A Theory of Composition for Differential Obliviousness

Mingxun Zhou[1], Elaine Shi[1], T-H. Hubert Chan[2], and Shir Maimon[3] *

[1] Carnegie Mellon University
[2] The University of Hong Kong
[3] Cornell University

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). 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.

1 Introduction

It is well-known that access patterns to even encrypted or secret-shared data can leak sensitive information [15,46,47,50,52,60]. Initiated by Goldreich and Ostrovsky [42,43], *oblivious* algorithms is a line of work that aims to provably obfuscate

* Randomized order. This paper subsumes part of the results in an unpublished manuscript [73] written by a subset of the authors. Full version of this paper: [74].

a program's access patterns without incurring too much slowdown. In particular, *obliviousness* (also referred to as *full obliviousness*) requires that a program's access patterns be indistinguishable for any two inputs. It is well-known that oblivious algorithms have broad applications, including in multi-party computation [44, 55], secure processors [54, 57, 59, 63, 69], secure outsourcing [67, 72], databases [6, 28, 35], blockchains [16], and so on. In the past decade, our community have significantly improved the efficiency of oblivious algorithms [65, 68, 70], leading to large-scale real-world adoption such as Signal's private contact discovery [25]. However, as we discuss below, in some applications, the overhead of full obliviousness may still be unacceptable.

Differential Obliviousness (DO), defined by Chan, Chung, Maggs, and Shi [17], is relaxed notion of access pattern privacy. DO requires that the program's access patterns satisfy only differential privacy (DP) [30], as opposed to a simulation-based notion like in full obliviousness [42, 43, 65]. Recent works [10, 12, 17, 24, 45] explored DO and showed how DO can allow us to circumvent fundamental performance barriers pertaining to full obliviousness:

- Chan et al. [17] 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 [3, 36, 51, 53] (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 [24], 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 [24], 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 [64]. In such cases, prior works showed that DO can achieve polynomial speedup over any fully oblivious algorithm [17, 24]!

Basic 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. 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 [17] 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[4]:

```
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. [17] to realize each `Select` operator. In stable 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. [17]'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. [17]. The standard DO notion requires the following.

Definition 1.1 (Basic differential obliviousness [17]). *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 basic DO notion (of M_1) does *not* guarantee

[4] Here we write the two `Select` statements in a single query for convenience, In practice, 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.

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 basic 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. [17]. 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. The outputs have Hamming distance more than 1. 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 basic DO by Chan et al. [17]. 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 basic 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 basic-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 basic 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 [48, 49]. 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 of the full version [74], 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. 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 of the full version [74], 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. [11] in an empirical work, has become a popular model for implementing distributed differentially private mechanisms. The model assumes 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 that the server can only see the union of all messages, without knowing the source of an individual message. Numerous earlier works [8, 22, 23, 39, 40] 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 [11] or using cryptographic protocols [1, 2, 9, 14, 20, 21, 26, 27, 29, 34, 38, 61, 62, 66, 75]. This makes the shuffle model a compelling approach not just in theory, but also in practical applications such as federated learning [41].

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 [8, 23, 33] focused on proving privacy amplification theorems for the shuffle model, culminating in the recent work by Feldman et al. [37], 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 the shuffle model can be expressed with our DO composition framework. Consider a composed mechanism $\mathsf{S} \circ \mathsf{M}_1$. $\mathsf{M}_1 : \mathcal{X}^n \to \mathcal{Y}^n$ is a 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 a shuffling mechanism that takes a message sequence $(y_1, \ldots, y_n)$ and outputs a random permutation of the sequence. All computation in M_1 are done by the clients locally, so we define $\mathsf{View}^{\mathsf{M}_1} := \emptyset$. We define the view in S as exactly its output: the random permutation. 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 by $\mathsf{S} \circ \mathsf{M}_1$ being (ϵ, δ)-DO w.r.t the input neighboring notion $\sim_{\mathcal{X}}$ where $\mathbf{x} \sim_{\mathcal{X}} \mathbf{x}'$ iff the Hamming distance is at most 1.

Can we replace the shuffler with a DO-shuffler? A couple very recent works [5, 13, 45] 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. [45] and Bünz et al. [13] 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 questions:

- *If we were to replace the shuffler in shuffle-model differentially private 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. [37]?*

The pioneering work of Gordon et al. [45] was the first to explore how to use a DO shuffler to design distributed differentially private mechanisms. Gordon et al. [45] 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. [8].

Our results. We prove a privacy amplification theorem for *any LDP mechanism* that achieves *optimal* parameters, tightly matching Feldman et al. [37]'s privacy amplification parameters for the shuffle model. This result improves work of Gordon et al. [45] 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 M_1 is (ϵ, δ)-NPDO w.r.t the output neighboring notion being exactly the DO-shuffler's input neighboring notion. Then, when 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* $\epsilon_0 \leq \log\left(\frac{n}{16\log(2/\delta)}\right)$. *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 [5, 45], then the composed protocol is also secure in a semi-honest corruption model. Similarly, if the underlying DO-shuffle satisfies malicious security (e.g., [5, 13]), 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 use the standard word-RAM model where the word size is logarithmic in the space. We assume that addition, multiplication, and boolean operations on words can be done in constant time. We also assume that sampling from truncated geometric distributions can be done in constant time[5]. 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 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. We assume that every extraneous location on the input tape stores a filler symbol $\perp$.

[5] Based on Appendix B of [24], we can obliviously sample a truncated geometric variable in expected time of $O(\frac{1}{\epsilon}\log\frac{1}{\delta})$. Further, we can sample M truncated geometric variables in time $O((M/\epsilon)\log(1/\delta))$ with probability $1 - \mathsf{negl}(M)$. Our algorithms described in Section 5 only need to sample $\frac{\epsilon L}{(\log^2 L \log(1/\delta))} + O(1)$ truncated geometrics. Therefore, all our runtime bounds(i.e., Theorem 5.2 and Theorem 5.6) hold in expectation and with high probability without assuming sampling in constant time.

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 $m' > m$ 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. By doing so, 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 $\mathsf{View}^{\mathsf{M}}(x) : \mathcal{X} \to \mathcal{V}$ 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 $\mathsf{Exec}^{\mathsf{M}}(x) : \mathcal{X} \to \mathcal{V} \times \mathcal{Y}$ 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

Definition and Composition Theorem Earlier, in Section 1, we argued why basic 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 strongly neighbor-preserving 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).

Composition Examples

Example 1. Recall that in Section 1, we gave a natural SQL database example that required applying one compaction algorithm on the output of another compaction algorithm. We pointed out that two sequential instances of Chan et al.'s DO compaction algorithm [17] do not give (tight) composable guarantees. In Example 1, we will see that if we replace the second instance with a modification of the compaction algorithm such that it is DO w.r.t. edit distance (as opposed to Hamming distance), the two instances would compose nicely.

Specifically, let M_1 be Chan et al.'s DO compaction algorithm [17]. 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[6] 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 DO merge algorithm was proposed by Chan et al. [17], 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. $\overset{2}{\sim}_E$.

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

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 [32], 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.

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. 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, like Examples 1 and 2.

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 [7,71]: given an input array x, we want to sample each entry with probability p, and generate a new array with only the sampled elements. Consider a 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. The key differences are highlighted by underlining.

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 encountered, 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 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 later prove that Examples 3 and 4 satisfy DO with our new 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**:

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

Definition 3.4 ((ϵ, δ)-NPDO). *Given a mechanism $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ with view space $\mathcal{V}$, 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 some 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 theorem.

Theorem 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.* $\mathcal{X}$ *.*

The proof is deferred to Appendix B.1 of the full version [74].

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 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 lemmas provide helpful tools for 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.*

The proof is deferred to Appendix B.1 of the full version [74].

(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 $Y \subseteq \mathcal{Y}$,*

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

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

Intuitively, (ϵ, δ)-NP requires 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 $\sim_{\mathcal{Y}}$.*

The proof is deferred to Appendix B.1 of the full version [74].

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,

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

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* $S \subseteq \Omega$, $\Pr[A \in S] \leq e^{\epsilon} \cdot \Pr[B \in \mathcal{N}(S)] + \delta$, *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* $a \in \Omega$, $\sum_{b \in \Omega, b \sim a} w(a,b) \leq \Pr[A = a]$;
   (c) *For all* $b \in \Omega$, $\sum_{a \in \Omega, a \sim b} w(a,b) \leq e^{\epsilon} \cdot \Pr[B = b]$;
   (d) $\sum_{a,b \in \Omega} w(a,b) \geq 1 - \delta$.

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 $|\mathcal{N}(S)| \geq |S|$, 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**. The full proof of Lemma 4.1 is deferred to Appendix B.2 of the full version [74]. Below, we prove our main composition theorem assuming that Lemma 4.1 holds.

Proof (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 $S \subseteq \mathcal{V}_1 \times \mathcal{V}_2 \times \mathcal{Z}$,

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

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,

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

$$\leq \sum_{(v_1,y)\in\mathcal{V}_1\times\mathcal{Y},y'\sim_{\mathcal{Y}}y} w\left((v_1,y),(v_1,y')\right)\cdot\left(e^{\epsilon_2}\Pr[\mathsf{Exec}^{\mathsf{M}_2}(y')\in\mathcal{N}(S_{v_1})]+\delta_2\right)+\delta_1$$

(Use condition (b) of the matching)

$$\leq \sum_{(v_1,y)\in\mathcal{V}_1\times\mathcal{Y},y'\sim_{\mathcal{Y}}y} w\left((v_1,y),(v_1,y')\right)\cdot\left(e^{\epsilon_2}\Pr[\mathsf{Exec}^{\mathsf{M}_2}(y')\in\mathcal{N}(S_{v_1})]\right)+\delta_2+\delta_1$$

(Use condition (c) of the matching)

$$\leq \sum_{(v_1,y')\in\mathcal{V}_1\times\mathcal{Y}} e^{\epsilon_1}\Pr[\mathsf{Exec}^{\mathsf{M}_1}(x')=(v_1,y')]\cdot\left(e^{\epsilon_2}\Pr[\mathsf{Exec}^{\mathsf{M}_2}(y')\in\mathcal{N}(S_{v_1})]\right)+\delta_2+\delta_1$$

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

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. [17] showed how to construct a stable compaction algorithm that is (ϵ,δ)-DO w.r.t. the *Hamming* distance [17], 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. [17]. 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 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. The 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 [18, 53] 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 [19,31] 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 [17], 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 [17] 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 $\mathsf{RandBin}$ 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 $\mathsf{CompactBin}$ 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 $\mathsf{RandBin}$ 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 $s = \frac{15}{\epsilon} \log^2 L \log \frac{1}{\delta}$ 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 $\rho_i \xleftarrow{\$} s + \mathcal{G}(\epsilon_2, \delta_2) \in [\frac{Z}{2} \ldots Z]$. Let $\boldsymbol{\rho} := (\rho_1, \rho_2, \ldots, \rho_B)$.
- 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 up to 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 revealing 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 always outputs the correct bin representation: For all $i \in [B]$, 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 are moved to the bins.*

Theorem 5.2. *Assuming $|I| \ge \frac{c}{\epsilon}\log\frac{1}{\delta}$ for any fix constant c,* RandBin *has a worst-case runtime of $O\left(|I|\left(\log\log|I| + \log\frac{1}{\epsilon} + \log\log\frac{1}{\delta}\right)\right)$.*

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

The above theorems' proofs are in Appendix B.3 of the full version [74].

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.

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

- Let $s = \frac{4}{\epsilon_1} \log^2 B \log \frac{2}{\delta_1}$, 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 $\mathsf{CompactBin}$ 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* $\mathsf{CompactBin}$ *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.*

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

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

The theorems' proofs are deferred to Appendix B.3 of the full version [74].

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 $|I| \geq \frac{c}{\epsilon} \log \frac{1}{\delta}$ for any fix constant c, 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 Differentially Oblivious Shuffle Model

In this section, we use our composition framework to prove a privacy amplification theorem for the differentially oblivious shuffle (DO-shuffle) model.

Background. Consider a distributed setting with n clients and an untrusted server who wants to learn some statistics of the clients' private data. To achieve this, the differential privacy literature proposed two models for achieving this. In the *local model*, each client adds some noise to its own data by running an ϵ_0-locally differentially private (LDP) mechanism, and sends the noisy result the server. The server then computes the desired statistics using each client's noisy input. In the local model, the server's view satisfies ϵ_0-DP.

Some more recent works [22, 23, 39, 40] considered a new model called the *shuffle model*. In this model, we assume that a trusted shuffler can shuffle all of the clients' messages, and the server only sees the permuted messages (without learning the permutation). Interestingly, earlier works [22, 23, 39, 40] showed that the shuffle model can *amplify* privacy. In particular, suppose each client still runs an ϵ_0-locally differentially private (LDP) mechanism before sending the noisy outcomes to the shuffler, then the server's view would satisfy (ϵ, δ)-DP where ϵ can be much smaller than ϵ_0. Notably, the recent work of Feldman et al. [37] 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).

To benefit from the shuffle model, we need to realize the shuffle either using trusted hardware or through a cryptographic protocol. Some recent works [5, 13, 45] showed that it may be asymptotically more efficient to realize a relaxed shuffler that satisfies differential obliviousness than a perfectly secure shuffler. Therefore, a natural question is whether we can also enjoy the same degree of privacy amplification with a differentially oblivious shuffler rather than a perfectly secure shuffler. We explore this question in the remainder of this section.

6.1 Definitions

Suppose that the server and the clients jointly execute a protocol to realize a shuffler. The syntax of a shuffle protocol is defined below.

Definition 6.1 (Syntax of a 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.*

Before defining security, we need to define the adversary's capabilities and the view of the adversary. 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. [45] assumes a semi-honest adversary cannot observe honest-to-honest communication, whereas Bünz et al. [13] 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.

Next, we define the notion of differential obliviousness for a shuffle protocol. We first need to define what neighboring means.

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 (Differential obliviousness of a shuffle protocol). *A shuffle protocol is said to satisfy statistical (ϵ, δ)-differential 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*

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

If we set $\epsilon = 0$ and $\delta = 0$, the above notion becomes equivalent to the security of a perfectly secure shuffle.

We next define a computational variant of the DO notion, since some known DO shuffler instantiations enjoy computational security [5, 13, 45].

Definition 6.4 (Computational DO for a shuffle protocol). *A shuffle protocol Φ is said to satisfy computational (ϵ, δ)-differential obliviousness in the*

presence of $t \le n$ corruptions, iff for any non-uniform 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

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

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.

Next, we define the notion of a locally differentially private (LDP) mechanism. The main privacy amplification theorem we want to prove in this section asks the following question: if each client computes its message by applying an ϵ_0-LDP mechanism to its private input, and then a DO shuffler shuffles all clients' messages before revealing the shuffled result to the server, can we prove that the server's view satisfies (ϵ, δ)-DP where ϵ is much smaller than ϵ_0?

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

6.2 Privacy Amplification in the DO-Shuffle Model

We formally restate Theorem 1.4 when the DO-shuffler Φ statisfies statistical DO as the following theorem. In Appendix A.1 of the full version [74], we will extend our composition framework to support the case when the DO-shuffler satisfies computational differential obliviousness (Definition 6.4).

Theorem 6.6. *Let $\mathcal{R} : \mathcal{X} \to \mathcal{Y}$ be an ϵ_0-LDP mechanism to be run by each client over its private input. Suppose $\mathcal{A}$ is an adversary controlling the server and at most t clients. Let Φ be a statistical (ϵ_1, δ_1)-DO shuffling protocol. Define the random experiment $\mathsf{Expt}^{\mathcal{A}}(x_1, \ldots, x_n)$ as the following where each $x_i \in \mathcal{X}$ denotes the private input of client $i \in [n]$:*

1. *Each honest client i treats the output of $\mathcal{R}(x_i)$ as the input in the next step;*
2. *Execute the DO-shuffling protocol Φ with the presence of the adversary $\mathcal{A}$ and let $\mathcal{A}$ also observe the outcome of the shuffling.*
3. *Output whatever $\mathcal{A}$ outputs.*

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. For any $\delta > 0$ that $\epsilon_0 \le \log\left(\frac{n-t}{16\log(2/\delta)}\right)$, we have that

$$\Pr\left[\mathsf{Expt}^{\mathcal{A}}(\mathbf{x}) = 1\right] \le e^{\epsilon+\epsilon_1} \cdot \Pr\left[\mathsf{Expt}^{\mathcal{A}}(\mathbf{x}') = 1\right] + \delta + \delta_1$$

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

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).

It is easy to see that the adversary has the same view in the composed mechanism $\mathsf{M}_2 \circ \mathsf{M}_1$ and in the random experiment described in Theorem 6.6. So we only need to prove that $\mathsf{M}_2 \circ \mathsf{M}_1$ is $(\epsilon+\epsilon_1, \delta+\delta_1)$-DO. The crux is to show that M_1 satisfies (ϵ, δ)-NPDO when at most t clients are corrupted, as more formally stated in the following lemma:

Lemma 6.7. *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).*

We can directly apply our composition theorem (Theorem 3.6)and Lemma 6.7 to get the desired result. The proof is in Appendix B.4 of the full version [74].

Acknowledgments

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. T-H. Hubert Chan was partially supported by the Hong Kong RGC under the grants 17201220, 17202121 and 17203122.

References

1. Abe, M.: Mix-networks on permutation networks. In: Lam, K.Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT'99. LNCS, vol. 1716, pp. 258–273. Springer, Heidelberg (Nov 1999). https://doi.org/10.1007/978-3-540-48000-6_21
2. Abraham, I., Pinkas, B., Yanai, A.: Blinder - scalable, robust anonymous committed broadcast. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 1233–1252. ACM Press (Nov 2020). https://doi.org/10.1145/3372297.3417261
3. Afshani, P., Freksen, C.B., Kamma, L., Larsen, K.G.: Lower bounds for multiplication via network coding. In: Baier, C., Chatzigiannakis, I., Flocchini, P., Leonardi, S. (eds.) ICALP 2019. LIPIcs, vol. 132, pp. 10:1–10:12. Schloss Dagstuhl (Jul 2019). https://doi.org/10.4230/LIPIcs.ICALP.2019.10

4. Aharoni, R., Berger, E., Georgakopoulos, A., Perlstein, A., Sprüssel, P.: The max-flow min-cut theorem for countable networks. Journal of Combinatorial Theory, Series B **101**(1), 1–17 (2011). https://doi.org/10.1016/j.jctb.2010.08.002
5. Ando, M., Lysyanskaya, A., Upfal, E.: Practical and provably secure onion routing. In: Chatzigiannakis, I., Kaklamanis, C., Marx, D., Sannella, D. (eds.) ICALP 2018. LIPIcs, vol. 107, pp. 144:1–144:14. Schloss Dagstuhl (Jul 2018). https://doi.org/10.4230/LIPIcs.ICALP.2018.144
6. Arasu, A., Kaushik, R.: Oblivious query processing. In: Proc. 17th International Conference on Database Theory (ICDT), Athens, Greece, March 24-28, 2014. pp. 26–37. OpenProceedings.org (2014). https://doi.org/10.5441/002/icdt.2014.07
7. Balle, B., Barthe, G., Gaboardi, M.: Privacy amplification by subsampling: Tight analyses via couplings and divergences. NeurIPS (2018). https://doi.org/10.5555/3327345.3327525
8. Balle, B., Bell, J., Gascón, A., Nissim, K.: The privacy blanket of the shuffle model. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 638–667. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26951-7_22
9. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 263–280. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_17
10. Beimel, A., Nissim, K., Zaheri, M.: Exploring differential obliviousness. In: Approx/Random. LIPIcs, vol. 145, pp. 65:1–65:20 (2019). https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2019.65, https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2019.65
11. Bittau, A., Erlingsson, U., Maniatis, P., Mironov, I., Raghunathan, A., Lie, D., Rudominer, M., Kode, U., Tinnes, J., Seefeld, B.: Prochlo: Strong privacy for analytics in the crowd. In: SOSP. p. 441–459. Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3132747.3132769, https://doi.org/10.1145/3132747.3132769
12. Bogatov, D., Kellaris, G., Kollios, G., Nissim, K., O'Neill, A.: ϵpsolute: Efficiently querying databases while providing differential privacy. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021. pp. 2262–2276. ACM Press (Nov 2021). https://doi.org/10.1145/3460120.3484786
13. Bünz, B., Hu, Y., Matsuo, S., Shi, E.: Non-interactive differentially anonymous router. Cryptology ePrint Archive, Report 2021/1242 (2021), https://eprint.iacr.org/2021/1242
14. Camenisch, J., Lysyanskaya, A.: A formal treatment of onion routing. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 169–187. Springer, Heidelberg (Aug 2005). https://doi.org/10.1007/11535218_11
15. Cash, D., Grubbs, P., Perry, J., Ristenpart, T.: Leakage-abuse attacks against searchable encryption. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015. pp. 668–679. ACM Press (Oct 2015). https://doi.org/10.1145/2810103.2813700
16. Cecchetti, E., Zhang, F., Ji, Y., Kosba, A.E., Juels, A., Shi, E.: Solidus: Confidential distributed ledger transactions via PVORM. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 701–717. ACM Press (Oct / Nov 2017). https://doi.org/10.1145/3133956.3134010
17. Chan, T.H.H., Chung, K.M., Maggs, B.M., Shi, E.: Foundations of differentially oblivious algorithms. In: Chan, T.M. (ed.) 30th SODA. pp. 2448–2467. ACM-SIAM (Jan 2019). https://doi.org/10.1137/1.9781611975482.150

18. Chan, T.H.H., Guo, Y., Lin, W.K., Shi, E.: Cache-oblivious and data-oblivious sorting and applications. In: Czumaj, A. (ed.) 29th SODA. pp. 2201–2220. ACM-SIAM (Jan 2018). https://doi.org/10.1137/1.9781611975031.143
19. Chan, T.H.H., Shi, E., Song, D.: Private and continual release of statistics. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010, Part II. LNCS, vol. 6199, pp. 405–417. Springer, Heidelberg (Jul 2010). https://doi.org/10.1007/978-3-642-14162-1_34
20. Chaum, D.: The dining cryptographers problem: Unconditional sender and recipient untraceability. Journal of Cryptology **1**(1), 65–75 (Jan 1988). https://doi.org/10.1007/BF00206326
21. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM **24**(2), 84–90 (Feb 1981). https://doi.org/10.1145/358549.358563
22. Cheu, A.: Differential privacy in the shuffle model: A survey of separations. arXiv preprint arXiv:2107.11839 (2021). https://doi.org/10.48550/arXiv.2107.11839
23. Cheu, A., Smith, A.D., Ullman, J., Zeber, D., Zhilyaev, M.: Distributed differential privacy via shuffling. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 375–403. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17653-2_13
24. Chu, S., Zhuo, D., Shi, E., Chan, T.H.: Differentially oblivious database joins: Overcoming the worst-case curse of fully oblivious algorithms. In: Tessaro, S. (ed.) ITC (2021). https://doi.org/10.4230/LIPIcs.ITC.2021.19
25. Connell, G.: Technology deep dive: Building a faster oram layer for enclaves. https://signal.org/blog/building-faster-oram/
26. Corrigan-Gibbs, H., Boneh, D., Mazières, D.: Riposte: An anonymous messaging system handling millions of users. In: 2015 IEEE Symposium on Security and Privacy. pp. 321–338. IEEE Computer Society Press (May 2015). https://doi.org/10.1109/SP.2015.27
27. Corrigan-Gibbs, H., Ford, B.: Dissent: accountable anonymous group messaging. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM CCS 2010. pp. 340–350. ACM Press (Oct 2010). https://doi.org/10.1145/1866307.1866346
28. Crooks, N., Burke, M., Cecchetti, E., Harel, S., Agarwal, R., Alvisi, L.: Obladi: Oblivious serializable transactions in the cloud. In: 13th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2018, Carlsbad, CA, USA, October 8-10, 2018. pp. 727–743. USENIX Association (2018). https://doi.org/10.5555/3291168.3291222
29. Degabriele, J.P., Stam, M.: Untagging Tor: A formal treatment of onion encryption. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 259–293. Springer, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78372-7_9
30. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (Mar 2006). https://doi.org/10.1007/11681878_14
31. Dwork, C., Naor, M., Pitassi, T., Rothblum, G.N.: Differential privacy under continual observation. In: Schulman, L.J. (ed.) 42nd ACM STOC. pp. 715–724. ACM Press (Jun 2010). https://doi.org/10.1145/1806689.1806787
32. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Found. Trends Theor. Comput. Sci. **9**(3–4), 211–407 (2014). https://doi.org/10.1561/0400000042, https://doi.org/10.1561/0400000042

33. Erlingsson, Ú., Feldman, V., Mironov, I., Raghunathan, A., Talwar, K., Thakurta, A.: Amplification by shuffling: From local to central differential privacy via anonymity. In: Chan, T.M. (ed.) 30th SODA. pp. 2468–2479. ACM-SIAM (Jan 2019). https://doi.org/10.1137/1.9781611975482.151
34. Eskandarian, S., Boneh, D.: Clarion: Anonymous communication from multiparty shuffling protocols. Cryptology ePrint Archive, Report 2021/1514 (2021), https://eprint.iacr.org/2021/1514
35. Eskandarian, S., Zaharia, M.: Oblidb: Oblivious query processing for secure databases. Proc. VLDB Endow. **13**(2), 169–183 (2019). https://doi.org/10.14778/3364324.3364331
36. Farhadi, A., Hajiaghayi, M., Larsen, K.G., Shi, E.: Lower bounds for external memory integer sorting via network coding. In: Charikar, M., Cohen, E. (eds.) 51st ACM STOC. pp. 997–1008. ACM Press (Jun 2019). https://doi.org/10.1145/3313276.3316337
37. Feldman, V., McMillan, A., Talwar, K.: Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. In: FOCS (2021). https://doi.org/10.1109/FOCS52979.2021.00096
38. Gertner, Y., Ishai, Y., Kushilevitz, E., Malkin, T.: Protecting data privacy in private information retrieval schemes. JCSS (2000). https://doi.org/10.1145/276698.276723
39. Ghazi, B., Golowich, N., Kumar, R., Pagh, R., Velingker, A.: On the power of multiple anonymous messages: Frequency estimation and selection in the shuffle model of differential privacy. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part III. LNCS, vol. 12698, pp. 463–488. Springer, Heidelberg (Oct 2021). https://doi.org/10.1007/978-3-030-77883-5_16
40. Ghazi, B., Kumar, R., Manurangsi, P., Pagh, R.: Private counting from anonymous messages: Near-optimal accuracy with vanishing communication overhead (2021). https://doi.org/10.48550/ARXIV.2106.04247
41. Girgis, A.M., Data, D., Diggavi, S., Kairouz, P., Suresh, A.T.: Shuffled model of federated learning: Privacy, accuracy and communication trade-offs. IEEE Journal on Selected Areas in Information Theory **2**(1), 464–478 (2021). https://doi.org/10.1109/JSAIT.2021.3056102
42. Goldreich, O.: Towards a theory of software protection and simulation by oblivious RAMs. In: Aho, A. (ed.) 19th ACM STOC. pp. 182–194. ACM Press (May 1987). https://doi.org/10.1145/28395.28416
43. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious rams. J. ACM **43**(3), 431–473 (may 1996). https://doi.org/10.1145/233551.233553, https://doi.org/10.1145/233551.233553
44. Gordon, S.D., Katz, J., Kolesnikov, V., Krell, F., Malkin, T., Raykova, M., Vahlis, Y.: Secure two-party computation in sublinear (amortized) time. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 2012. pp. 513–524. ACM Press (Oct 2012). https://doi.org/10.1145/2382196.2382251
45. Gordon, S.D., Katz, J., Liang, M., Xu, J.: Spreading the privacy blanket: - differentially oblivious shuffling for differential privacy. In: Ateniese, G., Venturi, D. (eds.) ACNS 22. LNCS, vol. 13269, pp. 501–520. Springer, Heidelberg (Jun 2022). https://doi.org/10.1007/978-3-031-09234-3_25
46. Grubbs, P., Lacharité, M.S., Minaud, B., Paterson, K.G.: Learning to reconstruct: Statistical learning theory and encrypted database attacks. In: 2019 IEEE Symposium on Security and Privacy. pp. 1067–1083. IEEE Computer Society Press (May 2019). https://doi.org/10.1109/SP.2019.00030

47. Grubbs, P., McPherson, R., Naveed, M., Ristenpart, T., Shmatikov, V.: Breaking web applications built on top of encrypted data. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 1353–1364. ACM Press (Oct 2016). https://doi.org/10.1145/2976749.2978351
48. Hall, P.: On representatives of subsets. Journal of the London Mathematical Society **1**(1), 26–30 (1935). https://doi.org/10.1112/jlms/s1-10.37.26
49. Hall Jr, M.: Distinct representatives of subsets. Bulletin of the American Mathematical Society **54**(10), 922–926 (1948). https://doi.org/10.1090/S0002-9904-1948-09098-X
50. Islam, M.S., Kuzu, M., Kantarcioglu, M.: Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In: NDSS 2012. The Internet Society (Feb 2012)
51. Jacob, R., Larsen, K.G., Nielsen, J.B.: Lower bounds for oblivious data structures. In: Chan, T.M. (ed.) 30th SODA. pp. 2439–2447. ACM-SIAM (Jan 2019). https://doi.org/10.1137/1.9781611975482.149
52. Kellaris, G., Kollios, G., Nissim, K., O'Neill, A.: Generic attacks on secure outsourced databases. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 1329–1340. ACM Press (Oct 2016). https://doi.org/10.1145/2976749.2978386
53. Lin, W.K., Shi, E., Xie, T.: Can we overcome the n log n barrier for oblivious sorting? In: Chan, T.M. (ed.) 30th SODA. pp. 2419–2438. ACM-SIAM (Jan 2019). https://doi.org/10.1137/1.9781611975482.148
54. Liu, C., Hicks, M., Harris, A., Tiwari, M., Maas, M., Shi, E.: Ghostrider: A hardware-software system for memory trace oblivious computation. In: ASPLOS (2015). https://doi.org/10.1145/2694344.2694385
55. Liu, C., Wang, X.S., Nayak, K., Huang, Y., Shi, E.: ObliVM: A programming framework for secure computation. In: 2015 IEEE Symposium on Security and Privacy. pp. 359–376. IEEE Computer Society Press (May 2015). https://doi.org/10.1109/SP.2015.29
56. Lochbihler, A.: A mechanized proof of the max-flow min-cut theorem for countable networks with applications to probability theory. Journal of Automated Reasoning pp. 1–26 (2022). https://doi.org/10.1007/s10817-022-09616-4
57. Maas, M., Love, E., Stefanov, E., Tiwari, M., Shi, E., Asanovic, K., Kubiatowicz, J., Song, D.: PHANTOM: practical oblivious computation in a secure processor. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013. pp. 311–324. ACM Press (Nov 2013). https://doi.org/10.1145/2508859.2516692
58. Mironov, I., Pandey, O., Reingold, O., Vadhan, S.P.: Computational differential privacy. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 126–142. Springer, Heidelberg (Aug 2009). https://doi.org/10.1007/978-3-642-03356-8_8
59. Mishra, P., Poddar, R., Chen, J., Chiesa, A., Popa, R.A.: Oblix: An efficient oblivious search index. In: 2018 IEEE Symposium on Security and Privacy. pp. 279–296. IEEE Computer Society Press (May 2018). https://doi.org/10.1109/SP.2018.00045
60. Naveed, M., Kamara, S., Wright, C.V.: Inference attacks on property-preserving encrypted databases. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015. pp. 644–655. ACM Press (Oct 2015). https://doi.org/10.1145/2810103.2813651
61. Ostrovsky, R., Shoup, V.: Private information storage (extended abstract). In: 29th ACM STOC. pp. 294–303. ACM Press (May 1997). https://doi.org/10.1145/258533.258606

62. Reed, M., Syverson, P., Goldschlag, D.: Anonymous connections and onion routing. IEEE Journal on Selected Areas in Communications **16**(4), 482–494 (1998). https://doi.org/10.1109/49.668972
63. Ren, L., Yu, X., Fletcher, C.W., van Dijk, M., Devadas, S.: Design space exploration and optimization of path oblivious RAM in secure processors. In: ISCA. pp. 571–582 (2013). https://doi.org/10.1145/2485922.2485971
64. Roughgarden, T.: Beyond the Worst-Case Analysis of Algorithms. Cambridge University Press (2020). https://doi.org/10.1017/9781108637435
65. Shi, E., Chan, T.H.H., Stefanov, E., Li, M.: Oblivious RAM with $O((\log N)^3)$ worst-case cost. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 197–214. Springer, Heidelberg (Dec 2011). https://doi.org/10.1007/978-3-642-25385-0_11
66. Shi, E., Wu, K.: Non-interactive anonymous router. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part III. LNCS, vol. 12698, pp. 489–520. Springer, Heidelberg (Oct 2021). https://doi.org/10.1007/978-3-030-77883-5_17
67. Stefanov, E., Shi, E.: ObliviStore: High performance oblivious cloud storage. In: 2013 IEEE Symposium on Security and Privacy. pp. 253–267. IEEE Computer Society Press (May 2013). https://doi.org/10.1109/SP.2013.25
68. Stefanov, E., van Dijk, M., Shi, E., Fletcher, C.W., Ren, L., Yu, X., Devadas, S.: Path ORAM: an extremely simple oblivious RAM protocol. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013. pp. 299–310. ACM Press (Nov 2013). https://doi.org/10.1145/2508859.2516660
69. Tinoco, A., Gao, S., Shi, E.: EnigMap: Signal should use oblivious algorithms for private contact discovery. Cryptology ePrint Archive, Report 2022/1083 (2022), https://eprint.iacr.org/2022/1083
70. Wang, X., Chan, T.H.H., Shi, E.: Circuit ORAM: On tightness of the Goldreich-Ostrovsky lower bound. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015. pp. 850–861. ACM Press (Oct 2015). https://doi.org/10.1145/2810103.2813634
71. Wang, Y.X., Balle, B., Kasiviswanathan, S.: Subsampled rényi differential privacy and analytical moments accountant. Journal of Privacy and Confidentiality **10**(2) (Feb 2021). https://doi.org/10.29012/jpc.723
72. Williams, P., Sion, R., Tomescu, A.: PrivateFS: a parallel oblivious file system. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 2012. pp. 977–988. ACM Press (Oct 2012). https://doi.org/10.1145/2382196.2382299
73. Zhou, M., Shi, E.: The power of the differentially oblivious shuffle in distributed privacy mechanisms. Cryptology ePrint Archive, Report 2022/177 (2022), https://eprint.iacr.org/2022/177
74. Zhou, M., Shi, E., Chan, T.H.H., Maimon, S.: A theory of composition for differential obliviousness. Cryptology ePrint Archive, Report 2022/1357 (2022), https://eprint.iacr.org/2022/1357
75. Zhuang, L., Zhou, F., Zhao, B.Y., Rowstron, A.: Cashmere: Resilient anonymous routing. In: NSDI (2005). https://doi.org/10.5555/1251203.1251225