A Theory of Composition for Differential Obliviousness

Mingxun Zhou
CMU

Elaine Shi
CMU

T-H. Hubert Chan
HKU

Shir Maimon*
Cornell University

mingxunz@andrew.cmu.edu, runting@cs.cmu.edu, hubert@cs.hku.hk, shir@cs.cornell.edu

Abstract

Differential obliviousness (DO) is a privacy notion which guarantees that the access patterns of a program satisfies differential privacy. Differential obliviousness was studied in a sequence of recent works as a relaxation of full obliviousness. Earlier works showed that DO not only allows us to circumvent the logarithmic-overhead barrier of fully oblivious algorithms, in many cases, it also allows us to achieve polynomial speedup over full obliviousness, since it avoids "padding to the worst-case" behavior of fully oblivious algorithms.

Despite the promises of differential obliviousness (DO), a significant barrier that hinders its broad application is the lack of composability. In particular, when we apply one DO algorithm to the output of another DO algorithm, the composed algorithm may no longer be DO (with reasonable parameters). More specifically, the outputs of the first DO algorithm on two neighboring inputs may no longer be neighboring, and thus we cannot directly benefit from the DO guarantee of the second algorithm.

In this work, we are the first to explore a theory of composition for differentially oblivious algorithms. We propose a refinement of the DO notion called (ϵ, δ)-neighbor-preserving-DO, or (ϵ, δ)-NPDO for short, and we prove that our new notion indeed provides nice compositional guarantees. In this way, the algorithm designer can easily track the privacy loss when composing multiple DO algorithms.

We give several example applications to showcase the power and expressiveness of our new NPDO notion. One of these examples is a result of independent interest: we use the compositional framework to prove an optimal privacy amplification theorem for the differentially oblivious shuffle model. In other words, we show that for a class of distributed differentially private mechanisms in the shuffle-model, one can replace the perfectly secure shuffler with a DO shuffler, and nonetheless enjoy almost the same privacy amplification enabled by a shuffler.

*Author order is randomized. This paper subsumes part of the results in an unpublished manuscript [ZS22] written by a subset of the authors.

1 Introduction

Differential Obliviousness (DO), defined by Chan, Chung, Maggs, and Shi [CCMS19], is a privacy notion for hiding a program's memory access patterns. In comparison with the classical notion of full obliviousness [GO96, Gol87, SCSL11], DO is a relaxation which requires that the program's access patterns satisfy only differential privacy (DP) [DMNS06], as opposed to a simulation-based notion like in full obliviousness [GO96, Gol87, SCSL11]. Several recent works [CCMS19, BKK+21, CZSC21, BNZ19, GKLX22] explored DO and illustrated its benefits:

- Chan et al. [CCMS19] showed a fundamental separation in terms of efficiency between DO and full obliviousness. Specifically, for a class of common tasks such as compaction, merging, and range query data structures, while full obliviousness is inherently subject to at least $\Omega(\log N)$ multiplicative overhead [LSX19, JLN19, AFKL19, FHLS19] (in comparison with the insecure baseline), using DO allows us to reduce the overhead to only $O(\log \log N)$ where N denotes the data size.
- Not only does DO allow us to overcome the logarithmic barrier for fully oblivious algorithms, another important aspect that is sometime overlooked is that DO allows us to overcome the "worst-case barrier" of fully oblivious algorithms [CZSC21], which leads to *polynomial* speedup over full obliviousness in many applications. Specifically, to achieve full obliviousness, we must pad the running time and output length to the worst case over all possible inputs (of some fixed length), whereas DO algorithms may reveal the *noisy* running time or output length. In many real-world scenarios such as database joins [CZSC21], the common case enjoys much shorter runtime and output length than the worst case. For exactly this reason, there is an entire line of work that focuses on designing algorithms optimized for the common rather than the worst case [Rou20]. In such cases, prior works showed that DO can achieve polynomial speedup over any fully oblivious algorithm [CCMS19, CZSC21]!

Vanilla DO does NOT lend to composition. Given the promises of DO, we would like to apply DO to more applications. Unfortunately, the status quo of DO hinders its broad applicability due to the lack of *compositional* guarantees. Specifically, when designing algorithms, it is customary to compose several algorithmic building blocks together. In such cases, it would be nice to say that the composed algorithm also satisfies DO with reasonable parameters as long as the underlying algorithmic building blocks also satisfy DO. Similarly, in some applications, we may need to apply a DO algorithm to the outcome of another (e.g., the SQL database application below). In such cases, we also want to be able to track the privacy loss over time. While the original full obliviousness notion indeed allows such composition, unfortunately, the standard DO notion [CCMS19] does not!

As an explicit example of composition, imagine that we want to build a differentially oblivious database supporting SQL queries. Consider the following natural SQL query where we want to select entries from a table which in itself is the result of a previous `Select` operation[1]:

```
Select (id, position) from
    (Select (id, dept, position) from Employees where salary > 200K)
where dept = "CS"
```

To support this query in a differentially oblivious manner, the most natural idea is to use the DO stable compaction algorithm of Chan et al. [CCMS19] to realize each `Select` operator. In stable

[1] Here we write the two `Select` statements in a single query for convenience, although in a practical interactive database, it could be that the first `Select` query is interactively issued and its result stored as a temporary table, and then the second `Select` query is interactively issued.

compaction, we obtain an input array where each element is either a *real* element or a *filler*, and we want to output an array containing all the *real* elements of the input and preserving the order they appear in the input. Unfortunately, this approach completely fails since Chan et al. [CCMS19]'s DO compaction algorithm does NOT compose.

To understand why, we will introduce some basic notation. Let $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ denote an algorithm, which takes in an input $x \in \mathcal{X}$, and produces an output $y \in \mathcal{Y}$. Consider some neighboring notion $\sim_{\mathcal{X}}$ defined over the input domain $\mathcal{X}$. For example, let $x, x' \in \mathcal{X}$ be two input arrays/tables where each entry corresponds to an individual user. One example is Hamming-distance neighboring: we say that $x \sim_{\mathcal{X}} x'$ iff the Hamming distance of x and x' is at most 1 — this is also the neighboring notion adopted by the DO compaction algorithm of Chan et al. [CCMS19]. The standard DO notion requires the following.

Definition 1.1 (Vanilla differential obliviousness [CCMS19])**.** We say that an algorithm M satisfies (ϵ, δ)-DO w.r.t. some symmetric relation $\sim_{\mathcal{X}}$ iff for any $x, x' \in \mathcal{X}$ such that $x \sim_{\mathcal{X}} x'$, for any subset S,

$$\Pr[\mathsf{View}^{\mathsf{M}}(x) \in S] \leq e^{\epsilon} \cdot \Pr[\mathsf{View}^{\mathsf{M}}(x') \in S] + \delta, \tag{1}$$

where $\mathsf{View}^{\mathsf{M}}(x)$ is a random variable denoting the the memory access patterns observed when running the algorithm M over the input x.

Now, imagine that we have two DO mechanisms $\mathsf{M}_1 : \mathcal{X}_1 \to \mathcal{X}_2$ and $\mathsf{M}_2 : \mathcal{X}_2 \to \mathcal{Y}$ (e.g., think of M_1 and M_2 as Chan et al.'s DO compaction algorithm). We want to apply M_2 to the output of M_1, and hope that the composed mechanism $\mathsf{M}_2 \circ \mathsf{M}_1(\cdot)$ satisfies DO. By the DO definition, we know that M_2 offers indistinguishability for two *neighboring* inputs from $\mathcal{X}_2$. Now, consider two neighboring inputs $x \sim_{\mathcal{X}_1} x'$ from $\mathcal{X}_1$, and consider running the mechanism M_1 over x and x', respectively. Unfortunately, the vanilla DO notion (of M_1) does *not* guarantee that the outputs $\mathsf{M}_1(x)$ and $\mathsf{M}_1(x')$ are also neighboring. Therefore, we may not be able to benefit from the DO property of M_2!

We stress that this is not just a deficiency of the vanilla DO definition. Natural designs of DO algorithms often do not guarantee that the outputs obtained from two neighboring inputs must be neighboring too. For example, consider the stable compaction algorithm of Chan et al. [CCMS19]. Given two input arrays $x = (1, 2, \perp, 3, 4)$ and $x' = (\perp, 2, \perp, 3, 4)$ with Hamming distance 1 where $\perp$ denotes a filler, the compacted outputs will be $(1, 2, 3, 4)$ and $(2, 3, 4)$, respectively. Obviously, the outputs have Hamming distance more than 1. One observation is while the outputs have large Hamming distance, the edit distance is only one — unfortunately, Chan et al.'s compaction algorithm provides privacy only for Hamming-distance neighboring and the guarantees do not generalize to edit-distance neighboring.

DP composition theorems do not work for DO. Since DO is essentially DP applied to the memory access patterns, a natural question is: *can we simply use DP composition theorems to reason about the composition DO mechanisms*? The answer is *no* because DP composition and composition of DO mechanisms are of different nature. In DP composition, we have multiple mechanisms $\mathsf{M}_1, \ldots, \mathsf{M}_k$ where M_i satisfies (ϵ_i, δ_i)-DP. The basic DP composition theorem says that the composed mechanism $\mathsf{M}(x) := (\mathsf{M}_1(x), \ldots, \mathsf{M}_k(x))$ satisfies $(\sum_{i=1}^{k} \epsilon_i, \sum_{i=1}^{k} \delta_i)$-DP. Here, all these mechanisms are applied to the *same input* x. In DO composition, we want to apply M_2 to the *output* of of M_1 instead. More generally, if there are k DO mechanisms $\mathsf{M}_1, \ldots, \mathsf{M}_k$, we want to know whether the composed mechanism $\mathsf{M}_k \circ \mathsf{M}_{k-1} \circ \ldots, \circ \mathsf{M}_1(x) = \mathsf{M}_k(\mathsf{M}_{k-1}(\ldots \mathsf{M}_1(x)))$ is also DO.

Given the status quo, we ask the following natural question:

Can we have suitable and useful refinements of differential obliviousness (DO) that lend to composition?

1.1 Main Contribution: A Theory of Composition for Differential Obliviousness

We are the first to initiate a formal exploration of the composability of differential obliviousness. In this sense, we make an important conceptual contribution: by laying the groundwork for the composition of DO algorithms. we hope that our work can allow DO to have wider applicability.

A new, composable DO notion. Our first contribution is to introduce a new, composable DO notion called *Neighbor-Preserving Differential Obliviousness (NPDO)* that can be viewed as a strengthening of the vanilla DO by Chan et al. [CCMS19]. Our NPDO notion is composition friendly in the following senses:

C1. If M_1 satisfies (ϵ_1, δ_1)-NPDO, and M_2 satisfies (ϵ_2, δ_2)-DO (the vanilla version), then the composed mechanism $\mathsf{M}_2 \circ \mathsf{M}_1$ satisfies $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DO.

C2. If M_1 satisfies (ϵ_1, δ_1)-NPDO, and M_2 satisfies (ϵ_2, δ_2)-NPDO, then the composed mechanism $\mathsf{M}_2 \circ \mathsf{M}_1$ satisfies $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-NPDO.

In the above, the first property allows us to apply any vanilla-DO algorithm M_2 to the output of an NPDO algorithm M_1, and the composed algorithm $\mathsf{M}_2 \circ \mathsf{M}_1$ would satisfy vanilla DO. The second property allows us to perform composition repeatedly. In particular, if both M_1 and M_2 are NPDO, then the composed algorithm $\mathsf{M}_2 \circ \mathsf{M}_1$ also satisfies NPDO, i.e., it can be further composed with other DO or NPDO algorithms.

Finding the right notion turned out to be non-trivial. We want to capture the intuition that "the algorithm should produce neighboring outputs for neighboring inputs". However, it is not obvious how to formally capture this idea of "neighbor-preserving" especially when the outputs of the DO algorithm may be randomized. Indeed, naïve ways to define "neighbor-preserving" turned out to be too stringent and preclude many natural and interesting algorithms (see Section 3.1). We instead suggest a more general version that allows us to capture a probabilistic notion of neighbor-preserving. More specifically, our NPDO notion requires that when one applies the algorithm M on two neighboring inputs x and x', the *joint distribution* of the adversary's view and the output must be distributionally close in some technical sense, where *closeness is parametrized by some output neighboring relation*. The formal definition is presented below:

Definition 1.2 ((ϵ, δ)-NPDO)**.** We say that an algorithm $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ with view space $\mathcal{V}$ satisfies (ϵ, δ)-NPDO w.r.t. input relation $\sim_{\mathcal{X}}$ and output relation $\sim_{\mathcal{Y}}$, if for any $x, x' \in \mathcal{X}$ such that $x \sim_{\mathcal{X}} x'$, for any subset $S \subseteq \mathcal{V} \times \mathcal{Y}$,

$$\Pr[\mathsf{Exec}^{\mathsf{M}}(x) \in S] \leq e^{\epsilon} \cdot \Pr[\mathsf{Exec}^{\mathsf{M}}(x') \in \mathcal{N}(S)] + \delta.$$

In the above, $\mathsf{Exec}^{\mathsf{M}}(x)$ samples a random execution of M on the input x, and returns the view (i.e., access patterns) as well as the algorithm's output. Further, the notation $\mathcal{N}(S)$, i.e., the neighboring set of S, is defined as follows:

$$\mathcal{N}(S) = \{(v, y) | \exists (v, y') \in S \text{ s.t. } y \sim_{\mathcal{Y}} y'\}$$

Expressiveness of our notion. We give various natural examples to demonstrate the expressiveness and power of our notion. We believe that our NPDO notion is indeed the right notion, given the simplicity in form and its broad applicability. Besides the motivating SQL database example mentioned earlier in this section, other notable examples include the design of a differentially oblivious subsampling algorithm, a stable compaction algorithm that is DO w.r.t. edit distance, and finally, proving an optimal privacy amplification theorem in the differentially oblivious shuffle model. Since the last application is of independent interest even as a standalone result, we will discuss the context and the implifications of this result separately in Section 1.2.

Proof of composition theorem. Our second contribution is to prove the composition theorem:

Theorem 1.3 (Composition theorem)**.** *The aforementioned compositional properties C1 and C2 hold, as long as the algorithm* M_1*'s view space and output space are finite or countably infinite.*

The proof of the composition theorem is rather non-trivial. A key step in the proof is to show the following equivalence (see Lemma 4.1). An algorithm $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ (with at most countably infinite view space $\mathcal{V}$ and output space $\mathcal{Y}$) satisfies (ϵ, δ)-NPDO w.r.t. $\sim_{\mathcal{X}}$ and $\sim_{\mathcal{Y}}$, *if and only if* for any neighboring inputs $x \sim_{\mathcal{X}} x'$, there exists an (ϵ, δ)-matching between the the probability spaces of the random variables $\mathsf{Exec}^{\mathsf{M}}(x) \in \mathcal{V} \times \mathcal{Y}$ and $\mathsf{Exec}^{\mathsf{M}}(x') \in \mathcal{V} \times \mathcal{Y}$. In an (ϵ, δ)-matching, imagine that we have a (possibly countably infinitely large) bipartite graph where one side has the sources, and the other side has the destinations. Both sources and destinations come from the space $\mathcal{V} \times \mathcal{Y}$. If there is an edge of weight w between some source and some destination, we may imagine that the source wants to send w amount of commodity to the destination. Now, each source $(v, y) \in \mathcal{V} \times \mathcal{Y}$ produces an amount of commodity equal to $\Pr\left[\mathsf{Exec}^{\mathsf{M}}(x) = (v, y)\right]$, and each destination (v, y') can receive at most $e^{\epsilon} \cdot \Pr\left[\mathsf{Exec}^{\mathsf{M}}(x') = (v, y')\right]$ amount of commodity. Furthermore, a source (v, y) can be matched with a desitination (v', y') only if they are neighboring, i.e., $v = v'$ and $y \sim_{\mathcal{Y}} y'$. We want to find a matching such that all but δ amount of commodity is delivered to the destinations. To prove this key equivalence lemma, we are inspired by techniques used to prove the Hall's marriage theorem [Hal35,HJ48]. Once we prove the key equivalence lemma, we then rely on it to prove the composition theorem.

In the main body, we primarily focus on proving the composition theorem for *statistical* notions of DO. In Appendix A, we further extend our composition theorem to support suitable, *computational* notions of differential obliviousness as well.

Finally, in our composition theorem, we assume that the view and output spaces of M_1 are at most countably infinitely large. This assumption is reasonable given that we primarily focus on the standard word-RAM model of execution. However, it is indeed an interesting open question whether we can remove this restriction and prove the composition theorem for uncountably large view and output spaces — this is useful if we consider RAM machines that can handle real arithmetic. In Appendix C, we discuss the additional technicalities that one might encounter if we wish to remove the countable restriction.

1.2 Additional Result: Optimal Privacy Amplification in the DO-Shuffle Model

As an application of our composition framework, we use it to prove an optimal privacy amplification theorem in the differentially oblivious shuffle (DO-shuffle) model. Since this result can be of independent interest on its own, we explain the motivation and context below.

Background: privacy amplification in the shuffle model. To understand the DO-shuffle model, let us first review some background on the so-called shuffle model. Imagine that a set of

clients each hold some private data, and an *untrusted* server wants to perform some analytics over the union of the clients' data, while preserving each individual client's privacy. Specifically, we want to guarantee that for two neighboring input configurations of the clients denoted $\mathbf{x}$ and $\mathbf{x}'$ respectively, the distributions of the server's view are "close".

The shuffle model, first proposed by Bittau et al. [BEM+17] in an empirical work, has become a popular model for implementing distributed differentially private mechanisms. In this model, we assume the existence of a trusted shuffler that takes the union of all clients' messages, randomly permutes them, and presents the shuffled result to the server. The server then performs some computation and outputs the analytics result. The trusted shuffler guarantees the anonymity of all messages, such that the server can only see the union of all messages, without knowing the source of an individual message. Numerous earlier works [BBGN19, CSU+19, Che21, GGK+21, GKMP20] have shown that the shuffle model often enables differentially private mechanisms whose utility approximates the best known algorithms in the *central model* (where the server is trusted and we only need privacy on the outcome of the analytics). Moreover, several works have shown that the trusted shuffler can be efficiently implemented either using trusted hardware [BEM+17] or using cryptographic protocols [Cha81, Abe99, BG12, Cha88, CGF10, APY20, CBM15, SW21, AKTZ17, OS97, GIKM00, ZZZR05, EB, GRS99, DS18, CL05]. This makes the shuffle model a compelling approach not just in theory, but also in practical applications such as federated learning [GDD+21].

A particular useful type of theorem in the shuffle model is called a privacy amplification theorem, which we explain below. Henceforth, let $\mathcal{R}(x_i)$ be some differentially private mechanism each client i applies to randomize its own private input x_i (often called a *locally differentially private (LDP)* randomizer). Roughly speaking, a privacy amplification theorem makes a statement of the following nature where $\mathcal{S}(\cdot)$ denotes the shuffler that outputs a random permutation of the inputs: if each client's LDP mechanism $\mathcal{R}$ consumes ϵ_0 privacy budget, then shuffler's outcome $\mathcal{S}(\mathcal{R}(x_1), \ldots, \mathcal{R}(x_n))$ satisfies (ϵ, δ)-DP for $\epsilon = \epsilon(\epsilon_0, \delta) \ll \epsilon_0$, i.e., privacy is amplified for the overall shuffle-model mechanism. A line of work [CSU+19, EFM+19, BBGN19] focused on proving privacy amplification theorems for the shuffle model, culminating in the recent work by Feldman et al. [FMT21], who proved a privacy amplification theorem for any LDP mechanism with optimal parameters.

Connection between the shuffle model and our DO composition framework. We realize that shuffle model can be expressed under our DO composition framework. Consider the composed mechanism $\mathsf{S} \circ \mathsf{M}_1$. $\mathsf{M}_1 : \mathcal{X}^n \to \mathcal{Y}^n$ is the local randomization mechanism that takes n clients' inputs $(x_1, \ldots, x_n)$ and outputs the message sequence $(y_1, \ldots, y_n)$ where $y_i = \mathcal{R}(x_i)$. $\mathsf{S} : \mathcal{Y}^n \to \mathcal{Y}^n$ is the shuffling mechanism that takes the message sequence $(y_1, \ldots, y_n)$ and outputs a random permutation of the sequence. Since all computation in M_1 are done by the clients locally, we define $\mathsf{View}^{\mathsf{M}_1} := \emptyset$. Also, we define the view in S as the same as its output: a random permutation of $(y_1, \ldots, y_n)$. Then, the view of the server in the shuffle model is exactly the same as the view of the adversary in $\mathsf{S} \circ \mathsf{M}_1$. Thus, (ϵ, δ)-shuffle-DP guarantee can be expressed as $\mathsf{S} \circ \mathsf{M}_1$ is (ϵ, δ)-DO w.r.t the input neighboring notion $\sim_{\mathcal{X}}$ such that $\mathbf{x} \sim_{\mathcal{X}} \mathbf{x}'$ iff the Hamming distance is at most 1.

Can we replace the shuffler with a DO-shuffler? A couple very recent works [GKLX22, BHMS, ALU18] have suggested a relaxed shuffler model called the *differentially oblivious shuffle* model (or *DO-shuffle model* for short). Unlike the traditional shuffle model which provides full anonymity on the clients' messages, the DO-shuffle model permutes the clients' messages but possibly allowing some differentially private leakage. More concretely, a DO-shuffle protocol guarantees that for two *neighboring* input vectors $\mathbf{x}_H$ and $\mathbf{x}'_H$ corresponding to the set of honest parties, the

adversary's views in the protocol execution are computationally or statistically close. The recent works by Gordon et al. [GKLX22] and Bünz et al. [BHMS] both show that the relaxed DO-shuffle can be asymptotically more efficient to cryptographically realize than a fully anonymous shuffle. It would therefore be desirable to use a DO-shuffler as a drop-in replacement of the perfectly secure shuffle. This raises a couple very natural question:

- *If we were to replace the shuffler in shuffle-model differentially private (DP) mechanisms with a DO-shuffler, can we still get comparable privacy-utility tradeoff?*
- *More specifically, can we prove an optimal privacy amplification theorem for the DO-shuffle model, matching the parameters of Feldman et al. [FMT21]?*

The pioneering work of Gordon et al. [GKLX22] was the first to explore how to use a DO shuffler to design distributed differentially private mechanisms. Gordon et al. [GKLX22] showed two novel results. First, they prove an optimal privacy amplification theorem for the randomized response mechanism in the DO-shuffle model, with parameters that tightly match the shuffle-model counterpart. Next, they generalize their first result, and prove a privacy amplification theorem for any local differentially private (LDP) mechanism — however, this more general result is *non-optimal*, since they rely on the non-optimal shuffle-model amplification theorem from Balle et al. [BBGN19].

Our results. We prove a privacy amplification theorem for *any LDP mechanism* that achieves *optimal* parameters, tightly matching Feldman et al. [FMT21]'s privacy amplification parameters for the shuffle model. This result improves work of Gordon et al. [GKLX22] in the following senses: 1) we asymptotically improve their privacy amplification theorem for any general LDP mechanism; and 2) their privacy amplification theorem for the specific randomized response mechanism can be viewed as a special case of our general theorem. More interestingly, we can prove our result fully under our DO composition framework. The curx of the proof is to show that the local randomization mechanism $\mathsf{M_1}$ is (ϵ, δ)-NPDO w.r.t the output neighboring notion being exactly the DO-shuffler's input neighboring notion. Then, when $\mathsf{M_1}$ composes with an (ϵ_1, δ_1)-DO shuffler, the composed mechanism will be $(\epsilon + \epsilon_1, \delta + \delta_1)$-DO.

Below, we give a more formal statement of our result. Let Φ denote a DO-shuffling protocol. Given an LDP-randomizer $\mathcal{R}(\cdot)$, we use the notation $\Pi(x_1, \ldots, x_n) := \Phi(\mathcal{R}(x_1), \ldots, \mathcal{R}(x_n))$ to denote the composed protocol where each of the n parties first applies the local randomizer $\mathcal{R}(\cdot)$ to its own private data, and then invokes an instance of the DO-shuffling protocol Φ on the outcome $\mathcal{R}(x_i)$.

Theorem 1.4 (Optimal privacy amplification for any LDP mechanism in the DO-shuffle model)**.** *Suppose $\epsilon_0 \le \log\left(\frac{n}{16\log(2/\delta)}\right)$. Given n copies of an ϵ_0-LDP randomizer $\mathcal{R}$ and an (ϵ_1, δ_1)-DO shuffler Φ resilient to t corrupted parties, the composed protocol $\Pi(x_1, \ldots, x_n) := \Phi(\mathcal{R}(x_1), \ldots, \mathcal{R}(x_n))$ is $(\epsilon + \epsilon_1, \delta + \delta_1)$-DO against up to t corrupted parties where*

$$\epsilon = O\left(\frac{(1 - e^{\epsilon_0})e^{\epsilon_0/2}\sqrt{\log(1/\delta)}}{\sqrt{n - t}}\right).$$

Furthermore, if the DO-shuffler satisfies computational (or statistical, resp.) DO, then the composed protocol satisfies computation (or statistical, resp.) DO.

Further, if the underlying DO-shuffle protocol satisfies semi-honest security [GKLX22, ALU18], then the composed protocol is also secure in a semi-honest corruption model. Similarly, if the underlying DO-shuffle satisfies malicious security (e.g., [ALU18, BHMS]), then the composed protocol is also secure in a malicious model.

2 Model and Preliminaries

2.1 Model of Computation

We consider a standard Random Access Machine (RAM) model of computation. We assume that the adversary can observe the memory access patterns of the algorithm, including which locations are read or written and in which time steps. The adversary cannot see the contents of the memory tape themselves, which also means that the adversary cannot see the contents of the input and output.

Format of input and output tape. We explain the format of the input and output tape — the modeling technicalities are without loss of generality, and matter if we want to mask the true input and output lengths.

In the most general model, *the algorithm may or may not be able to observe the input and output length, depending on the algorithm.* More specifically, we may assume that the input is written on an input tape — the input tape itself has *unbounded* length and the *actual length of the input is written on some dedicated location*, e.g., address 0, of the input tape. The algorithm can then read address 0 to learn the actual input length. During the execution, the algorithm *may read a random number of extraneous locations* on the input tape, such that the adversary may not be able to observe the exact input length. Without loss of generality, we may assume that every extraneous location on the input tape stores a filler symbol $\perp$.

Similarly, the algorithm must write the output on an output tape. Again, the algorithm, may *write a random number of extraneous locations* on the output tape. For example, if the actual output length is m, the algorithm may actually write write $m' > m$ locations on the output tape where m' is a random variable, to mask the true output length. To indicate the actual output length, the algorithm can write the actual output length m on some dedicated location of the output tape. Therefore, if the algorithm writes to a random number of extraneous locations on the input tape, the adversary may not be able to observe the exact output length.

2.2 Preliminaries

Mathematical tools. We introduce some basic mathematical tools.

Definition 2.1 (Symmetric geometric distribution)**.** Let $\alpha > 1$. The symmetric geometric distribution $\mathsf{Geom}(\alpha)$ takes integer values such that the probability mass function at k is $\frac{\alpha-1}{\alpha+1} \cdot \alpha^{-|k|}$.

In designing DO algorithms, we often pad the true output length with random fillers such that the adversary observes a randomized output length. Below, we define a shifted and truncated geometric distribution which is often used to sample the number of fillers used for padding. In particular, this distribution always gives non-negative and bounded random variables.

Definition 2.2 (Shifted and truncated geometric distribution)**.** Let $\epsilon > 0$ and $\delta \in (0, 1)$ and $\Delta \geq 1$. Let k_0 be the smallest positive integer such that $\Pr[|\mathsf{Geom}(e^{\frac{\epsilon}{\Delta}})| \geq k_0] \leq \delta$, where $k_0 = \frac{\Delta}{\epsilon} \ln \frac{2}{\delta} + O(1)$.

The shifted and truncated geometric distribution $\mathcal{G}(\epsilon, \delta, \Delta)$ has support $[0, 2(k_0 + \Delta - 1)]$, and is defined as:

$$\min\{\max\{0, k_0 + \Delta - 1 + \mathsf{Geom}(e^\epsilon)\}, 2(k_0 + \Delta - 1)\}$$

For the special case $\Delta = 1$, we write $\mathcal{G}(\epsilon, \delta) := \mathcal{G}(\epsilon, \delta, 1)$.

Common distance notions. We will also use a couple common distance notions in our examples, including Hamming distance and edit distance.

Definition 2.3 (Hamming distance neighboring $\sim_H$)**.** We say that two arrays x, x' are neighboring by the Hamming distance iff 1) they have the same length; and 2) they differ in at most one position.

Definition 2.4 (Edit distance neighboring $\sim_E$)**.** We say that two arrays x, x' are neighboring by the edit distance iff x' can be obtained from x through either one insertion, one deletion, or one substitution. Note that x and x' need not have the same length.

Notations for randomized execution. Given randomized mechanisms $\mathsf{M}_1 : \mathcal{X} \to \mathcal{Y}$ and $\mathsf{M}_2 : \mathcal{Y} \times \mathcal{Z}$, the composed mechanism $\mathsf{M}_2 \circ \mathsf{M}_1 : \mathcal{X} \to \mathcal{Z}$ works as follows: for input $x \in \mathcal{X}$, we first apply $\mathsf{M}_1(x)$ to produce an intermediate $y \in \mathcal{Y}$, and then we apply $\mathsf{M}_2(y)$.

Henceforth, given an algorithm $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$, and an input $x \in \mathcal{X}$, we often use the following random variables:

- The random variable $\underline{\mathsf{View}^{\mathsf{M}}(x) : \mathcal{X} \to \mathcal{V}}$ denotes the memory access patterns (also called the **view**) observed by the adversary when M receives the input x, where $\mathcal{V}$ is the view space for M.
- The notation $\underline{\mathsf{Exec}^{\mathsf{M}}(x) : \mathcal{X} \to \mathcal{V} \times \mathcal{Y}}$ is a random variable that outputs the view and the output over a random execution of $\mathsf{M}(x)$.

3 A Composition Framework for DO

In this section, we explore what kind of DO notions are composition-friendly. As a warmup, we first suggest a simple notion called strongly neighbor-preserving (or strongly NP for short), and show that any DO algorithm that is strongly NP lends to composition. The strong NP notion, however, is too stringent. We then propose a more general notion called (ϵ, δ)-neighbor-preserving differential obliviousness or (ϵ, δ)-NPDO for short, which captures a probabilistically approximate notion of neighbor-preserving. We then present our main composition theorem which states that any algorithm that satisfies NPDO lends to composition. Along the way, we give several simple motivating examples to demonstrate the usefulness our compositional framework.

3.1 Strongly Neighbor-Preserving

3.1.1 Definition and Composition Theorem

Earlier, in Section 1, we argued why vanilla DO algorithms do not lend to composition, because neighboring inputs may lead to very dissimilar outputs. One (somewhat imprecise) intuition is the following: if a DO mechanism is additionally *neighbor-preserving*, i.e., neighboring inputs lead to neighboring outputs, then it should lend to composition.

We first define a strong notion of neighbor-preserving, which requires that running the algorithm over two neighboring inputs produces neighboring outputs with probability 1.

Definition 3.1 (Strongly neighbor-preserving)**.** We say that a randomized algorithm $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ is strongly neighbor-preserving w.r.t. $\sim_{\mathcal{X}}$ and $\sim_{\mathcal{Y}}$, iff for any two inputs $x, x' \in \mathcal{X}$ such that $x \sim_{\mathcal{X}} x'$,

$$\Pr[y \leftarrow \mathsf{M}(x), y' \leftarrow \mathsf{M}(x') : y \sim_{\mathcal{Y}} y'] = 1.$$

We can prove that if an algorithm satisfies both DO and strongly neighbor-preserving, then it is composable, formally stated below.

Theorem 3.2 (Strongly neighbor-preserving + DO gives composition)**.** *Suppose that* $\mathsf{M}_1 : \mathcal{X} \to \mathcal{Y}$ *is* (ϵ_1, δ_1)*-DO w.r.t.* $\sim_{\mathcal{X}}$ *and strongly neighbor-preserving w.r.t.* $\sim_{\mathcal{X}}$ *and* $\sim_{\mathcal{Y}}$*, and moreover, suppose that* $\mathsf{M}_2 : \mathcal{Y} \to \mathcal{Z}$ *is* (ϵ_2, δ_2)*-DO w.r.t.* $\sim_{\mathcal{Y}}$*, then* $\mathsf{M}_2 \circ \mathsf{M}_1$ *satisfies* $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$*-DO w.r.t.* $\sim_{\mathcal{X}}$*.*

Furthermore, if M_2 *is additionally strongly neighbor-preserving w.r.t.* $\sim_{\mathcal{Y}}$ *and* $\sim_{\mathcal{Z}}$*, then* $\mathsf{M}_2 \circ \mathsf{M}_1$ *is also strongly neighbor-preserving w.r.t.* $\sim_{\mathcal{X}}$ *and* $\sim_{\mathcal{Z}}$*.*

Proof. Later in Lemma 3.7 of Section 3.4, we will prove that (ϵ_1, δ_1)-DO plus strongly neighbor-preserving is a special case of our more general notion (ϵ_1, δ_1)-NPDO. In this sense, this composition theorem can be viewed as a special case of our main composition theorem for NPDO (Theorem 3.6). □

3.1.2 Composition Examples

Example 1. Earlier in Section 1, we pointed out that two sequential instances of Chan et al.'s DO compaction algorithm [CCMS19] do not give (tight) composable guarantees. In Example 1, we will see that if we replace the second instance with a modification of Chan et al.'s compaction algorithm such that it is DO w.r.t. edit distance (as opposed to Hamming distance), then the two instances would compose nicely.

Specifically, let M_1 be Chan et al.'s DO compaction algorithm [CCMS19]. Recall that the algorithm receives an input array where each element is either a *real* element or a *filler*, and outputs an array containing all the real elements in the input and preserving the order they appear in the input. M_1 is (ϵ_1, δ_1)-DO w.r.t. $\sim_H$ (i.e., Hamming distance). Now, suppose we can construct another compaction algorithm denoted M_2 that is (ϵ_2, δ_2)-DO w.r.t. to $\sim_E$ (i.e., *edit* distance). How to construct such an M_2 while preserving efficiency turns out to be non-trivial, and we defer the construction to Section 5 — interestingly, designing M_2 itself demonstrates the usefulness of our composition framework, too.

Observe that given a fixed input array x, the output of $\mathsf{M}_1(x)$ must be an ordered list of real elements contained in x plus an appropriate number of fillers, and the total length of the output[2] is the same as the input x. Thus, for any neighboring inputs $x \sim_H x'$, it must be that $\mathsf{M}_1(x) \sim_E \mathsf{M}_2(x')$. Therefore, we conclude that M_1 is strongly neighbor-preserving w.r.t. the input relation $\sim_H$ and the output relation $\sim_E$. Applying Theorem 3.2, we conclude that the composed mechanism $\mathsf{M}_2 \circ \mathsf{M}_1$ satisfies $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DO.

Example 2. Let M_1 be an algorithm that merges two sorted input arrays (x_0, x_1), where each element in the input array has a payload besides the sort-key. Suppose that M_1 satisfies (ϵ_1, δ_1) differential obliviousness w.r.t. $\overset{2}{\sim}_E$, i.e., two inputs (x_0, x_1) and (x_0', x_1') are considered neighboring iff for $b \in \{0, 1\}$, $|x_b| = |x_b'|$, and x_b and x_b' have edit distance at most 2 (i.e., $x_b \overset{2}{\sim}_E x_b'$). Such an

[2]Even though the algorithm M_1 itself is randomized, the output of M_1 is deterministic and unique given the input.

DO merge algorithm was proposed by Chan et al. [CCMS19], and moreover, their algorithm always outputs an array whose length is the sum of the input arrays. Notice that for neighboring inputs, M_1 always produces outputs that have edit distance at most 4.

Let M_2 be a stable tight compaction algorithm that selects elements from the input array whose payload string satisfies a certain predicate (e.g., entries corresponding to students in the computer science department). Suppose that M_2 satisfies (ϵ_2, δ_2)-DO w.r.t. $\overset{4}{\sim}_E$, i.e., where neighboring inputs are those with edit distance at most 4 — such an M_2 is described in Section 5.

By Theorem 3.2, we conclude that the composed mechanism $\mathsf{M}_2 \circ \mathsf{M}_1$ satisfies $(\epsilon_1+\epsilon_2, \delta_1+\delta_2)$-DO w.r.t. $\overset{2}{\sim}_E$.

Remark 3.3 (Capturing k-neighboring relations)**.** Recall that our strongly neighbor-preserving definition (i.e., Definition 3.1) is parametrized with the input and output relations. Example 2 is used to illustrate the case when the these input/output relations are parametrized with a k-neighboring notion (rather than 1-neighboring) — this shows the generality of the approach. For example, later in Section 5 , we will construct an efficient stable compaction algorithm that is (ϵ, δ)-DO w.r.t. to $\overset{1}{\sim}_E$ neighboring. Applying the standard group privacy theorem of differential privacy [DR14], we can get a a compaction algorithm that is $(4\epsilon, 4e^{4\epsilon}\delta)$-DO w.r.t. to $\overset{4}{\sim}_E$ neighboring.

3.1.3 Limitations

The strong neighbor-preserving requirement (i.e., Definition 3.1) is natural and directly captures our intuition that if a DO mechanism maps neighboring inputs to neighboring outputs, then it is composable. In particular, the strongly neighbor-preserving requirement is often suitable when the output computed by the algorithm is deterministic (i.e., uniquely determined by the inputs), even though the algorithm itself may be randomized — Examples 1 and 2 fit this case.

However, the strongly neighbor-preserving requirement may be too stringent especially when the output of the algorithm may be randomized. For example, consider the following DO subsampling algorithm.

Example 3. We consider the task of subsampling, which is widely used in private data analytics [BBG18, WBK19]: given an input array x, we want to sample each entry with probability p, and generate a new array that contains only the sampled elements. Consider the following subsampling algorithm where n denotes the length of the input array x:

1. Call $\mathsf{M}_1(x) := \mathsf{InPlaceSample}(x)$ which is defined as follows: Scan the input array x. For each real element encountered, append it to the output tape with probability p and append a filler element otherwise. For each filler element encountered, just append a filler to the output tape.
2. Apply M_2, a compaction algorithm that is (ϵ', δ')-DO w.r.t. $\sim_H$ to the output of the above step.

We want to prove that the above algorithm satisfies DO w.r.t. $\sim_H$ through composition — intuitively, this should be true. In particular, the first subroutine $\mathsf{M}_1 := \mathsf{InPlaceSample}$ has deterministic access patterns. We explicitly denote $\mathsf{M}_1(\cdot; \rho)$ to fix the random tape ρ consumed by M_1. For any fixed random tape ρ, and any neighboring inputs $x \sim_H x'$, $\mathsf{M}_1(x; \rho)$ and $\mathsf{M}_1(x'; \rho)$ output two arrays with Hamming distance 1. Therefore, intuitively, as long as the compaction algorithm in the second step is (ϵ', δ')-DO w.r.t. Hamming distance, the entire subsampling algorithm should be (ϵ', δ')-DO as well. Unfortunately, we cannot directly use strong neighbor-preserving to prove this composition here, since a random execution of $\mathsf{M}_1(x)$ and a random execution of $\mathsf{M}_1(x')$ are

not guaranteed to always output Hamming-distance-neighboring outputs — it depends on which subset of elements are selected.

This motivates us to relax the strongly neighbor-preserving to make it more general, such that our compositional framework can be more expressive. However, before we do so, we introduce another more general example, Example 4, which is a variation of Example 3. Specifically, in Example 3, although the output of $\mathsf{M}_1 := \mathsf{InPlaceSample}$ is randomized, the view of M_1 is deterministic. In Example 4, both the view and the output of the first algorithm are randomized.

Example 4. The main difference between Examples 3 and 4 is that Example 4 aims to have a subsampling algorithm that is DO w.r.t. *edit distance*, whereas Example 3 aims to be DO w.r.t. *Hamming distance*. To achieve this, in Example 4, we need to mask the true length of the input and output by reading/writing a random number of extraneous locations on the input tape, Further, the compaction algorithm we call must now be DO w.r.t. edit distance too. The detailed algorithm is described below, where the key differences are highlighted in blue.

1. Call $\mathsf{M}_1(x) := \mathsf{InPlaceSample}_{\epsilon,\delta}(x)$ which is defined as follows:
   - Sample $r \xleftarrow{\$} \mathcal{G}(\epsilon, \delta, \Delta = 1)$, let $n' = n + r$ be the noisy input length.
   - Scan n' locations on the input tape. For each real element encountered, append it to the output tape with probability p and append a filler element otherwise. For each filler element encountered, just append a filler to the output tape.
   - The output array is defined to be the first n elements of the output tape. Write down its length n at a fixed dedicated location (e.g., location 0) on the output tape.
2. Apply M_2, a compaction algorithm that is (ϵ', δ')-DO w.r.t. $\sim_E$ to the output of the above step, i.e., the compaction algorithm treats the output tape of M_1 as its own input tape.

We shall later prove that Examples 3 and 4 satisfy DO using our new compositional framework.

3.2 (ϵ, δ)-Neighbor-Preserving Differential Obliviousness (NPDO)

Recognizing the limitations of strongly neighbor-preserving (Definition 3.1), we would like to make the compositional framework more general. In particular, the above Examples 3 and 4 can serve as simple motivating examples.

Given a mechanism M whose view space is $\mathcal{V}$ and output space is $\mathcal{Y}$, given some symmetric relation $\sim_{\mathcal{Y}}$ over the output space, and given a set $S \subseteq \mathcal{V} \times \mathcal{Y}$, we define the following notation for denoting **neighbor sets**:

$$\mathcal{N}(S) := \left\{(v, y') | \exists (v, y) \in S \text{ s.t. } y' \sim_{\mathcal{Y}} y\right\}$$

Definition 3.4 ((ϵ, δ)-NPDO)**.** Given a mechanism $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ whose view space is $\mathcal{V}$, we say that it satisfies (ϵ, δ)-neighbor-preserving differential obliviousness, or (ϵ, δ)-NPDO for short, w.r.t. symmetric relations $\sim_{\mathcal{X}}$ and $\sim_{\mathcal{Y}}$, respectively, iff for all $x \sim_{\mathcal{X}} x'$, for every $S \subseteq \mathcal{V} \times \mathcal{Y}$,

$$\Pr[\mathsf{Exec}^{\mathsf{M}}(x) \in S] \leq e^{\epsilon} \cdot \Pr[\mathsf{Exec}^{\mathsf{M}}(x') \in \mathcal{N}(S)] + \delta \tag{2}$$

Our NPDO definition looks similar in form as the standard differential privacy notion, with a couple important observations: 1) the notion is defined over the Cartesian product of the view and the output of the mechanism, which is important for composition to hold; 2) on the right-hand-side of Equation (2), we consider the probability of $\mathsf{M}(x')$ landing in the *neighboring* set $\mathcal{N}(S)$ on a neighboring input $x' \sim_{\mathcal{X}} x$ — this is important for capturing a probabilistic notion of neighbor-preserving.

It is not hard to see that if an algorithm satisfies (ϵ, δ)-NPDO, it must satisfy (ϵ, δ)-DO, as stated in the following fact.

Fact 3.5. *Suppose that $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ satisfies (ϵ, δ)-NPDO w.r.t. $\mathcal{X}$ and $\mathcal{Y}$. Then, M satisfies (ϵ, δ)-DO w.r.t. $\mathcal{X}$.*

Proof. Let $\mathcal{V}$ be the sample space of $\mathsf{View}^{\mathsf{M}}$. Consider all those $S_v \subseteq \mathcal{V}$. We know $\mathcal{N}(S_v \times \mathcal{Y}) = S_v \times \mathcal{Y}$. Therefore,

$$\begin{aligned}&\Pr[\mathsf{View}^{\mathsf{M}}(x) \in S_v] = \Pr[\mathsf{Exec}^{\mathsf{M}}(x) \in S_v \times \mathcal{Y}] \\ \leq & e^{\epsilon} \Pr[\mathsf{Exec}^{\mathsf{M}}(x') \in \mathcal{N}(S_v \times \mathcal{Y})] + \delta = e^{\epsilon} \Pr[\mathsf{View}^{\mathsf{M}}(x') \in S_v] + \delta.\end{aligned}$$

□

3.3 Main Composition Theorem

One main technical contribution of our paper is to prove a composition theorem for our NPDO notion, as stated below.

Theorem 3.6 (Main composition theorem)**.** *Suppose that an algorithm $\mathsf{M}_1 : \mathcal{X} \to \mathcal{Y}$ satisfies (ϵ_1, δ_1)-NPDO w.r.t. $\sim_{\mathcal{X}}$ and $\sim_{\mathcal{Y}}$. Further, suppose that the algorithm M_1's view space $\mathcal{V}$ and the output space $\mathcal{Y}$ are finite or countably infinite. Then, the following composition statements hold:*

1. *Suppose that $\mathsf{M}_2 : \mathcal{Y} \to \mathcal{Z}$ satisfies (ϵ_2, δ_2)-DO w.r.t. $\sim_{\mathcal{Y}}$. Then, the composed mechanism $\mathsf{M}_2 \circ \mathsf{M}_1 : \mathcal{X} \to \mathcal{Z}$ satisfies $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DO.*
2. *Suppose that $\mathsf{M}_2 : \mathcal{Y} \to \mathcal{Z}$ satisfies (ϵ_2, δ_2)-NPDO w.r.t. $\sim_{\mathcal{Y}}$ and $\sim_{\mathcal{Z}}$. Then, the composed mechanism $\mathsf{M}_2 \circ \mathsf{M}_1 : \mathcal{X} \to \mathcal{Z}$ satisfies $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-NPDO.*

The proof of Theorem 3.6 is non-trivial and presented in Section 4. We can use Theorem 3.6 to prove that the algorithms in the earlier Examples 1 to 4 satisfy DO. Before doing so, let us first introduce some helpful tools for proving an algorithm NPDO.

3.4 Helpful Tools for Proving NPDO

To use our main composition theorem, we need to prove that some algorithm satisfies NPDO. The following couple lemmas can often lend to this purpose.

Strongly NP + DO $\Longrightarrow$ NPDO. First, it is not hard to see that if an algorithm satisfies the earlier strongly neighbor-preserving notion (Definition 3.1) as well as DO, then it also satisfies NPDO as stated below:

Lemma 3.7 (Strongly NP and DO imply NPDO)**.** *Suppose that an algorithm $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ is strongly neighbor-preserving w.r.t. $\sim_{\mathcal{X}}$ and $\sim_{\mathcal{Y}}$, as well as (ϵ, δ)-DO w.r.t. $\sim_{\mathcal{X}}$. Then, M satisfies (ϵ, δ)-NPDO.*

Proof. Fix a pair of neighboring input x, x'. By the strongly neighbor-preserving definition, for any $y, y' \in \mathcal{Y}$ such that $\Pr[\mathsf{M}(x) = y] > 0$ and $\Pr[\mathsf{M}(x') = y'] > 0$, it must be that $y \sim_{\mathcal{Y}} y'$. For any subset $S \subseteq \mathcal{V} \times \mathcal{Y}$, define the partial set $V(S) := \{v | \exists (v, y) \in S\}$. Then, we have

$$\begin{aligned}&\Pr[\mathsf{Exec}^{\mathsf{M}}(x) \in S] \leq \Pr[\mathsf{View}^{\mathsf{M}}(x) \in V(S)] \\ \leq& e^{\epsilon} \Pr[\mathsf{View}^{\mathsf{M}}(x') \in V(S)] + \delta = e^{\epsilon} \Pr[\mathsf{Exec}^{\mathsf{M}}(x') \in \mathcal{N}(S)] + \delta.\end{aligned}$$

□

(ϵ, δ)-NP. Next, we define another notion that captures the idea of "probabilistically approximate neighbor-preserving" called (ϵ, δ)-neighbor-preserving, or (ϵ, δ)-NP for short. We show that if an algorithm satisfies (ϵ, δ)-NP as well as (ϵ', δ')-DO, then it also satisfies $(\epsilon + \epsilon', \delta + \delta')$-NPDO.

Definition 3.8 ((ϵ, δ)-NP)**.** Given a mechanism $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ whose view space is $\mathcal{V}$, we say that it satisfies (ϵ, δ)-neighbor-preserving, or (ϵ, δ)-NP for short, w.r.t. $\sim_{\mathcal{X}}$ and $\sim_{\mathcal{Y}}$, iff for all $x \sim_{\mathcal{X}} x'$, for every view $v^* \in \mathcal{V}$ that happens with non-zero probability in $\mathsf{Exec}^{\mathsf{M}}(x)$ as well as $\mathsf{Exec}^{\mathsf{M}}(x')$, for every $Y \subseteq \mathcal{Y}$,

$$\begin{aligned}&\Pr[(v, y) \leftarrow \mathsf{Exec}^{\mathsf{M}}(x) : y \in Y \,|v = v^*] \\ &\qquad \leq e^{\epsilon} \cdot \Pr[(v', y') \leftarrow \mathsf{Exec}^{\mathsf{M}}(x') : y' \in \mathcal{N}(Y) \,\big|v' = v^*] + \delta \quad (3)\end{aligned}$$

where $\mathcal{N}(Y)$ contains all y' such that $y' \sim_{\mathcal{Y}} y$ for all $y \in Y$.

Intuitively, (ϵ, δ)-NP captures the idea that *conditioned on any view*, the algorithm, on neighboring inputs, must output probabilistically approximately close outputs.

Lemma 3.9 ((ϵ, δ)-NP and DO imply NPDO)**.** *Suppose that an algorithm $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ is (ϵ_1, δ_1)-DO and (ϵ_2, δ_2)-neighbor-preserving w.r.t. $\sim_{\mathcal{X}}$ and $\sim_{\mathcal{Y}}$. Then, M satisfies $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-NPDO w.r.t $\sim_{\mathcal{X}}$ and $\sim_{\mathcal{Y}}$.*

Proof. We slightly abuse the notation of $\mathcal{N}(\cdot)$ to denote the "neighbor set" for $\mathcal{Y}$ and also the $\mathcal{V} \times \mathcal{Y}$, as defined in Definition 3.8 and Definition 3.4. Fix a pair of neigboring input x, x'. For any $S \subseteq \mathcal{V} \times \mathcal{Y}$, denote the partial set $S_v = \{y \,|\exists (v, y) \in S\}$. Let $\mu(v) = \min(\Pr[\mathsf{View}^{\mathsf{M}}(x) = v], e^{\epsilon_1} \Pr[\mathsf{View}^{\mathsf{M}}(x') = v])$. It's trivial that $\sum_{v \in \mathcal{V}} \mu(v) \leq 1$. Also, since M is (ϵ_1, δ_1)-DO, we have $\sum_{v \in \mathcal{V}} \mu(v) \geq 1 - \delta_1$. Then,

$$\begin{aligned}&\Pr[\mathsf{Exec}^{\mathsf{M}}(x) \in S] \\ =& \sum_{v^* \in \mathcal{V}} \Pr[\mathsf{View}^{\mathsf{M}}(x) = v^*] \Pr[(v, y) \leftarrow \mathsf{Exec}^{\mathsf{M}}(x) : y \in S_{v^*} \,|v = v^*] \\ \leq& \delta_1 + \sum_{v^* \in \mathcal{V}} \mu(v^*) \Pr[(v, y) \leftarrow \mathsf{Exec}^{\mathsf{M}}(x) : y \in S_{v^*} \,|v = v^*] \\ \leq& \delta_1 + \sum_{v^* \in \mathcal{V}} \mu(v^*)(e^{\epsilon} \Pr[(v', y') \leftarrow \mathsf{Exec}^{\mathsf{M}}(x') : y' \in \mathcal{N}(S_{v^*}) \,\big|v' = v^*] + \delta_2) \\ \leq& \delta_1 + \delta_2 + \sum_{v^* \in \mathcal{V}} \mu(v^*)(e^{\epsilon} \Pr[(v', y') \leftarrow \mathsf{Exec}^{\mathsf{M}}(x') : y' \in \mathcal{N}(S_{v^*}) \,\big|v' = v^*]) \\ \leq& \delta_1 + \delta_2 + \sum_{v^* \in \mathcal{V}} e^{\epsilon_1} \Pr[\mathsf{View}^{\mathsf{M}}(x') = v^*] e^{\epsilon_2} \Pr[(v', y') \leftarrow \mathsf{Exec}^{\mathsf{M}}(x') : y' \in \mathcal{N}(S_{v^*}) \,\big|v' = v^*] \\ =& \delta_1 + \delta_2 + e^{\epsilon_1 + \epsilon_2} \Pr[\mathsf{Exec}^{\mathsf{M}}(x') \in \mathcal{N}(S)].\end{aligned}$$

□

3.5 Our Composition Theorem in Action

Using the simple motivating examples introduced so far, we can see our composition theorems in action.

Examples 1 and 2. As mentioned earlier, the first algorithm M_1 in either Example 1 or Example 2 satisfies strongly neighbor-preserving as well as (ϵ_1, δ_1)-DO. Therefore, they can be viewed as a special case of (ϵ, δ)-NPDO. Since M_2 in Example 1 or 2 satisfies (ϵ_2, δ_2)-DO, by our main composition theorem, we immediately reach the conclusion that the composed algorithm $\mathsf{M}_2 \circ \mathsf{M}_1$ satisfies $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DO.

Example 3. We can use Theorem 3.6 to prove that the subsampling algorithm of Example 3 satisfies (ϵ', δ')-DO w.r.t. $\sim_H$. To accomplish this, it suffices to show that the first algorithm, $\mathsf{M}_1 := \mathsf{InPlaceSample}$, satisfies $(0, 0)$-NPDO w.r.t. $\sim_H$ and $\sim_H$. Observe that in Example 3, two inputs are neighboring if their Hamming distance is at most 1, which implies that neighboring inputs must have the same length. Also, M_1 always generates a deterministic view that depends only on the length of the input. Therefore, to prove that $\mathsf{M}_1(x)$ satisfies $(0, 0)$-NPDO, it suffices to show that for any pair of neighboring inputs $x \sim_H x'$, for any subset of outputs $Y \subseteq \mathcal{Y}$ where $\mathcal{Y}$ is the output space of M_1,

$$\Pr[\mathsf{M}_1(x) \in Y] \leq \Pr[\mathsf{M}_1(x') \in \mathcal{N}(Y)], \tag{4}$$

where $\mathcal{N}(Y)$ denotes the set of all output arrays that are neighboring to some array in Y. Observe also that for any possible output y of $\mathsf{M}_1(x)$, let ρ be the random coins used for subsampling that led to the result y, then, if the same random coins ρ is encountered in an execution of $\mathsf{M}_1(x')$ on some neighboring $x' \sim_H x$, the outcome must be neighboring to y. Therefore, Equation (4) holds.

Example 4. Similarly, we can use Theorem 3.6 to prove that the subsampling algorithm of Example 4 satisfies $(\epsilon + \epsilon', \delta + \delta')$-DO w.r.t. $\sim_E$. By Theorem 3.6, it suffices to show that the $\mathsf{M}_1 := \mathsf{InPlaceSample}_{\epsilon,\delta}$ algorithm in Example 4 satisfies (ϵ, δ)-NPDO w.r.t. $\sim_E$ being both of the input and output neighboring notion. Recall that M_1 pads the input array with a random number of elements, such that the noisy length is n'. Then, it simply scans through the n' elements and either writes down the element if it is a real element and has been sampled, or writes down $\perp$. To show that M_1 satisfies (ϵ, δ)-NPDO, we will prove that M_1 satisfies $(0, 0)$-NP and (ϵ, δ)-DO, respectively, and then the conclusion follows from Lemma 3.9. It is easy to prove that M_1 satisfies (ϵ, δ)-DO. To see this, observe that the view depends only on the noisy input length where the noise is sampled according to a truncated geometric distribution.

Therefore, we focus on showing that M_1 satisfies $(0, 0)$-NP. Observe that in M_1, the random coins that determine the view and those that determine the output are independent. Therefore, it suffices to show that for any $x \sim_E x'$, for any $Y \subseteq \mathcal{Y}$ where $\mathcal{Y}$ is the output space of M_1,

$$\Pr[\mathsf{M}_1(x) \in Y] \leq \Pr[\mathsf{M}_1(x') \in \mathcal{N}(Y)].$$

Since $x \sim_E x'$, there can be at most one element in x that is not in x' (e.g., the element that is added or modified in x), and vice versa. Henceforth, we use $\mathsf{Common}(x, x')$ to denote the list of common elements that appear both in x and x'. Let $G(Y)$ be the event that there exists some $y \in Y$, such that the elements in $\mathsf{Common}(x, x')$ receive the same sampling decision as in y. We also say that $G(Y)$ represents the event that $\mathsf{Common}(x, x')$ receive coins compatible with Y. Therefore, we have that

$$\Pr[\mathsf{M}_1(x) \in Y] \leq \Pr[\mathsf{M}_1(x) : G(Y)] \leq \Pr[\mathsf{M}_1(x') \in \mathcal{N}(Y)].$$

In the above, the second inequality holds since conditioned on $\mathsf{Common}(x, x')$ receiving coins compatible with Y in a random execution of $\mathsf{M}_1(x')$, the outcome must be neighboring to some element in Y with probability 1.

Additional applications. Later in Section 5 , we use our composition framework to design a differentially oblivious stable compaction algorithm w.r.t. the edit distance — this building block was needed in Examples 1, 2, 4. Last but not the least, in Section 6 , we use our composition framework to prove an optimal privacy amplification theorem for the DO-shuffle model.

4 Proof of Main Composition Theorem

In this section, we shall prove our main composition theorem, that is, Theorem 3.6. A key stepping stone is the following equivalence lemma.

Lemma 4.1 (Equivalence of (ϵ, δ)-NPDO and existence of an (ϵ, δ)-matching)**.** *Assume the axiom of choice. Given a finite or countable infinite sample space Ω and a symmetric relation $\sim$ on Ω, consider two random variables $A, B \in \Omega$. The following statements are equivalent:*

1. *For every $S \subseteq \Omega$, $\Pr[A \in S] \le e^\epsilon \cdot \Pr[B \in \mathcal{N}(S)] + \delta$, where the neighbor set $\mathcal{N}(S)$ is defined as $\mathcal{N}(S) := \{b \in \Omega | \exists a \in S, a \sim b\}$.*
2. *There exists an (ϵ, δ)-matching $w : \Omega \times \Omega \to [0, 1]$ satisfying the following conditions:*
   (a) *For all $a, b \in \Omega$, $w(a, b) > 0$ only if $a \sim b$;*
   (b) *For all $a \in \Omega$, $\sum_{b \in \Omega, b \sim a} w(a, b) \le \Pr[A = a]$;*
   (c) *For all $b \in \Omega$, $\sum_{a \in \Omega, a \sim b} w(a, b) \le e^\epsilon \cdot \Pr[B = b]$;*
   (d) *$\sum_{a,b \in \Omega} w(a, b) \ge 1 - \delta$.*

Graph interpretation. Lemma 4.1 has a similar flavor as the Hall's theorem for bipartite graphs. The Hall's theorem says that if for each subset S of one component of a bipartitie graph, the size of its neighbor set satisfies $|\mathcal{N}(S)| \ge |S|$, then we can find a perfect matching in the graph. The proof of Lemma 4.1 is also inspired by the proof of the Hall's theorem.

We think of a bipartite graph where vertices on the left and right both come from the set Ω, and $w(a, b)$ defines the weight on edge (a, b). Imagine that each vertex $a \in \Omega$ on the left is factory that produces $\Pr[A = a]$ amount of produce, and each vertex $b \in \Omega$ on the right is a warehouse that can store up to $e^\epsilon \cdot \Pr[B = b]$ amount of produce. Condition (a) says that a factory is only allowed to route its produce to neighboring warehouses. The function w effectively defines a fractional flow such that almost all, i.e., $1 - \delta$ amount of produce is routed to some warehouse, and moreover, none of the warehouses exceed their capacity. For this reason, we also call w an (ϵ, δ)**-matching**.

Below, we prove our main composition theorem assuming that Lemma 4.1 holds.

Proof of Theorem 3.6. We directly prove the more general case when M_2 is (ϵ_2, δ_2)-NPDO. When M_2 is only (ϵ_2, δ_2)-DO, we can prove $\mathsf{M}_2 \circ \mathsf{M}_1$ is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DO with nearly the same argument.

Fix any neighboring input x, x'. By Lemma 4.1, there exists an (ϵ_1, δ_1)-matching $w : (\mathcal{V}_1 \times \mathcal{Y}) \times (\mathcal{V}_1 \times \mathcal{Y}) \to [0, 1]$ w.r.t the natural neighbor notion $\sim$ in the product space $\mathcal{V}_1 \times \mathcal{Y}$: $(v_1, y) \sim (v_1', y')$ when $v_1 = v_1'$ and $y \sim_{\mathcal{Y}} y'$. We want to prove that, for any subset $S \subseteq \mathcal{V}_1 \times \mathcal{V}_2 \times \mathcal{Z}$,

$$\Pr[\mathsf{Exec}^{\mathsf{M}_2 \circ \mathsf{M}_1}(x) \in S] \le e^{\epsilon_1 + \epsilon_2} \Pr[\mathsf{Exec}^{\mathsf{M}_2 \circ \mathsf{M}_1}(x') \in \mathcal{N}(S)] + \delta_1 + \delta_2.$$

Define the partial set $S_{v_1} := \{(v_2, z) | \exists (v_1, v_2, z) \in S\}$ for any $v_1 \in \mathcal{V}_1$. Then,

$$
\begin{aligned}
&\Pr[\mathsf{Exec}^{\mathsf{M}_2 \circ \mathsf{M}_1}(x) \in S] \\
=& \sum_{(v_1,y) \in \mathcal{V}_1 \times \mathcal{Y}} \Pr[\mathsf{Exec}^{\mathsf{M}_1}(x) = (v_1, y)] \cdot \Pr[\mathsf{Exec}^{\mathsf{M}_2}(y) \in S_{v_1}] \\
&\textit{(Use condition (a), (b) and (d) of the matching)} \\
\leq& \sum_{(v_1,y) \in \mathcal{V}_1 \times \mathcal{Y}, y' \sim_{\mathcal{Y}} y} w\left((v_1, y), (v_1, y')\right) \cdot \Pr[\mathsf{Exec}^{\mathsf{M}_2}(y) \in S_{v_1}] + \delta_1 \\
\leq& \sum_{(v_1,y) \in \mathcal{V}_1 \times \mathcal{Y}, y' \sim_{\mathcal{Y}} y} w\left((v_1, y), (v_1, y')\right) \cdot \left(e^{\epsilon_2} \Pr[\mathsf{Exec}^{\mathsf{M}_2}(y') \in \mathcal{N}(S_{v_1})] + \delta_2\right) + \delta_1 \\
&\textit{(Use condition (b) of the matching)} \\
\leq& \sum_{(v_1,y) \in \mathcal{V}_1 \times \mathcal{Y}, y' \sim_{\mathcal{Y}} y} w\left((v_1, y), (v_1, y')\right) \cdot \left(e^{\epsilon_2} \Pr[\mathsf{Exec}^{\mathsf{M}_2}(y') \in \mathcal{N}(S_{v_1})]\right) + \delta_2 + \delta_1 \\
&\textit{(Use condition (c) of the matching)} \\
\leq& \sum_{(v_1,y') \in \mathcal{V}_1 \times \mathcal{Y}} e^{\epsilon_1} \Pr[\mathsf{Exec}^{\mathsf{M}_1}(x') = (v_1, y')] \cdot \left(e^{\epsilon_2} \Pr[\mathsf{Exec}^{\mathsf{M}_2}(y') \in \mathcal{N}(S_{v_1})]\right) + \delta_2 + \delta_1 \\
=& e^{\epsilon_1 + \epsilon_2} \Pr[\mathsf{Exec}^{\mathsf{M}_2 \circ \mathsf{M}_1}(x') \in \mathcal{N}(S)] + \delta_2 + \delta_1.
\end{aligned}
$$

□

Proof of Lemma 4.1. For any matching w, we use the notation of the partial sum of the matching: for any $a \in \Omega$, $w(a, \cdot) := \sum_{b \sim a} w(a, b)$. Similarly, for any $b \in \Omega$, we have $w(\cdot, b) := \sum_{a \sim b} w(a, b)$. We also define the size of a matching w as $|w| := \sum_{a,b,a \sim b} w(a, b)$.

We first prove that if there is an (ϵ, δ)-matching, the first statement holds. We prove it by contradiction. Suppose there is a set S violates the first condition, i.e., $\Pr[A \in S] > e^{\epsilon} \Pr[B \in \mathcal{N}(S)] + \delta$, while there is an (ϵ, δ)-matching w. We have that $\Pr[A \in S] - \sum_{a \in S, b \sim x} w(a, b) \geq \Pr[A \in S] - e^{\epsilon} \Pr[B \in \mathcal{N}(S)] > \delta$. Then, $1 - \sum_{a,b,a \sim b} w(a, b) = \left(\Pr[A \in S] - \sum_{a \in S, b \sim a} w(a, b)\right) + \left(\Pr[A \in \Omega / S] - \sum_{a \in \Omega / S, b \sim a} w(a, b)\right) > \delta + 0$. So $\sum_{a,b} w(a, b) < 1 - \delta$, which contradicts the condition (d) of the matching.

Now, we prove that if the first statement holds, we can find an (ϵ, δ)-matching w. This direction is more challenging. We first prove it assuming Ω is finite. Let w^* be the "max legal" matching. When we say w^* is legal, we mean that it satisfies the condition (a)-(c) for (ϵ, δ)-matching. Also, the maximality of w^* means the the size of w^* is more than or equal to any other leagl matching w. We prove that, if $|w^*| < 1 - \delta$, then we can find a set S that violates the condition in the first statement. Let S_0 be the set of all those $a \in \Omega$ that are not "saturated", i.e., $\sum_{b \sim a} w^*(a, b) < \Pr[A = a]$. Now, for each iteration, we keep expanding the set S_i to S_{i+1}. Define $T_i := \{a \in A | \exists b \in \mathcal{N}(S_i), w^*(a, b) > 0\}$. Let $S_{i+1} = S_i \cup T_i$. We keep expanding the set. Since Ω is finite, there exists a maximal S_i that cannot be expanded, i.e., $S_i \cup T_i = S_i$. Notice that for i, all $b \in \mathcal{N}(S_{i+1})$ are saturated. Otherwise, we can find a pair of unsaturated vertices a_0, b_i, such that there exists an alternating path $a_0, b_0, a_1, b_1, \cdots, a_i, b_i$ where for all $j \in [i - 1]$, $a_j \sim b_j$ and $w^*(a_{j+1}, b_j) > 0$. Now, let $\Delta = \min\{\Pr[A = a] - w^*(a, \cdot), e^{\epsilon} \Pr[B = b] - w^*(\cdot, b), \min_{j \in [i]}\{w^*(a_{j+1}, b_j)\}\}$. We know $\Delta > 0$. For each $j \in [i - 1]$, we simply raise $w^*(a_j, b_j)$ by Δ and decrease $w^*(a_{j+1}, b_j)$ by Δ. This procedure raises the size of w^* by Δ while preserving all legal conditions, which breaks the maximality of w^*.

Now, we argue that this maximal set S_i violates the first statement. We first notice that all $a \in \Omega/S_i$ are saturated. So $\Pr[A \in \Omega/S_i] - \sum_{a\in\Omega/S_i} w^*(a,\cdot) = 0$. Then, $\Pr[A \in S_i] - \sum_{a\in S_i} w^*(a,\cdot) = (\Pr[A \in S_i] - \sum_{a\in S_i} w(a,\cdot)) + (\Pr[A \in \Omega/S_i] - \sum_{a\in\Omega/S_i} w(a,\cdot)) = 1 - |w^*| > \delta$. Now, we know that all $b \in \mathcal{N}(S_i)$ are saturated. Also, $\mathcal{N}(S_i)$ are only saturated by the vertices in S_i, otherwise we can expand S_i more. So we have that $\Pr[A \in S_i] - e^\epsilon \Pr[B \in \mathcal{N}(S_i)] = \Pr[A \in S_i] - \sum_{a\in S_i, b\in\mathcal{N}(S_i), a\sim b} w(a,b) > \delta$. Therefore, S_i violates the condition in the first statement.

Now we discuss the case when Ω is countable infinite. We will try to use the same argument as the finite case, but we need to handle a few additional details. First, is the maximal matching w^* well-defined and attainable in the infinite case? This turns out to be non-trivial. Luckily, our matching problem can be seen as a "locally finite" network flow problem in a countable inifinite graph. Due to the Theorem 5.1 in Aharoni et al. [ABG+11][3], the maximum flow is attainable for any countable infinite and locally finite graph, which implies the maximum matching in our setting is well-defined and attainable in our case. Notice that the axiom of choice is assumed in the proof from Aharoni et al. Also, the expanding procedure could take infintie number of steps to stop. Now, we can simply define the set S_{final} that violates the condition as the union of all S_i for $i \in 0, 1, 2 \ldots$. This maximal set S_{final} is well-defined by Zorn's Lemma. Next, we know that all $y \in \mathcal{N}(S_{\text{final}})$ are saturated. Although the number of vertices in $\mathcal{N}(S_{\text{final}})$ could be infinite, there exists a finite-length alternative path from an unsaturated x to each y in $\mathcal{N}(S_{\text{final}})$, since our expanding prodcedure goes in round $i \in 0, 1, 2 \cdots$. So if there is any unsaturated $y \in \mathcal{N}(S_{\text{final}})$, the maximality of w^* are broken. Every other parts of the proof still hold in the countable infinite case.

□

5 Application: DO Compaction w.r.t. Edit Distance

Earlier in our Examples 1, 2, and 4, we assumed a stable compaction algorithm that is differentially oblivious w.r.t. the *edit* distance. Chan et al. [CCMS19] showed how to construct a stable compaction algorithm that is (ϵ, δ)-DO w.r.t. the *Hamming* distance [CCMS19], taking $O(n(\log\log n + \log\log\frac{1}{\delta}))$ time to compact an array of size n (assuming that ϵ is a constant). However, we are not aware of any straightforward way to modify their algorithm to work for edit distance. Another naïve approach is to use oblivious sorting directly but this would incur $\Theta(n \log n)$ runtime which is asymptotically worse. In this section, we fill in this missing piece that is needed by Examples 1, 2, and 4. We will describe a stable compaction algorithm that works for edit distance and it preserves the runtime of Chan et al. [CCMS19]. Intriguingly, the design of our new compaction algorithm turns out to be a great example that demonstrates the power of our compositional framework.

5.1 Additional Preliminaries

Stable compaction. Recall that in stable compaction, we are given an input array which is written on an input tape. Some elements in the input array are *real* elements, and others are *fillers*. We want to output an array that contains only the real elements, and moreover, they must appear in the same order as the input array. We assume that the input array is written on the input tape, and its true length is written on some designated location on the input tape. Similarly, the

[3] A followup paper by Lochbihler [Loc22] pointed out and fixed some technical issues of the proof in Aharoni et al. [ABG+11].

algorithm should write the output array to an output tape, and the true length of the output array should be written to some dedicated location on the output array.

Stable Oblivious Sorting. Suppose we are given an input array I containing a list of m elements with a key attached to each element. Earlier works [CGLS18, LSX19] showed how to oblivious sort the array according to the keys in $O(m \log m)$ runtime while maintaining the stable property: the elements will be ordered by their relative order in the original array when their keys are the same.

Differentially private prefix sum. Given an input array I containing a list of m integers, we want to its prefix sums. We say that two inputs I, and I' are neighboring iff 1) they have the same length and 2) they differ in at most one position j, and $|I[j] - I'[j]| \leq 1$. Earlier works [DNPR10, CSS10, CSS11] showed how to construct a prefix sum mechanism that satisfies (ϵ, δ)-differential privacy, and moreover, the mechanism satisfies the following properties: 1) The access patterns (i.e., view) of the algorithm depend only on the input length; 2) The additive error is upper bounded by $O\left(\frac{1}{\epsilon}(\log |I|)^{1.5} \log \frac{1}{\delta}\right)$ with probability 1.

5.2 Roadmap and Intuition

Our algorithm $\mathsf{Compact}$ is the composition of the following two algorithms, i.e., $\mathsf{Compact}(\cdot) = \mathsf{CompactBin} \circ \mathsf{RandBin}(\cdot)$. Suppose we can prove that $\mathsf{RandBin}$ is (ϵ_1, δ_1)-NPDO and prove that $\mathsf{CompactBin}$ is (ϵ_2, δ_2)-NPDO, we have $\mathsf{Compact}$ is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-NPDO due to our main composition theorem 3.6.

1. $\mathsf{RandBin}$: Given an input array I containing real elements and fillers, and whose true length is stored in a dedicated location on the input tape, $\mathsf{RandBin}$ outputs a list of B bins denoted $(\mathsf{Bin}_i^{(Z)} : i \in [B])$, each of capacity Z. Each bin contains a random number of real elements and the rest are fillers. Furthermore, the ordered list of all real elements in all bins is the same as the ordered list of real elements in the input. The algorithm should output the parameters B and Z to some dedicated location on the output tape.

2. $\mathsf{CompactBin}$: Given a list of B bins denoted $(\mathsf{Bin}_i^{(Z)} : i \in [B])$ each of capacity Z, where the parameters B and Z are stored in some dedicated location on the input tape, the $\mathsf{CompactBin}$ algorithm outputs a compacted array containing only the real elements in the input bins, and preserving the same order they appear in the input bins. The algorithm outputs the true output length to some dedicated location on the output tape.

In short, $\mathsf{RandBin}$ is a pre-processing step that takes the input array and converts it into bin format, and $\mathsf{CompactBin}$ takes the bin representation, and performs the actual compaction. The informal intuition is as follows. From earlier work [CCMS19], we know how to construct an efficient DO stable compaction algorithm for Hamming distance. However, in our case, we have two inputs I and I' that have *edit* distance 1. The difficulty with edit distance is when I is obtained by inserting an extra element into I' at position j, the two inputs I and I' will differ in *every* position after j. Our idea is to leverage $\mathsf{RandBin}$ to "probabilistically localize" this difference caused by a single insertion operation. In particular, if some bin representation occurs for input I with some probability p, we want that under the neighboring input I', with probability close to p, we should encounter a similar bin representation where the difference is localized to only one or two bins. If we can accomplish this, then hopefully we can adapt ideas that worked for Hamming distaince [CCMS19] to compact the resulting bin representation.

Below, we will define an appropriate neighboring notion $\sim_B$ on the bin representation. We want to show that the RandBin pre-processing step satisfies NPDO w.r.t. the input relation $\sim_E$ and output relation $\sim_B$ for the bin representation. Further, we want to show that CompactBin satisfies NPDO w.r.t. $\sim_B$ and $\sim_H$. Then, the composed algorithm should be NPDO by our composition theorem.

Neighboring relation for bin representation $\sim_B$. Specifically, the neighboring relation $\sim_B$ is defined as below. Two lists of bins $(\mathsf{Bin}_i^{(Z)} : i \in [B])$ and $(\mathsf{Bin}_i^{'(Z')} : i \in [B'])$ are said to be neighboring, iff the following all hold:

- they have compatible dimensions, i.e., $B = B'$ and $Z = Z'$;
- After removing all fillers and concatenating the real elements in the list of bins, the resulting outcomes have edit distance at most one;
- There are at most *two* bins that have different bin loads (defined to be the number of real elements in the bin), further, for both of them, the difference in load is at most one.

5.3 RandBin Algorithm

We now describe the RandBin algorithm, which preprocesses the input array into a bin representation.

$\underline{\mathsf{RandBin}^{\epsilon,\delta}(I)}$: // *Let* $\epsilon_1 = \epsilon_2 = \epsilon_3 = \frac{\epsilon}{3}$, *and* $\delta_1 = \delta_2 = \delta_3 = \frac{\delta}{3}$.

- Sample $G \xleftarrow{\$} \mathcal{G}(\epsilon_1, \delta_1)$. Let $L = |I| + G$ be the noisy input length. Let $s = \Theta(\frac{1}{\epsilon} \log^2 L \log \frac{1}{\delta})$ be an upper bound on the support of $\mathcal{G}(\epsilon_2, \delta_2)$ and also the additive error of $\mathsf{PrefixSum}^{\epsilon_3,\delta_3}$ on at most L integers. Let the maximum bin load be $Z = 2s$ and $B = \lceil \frac{2L}{Z} \rceil + 1$ be the number of bins.
- For $i = 1$ to B, let $\rho_i \xleftarrow{\$} s + \mathcal{G}(\epsilon_2, \delta_2) \in [\frac{Z}{2} \dots Z]$. Let $\boldsymbol{\rho} := (\rho_1, \rho_2, \dots, \rho_B)$.
- Let $\mathsf{cnt} := \mathsf{PrefixSum}^{\epsilon_3,\delta_3}(\boldsymbol{\rho}) \in \mathbb{Z}^B$. Let $\mathbf{Buf} := \emptyset$ be a working buffer.
- For $i = 1$ to B:
  - fetch the unvisited elements in the input array up to index[a] $\mathsf{cnt}[i] + s$ and add them to Buf; mark them as visited.
  - if the current length of Buf is less than Z, append enough fillers such that its length is at least Z;
  - perform stable oblivious sorting on Buf such that the first Z positions contains only the real elements coming from the first $\sum_{j \le i} \rho_i$ positions in the input and fillers; all remaining elements are moved to the end of Buf.
  - pop the first Z elements of Buf to Bin_i.
  - perform stable oblivious sorting on Buf to move all the fillers to the end, if necessary, truncate Buf such that its length is at most $2s$.
- Output the bin representation $(\mathsf{Bin}_i^{(Z)} : i \in [B])$, and store the parameters B and Z in some dedicated location on the output tape.

[a] We may assume that any location in the input array beyond the original length $|I|$ is occupied by a filler.

Roughly speaking, the RandBin algorithm generates a list of random counts $\boldsymbol{\rho} := (\rho_1, \ldots, \rho_B)$. Then, all real elements contained in the first ρ_1 positions of the input are moved into Bin_1, all real elements contained in the next ρ_2 positions of the input are moved into Bin_2, and so on. To guarantee differential obliviousness, the algorithm cannot directly reveal the vector $\boldsymbol{\rho}$ — instead, it reveals only the noisy prefix sum of $\boldsymbol{\rho}$. Specifically, we apply an (ϵ_3, δ_3)-differentially private prefix sum algorithm to the vector y, i.e., $\mathsf{cnt} := \mathsf{PrefixSum}^{\epsilon_3,\delta_3}(Y)$. In other words, $\mathsf{cnt}[i]$ stores a noisy version of $\sum_{j\le i}\rho_i$, and it is guaranteed that the estimation error is at most s. Now, in each step i of the algorithm, we want to populate Bin_i. To do so, we simply fetch the next batch of elements in the input array upto position $\mathsf{cnt}[i]$ into a poly-logarithmically sized working buffer Buf. Buf also contains previously fetched elements that have not been placed into any bin yet. We can now obliviously sort Buf to create the next Bin_i. At the end of each step i, it is guaranteed that there are at most $2s$ real elements remaining in Buf. Therefore, we can obliviously sort Buf and compact its length to $2s$. This makes sure that Buf is always poly-logarithmic in size. Finally, to make the algorithm secure, we also need to mask the true input length, and we can accomplish this by adding a truncated geometric random noise to the true length, and reveal only the noisy length. Note that the number of bins B is a random variable that depends only on the noisy input length.

Theorem 5.1. *The* RandBin *algorithm outputs the correct bin representation with probability 1. That is, for every $i \in [B]$, all real elements from $I\left[\sum_{j<i}\rho_j + 1\right]$ to $I\left[\sum_{j\le i}\rho_j\right]$ are moved to Bin_i. Also, all real elements in the input array will be moved to the bins.*

Proof. We prove the correctness statement by induction. We assume that for the first $i-1$ iteration, $\mathsf{Bin}_1 \ldots \mathsf{Bin}_{i-1}$ have the correct real elements and also, no real elements have been truncated.

Now, in the i-th iteration, we will read the input array up to index $\mathsf{cnt}[i] + s$. Since s is the upper bound on the additive error of the prefix-sum mechanism. we have $|\mathsf{cnt}[i] - \sum_{j\le i}\rho_j| \le s$ and thus, $\mathsf{cnt}[i] + s \ge \sum_{j\le i}\rho_j$. Then, we know the real elements between $I\left[\sum_{j<i}\rho_j + 1\right]$ to $I\left[\sum_{j\le i}\rho_j\right]$ will be visited. By the assumption, they are not moved to the bins before, so they must still be in the buffer. Then, the algorithm obliviously sort the buffer and correctly move those real elements to Bin_i. Now, the real elements from $I[\sum_{j\le i}\rho_j + 1]$ to $I[\mathsf{cnt}[i] + s]$ are in Buf. Again, since $|\mathsf{cnt}[i] - \sum_{j\le i}\rho_j| \le s$, we have $\mathsf{cnt}[i] + s - \sum_{j\le i}\rho_j \le 2s$. So there will be no more than $2s$ real elements and after another round of oblivious sorting, no real elements will be truncated.

Finally, we have $\rho_i \ge Z/2$, so $\sum_{i\le B}\rho_i \ge BZ/2 \ge L \ge |I|$. Thus, we will visit the whole input and all the real elements will be moved to the correct bins. □

Theorem 5.2. *Assuming $|I| = \Omega(\frac{1}{\epsilon}\log\frac{1}{\delta})$,* RandBin *has a worst-case runtime of $O\left(|I|\left(\log\log|I| + \log\frac{1}{\epsilon} + \log\log\frac{1}{\delta}\right)\right)$.*

Proof. Before each iteration, Buf has size at most $2s$. For each iteration i, we will move at most $\mathsf{cnt}[i] - \mathsf{cnt}[i-1] + s$ elements to the buffer. Since $|\mathsf{cnt}[i] - \sum_{j\le i}\rho_j| \le s$ and $|\mathsf{cnt}[i-1] - \sum_{j\le i-1}\rho_j| \le s$, we have $\mathsf{cnt}[i] - \mathsf{cnt}[i-1] \le \rho_i + 2s$. Then, $\mathsf{cnt}[i] - \mathsf{cnt}[i-1] + s \le \rho_i + 2s \le Z + 2s \le 4s$. Thus, we know the buffer size cannot be more than $6s$, which implies that each oblivious sorting on Buf takes $O(s\log s)$ time. Also, we know $Z = 2s$, so we have $B = \lceil\frac{L}{s}\rceil + 1$. The total time complexity will be $O(Bs\log s) = O(L\log s)$. Since $|I| = \Omega\left(\frac{1}{\epsilon}\log\frac{1}{\delta}\right)$, we have that $L \le |I| + O\left(\frac{1}{\epsilon}\log\frac{1}{\delta}\right) = O(I)$. Finally, since $s = \Theta(\frac{1}{\epsilon}\log^2 L \cdot \log\frac{1}{\delta})$, the time complexity is $O\left(|I|\left(\log\log|I| + \log\frac{1}{\epsilon} + \log\log\frac{1}{\delta}\right)\right)$. □

Theorem 5.3. RandBin *is (ϵ, δ)-NPDO w.r.t. the input neighboring notion $\sim_E$ and the output neighboring notion $\sim_B$.*

Proof. Recall that we need to prove the following statement: for any subset S of (view, output) pairs, for any neighboring input pair I, I', it holds that

$$\Pr[\mathsf{Exec}^{\mathsf{RandBin}}(I) \in S] \leq e^{\epsilon} \Pr[\mathsf{Exec}^{\mathsf{RandBin}}(I') \in \mathcal{N}(S)] + \delta. \tag{5}$$

where $\mathsf{Exec}^{\mathsf{RandBin}}$ returns the view and the output bin representation, and the neighboring set $\mathcal{N}$ is defined w.r.t. the relation $\sim_B$ (i.e., neighboring of bin representation).

First, observe that the adversary's view is fully determined knowing the noisy input length L and the noisy bin loads cnt. Therefore, henceforth, we may use (L, cnt) to denote the view. Second, observe that the output of the algorithm is fully determined by the input I, the noisy input length L (which determines the parameters Z and B), and the actual random bin load vector $\boldsymbol{\rho}$. Now, we define the notation $\overline{\mathsf{Exec}}(I)$ to output L, cnt, as well as $\boldsymbol{\rho}$ — notice that here, we have switched the output to just the bin load vector $\boldsymbol{\rho}$ rather than the actual bin representation output. Henceforth, let $\mathcal{Y}$ be the support of the variable $\boldsymbol{\rho}$, and we abuse notation slightly and use $\mathcal{V}$ to denote the support of the pair (L, cnt). For any given subset $S \subseteq \mathcal{V} \times \mathcal{Y}$, we define the neighboring set notation $\overline{\mathcal{N}}$ (parametrized by I and I') as follows:

$$\overline{\mathcal{N}}(S) = \{(L, \mathsf{cnt}, \boldsymbol{\rho}') \mid \exists \boldsymbol{\rho}' \text{ s.t. } (L, \mathsf{cnt}, \boldsymbol{\rho}) \in S \text{ and } \mathsf{out}(I, L, \boldsymbol{\rho}) \sim_B \mathsf{out}(I', L, \boldsymbol{\rho}')\}$$

where $\mathsf{out}(I, L, \boldsymbol{\rho})$ outputs the unique bin representation that is fully determined by the input I, the noisy input length L, and the bin load vector $\boldsymbol{\rho}$.

Now, proving Equation (5) is equivalent to proving that for any $S \subseteq \mathcal{V} \times \mathcal{Y}$, any neighboring inputs I, I', it holds that

$$\Pr[\overline{\mathsf{Exec}}(I) \in S] \leq e^{\epsilon} \Pr[\overline{\mathsf{Exec}}(I') \in \overline{\mathcal{N}}(S)] + \delta.$$

We consider these following cases.

Case 1. A substitution transforms I to I'. This is an easy case. Fix any $(L, \mathsf{cnt}, \boldsymbol{\rho}) \in \mathcal{V} \times \mathcal{Y}$ such that $\Pr[\overline{\mathsf{Exec}}(I) = (L, \mathsf{cnt}, \boldsymbol{\rho})] > 0$. Let's assume that the substituted elements x are placed in Bin_i in the bin representation $\mathsf{out}(I, L, \boldsymbol{\rho})$. Since I and I' only differ by a substitution, x will be in Bin'_i in the bin representation $\mathsf{out}(I', L, \boldsymbol{\rho})$ with the same index i. All other bins are exactly the same. Also, the i-th bin will have real bin loads differ at most one. Therefore, by the definition of neighboring relation for bin representation, we know $\mathsf{out}(I, L, \boldsymbol{\rho}) \sim_B \mathsf{out}(I', L, \boldsymbol{\rho})$. Thus, in this case, given any $S \subseteq \mathcal{V} \times \mathcal{Y}$, $S \subseteq \overline{\mathcal{N}}(S)$. Moreover, the equation $\Pr[\overline{\mathsf{Exec}}(I) = (L, \mathsf{cnt}, \boldsymbol{\rho})] = \Pr[\overline{\mathsf{Exec}}(I') = (L, \mathsf{cnt}, \boldsymbol{\rho})]$ trivially holds in this case because $|I| = |I'|$.

Case 2. An insertion or deletion transforms I to I'.

Fix any $(L, \mathsf{cnt}, \boldsymbol{\rho}) \in \mathcal{V} \times \mathcal{Y}$ such that $\Pr[\overline{\mathsf{Exec}}(I) = (L, \mathsf{cnt}, \boldsymbol{\rho})] > 0$.

(i) Suppose an element x is deleted from I to form I'. Suppose i is the index of the bin in $\mathsf{out}(I, L, \boldsymbol{\rho})$ that contains x; observe the choice of B ensures that $i < B$. We write $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_B)$. We consider a $\boldsymbol{\rho}' = (\rho'_1, \ldots, \rho'_B)$, such $\rho'_{i+1} = \rho_{i+1} - 1$ and $\rho'_j = \rho_j$ for $j \neq i+1$. We first prove that $\mathsf{out}(I, L, \boldsymbol{\rho}) \sim_B \mathsf{out}(I', L, \boldsymbol{\rho}')$. We only need to check the bin load condition and the other two conditions are naturally satisfied. We write $\mathsf{out}(I, L, \boldsymbol{\rho}) = (\mathsf{Bin}_i : i \in [B])$ and $\mathsf{out}(I', L, \boldsymbol{\rho}') = (\mathsf{Bin}'_i : i \in [B])$. It is easy to see that $\mathsf{Bin}_1 = \mathsf{Bin}'_1, \ldots, \mathsf{Bin}_{i-1} = \mathsf{Bin}'_{i-1}$. Also, $\mathsf{Bin}_{i+2} = \mathsf{Bin}'_{i+2}, \ldots, \mathsf{Bin}_B = \mathsf{Bin}'_B$, because $\rho'_{i+1} = \rho_{i+1} - 1$ offsets the influence of the deletion of x for all following bins. Thus, only bin i and $i+1$ will be different in $\mathsf{out}(I, L, \boldsymbol{\rho})$ and $\mathsf{out}(I, L, \boldsymbol{\rho}')$. For example, suppose $I = a\bot bcd\bot \ldots$, $I' = abcd\bot \ldots$ where a, b, c, d are the real elements and $\bot$ denotes the filler elements. Assuming $\boldsymbol{\rho} = (3, 3, \ldots)$ and $\boldsymbol{\rho}' = (3, 2, \ldots)$, the corresponding bin representation will be $\underline{a\bot b}$ $\underline{cd\bot}$ and $\underline{abc}$ $\underline{d\bot\bot}$, where the first two bins will have different bin loads and the differences are at most one for each bin. More formally, since $\rho_i = \rho'_i$ and I' is missing the element x, Bin_i has the

particular element x, while Bin'_i has the element $I[\sum_{j\le i} y_j + 1]$ instead (which will be the same as $I'[\sum_{j\le i} y_j]$). Thus, the real element numbers in Bin_i and Bin'_i may differ by at most one. Also, Bin'_{i+1} simply misses the first element in Bin_{i+1} and every other elements are the same. So the real element numbers in Bin_{i+1} and Bin'_{i+1} may differ by at most one. Henceforce, we prove that $\mathsf{out}(I, L, \boldsymbol{\rho}) \sim_B \mathsf{out}(I', L, \boldsymbol{\rho}')$. For convenience, we define a mapping $\varphi_L : \mathcal{Y} \to \mathcal{Y}$, such that it maps every $\boldsymbol{\rho}$ to $\boldsymbol{\rho}'$ as the construction above. Notice that φ_L relies on L because the bin size Z and the bin number B relies on a fixed L, and then we restrict $\mathsf{dom}(\varphi_L)$ to those $\boldsymbol{\rho}$ that have exactly B bins of size Z.

We then prove $\varphi_L(\boldsymbol{\rho})$ is injective by contradiction. Suppose there are $\boldsymbol{\rho}^0, \boldsymbol{\rho}^1 \in \mathsf{dom}(\varphi_L)$ such that $\boldsymbol{\rho}^0 \neq \boldsymbol{\rho}^1$ but $\varphi_L(\boldsymbol{\rho}^0) = \varphi_L(\boldsymbol{\rho}^1)$. Assume x is in Bin_{i_0} in $\mathsf{out}(I, L, \boldsymbol{\rho}^0)$ and in Bin_{i_1} in $\mathsf{out}(I, L, \boldsymbol{\rho}^1)$, and $i_0 < i_1$ without loss of generality. We know that the prefix $\boldsymbol{\rho}^0_{1\ldots i_0}$ are the same as the prefix $\varphi_L(\boldsymbol{\rho}^0)_{1,\ldots,i_0}$ by the construction of φ_L. Also, the prefix $\boldsymbol{\rho}^1_{1,\ldots,i_1}$ are the same as the prefix $\varphi_L(\boldsymbol{\rho}^1)_{1,\ldots,i_1}$. Thus, the prefix $\boldsymbol{\rho}^0_{1,\ldots,i_0}$ are the same as the prefix $\boldsymbol{\rho}^1_{1,\ldots,i_0}$ and by the definition of out, x should be in Bin_{i_0} of $\mathsf{out}(I, L, \boldsymbol{\rho}^1)$. This is a contradiction.
(ii) Suppose an element x is inserted into I to from I'. Consider the bin representation $\mathsf{out}(I, L, \rho)$. Suppose x is in Bin_i. We can define an injective mapping φ similarly as the mapping in part (i): $\varphi_L(\boldsymbol{\rho}) = \boldsymbol{\rho}'$ such that $\rho'_{i+1} = \rho_{i+1} + 1$ and $\rho'_j = \rho_j$ for $j \neq i + 1$. We can prove φ_L is injective and the correponding bin representation are neighbors by similar arguments in part (i).

It remains to show the (ϵ, δ)-differentially private inequality. We can use the same proof for insertion and deletion, because $\|\boldsymbol{\rho} - \varphi_L(\boldsymbol{\rho})\|_1 = 1$ in both cases.

We abused the notation before and now we make a clear distinction: the notations $L, \mathsf{cnt}, \boldsymbol{\rho}$ denote the random variables, while the notations ℓ, cnt, ρ denote the particular point in the sample space.

For any measurable $S \subseteq \mathcal{V} \times \mathcal{Y}$ and $\ell \in \mathbb{Z}_+, \rho \in \mathcal{Y}$, denote $S_{\ell,\rho} := \{cnt : (\ell, cnt, \rho) \in S\}$. We also denote $\varphi(S) = \{(\ell, cnt, \varphi_\ell(\rho)) \,|(\ell, cnt, \rho) \in S\}$.

Observe that following properties:

1. First, since $d_{\mathsf{E}}(I, I') \le 1$, for all $\ell \in \mathbb{Z}_+$, $\Pr[L = \ell] \le e^{\epsilon_1} \Pr[L' = \ell] + \delta_1$.

2. Because of the truncated geometric distribution $\mathsf{Geom}(\epsilon_2, \delta_2)$ and the choice of Z, we have $\Pr[\boldsymbol{\rho} \notin \mathsf{dom}(\varphi_\ell) | L = \ell] \le \delta_2$.

3. For $\rho \in \mathsf{dom}(\varphi_\ell)$, since ρ and ρ' in both executions have the same distribution, we have $\Pr[\boldsymbol{\rho} = \rho | L = \ell] \le e^{\epsilon_2} \cdot \Pr[\boldsymbol{\rho}' = \varphi_\ell(\rho) | L' = \ell]$, noting that $\|\rho - \varphi_\ell(\rho)\|_1 = 1$.

4. For $\rho \in \mathsf{dom}(\varphi_\ell)$, again we have $\|\rho - \varphi_\ell(\rho)\| = 1$. Then, the (ϵ_3, δ_3)-differentially private prefix sum mechanism implies that $\Pr[\mathsf{cnt} \in S_{\ell,y} | \boldsymbol{\rho} = \rho, L = \ell] \le e^{\epsilon_3} \cdot \Pr[\mathsf{cnt}' \in S_{\ell,\rho} | \boldsymbol{\rho}' = \varphi_\ell(\rho), L' = \ell] + \delta_3$.

   Then, we have the following inequalities:

$$
\begin{aligned}
& \Pr[\overline{\mathsf{Exec}}(I) = (L, \mathsf{cnt}, \boldsymbol{\rho}) \in S | L = \ell] \\
\leq & \sum_{\rho \in \mathsf{dom}(\varphi_\ell)} \Pr[\boldsymbol{\rho} = \rho | L = \ell] \cdot \Pr[\mathsf{cnt} \in S_{\ell,\rho} | \boldsymbol{\rho} = \rho, L = \ell] + \delta_2 \\
\leq & \sum_{\rho \in \mathsf{dom}(\varphi_\ell)} \Pr[\boldsymbol{\rho} = \rho | L = \ell] \cdot (e^{\epsilon_3} \cdot \Pr[\mathsf{cnt}' \in S_{\ell,\rho} | \boldsymbol{\rho}' = \varphi_\ell(\rho), L' = \ell] + \delta_3) + \delta_2 \\
\leq & \sum_{\rho \in \mathsf{dom}(\varphi_\ell)} e^{\epsilon_3 + \epsilon_2} \Pr[\boldsymbol{\rho}' = \varphi_\ell(\rho) | L' = \ell] \cdot \Pr[\mathsf{cnt}' \in S_{\ell,\rho} | \boldsymbol{\rho}' = \varphi_\ell(\rho), L' = \ell] + \delta_3 + \delta_2 \\
= & \sum_{\rho' \in \mathsf{range}(\varphi_\ell)} e^{\epsilon_3 + \epsilon_2} \Pr[\boldsymbol{\rho}' = \rho' | L' = \ell] \cdot \Pr[\mathsf{cnt}' \in S_{\ell,\varphi_\ell^{-1}(\rho')} | \boldsymbol{\rho}' = \rho', L' = \ell] + \delta_3 + \delta_2 \\
= & \ e^{\epsilon_3 + \epsilon_2} \cdot \Pr[\overline{\mathsf{Exec}}(I') = (L', \mathsf{cnt}', \boldsymbol{\rho}') \in \varphi(S) | L' = \ell] + \delta_3 + \delta_2,
\end{aligned}
$$

where the last equality holds because for all $\rho' \in \mathsf{range}(\varphi_\ell)$, $cnt' \in S_{\varphi_\ell^{-1}(\rho')}$ iff $(\ell, cnt', \rho') \in \varphi(S)$.

Since $\Pr[L = \ell] \leq e^{\epsilon_1} \cdot \Pr[L' = \ell] + \delta_1$ for all ℓ, using a standard DP-composition argument, we have

$$\Pr[\overline{\mathsf{Exec}}(I) = (L, \mathsf{cnt}, \boldsymbol{\rho}) \in S] \leq e^{\epsilon_1 + \epsilon_2 + \epsilon_3} \cdot \Pr[\overline{\mathsf{Exec}}(I') = (L', \mathsf{cnt}', \boldsymbol{\rho}') \in \varphi(S)] + \delta_3 + \delta_2 + \delta_1.$$

Finally, since φ_ℓ is injective and we know for all $(\ell, cnt, \rho) \in S$, $(\ell, cnt, \varphi_\ell(\rho)) \in \overline{\mathcal{N}}(S)$, we have that

$$\Pr[\overline{\mathsf{Exec}}(I') = (L', \mathsf{cnt}', \boldsymbol{\rho}') \in \varphi(S)] \leq \Pr[\overline{\mathsf{Exec}}(I') = (L', \mathsf{cnt}', \boldsymbol{\rho}') \in \overline{\mathcal{N}}(S)],$$

and we finish the proof.

□

5.4 CompactBin Algorithm

We now describe the CompactBin algorithm which takes in a bin representation, outputs a compacted array, and writes the true length of the output to some dedicated location on the output tape.

$\underline{\mathsf{CompactBin}^{\epsilon,\delta}\left((\mathsf{Bin}_i^{(Z)} : i \in [B])\right)}$: // *Let* $\epsilon_1 = \frac{\epsilon}{2}$, $\delta_1 = \frac{\delta}{2(1+e^{\epsilon_1})}$.

- Let $s = \Theta(\frac{1}{\epsilon_1} \log^2 B \cdot \log \frac{1}{\delta_1})$, be an upper bound of the additive error of (ϵ_1, δ_1)-differentially private prefix sums on at most B integers.
- Let $R := (R_i : i \in [B])$, where R_i is the number of real elements in Bin_i. Call $\mathsf{cnt} := \mathsf{PrefixSum}^{\epsilon_1,\delta_1}(R)$.
- Let Buf and the output array be initially empty. For $i = 1$ to B:
  - Read the i-th bin and append it to the end of Buf.
  - Perform stable oblivious sorting on Buf such that all real elements are moved to the front.
  - Let L be the current length of the output array. Remove an appropriate number of

elements from the beginning of Buf and append them to the output array, such that the output array has length exactly $\max(\mathsf{cnt}[i] - s, L)$.

– Truncate Buf if necessary such that its length is at most $2s$.

- Append Buf to the end of the output array. Write the true output length $\sum_{i\in[B]} R_i$ to some dedicated location on the output tape.

To gain some intuition, basically in each step i, the CompactBin algorithm reads the next bin i, and tries to copy the real elements in bin i to the end of the output array. To achieve differential obliviousness, the algorithm cannot reveal the true number of real elements inside each bin. Therefore, it calls a differentially private prefix sum mechanism to compute an array $\mathsf{cnt}[1:B]$ where $\mathsf{cnt}[i]$ is an estimate of the number of real elements contained in the first i bins. The prefix sum algorithm guarantees that the estimation error is upper bounded by s. Therefore, at the end of the i-th step, the algorithm should have written exactly $\mathsf{cnt}[i] - s$ number of real elements to the output array. To accomplish this, the algorithm makes use of a temporary working buffer Buf that is used to store the real elements that have been fetched from the input bins but have not been appended to the output array. It guarantees that at the end of each step, there are at most $2s$ real elements leftover in Buf.

Theorem 5.4. *With probability 1, the output of* CompactBin *includes all the real elements from the B input bins with their order preserved and the filler elements in the output array only appear after the last real element.*

Proof. We need to show that no filler element will be output before the last real element and no real element will be truncated. We know fact that for every $i \in [B]$, $|\sum_{j\le i} R_i - \mathsf{cnt}[i]| \le s$ since s is the upper bound on error of the prefix sum mechanism.

Let i^* be the smallest i such that $\mathsf{cnt}[i] - s \ge 0$. If all $\mathsf{cnt}[i] < s$, then let $i^* = B + 1$. First, we observe that during the first $i^* - 1$ iteration, no element will be output. Also, at the end of the iteration i^*-1, Buf contains $\sum_{j\le i^*-1} R_j$ real elements. We know that $\sum_{j\le i^*-1} R_j \le \mathsf{cnt}[i^*-1]+s \le 2s$, so no real elements get truncated.

Starting from the i^* iteration, for each iteration i, after reading the i-th bin into Buf, the algorithm has read $\sum_{j\le i} R_i$ real elements. Then, if we know at the end of the iteration, the output array does not have more than $\sum_{j\le i} R_i$ elements, then all the elements that in the output tape are still real. We simply have that for $i \ge i^*$, the output array will be appended to the length of $\mathsf{cnt}[i] - s$ at the end of the iteration and $\mathsf{cnt}[i] - s \le \sum_{j\le i} R_i$. Also, we know that after the output step, Buf contains exactly $\sum_{j\le i} R_i - (\mathsf{cnt}[i] - s) \le 2s$ real elements. So no real elements will be truncated.

At the end of the algorithm, we append the whole Buf to the output. Since this Buf is sorted, the first filler element will be after the last real element. □

Theorem 5.5. $\mathsf{CompactBin}^{\epsilon,\delta}$ *is (ϵ, δ)-NPDO w.r.t. the input neighboring relation $\sim_B$ and output neighboring relation $\sim_E$.*

Proof. First, due to Theorem 5.4 and the definition of $\sim_B$, we know that CompactBin is strongly-NP. Also, notice that the view of the adversary fully depends on the (ϵ_1, δ_1)-DP prefix sum mechanism result cnt. By the definition of $\sim_B$, we know there are at most two bins that have difference at most one given the neighboring input. Notice that we set $\epsilon_1 = \epsilon/2, \delta_1 = \delta/(2(1 + e^{\epsilon_1}))$. With group privacy theorem, it is straightforward that releasing cnt to the adversary is (ϵ, δ)-DP and thus CompactBin is (ϵ, δ)-DO. Therefore, we have CompactBin is (ϵ, δ)-NPDO. □

Theorem 5.6. CompactBin *has a worst-case runtime of $O(B(Z + s)\log(Z + s))$.*

Proof. We have B iterations. Before each iteration, the Buf has size at most $2s$. When we read the i-th bin into Buf, its size will be at most $2s + Z$. So performing oblivious sorting on Buf takes $O((Z+s)\log(Z+s))$ time. In total, the time complexity will be $O(B(Z+s)\log(Z+s))$. □

From the $\mathsf{RandBin}$ algorithm, $BZ = O(|I|)$, $Z = \Theta(s)$, and $s = O(\frac{1}{\epsilon}\log^2|I|\log\frac{1}{\delta})$. By Theorem 5.2 and Theorem 5.6, the following corollary holds:

Corollary 5.7. *Assuming* $|I| = \Omega(\frac{1}{\epsilon}\log\frac{1}{\delta})$*, then the full compaction algorithm* $\mathsf{Compact}^{\epsilon,\delta} := \mathsf{CompactBin}^{\epsilon/2,\delta/2} \circ \mathsf{RandBin}^{\epsilon/2,\delta/2}$ *has a worst-case runtime of* $O\left(|I|\left(\log\log|I| + \log\frac{1}{\epsilon} + \log\log\frac{1}{\delta}\right)\right)$.

6 Application: Optimal Privacy Amplification in the Differential Oblivious Shuffle Model

We use our composition framework to prove a privacy amplification theorem for the differentially oblivious shuffle (DO-shuffle) model. In particular, consider a distributed setting with n clients and one server. In the so-called local model, each client runs an ϵ_0-locally differentially private (LDP) mechanism on its own private data, and sends the result the server. Earlier works [CSU+19, Che21, GGK+21, GKMP20] showed that if we first rely on a trusted shuffler to shuffle all clients' messages, and reveal only the permuted messages to the server (without revealing the permutation), then, we can significantly amplify the privacy guarantees. Here, amplification means that we can have an (ϵ, δ)-DP guarantee with $\epsilon < \epsilon_0$. Notably, the recent work of Feldman et al. [FMT21] proved optimal parameters for privacy amplification in a perfectly secure shuffle model, that is, it can achieve (ϵ, δ)-DP with any $\delta > 0$ and $\epsilon = O\left((1-e^{-\epsilon_0})e^{\epsilon_0/2}\sqrt{\frac{\log(1/\delta)}{n}}\right)$. In this section, our goal is to show that the *perfectly secure* shuffle in privacy amplification can be replaced with a much weaker, (ϵ, δ)-*differentially oblivious* shuffle, without degrading the amplification guarantees (except for extra ϵ and δ additive factors that arise from the differentially oblivious shuffler itself).

6.1 Definitions

Definition 6.1 (Shuffle protocol)**.** A protocol between a server and n clients each with some input from $\mathcal{X}$ is said to be a shuffle protocol, iff under an *honest execution*, the server outputs a random permutation of the clients' inputs.

We assume that an adversary $\mathcal{A}$ may control up to t clients as well as the server, we define the random variable $\mathsf{View}^{\mathcal{A}}(\mathbf{x}_H)$ to mean the view of the adversary during an execution where the honest clients' inputs are $\mathbf{x}_H \in \mathcal{X}^{n-t}$. The view of the adversary $\mathcal{A}$ should include whatever the adversary can observe during the execution. Specifically, the view include the server's output, all messages sent and received by the corrupted clients and the server. Further, the view may include any additional information the adversary can observe. For example, if the adversary can observe honest-to-honest communication (e.g., a network adversary), then, the view should also include the honest-to-honest communication. For a protocol secure in the *semi-honest* model, we assume that the corrupt players will honestly follow the protocol. For the protocol secure in the *malicious* model, we assume that the corrupt players can send arbitrary messages and the adversary $\mathcal{A}$ controls the messages sent by corrupt players.

Remark 6.2. Different DO-shuffle protocols may provide security guarantees under differing adversarial power. For example, of Gordon et al. [GKLX22] assumes a semi-honest adversary cannot

observe honest-to-honest communication, whereas Bünz et al. [BHMS] assumes a malicious adversary who can observe the entire network communication. Our privacy amplification theorem does not care about the exact modeling choice made by the underlying DO-shuffle protocol, and the composed DO-shuffle-model mechanism essentially inherits the same assumptions as the underlying DO-shuffle.

Neighboring by swapping. Given some set $\mathcal{D}$ and two vectors $\mathbf{y}, \mathbf{y}' \in \mathcal{D}^m$, we say that $\mathbf{y} \sim_S \mathbf{y}'$, iff either $\mathbf{y} = \mathbf{y}'$, or $\mathbf{y}'$ can be obtained from $\mathbf{y}$ by swapping the values of two coordinates.

Definition 6.3 (DO-shuffle)**.** A shuffle protocol is said to satisfy *statistical* (ϵ, δ)-differentially obliviousness in the presence of $t \leq n$ corruptions, iff the following holds: for any adversary $\mathcal{A}$ controlling the server and at most t clients, for any two honest input configurations $\mathbf{y}_H, \mathbf{y}'_H \in \mathcal{Y}^{n-t}$ such that $\mathbf{y}_H \sim_S \mathbf{y}'_H$, for any subset $S \subseteq \mathcal{V}$ where $\mathcal{V}$ denotes the view space, it holds that

$$\Pr\left[\mathsf{View}^{\mathcal{A}}(\mathbf{y}_H) \in S\right] \leq e^{\epsilon} \cdot \Pr\left[\mathsf{View}^{\mathcal{A}}(\mathbf{y}'_H) \in S\right] + \delta$$

Definition 6.4 (Computational DO-shuffle)**.** A shuffle protocol Φ is said to satisfy computational (ϵ, δ)-differentially obliviousness in the presence of $t \leq n$ corruptions, iff for any probabilistic polynomial-time (p.p.t.) adversary $\mathcal{A}$ controlling the server and at most t clients, for any two neighboring honest input configurations $\mathbf{y}_H \sim_S \mathbf{y}'_H$, it holds that

$$\Pr\left[\mathsf{Expt}^{\mathcal{A}}(1^{\lambda}, \mathbf{y}_H) = 1\right] \leq e^{\epsilon} \cdot \Pr\left[\mathsf{Expt}^{\mathcal{A}}(1^{\lambda}, \mathbf{y}'_H) = 1\right] + \delta$$

where $\mathsf{Expt}^{\mathcal{A}}(1^{\lambda}, \mathbf{y})$ is the randomized experiment where we execute the protocol using security parameter λ and interacting with the adversary $\mathcal{A}$, and at the end we output whatever $\mathcal{A}$ outputs.

Definition 6.5 (ϵ_0-LDP mechanism)**.** The function $\mathcal{R} : \mathcal{X} \to \mathcal{Y}$ is an ϵ_0-LDP mechanism if for any $x, x' \in \mathcal{X}$ and any subset $S \subseteq \mathcal{Y}$, $\Pr[\mathcal{R}(x) \in S] \leq e^{\epsilon_0} \Pr[\mathcal{R}(x') \in S]$.

6.2 Privacy Amplification in the DO-Shuffle Model

Since we want to use our composition framework to prove optimal privacy amplification in the DO-shuffle model, we can define the first and second mechanism M_1 and M_2 as follows:

- The first mechanism $\mathsf{M}_1 : \mathcal{X}^n \to \mathcal{Y}^n$ is where the n clients each apply the ϵ_0-LDP mechanism $\mathcal{R} : \mathcal{X} \to \mathcal{Y}$ to their private data, respectively. The mechanism generates no view observable by the adversary, and moreover, its output is the concatenation of all clients' outputs.
- The second mechanism $\mathsf{M}_2 : \mathcal{Y}^n \to \mathcal{Y}^n$ is the DO-shuffler itself. Here, the view of the adversary is its view in the DO-shuffle protocol, and the output is the shuffled outcome. In the main body, we shall first assume that M_2 satisfies *statistical* differential obliviousness (Definition 6.3). Later in Appendix A.1, we will extend our composition framework to support the case when M_2 is a computationally differentially oblivious shuffler (Definition 6.4).

To prove an optimal privacy amplification theorem in the DO-shuffle model, the crux is to show that M_1 satisfies (ϵ, δ)-NPDO for $\epsilon = O\left((1 - e^{-\epsilon_0})e^{\epsilon_0/2}\sqrt{\frac{\log(1/\delta)}{n-t}}\right)$ with any $\delta > 0$ when at most t clients are corrupted, as more formally stated in the following lemma,

Lemma 6.6. *Suppose* $\epsilon_0 \leq \log\left(\frac{n-t}{16\log(2/\delta)}\right)$. *The above mechanism* M_1 *satisfies* (ϵ, δ)*-NPDO w.r.t. the input relation* $\sim_H$*, (i.e., two vectors are neighboring if they have the same length and differ in at most one position) and the output relation* $\sim_S$ *(i.e., neighboring by swapping).*

If we can prove Lemma 6.6, we can directly apply our composition theorem (Theorem 3.6) to get the desired result. Before we embark on proving Lemma 6.6, it is interesting to point out that Lemma 6.6 is a generalization of the following main theorem of Feldman et al. [FMT21], who proved an optimal privacy amplification theorem in a perfectly secure shuffle model.

Theorem 6.7 (Main theorem of Feldman et al. [FMT21])**.** *Let $\mathcal{R} : \mathcal{X} \to \mathcal{Y}$ be an ϵ_0-LDP mechanism to be run by each client over its private input. Let $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{X}^n$ and $\mathbf{x}' = (x'_1, \ldots, x'_n) \in \mathcal{X}^n$ be any two neighboring input configurations that differ in at most one client's input. Let $\mathsf{AllSets}^n(\mathcal{Y})$ be the family of all multi-sets of size n drawn from the space $\mathcal{Y}$. For any $\delta > 0$ that $\epsilon_0 \leq \log\left(\frac{n}{16\log(2/\delta)}\right)$, it must be that for any subset $S \subseteq \mathsf{AllSets}^n(\mathcal{Y})$,*

$$\Pr[\mathsf{MSet}(\mathcal{R}_1(x_1), \ldots, \mathcal{R}_n(x_n)) \in S] \leq e^{\epsilon} \cdot \Pr[\mathsf{MSet}(\mathcal{R}_1(x'_1), \ldots, \mathcal{R}_n(x'_n)) \in S] + \delta,$$

for $\epsilon = O\left((1 - e^{-\epsilon_0})e^{\epsilon_0/2}\sqrt{\frac{\log(1/\delta)}{n}}\right)$. In the above, the notation $\mathsf{MSet}(\cdot)$ converts the input elements into a multi-set representation.

We can view Feldman et al.'s main theorem (Theorem 6.7) as a special case of (ϵ, δ)-NPDO, where the output neighboring relation is set equivalence, i.e., two output vectors $\mathbf{y} \in \mathcal{Y}^n$ and $\mathbf{y}' \in \mathcal{Y}^n$ are considered neighboring iff $\mathsf{MSet}(\mathbf{y}) = \mathsf{MSet}(\mathbf{y}')$. Our Lemma 6.6 can be viewed as a strict generalization of Feldman et al.'s Theorem 6.7, since in our case, the output neighboring relation $\sim_S$ is more stringent than set equivalence. At a more intuitive level, Feldman et al.'s Theorem 6.7 says that secretly permuting the clients amplifies privacy, but whereas our Lemma 6.6 says that we need only the ability to secretly swap two clients without being noticed to amplify privacy, and the degree of amplification is almost as strong as permuting all clients.

6.3 Proof of Lemma 6.6

We provide the proof sketch and defer the proof to Appendix B.1.

We want to prove that the mechanism M_1 as defined above satisfies (ϵ, δ)-NPDO. Instead of directly proving this statement, we first define a related mechanism M'_1 which is otherwise the same as M_1, except that we augment the adversary's view with some extra information. Specifically, recall that the mechanism M_1 does not generate any view observable by the adversary. Now, in M'_1, we assume the adversary can observe some additional auxiliary information which we shall define shortly. We shall prove that M'_1 satisfies (ϵ, δ)-NPDO by proving that M'_1 satisfies (ϵ, δ)-DO and $(0, 0)$-NP. This immediately implies that the original M_1 satisfies (ϵ, δ)-NPDO as well.

Equivalent view of M_1. Without loss of generality, we may assume that client 1 is the client whose input differs in the two neighboring input configurations $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{x}' = (x'_1, x_2, \ldots, x_n)$. Feldman et al. [FMT21] showed that M_1 can be equivalently viewed as the following randomized experiment. Recall that in M_1, every client i runs a ϵ_0-DP local randomizer $\mathcal{R}$ on its private data x_i. An equivalent way to view $\mathcal{R}(x_i)$ where $i \neq 1$ is the following:

$$\mathcal{R}(x_i) = \begin{cases} \mathcal{R}(x_1) & w.p. \frac{1}{2e^{\epsilon_0}}, \\ \mathcal{R}(x'_1) & w.p.\ \frac{1}{2e^{\epsilon_0}}, \\ \mathcal{Q}_{\{x_1, x'_1\}}(x_i), & w.p.\ 1 - \frac{1}{e^{\epsilon_0}}. \end{cases}$$

Using this equivalent view, we can imagine that every other client besides client 1 first flips a random coin to decide whether it wants to be a clone of $\mathcal{R}(x_1)$ or $\mathcal{R}(x'_1)$. If so, it chooses to be

a clone of $\mathcal{R}(x_1)$ or $\mathcal{R}(x_1')$ with equal probability. If not, it will sample from a suitable leftover distribution $\mathcal{Q}_{\{x_1,x_1'\}}(x_i)$. The client 1 is always a clone of itself, i.e., of $\mathcal{R}(x_1)$ in the first world, and of $\mathcal{R}(x_1')$ in the second world.

Definition of M_1'. In M_1, the adversary's view is empty. We define an augmented mechanism that is almost identical to M_1 except that we give the adversary some extra information, i.e., we augment the adversary's view. In particular, we will give the adversary the extra information (U, V, T), where

- V contains the indices of the clones, i.e., the clients (including 1) who clone either $\mathcal{R}(x_1)$ or $\mathcal{R}(x_1')$,
- U is the multi-set of the messages (i.e., outcomes of the local randomizer) sent by the clones in V,
- T contains the messages sent by all the non-clones (i.e., clients not in V) along with their indices.

It suffices to prove that M_1' satisfies (ϵ, δ)-NPDO. To do so, we will prove that M_1' satisfies (ϵ, δ)-DO and $(0, 0)$-NP. It turns out that the fact that M_1' is (ϵ, δ)-DO is already implied by the key lemma of Feldman et al. [FMT21], as stated in Lemma B.1 of of Appendix B.1. Note that Feldman et al.'s main theorem Theorem 6.7 is also a corollary of Lemma B.1. The intuition is that if we can secretly permute all clients' messages, then client 1 is well-hidden among the clones — in particular, observe that even when we directly tell the adversary all the messages sent by non-clones, client 1 is still well-hidden.

Therefore, the crux of our proof is to show that M_1' satisfies $(0, 0)$-NP.

Proving that M_1' satisfies $(0, 0)$-NP. This is the most technical part of our proof. To prove that M_1' satisfies $(0, 0)$-NP, we will be using Lemma 4.1. Specifically, we will consider the probability space conditioned on the adversary's view (U, V, T), and we want to find an $(0, 0)$-matching in a bipartite graph where the vertices on both sides represent a point in the output space of M_1'. One can think of the vertices on the left as producers, and the amount of commodity produced by each producer is exactly the *conditional* probability of this output. The vertices on the right are consumers, and each consumer's maximum capacity is also the *conditional* probability of this output. Now, each producer can only route to neighboring consumers, and we want to make sure that all but $1 - \delta$ amount of the commodity is routed to some consumer. In our proof, it is equivalent to just ignore the term T, and imagine that the output space is merely all permutations of the messages in U. This part of the proof is somewhat more involved, and we defer the actual proof to Appendix B.1.

Acknowledgment

This work is in part supported by a grant from ONR, a gift from Cisco, NSF awards under grant numbers 2128519 and 2044679, and a Packard Fellowship.

References

[Abe99] Masayuki Abe. Mix-networks on permutation networks. In *ASIACRYPT*, 1999.

[ABG+11] Ron Aharoni, Eli Berger, Agelos Georgakopoulos, Amitai Perlstein, and Philipp Sprüssel. The max-flow min-cut theorem for countable networks. *Journal of Combinatorial Theory, Series B*, 101(1):1–17, 2011.

[AFKL19] Peyman Afshani, Casper Benjamin Freksen, Lior Kamma, and Kasper Green Larsen. Lower bounds for multiplication via network coding. In *ICALP*, pages 10:1–10:12, 2019.

[AKTZ17] Nikolaos Alexopoulos, Aggelos Kiayias, Riivo Talviste, and Thomas Zacharias. Mcmix: Anonymous messaging via secure multiparty computation. In *Usenix Security*, 2017.

[ALU18] Megumi Ando, Anna Lysyanskaya, and Eli Upfal. Practical and provably secure onion routing. In *ICALP*, 2018.

[APY20] Ittai Abraham, Benny Pinkas, and Avishay Yanai. Blinder: Mpc based scalable and robust anonymous committed broadcast. In *ACM CCS*, 2020.

[BBG18] Borja Balle, Gilles Barthe, and Marco Gaboardi. Privacy amplification by subsampling: Tight analyses via couplings and divergences. *NeurIPS*, 2018.

[BBGN19] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. In *CRYPTO*, 2019.

[BEM+17] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *SOSP*, 2017.

[BG12] Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *Eurocrypt*, volume 7237, pages 263–280, 2012.

[BHMS] Benedikt Bünz, Yuncong Hu, Shin'ichiro Matsuo, and Elaine Shi. Non-interactive differentially anonymous router. Cryptology ePrint, 2021/1242.

[BKK+21] Dmytro Bogatov, Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. ϵpsolute: Efficiently querying databases while providing differential privacy. In *CCS*, 2021.

[BNZ19] Amos Beimel, Kobbi Nissim, and Mohammad Zaheri. Exploring differential obliviousness. In *APPROX/RANDOM*, 2019.

[CBM15] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *S & P*, 2015.

[CCMS19] T-H. Hubert Chan, Kai-Min Chung, Bruce M. Maggs, and Elaine Shi. Foundations of differentially oblivious algorithms. In *SODA*, 2019.

[CGF10] Henry Corrigan-Gibbs and Bryan Ford. Dissent: Accountable anonymous group messaging. In *CCS*, page 340–350, 2010.

[CGLS18] TH Hubert Chan, Yue Guo, Wei-Kai Lin, and Elaine Shi. Cache-oblivious and data-oblivious sorting and applications. In *SODA*, 2018.

[Cha81] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, February 1981.

[Cha88] David L. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1988.

[Che21] Albert Cheu. Differential privacy in the shuffle model: A survey of separations. *arXiv preprint arXiv:2107.11839*, 2021.

[CL05] Jan Camenisch and Anna Lysyanskaya. A formal treatment of onion routing. In *CRYPTO*, 2005.

[CSS10] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. In *ICALP*, 2010.

[CSS11] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *TISSEC*, 14(3):26, 2011.

[CSU+19] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *EUROCRYPT*, 2019.

[CZSC21] Shumo Chu, Danyang Zhuo, Elaine Shi, and T.-H. Hubert Chan. Differentially oblivious database joins: Overcoming the worst-case curse of fully oblivious algorithms. In *ITC*, 2021.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

[DNPR10] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *STOC*, 2010.

[DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[DS18] Jean Paul Degabriele and Martijn Stam. Untagging tor: A formal treatment of onion encryption. In *EUROCRYPT*, 2018.

[EB] Saba Eskandarian and Dan Boneh. Clarion: Anonymous communication from multiparty shuffling protocols. Cryptology ePrint Archive, Report 2021/1514.

[EFM+19] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *SODA*, 2019.

[FHLS19] Alireza Farhadi, MohammadTaghi Hajiaghayi, Kasper Green Larsen, and Elaine Shi. Lower bounds for external memory integer sorting via network coding. In *STOC*, 2019.

[FMT21] Vitaly Feldman, Audra McMillan, and Kunal Talwar. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. In *FOCS*, 2021.

[GDD+21] Antonious Girgis, Deepesh Data, Suhas Diggavi, Peter Kairouz, and Ananda Theertha Suresh. Shuffled model of differential privacy in federated learning. In *AISTATS*. PMLR, 2021.

[GGK+21] Badih Ghazi, Noah Golowich, Ravi Kumar, Rasmus Pagh, and Ameya Velingker. On the power of multiple anonymous messages: Frequency estimation and selection in the shuffle model of differential privacy. In *EUROCRYPT*, 2021.

[GIKM00] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. *JCSS*, 2000.

[GKLX22] S. Dov Gordon, Jonathan Katz, Mingyu Liang, and Jiayu Xu. Spreading the privacy blanket: - differentially oblivious shuffling for differential privacy. In *ACNS*, 2022.

[GKMP20] Badih Ghazi, Ravi Kumar, Pasin Manurangsi, and Rasmus Pagh. Private counting from anonymous messages: Near-optimal accuracy with vanishing communication overhead. In *ICML*, 2020.

[GO96] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. *J. ACM*, 1996.

[Gol87] O. Goldreich. Towards a theory of software protection and simulation by oblivious RAMs. In *STOC*, 1987.

[GRS99] David Goldschlag, Michael Reed, and Paul Syverson. Onion routing for anonymous and private internet connections. *CACM*, 1999.

[Hal35] P Hall. On representatives of subsets. *Journal of the London Mathematical Society*, 1(1):26–30, 1935.

[HJ48] Marshall Hall Jr. Distinct representatives of subsets. *Bulletin of the American Mathematical Society*, 54(10):922–926, 1948.

[JLN19] Riko Jacob, Kasper Green Larsen, and Jesper Buus Nielsen. Lower bounds for oblivious data structures. In *SODA*, 2019.

[Loc22] Andreas Lochbihler. A mechanized proof of the max-flow min-cut theorem for countable networks with applications to probability theory. *Journal of Automated Reasoning*, pages 1–26, 2022.

[LSX19] Wei-Kai Lin, Elaine Shi, and Tiancheng Xie. Can we overcome the $n \log n$ barrier for oblivious sorting? In *SODA*, 2019.

[MPRV09] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. Computational differential privacy. In *CRYPTO*, pages 126–142. Springer, 2009.

[OS97] Rafail Ostrovsky and Victor Shoup. Private information storage. In *STOC*, 1997.

[Rou20] Tim Roughgarden. *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2020.

[SCSL11] Elaine Shi, T.-H. Hubert Chan, Emil Stefanov, and Mingfei Li. Oblivious RAM with $O((\log N)^3)$ worst-case cost. In *ASIACRYPT*, 2011.

[SW21] Elaine Shi and Ke Wu. Non-interactive anonymous router. In *Eurocrypt*, 2021.

[WBK19] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. Subsampled rényi differential privacy and analytical moments accountant. In *AISTATS*, 2019.

[ZS22] Mingxun Zhou and Elaine Shi. The power of the differentially oblivious shuffle in distributed privacy mechanisms. Cryptology ePrint Archive, Paper 2022/177, 2022.

[ZZZR05] Li Zhuang, Feng Zhou, Ben Y. Zhao, and Antony Rowstron. Cashmere: Resilient anonymous routing. In *NSDI*, 2005.

A Computational Differentially Obliviousness and Composition

In the main body, we focused on *statistical* notions of NPDO and DO. In this section, we discuss how to extend our composition theorem to computational notions of NPDO and DO.

As a first step (Appendix A.1), we consider the case where the second mechanism satisfies a standard, indistinguishability-based computational DO notion, and the first mechanism still satisfies statistical NPDO. We particularly care about this scenario, since in our DO-shuffle-model application, M_1 is the step where the clients each run their LDP algorithm, and indeed M_1 enjoys statistical security. However, known instantiations [ALU18, GKLX22, BHMS] of the DO-shuffler, i.e., M_2, satisfy only computational notions of security.

Next, in Appendix A.2 we will consider the case where both the first and the second mechanisms enjoy computationally security. To do so, we will have to first introduce a computational variant of our NPDO notion.

A.1 When M_2 is Computationally Secure

We first define an indistinguishability-based notion of DO, in the same flavor of the indistinguishability-based, computational DP notion of Mironov et al. [MPRV09].

Definition A.1 (CDO Mechanism)**.** A mechanism Φ is said to satisfy computational (ϵ, δ)-differentially obliviousness iff for any probabilistic polynomial-time (p.p.t.) adversary $\mathcal{A}$ for any two neighboring input configurations $x \sim_{\mathcal{X}} x$, it holds that

$$\Pr\left[\mathsf{Expt}_{\Phi}^{\mathcal{A}}(1^{\lambda}, x) = 1\right] \leq e^{\epsilon} \cdot \Pr\left[\mathsf{Expt}_{\Phi}^{\mathcal{A}}(1^{\lambda}, x') = 1\right] + \delta + \mathsf{negl}(\lambda)$$

where $\mathsf{Expt}_{\Phi}^{\mathcal{A}}(1^{\lambda}, x)$ is the randomized experiment where we execute the mechanism Φ using security parameter λ and interacting with the adversary $\mathcal{A}$, and at the end we output whatever $\mathcal{A}$ outputs.

We extend our composition theorem to work even when the second mechanism M_2 satisfies the above CDO notion.

Corollary A.2 (Composition when M_2 satisfies CDO)**.** *Suppose that an algorithm $\mathsf{M}_1 : \mathcal{X} \to \mathcal{Y}$ satisfies (ϵ_1, δ_1)-NPDO w.r.t. $\sim_{\mathcal{X}}$ and $\sim_{\mathcal{Y}}$, and moreover, the view space $\mathcal{V}$ and output space $\mathcal{Y}$ of M_1 is finite or countably infinite. Suppose that $\mathsf{M}_2 : \mathcal{Y} \to \mathcal{Z}$ satisfies (ϵ_2, δ_2)-CDO w.r.t. $\sim_{\mathcal{Y}}$. Then, the composed mechanism $\mathsf{M}_2 \circ \mathsf{M}_1 : \mathcal{X} \to \mathcal{Z}$ satisfies $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-CDO.*

Proof. Fix any neighboring input x, x'. By Lemma 4.1, there exists an (ϵ_1, δ_1)-matching $w : (\mathcal{V}_1 \times \mathcal{Y}) \times (\mathcal{V}_1 \times \mathcal{Y}) \to [0, 1]$ w.r.t natural neighbor notion $\sim$ in the product space $\mathcal{V}_1 \times \mathcal{Y}$: $(v_1, y) \sim (v_1', y')$ when $v_1 = v_1'$ and $y \sim_{\mathcal{Y}} y'$. We want to prove that, for any p.p.t. adversary $\mathcal{A}$,

$$\Pr\left[\mathsf{Expt}_{\mathsf{M}_2 \circ \mathsf{M}_1}^{\mathcal{A}}(1^{\lambda}, x) = 1\right] \leq e^{\epsilon_1 + \epsilon_2} \cdot \Pr\left[\mathsf{Expt}_{\mathsf{M}_2 \circ \mathsf{M}_1}^{\mathcal{A}}(1^{\lambda}, x') = 1\right] + \delta_1 + \delta_2 + \mathsf{negl}(\lambda).$$

For any p.p.t. adversary $\mathcal{A}$, we denote adversary $\mathcal{A}(v_1)$, whose behavior is the same as the adversary $\mathcal{A}$ when it is given the auxiliary information v_1 (which should be polynomial-length).

We have

$$
\begin{aligned}
&\Pr\left[\mathsf{Expt}^{\mathcal{A}}_{\mathsf{M}_2\circ\mathsf{M}_1}(1^\lambda, x) = 1\right] \\
=& \sum_{(v_1,y)\in\mathcal{V}_1\times\mathcal{Y}} \Pr[\mathsf{Exec}^{\mathsf{M}_1}(x) = (v_1, y)] \cdot \Pr\left[\mathsf{Expt}^{\mathcal{A}(v_1)}_{\mathsf{M}_2}(1^\lambda, y) = 1\right] \\
&\textit{(Use condition (b) and (d) of the matching)} \\
\leq& \sum_{(v_1,y)\in\mathcal{V}_1\times\mathcal{Y}, y'\sim_{\mathcal{Y}} y} w\left((v_1, y), (v_1, y')\right) \cdot \Pr\left[\mathsf{Expt}^{\mathcal{A}(v_1)}_{\mathsf{M}_2}(1^\lambda, y) = 1\right] + \delta_1 \\
&(\mathsf{M}_2 \textit{ is } (\epsilon_2, \delta_2)\textit{-CDO}) \\
\leq& \sum_{(v_1,y)\in\mathcal{V}_1\times\mathcal{Y}, y'\sim_{\mathcal{Y}} y} w\left((v_1, y), (v_1, y')\right) \cdot \left(e^{\epsilon_2} \Pr\left[\mathsf{Expt}^{\mathcal{A}(v_1)}_{\mathsf{M}_2}(1^\lambda, y') = 1\right] + \delta_2 + \mathsf{negl}(\lambda)\right) + \delta_1 \\
&\textit{(Use condition (b) of the matching)} \\
\leq& \sum_{(v_1,y)\in\mathcal{V}_1\times\mathcal{Y}, y'\sim_{\mathcal{Y}} y} w\left((v_1, y), (v_1, y')\right) \cdot \left(e^{\epsilon_2} \Pr\left[\mathsf{Expt}^{\mathcal{A}(v_1)}_{\mathsf{M}_2}(1^\lambda, y') = 1\right]\right) + \mathsf{negl}(\lambda) + \delta_2 + \delta_1 \\
&\textit{(Use condition (c) of the matching)} \\
\leq& \sum_{(v_1,y')\in\mathcal{V}_1\times\mathcal{Y}} e^{\epsilon_1} \Pr[\mathsf{Exec}^{\mathsf{M}_1}(x') = (v_1, y')] \cdot \left(e^{\epsilon_2} \Pr\left[\mathsf{Expt}^{\mathcal{A}(v_1)}_{\mathsf{M}_2}(1^\lambda, y') = 1\right]\right) + \mathsf{negl}(\lambda) + \delta_2 + \delta_1 \\
=& e^{\epsilon_1+\epsilon_2} \Pr\left[\mathsf{Expt}^{\mathcal{A}}_{\mathsf{M}_2\circ\mathsf{M}_1}(1^\lambda, x') = 1\right] + \mathsf{negl}(\lambda) + \delta_2 + \delta_1.
\end{aligned}
$$

□

A.2 When Both M_1 and M_2 Are Computationally Secure

We now consider the case when both M_1 and M_2 are computationally secure. To understand this case, we have to first define a computational variant of our NPDO notion. The most straightforward way is to use a simulation-based notion, i.e., an algorithm satisfies (ϵ, δ)-computational-NPDO iff the view (i.e., access patterns) and the output generated by the algorithm can be simulated by those generated by another algorithm which satisfies (ϵ, δ)-statistical-NPDO. We first define the computationally indistinguishability of mechanisms:

Definition A.3 (Computationally indistinguishable mechanisms)**.** We say two mechanisms M and M' are computationally indistinguishable iff for any p.p.t. adversary $\mathcal{A}$,

$$
\begin{aligned}
\Big| \Pr\left[x \leftarrow \mathcal{A}; (v, y) \leftarrow \mathsf{Exec}^{\mathsf{M}}(1^\lambda, x) : \mathcal{A}(v, y) = 1\right] \\
- \Pr\left[x \leftarrow \mathcal{A}; (v, y) \leftarrow \mathsf{Exec}^{\mathsf{M}'}(1^\lambda, x) : \mathcal{A}(v, y) = 1\right] \Big| \leq \mathsf{negl}(\lambda).
\end{aligned}
$$

Definition A.4 ((ϵ, δ)-computational-NPDO)**.** We say that an algorithm $\mathsf{M}(1^\lambda, \cdot) : \mathcal{X} \to \mathcal{Y}$ with view space $\mathcal{V}$ satisfies (ϵ, δ)-computational-NPDO, iff there exists an (ϵ, δ)-statistical-NPDO algorithm $\mathsf{M}'(1^\lambda, \cdot) : \mathcal{X} \to \mathcal{Y}$ such that M and M' are computationally indistinguishable.

We now extend our composition framework to the case when both M_1 and M_2 satisfy computational notions of security.

Corollary A.5 (Composition when both M_1 and M_2 have computational security)**.** *Suppose that an algorithm $\mathsf{M}_1 : \mathcal{X} \to \mathcal{Y}$ satisfies (ϵ_1, δ_1)-computational-NPDO w.r.t. $\sim_{\mathcal{X}}$ and $\sim_{\mathcal{Y}}$, and moreover, the view space $\mathcal{V}$ and output space $\mathcal{Y}$ of M_1 is finite or countably infinite. The following two statement holds:*

1. *If $\mathsf{M}_2 : \mathcal{Y} \to \mathcal{Z}$ satisfies (ϵ_2, δ_2)-CDO w.r.t. $\sim_{\mathcal{Y}}$, then, the composed mechanism $\mathsf{M}_2 \circ \mathsf{M}_1 : \mathcal{X} \to \mathcal{Z}$ satisfies $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-CDO.*
2. *If M_2 satisfies (ϵ_2, δ_2)-computational-NPDO w.r.t. $\sim_{\mathcal{Y}}$ and $\sim_{\mathcal{Z}}$, then $\mathsf{M}_2 \circ \mathsf{M}_1$ satisfies $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-computation-NPDO.*

Before we prove this corollary, we first provide the following lemma:

Lemma A.6. *Supopse mechanism M is computationally indistinguishable from mechanism M' and M' is (ϵ, δ)-CDO, then M is also (ϵ, δ)-CDO.*

Proof. Since M is computationally indistinguishable from mechanism M', we have that for any p.p.t. $\mathcal{A}$ and any input x,

$$\Pr\left[\mathsf{Expt}^{\mathcal{A}}_{\mathsf{M}}(1^\lambda, x) = 1\right] \leq \Pr\left[\mathsf{Expt}^{\mathcal{A}}_{\mathsf{M}'}(1^\lambda, x) = 1\right] + \mathsf{negl}(\lambda).$$

Thus, for any neighboring input $x \sim_{\mathcal{X}} x'$

$$\begin{aligned}
&\Pr\left[\mathsf{Expt}^{\mathcal{A}}_{\mathsf{M}}(1^\lambda, x) = 1\right] \\
\leq &\Pr\left[\mathsf{Expt}^{\mathcal{A}}_{\mathsf{M}'}(1^\lambda, x) = 1\right] + \mathsf{negl}(\lambda) \\
\leq &e^\epsilon \left(\Pr\left[\mathsf{Expt}^{\mathcal{A}}_{\mathsf{M}'}(1^\lambda, x') = 1\right]\right) + \delta + \mathsf{negl}(\lambda) \\
\leq &e^\epsilon \left(\Pr\left[\mathsf{Expt}^{\mathcal{A}}_{\mathsf{M}}(1^\lambda, x') = 1\right] + \mathsf{negl}(\lambda)\right) + \delta + \mathsf{negl}(\lambda) \\
\leq &e^\epsilon \Pr\left[\mathsf{Expt}^{\mathcal{A}}_{\mathsf{M}}(1^\lambda, x') = 1\right] + \delta + \mathsf{negl}(\lambda).
\end{aligned}$$

□

Proof of Corollary A.5. When M_2 is CDO: By the definition of computational-NPDO, we know that there exists an (ϵ_1, δ_1)-statistical NPDO mechanism M'_1 that is computationally indistinguishable from M_1. We now prove that $\mathsf{M}_2 \circ \mathsf{M}_1$ is computationally indistinguishable from $\mathsf{M}_2 \circ \mathsf{M}'_1$ with a standard argument for "post-processing" (where M_2 is the post-process step). Suppose there is an p.p.t. adversary $\mathcal{A}$ that can distinguish $\mathsf{M}_2 \circ \mathsf{M}_1$ and $\mathsf{M}_2 \circ \mathsf{M}'_1$ with non-negligible advantage. Then, we simply build an p.p.t. adversary $\mathcal{B}$ that takes the challenge input x from $\mathcal{A}$, runs the execution of the first mechanism, $\mathsf{Exec}^{\mathsf{M}_1}(x)$, takes the output of the first mechanism as y and receives an execution from the environment ($\mathsf{Exec}^{\mathsf{M}_2}(1^\lambda, y)$ or $\mathsf{Exec}^{\mathsf{M}'_2}(1^\lambda, y)$, depending on the experiment). It then sends the view in $\mathsf{Exec}^{\mathsf{M}_1}(x)$ and the whole execution of the second mechanism to $\mathcal{A}$ and outputs what $\mathcal{A}$ outputs. $\mathcal{B}$ correctly simulates the input for $\mathcal{A}$, so $\mathcal{B}$ can distinguish M_1 and M'_1 with non-negligible advantage, which is a contradition. Thus, we know that $\mathsf{M}_2 \circ \mathsf{M}_1$ is computationally indistinguishable from $\mathsf{M}_2 \circ \mathsf{M}'_1$, which is an $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-CDO mechanism by Corollary A.2. Finally, by Lemma A.6, $\mathsf{M}_2 \circ \mathsf{M}_1$ is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-CDO.

When M_2 is computatiaon-NPDO: By the definition of computational-NPDO, we know that there exists an (ϵ_1, δ_1)-statistical NPDO mechanism M'_1 that is computationally indistinguishable from M_1. Also, there exists an (ϵ_2, δ_2)-statistical NPDO mechanism M'_2 that is computationally

indistinguishable from M_2. We only need to show that $\mathsf{M}_2 \circ \mathsf{M}_1$ is computationally indistinguishable from $\mathsf{M}_2' \circ \mathsf{M}_1'$ and then by our main composition theorem for statistical NPDO mechanism 3.6, we can prove the statement.

Let a hybrid mechanism be $\mathsf{M}_2' \circ \mathsf{M}_1$. By the similar argument as the first case in this proof, we have that $\mathsf{M}_2' \circ \mathsf{M}_1$ is computatiaonlly indistinguishable from $\mathsf{M}_2' \circ \mathsf{M}_1'$. We only need to prove that $\mathsf{M}_2' \circ \mathsf{M}_1$ is computationally indistinguishable from $\mathsf{M}_2 \circ \mathsf{M}_1$. Suppose there is a p.p.t. $\mathcal{A}$ that distinguishes $\mathsf{M}_2' \circ \mathsf{M}_1$ and $\mathsf{M}_2 \circ \mathsf{M}_1$. We can build a p.p.t. $\mathcal{B}$ that distinguishes M_2' and M_2 with non-negligible advantage: $\mathcal{B}$ takes the challenge input x from $\mathcal{A}$, samples an execution from M_1 and then submit the output of M_1 as the challenge input. When $\mathcal{B}$ gets the view and the output from M_2 or M_2', it sends the view, the output and also the view of $\mathsf{Exec}^{\mathsf{M}_1}(x)$ to $\mathcal{A}$. Then $\mathcal{B}$ outputs what $\mathcal{A}$ outputs. Since $\mathcal{B}$ correctly simulates the input distribution for $\mathcal{A}$, $\mathcal{B}$ can distinguish M_2' and M_2 with non-negligible advantage, which is a contradiction. □

B Deferred Proofs

B.1 Proofs for the privacy amplification theorem by DO-shuffle

In Section 6, we mentioned that the it is suffice to prove the mechanism M_1' that provides the adversary with auxiliary information is (ϵ, δ)-NPDO w.r.t the input neighboring relation of $\sim_H$ (neighboring by Hamming distance at most 1) and output neighboring relation of $\sim_S$ (neighboring by swapping two messages).

Equivalent view of M_1. Without loss of generality, fix a pair of neighboring input configurations $\mathbf{x} = (x_1, x_2 \ldots, x_n)$ and $\mathbf{x}' = (x_1', x_2 \ldots, x_n)$ that only differ in the first client's input. Feldman et al. [FMT21] defined the following equivalent view of M_1. For client 1, its message are generated as following: for $\mathbf{x}$, the message y_1 will be sampled from $\mathcal{R}(x_1)$; for $\mathbf{x}'$, y_1 is sampled from $\mathcal{R}(x_1')$. For each client i from 2 to n, it behaves the same given the input being $\mathbf{x}$ or $\mathbf{x}'$. That is, y_i is sampled from $\mathcal{R}(x_i)$. Since $\mathcal{R}$ is an ϵ_0-DP mechanism, for any $x, x' \in \mathcal{X}$ where $\mathcal{X}$ is the input domain and any $y \in \mathcal{Y}$ where $\mathcal{Y}$ is the randomizer's output domain, we have $\frac{\Pr[\mathcal{R}(x)=y]}{\Pr[\mathcal{R}(x')=y]} \geq e^{-\epsilon_0}$. Thus, we can "decompose" the distribution of $\mathcal{R}(x')$ as $\mathcal{R}(x') = e^{-\epsilon_0}\mathcal{R}(x) + (1 - e^{-\epsilon_0})\mathcal{Q}_x(x')$. It means that the distribution of $\mathcal{R}(x')$ can be seen as a mixture of $\mathcal{R}(x)$ and some appropriate leftover distribution $\mathcal{Q}_x(x')$. Similarly, for all x_i where $i = 2, \ldots, n$, we can decompose the output distribution of $\mathcal{R}(x_i)$ as

$$\mathcal{R}(x_i) = \begin{cases} \mathcal{R}(x_1) & w.p. \frac{1}{2e^{\epsilon_0}}, \\ \mathcal{R}(x_1') & w.p.\ \frac{1}{2e^{\epsilon_0}}, \\ \mathcal{Q}_{\{x_1, x_1'\}}(x_i), & w.p.\ 1 - \frac{1}{e^{\epsilon_0}}. \end{cases}$$

Here, $\mathcal{Q}_{\{x_1,x_1'\}}(x_i)$ is also some appropriate left-over distribution after the decomposition. The properties of $\mathcal{Q}_{\{x_1,x_1'\}}(x_i)$ are irrelevant in this proof, so we only need to know $\mathcal{Q}_{\{x_1,x_1'\}}(x_i)$ is a legal and well-defined distribution. In other words, with probability $1/e^{\epsilon_0}$, each client from 2 to n will clone either $\mathcal{R}(x_1)$ or $\mathcal{R}(x_1')$ with equal probability, and with the remaining probability, it will sample from the leftover distribution $\mathcal{Q}_{\{x_1,x_1'\}}(x_i)$.

Definition of M_1'. Recall that in M_1, the adversary's view is empty. We will define an augmented algorithm M_1' that is almost identical as M_1 except that we add some extra information to the adversary's view. We define the view of the mechanism M_1' to contain the random varaibles (U, V, T). Here, V is the set of the indices of those clients that sample their messages from $\mathcal{R}(x_1)$ or $\mathcal{R}(x_1')$.

V always contains index 1 and it may contain some indices from 2 to n, depending on those clients' decision on sampling from the corresponding decomposed distribution. U is the multi-set of the messages sent by those clients in V. T contains the messages sent by all the non-clones (i.e., clients not in V) along with their indices.

Later in our proof, we will consider a probability space conditioned on the adversary's view (U, V, T). Thus, it may be helpful to think of M_1' as first sampling the variables (U, V, T), and then sampling the output $(y_1, \ldots, y_n)$ accordingly. Specifically, consider the following equivalent way of sampling the experiment M_1'.

- First, decide out of the $n-1$ clients, how many clients sample from $\mathcal{R}(x_1)$ or $\mathcal{R}(x_1')$. We sample this quantity, denoted as c, from $\mathsf{Bin}(n-1, e^{-\epsilon_0})$.
- Then, conditioned on c clients sampling from $\mathcal{R}(x_1)$ or $\mathcal{R}(x_1')$, we sample the number of the clients cloning $\mathcal{R}(x_1)$, denoted as H, from $\mathsf{Bin}(c, 1/2)$. Let $H' = c - H$ be the number of clients cloning $\mathcal{R}(x_1')$.
- We now denote (m, m') be the total numbers of clients (including client 1) that are sampling from $\mathcal{R}(x_1)$ and $\mathcal{R}(x_1')$, repectively. When the input is $\mathbf{x}$, $(m, m') = (H+1, H')$. When the input is $\mathbf{x}'$, $(m, m') = (H, H'+1)$.
- Generate U, V as following: Sample m messages independently from $\mathcal{R}(x_1)$ and sample m' messages independently from $\mathcal{R}(x_1')$. Let U be the multi-set of these $m+m'$ messages. Let V be a subset of the indicies of $m+m'$ clients picked uniformly at random from $[n]$ that must include client 1.
- Generate T as following: for each $i \notin V$, sample a message y_i from $\mathcal{Q}_{\{x_1, x_1'\}}(x_i)$ and add the index-message pair (i, y_i) to T.
- Finally, generate the output (each client's message) in the following way.
  1. If the input is $\mathbf{x}$, sample a subset of m clients from V uniformly at random conditioned on containing client 1. If the input is $\mathbf{x}'$, the condition will be changed to exclude client 1.
  2. The clients in this set will be assigned messages from the m messages that are sampled earlier from $\mathcal{R}(x_1)$ (for instance, in a uniformly random order), and the remaining clients in V will be assigned messages from the remaining m' messages sampled from $\mathcal{R}(x_1')$.
  3. For a client $i \notin V$, its message is already determined by T.

A key lemma from Feldman et al. [FMT21] shows that releasing (m, m'), i.e.,the number of clients (including client 1) sampling from $\mathcal{R}(x_1)$ and $\mathcal{R}(x_1')$, is (ϵ, δ) differentially private, where ϵ is much smaller than ϵ_0. Formally, they prove the following result.

Lemma B.1 ([FMT21]). *Let $c \xleftarrow{\$} \mathsf{Bin}(n-1, e^{-\epsilon_0})$, $H \xleftarrow{\$} \mathsf{Bin}(c, 1/2)$ and $H' = c - H$. For any $\delta > 0$ and $\epsilon_0 \le \log\left(\frac{n}{16\log(2/\delta)}\right)$, the joint distributions on $(H+1, H')$ and $(H, H'+1)$ are (ϵ, δ)-close where $\epsilon = O\left(\frac{(1-e^{-\epsilon_0})e^{\epsilon_0/2}\sqrt{\log(1/\delta)}}{\sqrt{n}}\right)$.*

Here, two distributions $\mathcal{D}$ and $\mathcal{D}'$ are (ϵ, δ)-close means that for any subset S in the sample space, $\Pr[x \xleftarrow{\$} \mathcal{D} : x \in S] \le e^{\epsilon} \Pr[x' \xleftarrow{\$} \mathcal{D}' : x' \in S] + \delta$ and vice versa.

Notice that conditioned on (m, m'), whose distribution is (ϵ, δ)-close, the generation of the view (U, V, T) is the same regardless of the input being $\mathbf{x}$ or $\mathbf{x}'$. Thus, by post-processing theorem and Lemma B.1, we can easily show that M_1' is (ϵ, δ)-DO. Also, suppose there are t clients that are

corrupted by the adversary. Notice that all honest clients run $\mathcal{R}()$ locally. Then, we can simply just consider the honest client set, which has size $n-t$ and the aforementioned argument still holds, as long as we plug $n-t$ into Lemma B.1. We have:

Corollary B.2. *For any $\delta > 0$ and $\epsilon_0 \leq \log\left(\frac{n-t}{16\log(2/\delta)}\right)$, M_1' is (ϵ, δ)-DO with $\epsilon = O\left(\frac{(1-e^{-\epsilon_0})e^{\epsilon_0/2}\sqrt{\log(1/\delta)}}{\sqrt{n-t}}\right)$ when at most t clients are corrupted by the adversary.*

Now, we only need to prove the following key lemma and the proof is finished. In this proof, we will use techniques from Gordon et al. [GKLX22] in a non-black box way.

Lemma B.3. *M_1' is $(0,0)$-NP.*

Proof. Given any view of the adversary (U, V, T) that has non-zero probability when the input is $\mathbf{x}$ or $\mathbf{x}'$, it suffices to give a $(0,0)$-matching on the output space of M_1' conditioned on (U, V, T) as in Lemma 4.1.

If client i is sampling from $\mathcal{R}(x_1)$ or $\mathcal{R}(x_1')$, we call that client i is *cloning* $\mathcal{R}(x_1)$ or $\mathcal{R}(x_1')$. Recall that V is the subset of clients (including client 1) that clone either $\mathcal{R}(x_1)$ or $\mathcal{R}(x_1')$, and U is the multi-set of the messages they generate.

Without loss of generality, we assume that all clients in V are honest; otherwise, we remove dishonest clients from V (and also their messages from U). Denote $\ell = |V|$ and we can rename the clients such that $V = [\ell]$ without loss of generality. For clients $i \notin V$, their messages in the output are already determined by T. For the rest of the proof, since we will always conditioned on T, we will focus on the part of the output space $\mathcal{Y}^\ell$ corresponding to the clients in V.

To recap the setup, we have fixed some neighboring inputs $\mathbf{x}$ and $\mathbf{x}'$ and conditioned on some subset $V = [\ell]$ of clients that clone either $\mathcal{R}(x_1)$ or $\mathcal{R}(x_1')$, and their multi-set U of output messages (such that the view (U, V) has non-zero probability in both executions). We use SEQ_U to denote the multi-set of the $\ell!$ permutations of messages in U, where each permutation can be interpreted as an output in $\mathcal{Y}^\ell$. Since U itself is a multi-set, then so is SEQ_U; for the purpose of distinguishing repeated elements in U, we can artificially attach labels on them which will help to resolve any ambiguity. Observe that conditioning on (U, V), only outputs in SEQ_U can have non-zero probabilities in either executions. So we only need to build the mappings $w(\mathbf{y}, \mathbf{y}')$ for $\mathbf{y}, \mathbf{y}' \in \mathsf{SEQ}_U$.

Conditional Probability Calculations. Before we build the mapping, we need to figure out the conditional probabilities of the outputs. For each $\mathbf{y} \in \mathsf{SEQ}_U$, we will calculate $\Pr[\mathsf{M}_1'(\mathbf{x}) = \mathbf{y} | \mathsf{View}^{\mathsf{M}_1'}(\mathbf{x}) = (U, V, T)]$. For each $\mathbf{y}' \in \mathsf{SEQ}_U$, we will calculate $\Pr[\mathsf{M}_1'(\mathbf{x}') = \mathbf{y}' | \mathsf{View}^{\mathsf{M}_1'}(\mathbf{x}') = (U, V, T)]$. For simpilicity, we use $\Pr_{\mathbf{x}}[\cdot]$ to denote probabilities associated with the execution given input $\mathbf{x}$, and $\Pr_{\mathbf{x}'}[\cdot]$ to denote those associated given input $\mathbf{x}'$. So we write the previous conditional probabilities as $\Pr_{\mathbf{x}}[\mathbf{y} \mid U, V, T], \Pr_{\mathbf{x}'}[\mathbf{y}' \mid U, V, T]$ for short.

We use μ and μ' to denote the distribution for $\mathcal{R}(x_1)$ and $\mathcal{R}(x_1')$, respectively. Also, we define $\mu(\cdot)$ and $\mu'(\cdot)$ as their probability density functions. For $i \in V$ such that $i \neq 1$, client i will clone either $\mathcal{R}(x_1)$ or $\mathcal{R}(x_1')$ with $\frac{1}{2}$ probability (conditioned on only V); in other words, it will generate a message sampled from the mixture distribution $\omega = \frac{1}{2}\mu + \frac{1}{2}\mu'$. We also denote $\omega(\cdot)$ as its probability density function.

We first compute the probabilities corresponding to the execution on $\mathbf{x}$. Conditioned on V, for each $\mathbf{y} \in \mathsf{SEQ}_U$, we have

$$\Pr_{\mathbf{x}}[\mathbf{y} \mid V, T] = \mu(y_1)\omega(y_2)\cdots\omega(y_l) = \frac{\mu(y_1)}{\omega(y_1)} \prod_{u \in U} \omega(u).$$

By enumerating all $l!$ sequences in SEQ_U, we have $\Pr_{\mathbf{x}}[U \mid V, T] = \sum_{\mathbf{y} \in \mathsf{SEQ}_U} \Pr_{\mathbf{x}}[\mathbf{y} \mid V, T]$.

Observe that the above probability is the sum of $\ell!$ terms, each of which corresponds to a sequence in SEQ_U. Hence, for each $\mathbf{y} \in \mathsf{SEQ}_U$,

$$\Pr_{\mathbf{x}}[\mathbf{y} \mid (U,V,T)] = \frac{\Pr_{\mathbf{x}}[\mathbf{y} \mid V,T]}{\Pr_{\mathbf{x}}[U \mid V,T]} = \frac{\frac{\mu(y_1)}{\omega(y_1)} \prod_{u \in U} \omega(u)}{\sum_{\mathbf{z} \in \mathsf{SEQ}_U} \frac{\mu(z_1)}{\omega(z_1)} \prod_{u \in U} \omega(u)} = \frac{1}{(\ell-1)!} \frac{\mu(y_1)/\omega(y_1)}{\sum_{z \in U} \mu(z)/\omega(z)},$$

where the last equality holds, because we can group the $\ell!$ possible sequences in SEQ_U according to the message sent from client 1, i.e., z_1. Hence, there are ℓ groups and each group has $(\ell-1)!$ sequences.

By the same calculation, for the execution on $\mathbf{x}'$, we have:

$$\Pr_{\mathbf{x}'}[\mathbf{y} \mid (U,V,T)] = \frac{1}{(\ell-1)!} \frac{\mu'(y_1)/\omega(y_1)}{\sum_{z \in U} \mu'(z)/\omega(z)}.$$

Building the $(0,0)$*-matching.* We are now ready to build the $(0,0)$-matching $w : \mathcal{Y}^\ell \times \mathcal{Y}^\ell \to [0,1]$. Notice that we only need to assign non-zero value to $w(\mathbf{y}, \mathbf{y}')$ where $\mathbf{y}, \mathbf{y}' \in \mathsf{SEQ}_U$, because only those sequences will have non-zero probability in the execution conditioned on (U,V,T).

Given a vector $\mathbf{y} \in \mathcal{Y}^\ell$ and some index $k \in [\ell]$, we use $\mathbf{y}_{1 \leftrightarrow k}$ as the sequence produced by interchanging coordinates between the 1st and the kth positions. For each $\mathbf{y} \in \mathsf{SEQ}_U$, it has ℓ neighbors: $\mathbf{y}_{1 \leftrightarrow 1}, \mathbf{y}_{1 \leftrightarrow 2}, \ldots, \mathbf{y}_{1 \leftrightarrow \ell}$. We let

$$w(\mathbf{y}, \mathbf{y}_{1 \leftrightarrow k}) = \frac{1}{(\ell-1)!} \frac{\mu(y_1)/\omega(y_1)}{\sum_{z \in U} \mu(z)/\omega(z)} \times \frac{\mu'(y_k)/\omega(y_k)}{\sum_{z \in U} \mu'(z)/\omega(z)}$$

It's easy to check that this matching satisfies all the conditions in Lemma 4.1:

(a) We know that $\mathbf{y} \sim \mathbf{y}_{1 \leftrightarrow k}$, so $w(\mathbf{y}, \mathbf{y}')$ will only be non-zero if $\mathbf{y} \sim \mathbf{y}'$.

(b) For each $\mathbf{y} \in \mathsf{SEQ}_u$, we have that

$$\sum_{k \in [\ell]} w(\mathbf{y}, \mathbf{y}_{1 \leftrightarrow k}) = \frac{1}{(\ell-1)!} \frac{\mu(y_1)/\omega(y_1)}{\sum_{z \in U} \mu(z)/\omega(z)} \frac{\sum_{k \in [\ell]} \mu'(y_k)/\omega(y_k)}{\sum_{z \in U} \mu'(z)/\omega(z)} = \frac{1}{(\ell-1)!} \frac{\mu(y_1)/\omega(y_1)}{\sum_{z \in U} \mu(z)/\omega(z)}.$$

The last equality holds because $y_1, \ldots, y_\ell$ is a permutation of U.

So we get that $\sum_{k \in [\ell]} w(\mathbf{y}, \mathbf{y}_{1 \leftrightarrow k}) = \Pr_{\mathbf{x}}[\mathbf{y} \mid (U,V,T)]$.

(c) For all $\mathbf{y}' = (y'_1, \ldots, y'_\ell) \in \mathsf{SEQ}_U$, it also has ℓ neighbors $\mathbf{y}'_{1 \leftrightarrow 1}, \mathbf{y}'_{1 \leftrightarrow 2}, \ldots, \mathbf{y}'_{1 \leftrightarrow \ell}$ and only those neighbors will satisfy $w(\mathbf{y}'_{1 \leftrightarrow k}, \mathbf{y}') > 0$. We simply check that

$$\sum_{k \in [\ell]} w(\mathbf{y}'_{1 \leftrightarrow k}, \mathbf{y}') = \frac{1}{(\ell-1)!} \frac{\mu'(y'_1)/\omega(y'_1)}{\sum_{z \in U} \mu'(z)/\omega(z)} \frac{\sum_{k \in [\ell]} \mu(y'_k)/\omega(y'_k)}{\sum_{z \in U} \mu(z)/\omega(z)} = \frac{1}{(\ell-1)!} \frac{\mu'(y'_1)/\omega(y'_1)}{\sum_{z \in U} \mu'(z)/\omega(z)}.$$

The last equality holds because $y'_1, \ldots, y'_\ell$ is a permutation of U.

So we get that $\sum_{k \in [\ell]} w(\mathbf{y}'_{1 \leftrightarrow k}, \mathbf{y}') = \Pr_{\mathbf{x}'}[\mathbf{y}' \mid (U,V,T)]$.

(d) We simply have that

$$\sum_{\mathbf{y}, \mathbf{y}' \in \mathsf{SEQ}_U} w(\mathbf{y}, \mathbf{y}') = \sum_{\mathbf{y} \in \mathsf{SEQ}_U} \sum_{k \in [\ell]} w(\mathbf{y}, \mathbf{y}_{1 \leftrightarrow k}) = \sum_{\mathbf{y} \in \mathsf{SEQ}_U} \Pr_{\mathbf{x}}[\mathbf{y} \mid (U,V,T)] = 1.$$

All conditions hold for the $(0,0)$-matching, so we know that M'_1 is $(0,0)$-NP.

□

C Discussion: Generalizing to Uncountable Sample Spaces

In this section, we describe how Lemma 4.1 may be generalized to uncountable sample spaces. We show how Lemma 4.1 can be derived from a couple of conjectures that can be viewed as uncountable variants of some well-known primal-dual results on bipartite matching and vertex cover. Proving these conjectures is an interesting open question.

We first introduce generalized bipartite matching on measure spaces. Note that we use notation from measure theory that can capture both discrete and uncountable spaces.

Definition C.1 (Generalized Bipartite Matching)**.** Consider the following bipartite matching problem.
Bipartite graph. Suppose (Ω_1, μ_1) and (Ω_2, μ_2) are measure spaces, each of which represents a vertex set in a bipartite graph. Suppose $\mathcal{E} \subseteq \Omega_1 \times \Omega_2$ is a (measurable) subset representing the edges of the bipartite graph, where for each $x \in \Omega_1$, $\mathcal{E}(x) := \{y \in \Omega_2 : (x, y) \in \mathcal{E}\}$ is a measurable subset denoting the *neighbors* of x in Ω_2, and $\mathcal{E}(y) \subseteq \Omega_1$ is defined analogously for $y \in \Omega_2$.
Vertex capacities. Let $p : \Omega_1 \to \mathbb{R}_+$ and $q : \Omega_2 \to \mathbb{R}_+$ be measurable functions denoting vertex capacities. We consider the special case that $\int_{x\in\Omega_1} p(x)d\mu_1(x) < +\infty$ and $\int_{y\in\Omega_2} q(y)d\mu_2(y) < +\infty$.
Feasible matching. A feasible matching is a measurable function $w : \mathcal{E} \to \mathbb{R}_+$ satisfying the vertex capacity constraints:

- For all $x \in \Omega_1$, $\int_{y\in\mathcal{E}(x)} w(x, y)d\mu_2(y) \le p(x)$.
- For all $y \in \Omega_2$, $\int_{x\in\mathcal{E}(y)} w(x, y)d\mu_1(x) \le q(y)$.

Maximum matching. The goal is to "find" a feasible matching w that maximizes the objective function:
$\int_{(x,y)\in\mathcal{E}} w(x, y)d\mu_1(x)d\mu_2(y)$.

Interpreting Lemma 4.1 with Definition C.1. Definition C.1 is related to the second statement in Lemma 4.1. The edges in $\mathcal{E}$ corresponds to neighboring elements in the relation $\sim$. For the discrete spaces, the capacity for a vertex a in the first space is $p(a) = \Pr[A = a]$, and the capacity for b in the second space is $q(b) = e^\epsilon \cdot \Pr[B = b]$. Statement 2(d) means that there is a feasible matching w with objective value at least $1 - \delta$.

As in the finite case, the dual problem for Definition C.1 is the generalized vertex cover problem.

Definition C.2 (Generalized Vertex Cover Problem)**.** Consider the bipartite graph in Definition C.1 between measure space (Ω_1, μ_1) and (Ω_2, μ_2) with edges in $\mathcal{E} \subseteq \Omega_1 \times \Omega_2$.
Vertex costs. The measurable functions $p : \Omega_1 \to \mathbb{R}_+$ and $q : \Omega_2 : \Omega_2 \to \mathbb{R}_+$ are interpreted as vertex costs.
Feasible vertex cover. A feasible vertex cover consists of a pair of measurable functions $\alpha : \Omega_1 \to [0, 1]$ and $\beta : \Omega_2 \to [0, 1]$ such that for all $(x, y) \in \mathcal{E}$, $\alpha(x) + \beta(y) \ge 1$.
Minimum vertex cover. The goal is to "find" a feasible vertex cover (α, β) that minimizes the cost objective:
$\int_{x\in\Omega_1} \alpha(x)p(x)d\mu_1(x) + \int_{y\in\Omega_2} \beta(y)q(y)d\mu_2(y)$.

One can easily check that weak duality holds for the problems in Definitions C.1 and C.2. Observe that $w \equiv 0$ is feasible for the primal problem and $\alpha \equiv \beta \equiv 1$ is feasible for the dual problem. The first conjecture is that strong duality also holds.

Conjecture C.3 (Strong Duality)**.** *There exist primal and dual feasible solutions in Definitions C.1 and C.2 with the same objective value, and hence, each is an optimum solution in its corresponding problem.*

Ease of proof. We actually believe that Conjecture C.3 is more like a claim, because strong duality for finite LP can be proved via Farkas' Lemma, which can be derived from the hyperplane separation theorem. Hence, it should be a fairly straightforward exercise to apply the corresponding Hahn-Banach separation theorem to general spaces, where one might need to add appropriate assumptions on the measure spaces (Ω_1, μ_1) and (Ω_2, μ_2).

Conjecture C.4 (Integrality Gap for Vertex Cover)**.** *For the vertex cover problem in Definition C.2, there exists an integral feasible solution $\alpha : \Omega_1 \to \{0, 1\}$ and $\beta : \Omega_2 \to \{0, 1\}$ that attains the minimum.*

Ease of proof. If $\mathcal{E}$ is finite, then this is a well-known result by considering the vertex cover polytope for bipartite graphs. However, we are not aware of any result in the literature that can readily extend the result to general uncountable bipartite graphs.

We next show that Lemma 4.1 is an easy corollary from Conjectures C.3 and C.4.

Deriving Lemma 4.1. We consider the more difficult direction of statement 1 implies statement 2. Proceeding with proof by contradiction, we assume that the primal objective of the maximum matching is strictly less than $1 - \delta$. By strong duality from Conjecture C.3, the exists a feasible dual solution whose objective is also strictly less than $1 - \delta$.

By Conjecture C.4, we may assume that the dual solution (α, β) is integral. Define $S := \{x \in \Omega_1 : \alpha(x) = 0\}$ and $T := \{y \in \Omega_2 : \beta(x) = 1\}$. Since (α, β) is a feasible vertex cover, it follows that all the neighbors in S are contained in T.

Recall that the cost functions satisfy $p(a) = \Pr[A = a]$ and $q(b) = e^{\epsilon} \Pr[B = b]$ (which are interpreted as probability density functions in the continuous case). Hence, the objective value of (α, β) is:

$\Pr[A \notin S] + e^{\epsilon} \cdot \Pr[B \in T] \geq 1 - \Pr[A \in S] + e^{\epsilon} \cdot \Pr[B \in \mathcal{N}(S)]$.

Finally, since the value of (α, β) is also strictly less than $1 - \delta$, we immediately have

$\Pr[A \in S] > e^{\epsilon} \cdot \Pr[B \in \mathcal{N}(S)] + \delta$, obtaining the required contradition.