{G}^{\mathsf{PH}}_{M_d,n,p}(\pi_d)$ is a $\mathsf{Perm\text{-}Enc}_n(\pi)$ graph, as $\pi = \pi_d \circ \cdots \circ \pi_1$ (Observation 10.2).

Finally, applying a hybrid argument (Lemma 3.4) and noting that by assumption all generators $\mathcal{G}^{\mathsf{PH}}_{M_i,n,p}$ are ε_0-$\mathsf{Perm}(M_i)$-Hiding completes the proof. □

References

[ACK19a] Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Polynomial pass lower bounds for graph streaming algorithms. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 265–276, 2019.

[ACK19b] Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for $(\Delta + 1)$ vertex coloring. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 767–786. SIAM, 2019.

[AHRV07] Omer Angel, Alexander E Holroyd, Dan Romik, and Bálint Virág. Random sorting networks. *Advances in Mathematics*, 215(2):839–868, 2007.

[AKL16] Sepehr Assadi, Sanjeev Khanna, and Yang Li. Tight bounds for single-pass streaming complexity of the set cover problem. In *48th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 698–711. Association for Computing Machinery, 2016.

[AKL17] Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1723–1742. SIAM, 2017.

[AKS83] Miklós Ajtai, János Komlós, and Endre Szemerédi. An $O(n \log n)$ sorting network. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing (STOC)*, pages 1–9, 1983.

[AKSY20] Sepehr Assadi, Gillat Kol, Raghuvansh R Saxena, and Huacheng Yu. Multi-pass graph streaming lower bounds for cycle counting, max-cut, matching size, and other problems. In *FOCS*. https://arxiv.org/pdf/2009.03038.pdf, 2020.

[AKZ19] Sepehr Assadi, Nikolai Karpov, and Qin Zhang. Distributed and streaming linear programming in low dimensions. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*, pages 236–253. ACM, 2019.

[AR20] Sepehr Assadi and Ran Raz. Near-quadratic lower bounds for two-pass graph streaming algorithms. In *FOCS*. https://arxiv.org/pdf/2009.01161.pdf, 2020.

[AW21] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *SODA*. https://arxiv.org/pdf/2010.05846.pdf, 2021.

[Bat68] Kenneth E Batcher. Sorting networks and their applications. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, pages 307–314, 1968.

[Beh46] Felix A. Behrend. On sets of integers which contain no three terms in arithmetical progression. *Proceedings of the National Academy of Sciences of the United States of America*, 32(12):331, 1946.

[BKKL17] Ruben Becker, Andreas Karrenbauer, Sebastian Krinninger, and Christoph Lenzen. Near-optimal approximate shortest paths and transshipment in distributed and streaming models. In *DISC*, volume 91, pages 7:1–7:16, 2017.

[BS15] Marc Bury and Chris Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. In *ESA*, pages 263–274. 2015.

[BWZ16] Christos Boutsidis, David P Woodruff, and Peilin Zhong. Optimal principal component analysis in distributed and streaming models. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing (STOC)*, pages 236–249, 2016.

[CC07] Timothy M. Chan and Eric Y. Chen. Multi-pass geometric algorithms. *Discret. Comput. Geom.*, 37(1):79–102, 2007.

[CCHM14] Rajesh Chitnis, Graham Cormode, Mohammad Taghi Hajiaghayi, and Morteza Monemizadeh. Parameterized streaming: Maximal matching and vertex cover. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 1234–1251. SIAM, 2014.

[CDK19] Graham Cormode, Jacques Dark, and Christian Konrad. Independent sets in vertex-arrival streams. In *46th International Colloquium on Automata, Languages, and Programming (ICALP)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[CFCHT20] Yi-Jun Chang, Martin Farach-Colton, Tsan-Sheng Hsu, and Meng-Tsung Tsai. Streaming complexity of spanning tree computation. In *STACS*, 2020.

[CGMV20] Amit Chakrabarti, Prantar Ghosh, Andrew McGregor, and Sofya Vorotnikova. Vertex ordering problems in directed graph streams. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1786–1802. SIAM, 2020.

[CW09] Kenneth L Clarkson and David P Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the forty-first annual ACM symposium on Theory of computing (STOC)*, pages 205–214, 2009.

[CW13] Kenneth L Clarkson and David P Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of Computing (STOC)*, pages 81–90, 2013.

[CW16] Amit Chakrabarti and Anthony Wirth. Incidence geometries and the pass complexity of semi-streaming set cover. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 1365–1373. SIAM, 2016.

[ER14] Yuval Emek and Adi Rosén. Semi-streaming set cover. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 453–464. Springer, 2014.

[FKM$^+$04] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 531–543. Springer, 2004.

[FKM$^+$09] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the data-stream model. *SIAM Journal on Computing*, 38(5):1709–1727, 2009.

[FLN$^+$02] Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing (STOC)*, pages 474–483, 2002.

[GGK$^+$18] Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrović, and Ronitt Rubinfeld. Improved massively parallel computation algorithms for mis, matching, and vertex cover. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 129–138, 2018.

[GKK12] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms (SODA)*, pages 468–485. SIAM, 2012.

[GKMP20] Mika Göös, Sajin Koroth, Ian Mertz, and Toniann Pitassi. Automating cutting planes is np-hard. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 68–77. ACM, 2020.

[GKMS19] Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson. Weighted matchings via unweighted augmentations. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 491–500, 2019.

[GO16] Venkatesan Guruswami and Krzysztof Onak. Superlinear lower bounds for multipass graph processing. *Algorithmica*, 76(3):654–683, 2016.

[Goo14] Michael T Goodrich. Zig-zag sort: A simple deterministic data-oblivious sorting algorithm running in $O(n \log n)$ time. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing (STOC)*, pages 684–693, 2014.

[GPW17] Mika Goos, Toniann Pitassi, and Thomas Watson. Query-to-communication lifting for BPP. In *FOCS*, 2017.

[HPIMV16] Sariel Har-Peled, Piotr Indyk, Sepideh Mahabadi, and Ali Vakilian. Towards tight bounds for the streaming set cover problem. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*, pages 371–383, 2016.

[HRR98] Monika Rauch Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. Computing on data streams. *External memory algorithms*, 50:107–118, 1998.

[JSWZ20] Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. Faster dynamic matrix inverse for faster lps. *arXiv preprint arXiv:2004.07470*, 2020.

[Kap13] Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 1679–1697. SIAM, 2013.

[LP09] László Lovász and Michael D Plummer. *Matching theory*, volume 367. American Mathematical Soc., 2009.

[LSZ20] S Cliff Liu, Zhao Song, and Hengjie Zhang. Breaking the n-pass barrier: A streaming algorithm for maximum weight bipartite matching. *arXiv preprint arXiv:2009.06106*, 2020.

[LW16] Yi Li and David P Woodruff. On approximating functions of the singular values in a stream. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing (STOC)*, pages 726–739, 2016.

[McG05] Andrew McGregor. Finding graph matchings in data streams. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 170–181. Springer, 2005.

[MN20] Sagnik Mukhopadhyay and Danupon Nanongkai. Weighted min-cut: sequential, cut-query, and streaming algorithms. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 496–509, 2020.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[NN13] Jelani Nelson and Huy L Nguyên. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *2013 ieee 54th annual symposium on foundations of computer science (FOCS)*, pages 117–126. IEEE, 2013.

[ON62] Daniel G O'connor and Raymond J Nelson. Sorting system with nu-line sorting switch, 1962. US Patent 3,029,413.

[Pat90] Michael S Paterson. Improved sorting networks with $O(\log N)$ depth. *Algorithmica*, 5(1-4):75–92, 1990.

[RS78] Imre Z Ruzsa and Endre Szemerédi. Triple systems with no six points carrying three triangles. *Combinatorics (Keszthely, 1976), Coll. Math. Soc. J. Bolyai*, 18:939–945, 1978.

[RSW18] Aviad Rubinstein, Tselil Schramm, and Seth Matthew Weinberg. Computing exact minimum cuts without knowing the graph. In *9th Innovations in Theoretical Computer Science (ITCS)*, page 39. Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2018.

[SWZ17] Zhao Song, David P Woodruff, and Peilin Zhong. Low rank approximation with entrywise l1-norm error. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 688–701, 2017.

[Zel11] Mariano Zelke. Intractability of min-and max-cut in streaming graphs. *Information Processing Letters*, 111(3):145–150, 2011.