Advanced Composition Theorems for Differential Obliviousness

Mingxun Zhou*† CMU
Mengshi Zhao‡ HKU
T-H. Hubert Chan§ HKU
Elaine Shi¶ CMU

Abstract

Differential obliviousness (DO) is a privacy notion which mandates that the access patterns of a program satisfy differential privacy. Earlier works have shown that in numerous applications, differential obliviousness allows us to circumvent fundamental barriers pertaining to fully oblivious algorithms, resulting in asymptotical (and sometimes even polynomial) performance improvements. Although DO has been applied to various contexts, including the design of algorithms, data structures, and protocols, its compositional properties are not explored until the recent work of Zhou et al. (Eurocrypt'23). Specifically, Zhou et al. showed that the original DO notion is not composable. They then proposed a refinement of DO called neighbor-preserving differential obliviousness (NPDO), and proved a basic composition for NPDO.

In Zhou et al.'s basic composition theorem for NPDO, the privacy loss is linear in k for k-fold composition. In comparison, for standard differential privacy, we can enjoy roughly $\sqrt{k}$ loss for k-fold composition by applying the well-known advanced composition theorem given an appropriate parameter range. Therefore, a natural question left open by their work is whether we can also prove an analogous advanced composition for NPDO.

In this paper, we answer this question affirmatively. As a key step in proving an advanced composition theorem for NPDO, we define a more operational notion called symmetric NPDO which we prove to be equivalent to NPDO. Using symmetric NPDO as a stepping stone, we also show how to generalize NPDO to more general notions of divergence, resulting in Rényi-NPDO, zero-concentrated-NPDO, Gassian-NPDO, and g-NPDO notions. We also prove composition theorems for these generalized notions of NPDO.

1 Introduction

Oblivious algorithms, first proposed by Goldreich and Ostrovsky [GO96, Gol87], are a family of privacy-preserving algorithms whose memory access patterns are provably obfuscated and do not leak any information about secret inputs. Oblivious algorithms have a broad class of applications, e.g., in cloud computing [SS13, WST12, sig22], blockchain applications [CZJ+17], secure processors [RYF+13, LHH+15, MLS+13, SBTL18], and multi-party computation [GKK+12, LWN+15, GHL+14]. They have also been deployed at a large-scale in practice (e.g., see Signal's deployment [sig22] of Path ORAM).

Although a line of works [SCSL11, SvDS+13, WCS15, WNL+14, BCP15, RS21] have made oblivious algorithms increasingly more efficient, oblivious algorithms nonetheless suffer from a couple drawbacks. First, it is known that for numerous computational tasks, achieving full obliviousness

*Randomized author ordering.
†mingxunz@andrew.cmu.edu
‡zmsxsl@connect.hku.hk
§hubert@cs.hku.hk
¶runting@gmail.com

must incur a *logarithmic slow-down* in comparison with insecure algorithms [GO96, Gol87, LN18]. Second, for scenarios where the (insecure) program's runtime depends on the secret input, full obliviousness demands that we pad the running time over every input to the worst-case over all inputs — such padding can often incur a *polynomial slow-down* (e.g., for database joins [CZSC21]).

To overcome these drawbacks, Chan, Chung, Maggs, and Shi [CCMS19] introduced a relaxed notion of access pattern privacy called differential obliviousness (DO). Unlike full obliviousness which requires that the access patterns over any two inputs of the same length be indistinguishable [GO96, Gol87], differential obliviousness (DO) requires that the access patterns satisfy the notion of differential privacy (DP) [DMNS06], i.e., the access patterns over two *neighboring inputs* should be *close in distribution* by some mathematical notion. Chan et al. [CCMS19] showed that for various computational tasks, DO allows us to circumvent the logarithmic barrier of full obliviousness; they and subsequent works [BNZ19, CZSC21] showed that DO avoids having to pad to the worst-case running time, and in this way, DO can achieve polynomial speedup over full oblivious algorithms.

Basic DO is not composable. When we design algorithms, it is customary to compose multiple algorithmic building blocks. Specifically, we often want to apply algorithm M_2 to the output of another algorithm M_1. For example, in SQL databases, we may want to make some `Select` query and store the result as a table T; later on, we may want to make another query over this table T. Ideally, if both algorithms M_1 and M_2 satisfy DO, we would like the composed algorithm $\mathsf{M}_2 \circ \mathsf{M}_1$ to be DO as well; moreover, we want a way to account for the privacy budget during such composition.

The original DO notion proposed by Chan et al. [CCMS19], however, did not lend to such composition. To understand why, we first review their definition. Consider a randomized algorithm $\mathsf{M} : \mathcal{X}_0 \to \mathcal{X}_1$ whose access patterns come from the view space $\mathcal{V}_1$. Let $\sim_0$ denote a suitable neighboring relation on the input domain $\mathcal{X}_0$ of M. The basic DO notion of Chan et al. [CCMS19] is described below:

Definition 1.1 (Basic DO [CCMS19])**.** *An algorithm (also called a mechanism)* M *is* (ϵ, δ)*-DO iff for any neighboring* $x \sim_0 x' \in \mathcal{X}_0$*, for any* $S \subseteq \mathcal{V}_1$*,*

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

where $\mathsf{View}^{\mathsf{M}}(x)$ *denotes the access patterns observed by the adversary when we execute* M *on the input* x*.*

To understand why basic DO does not lend to composition, observe that the above definition guarantees closeness in distribution of the views (i.e., access patterns) only when the algorithm M is executed on neighboring inputs. However, when we execute the first DO algorithm M_1 over two neighboring inputs x and x', the respective outputs $y := \mathsf{M}_1(x)$ and $y' := \mathsf{M}_1(x')$ may not be neighboring. Therefore, when we pass the respective outputs y and y' to the second algorithm M_2, the second mechanism M_2 may not provide any meaningful guarantee even though it satisfies DO. We also refer the reader to the work of Zhou et al. [ZSCM23] who gave several examples of natural DO algorithms where neighboring inputs produce outputs that are far in distance.

For this reason, *DO composition is intrinsically different from standard DP composition*, and we cannot use standard DP composition theorems [Vad17, DR14, DRV10] to reason about DO. Specifically, in DO, the second mechanism M_2 is applied to some hidden variable (i.e., the output of the first algorithm M_1) that is not in the adversary's view when M_1 is executed, and the basic DO notion does not provide any meaningful guarantee about this hidden variable.

A composable DO notion: neighbor-preserving DO (NPDO). Zhou et al. [ZSCM23] argued that for DO to have broader application, composition is a necessary feature — for example, one reason why standard DP is successful is due to its compositional properties. Therefore, Zhou et al. [ZSCM23] proposed a composable variant of the DO notion called neighbor-preserving DO. At a high level, they want to refine the basic DO notion to additionally require that "neighboring inputs produce neighboring outputs". However, the technicality is that many natural DO algorithms' outputs are randomized, and they want to define a probabilistic notion of output-neighboring that captures a broad class of natural DO algorithms. We now review their NPDO notion below — any algorithm that satisfies NPDO also satisfies the basic DO notion of Chan et al. [CCMS19].

Definition 1.2 (NPDO [ZSCM23])**.** *An algorithm $\mathsf{M} : \mathcal{X}_0 \to \mathcal{X}_1$ with view space $\mathcal{V}_1$ is (ϵ, δ)-NPDO (w.r.t. input relation $\sim_0$ and output relation $\sim_1$), if for any neighboring $x \sim_0 x' \in \mathcal{X}_0$, for any subset $S \subseteq \mathcal{V}_1 \times \mathcal{X}_1$,*

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

where $\mathsf{Exec}^{\mathsf{M}}(x)$ denotes the following experiment: execute M on the input x, and output the view (i.e., access patterns) observed by the adversary, as well as the outcome $\mathsf{M}(x)$; and the notation $\mathcal{N}(S)$, i.e., the neighboring set of S, is defined as follows:

$$\mathcal{N}(S) = \{(v, y) \in \mathcal{V}_1 \times \mathcal{X}_1 \mid \exists (v, y') \in S \text{ s.t. } y \sim_1 y'\}.$$

Zhou et al. [ZSCM23] showed that indeed, natural DO algorithms either satisfy or can easily be adapted to satisfy the above NPDO notion. They additionally proved the following composition theorem for NPDO.

Theorem 1.3 (NPDO composition [ZSCM23])**.** *Suppose $\mathsf{M}_1 : \mathcal{X}_0 \to \mathcal{X}_1$ is (ϵ_1, δ_1)-NPDO and $\mathsf{M}_2 : \mathcal{X}_1 \to \mathcal{X}_2$ is (ϵ_2, δ_2)-NPDO (w.r.t. each algorithm's input/output neighboring relations), and M_1 has discrete view and output spaces. Then, the composed algorithm $\mathsf{M}_2 \circ \mathsf{M}_1 : \mathcal{X}_0 \to \mathcal{X}_2$ is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-NPDO.*

The above composition theorem also suggests a method for privacy budget accounting when composing multiple NPDO algorithms. In particular, the privacy loss over time is additive, analogous to the basic composition theorem of DP [Vad17, DR14]. More specifically, if we perform the composition k times and each individual algorithm has the privacy parameter (ϵ, δ), then the k-fold composed algorithm consumes a privacy budget of $(k\epsilon, k\delta)$.

Question 1: advanced composition for NPDO? Recall that for (ϵ, δ)-DP, we have an advanced composition theorem [Vad17,DR14,DRV10] which proves a tighter bound for k-fold standard DP composition: suppose each individual mechanism satisfies (ϵ, δ)-DP, then k-fold composition results in a $(\epsilon \cdot \sqrt{2k \ln \frac{1}{\delta'}} + 2k\epsilon^2, k\delta + \delta')$-DP mechanism for an arbitrary $\delta' \in [0, 1]$. For a typical choice $\epsilon < 1$, the term $\epsilon \cdot \sqrt{2k \ln \frac{1}{\delta'}}$ dominates, and thus we can enjoy roughly a factor of $\sqrt{k}$ in the privacy loss using the advanced composition theorem, as opposed to the original factor k in the basic composition theorem.

A natural question is whether NPDO can also enjoy such an improved bound for k-fold composition. As mentioned, since DO composition is intrinsically different from DP composition, we cannot directly use the advanced composition theorem of DP to reason about NPDO composition. Further, for technical reasons to be discussed later in Section 1.1, we cannot extend Zhou et al. [ZSCM23]'s proof to get advanced composition in any easy way. Therefore, the following natural question is open:

Can we prove an advanced composition theorem for (ϵ, δ)-NPDO?

Question 2: beyond the (ϵ, δ)-closeness notion. While the (ϵ, δ)-closeness notion (also called δ-max divergence [DRV10]) is natural and intuitive, the standard DP literature recognized an important limitation of this notion. For most natural DP mechanisms, one can often tune the parameters to achieve a tradeoff between the parameters ϵ and δ. However, proving that a mechanism satisfies (ϵ, δ)-DP proves only a single point in this tradeoff curve, and fails to provide a knob to the practitioner who must set the concrete parameters in practice. Therefore, a line of works proposed to replace the (ϵ, δ)-closeness notion in DP with new divergence notions that can capture the entire tradeoff curve, e.g., Rényi DP [Mir17], zero concentrated DP [BS16], Gaussian DP [DRS22], and more recently, f-DP [DDR20, VZ23] which generalizes all of the above notions. The same works also proved composition theorems for these new notions. In particular, (ϵ, δ)-DP can be viewed as a special case of Rényi DP [Mir17], zero concentrated DP [BS16], and f-DP [DDR20]; and the advanced composition theorem of (ϵ, δ)-DP can be viewed as a corollary of the composition theorems for Rényi DP [Mir17], zero concentrated DP [BS16], and f-DP [DDR20].

This raises another natural question:

> *Can we also replace the divergence notion in NPDO and have generalized notions of NPDO that lend to composition?*

Applications of advanced composition. A direct motivation of our work is recent endeavors to build a differentially oblivious relational databases. Notably, the work of Qin et al. [QJS+22] designed DO relational database operators (e.g., compaction, group-by, and join), and they want to use these DO operators to build a DO relational database. Unfortunately, in their work, they did not address the issue of composition; as a result, they cannot compose these operators in database queries. Consider the following example of multi-fold composition in database applications. It is customary for an analyst to perform some SQL query, save the result in some table, and later perform more SQL queries on the result table — this process can be repeated multiple times. In these cases, our advanced composition theorems can allow the DO database engine to have tighter accounting of the privacy budget consumed so far, and thus achieve better privacy/utility tradeoff.

Note that our DO database application is analogous to how advanced composition can be useful in privacy budget accounting in traditional *differentially private* databases [McS10, MTS+12, NH12, KTH+19, WBNM22, WRRW23]. Therefore, advanced composition for DO can also be helpful in scenarios where advanced composition was useful for traditional DP. For example, there is an elegant line of work that uses formal methods to programmatically track the privacy loss of a differentially private program [BGHP16, ZK17, VRG20]. Therefore, we envision that our advanced composition theorems for DO will be helpful in a similar manner if we are to extend such formal methods to support DO reasoning.

1.1 Main Result 1: Advanced Composition Theorem for (ϵ, δ)-NPDO

Our first contribution is to prove an advanced composition theorem for (ϵ, δ)-NPDO, as stated below.

Theorem 1.4 (Advanced composition theorem for NPDO)**.** *Let $\epsilon \geq 0$, $\delta, \delta' \in [0, 1]$ and $k \geq 2$. Suppose for $i \in [k]$, the algorithm $\mathsf{M}_i : \mathcal{X}_{i-1} \to \mathcal{X}_i$ is (ϵ, δ)-NPDO with respect to the neighboring relations for its input and output spaces. Further, suppose that $\mathsf{M}_1, \ldots, \mathsf{M}_k$ have finite output and view spaces. Then, the composition $\mathsf{M}_k \circ \mathsf{M}_{k-1} \circ \cdots \circ \mathsf{M}_1$ is $(\epsilon', \delta' + k\delta)$-NPDO, where $\epsilon' = \epsilon\sqrt{2k \ln \frac{1}{\delta'}} + 2k\epsilon^2$.*

Challenges and technical highlight. The advanced composition for standard DP [Vad17, DR14, DRV10] provides the following mathematical tool for reasoning about the divergence of distributions. Given k pairs of distributions (W_1, U_1), $(W_2, U_2), \ldots,$ (W_k, U_k) such that for each $i \in [k]$, W_i and U_i are (ϵ, δ)-close (even when conditioned on $W_1, U_1, \ldots, W_{i-1}, U_{i-1}$), then the joint distribution $(W_1, W_2, \ldots, W_k)$ and $(U_1, U_2, \ldots, U_k)$ are $(\epsilon', \delta' + k\delta)$-close where ϵ' and δ' are defined as in Theorem 1.4.

Unfortunately, for NPDO composition, we are unable to directly leverage the existing mathematical tool, because the NPDO definition (Definition 1.2) is not formulated as a statement of divergence over two distributions, due to the interference of the $\mathcal{N}(\cdot)$ operator on the right-hand-side of Equation (2). To prove the advanced composition theorem for NPDO, we need the following key insights.

1. *A New, Equivalent formulation of NPDO: symmetric NPDO.* As a key stepping stone towards proving advanced composition for NPDO, we first propose an alternative, equivalent formulation of NPDO called *symmetric NPDO*. This step of the proof is techncially involved and rather non-trivial.

   At a very high level, recall that Zhou et al. showed that (ϵ, δ)-NPDO is equivalent to the existence of an (ϵ, δ)-matching in a bipartite graph where each vertex captures the probability of seeing a particular (view, output) pair in a randomized execution, and the two sides of the bipartite graph consider random executions over neighboring inputs, respectively. Zhou et al.'s definition of an (ϵ, δ)-matching treats the two sides of the bipartite graph in an *asymmetric* manner. We first define an alternative notion called (ϵ, δ)-symmetric-NPDO which can be viewed as a pair of (ϵ, δ)-matchings, one from each direction. To prove that (ϵ, δ)-symmetric-NPDO implies (ϵ, δ)-NPDO, we need to prove that as long as the algorithm's output and view spaces are finite, then 1) (ϵ, δ)-symmetric-NPDO implies (ϵ, δ)-NPDO; and 2) (ϵ, δ)-NPDO implies (ϵ, δ)-symmetric-NPDO, The first direction is the easier one; however, proving the second direction is rather challenging. At a very high level, we start out with an asymmetric matching, and we then iteratively modify the matching. We prove that the iterative adjustments will stop in finite number of steps, and the resulting matching converges to a symmetric one — see Section 3.2 and Appendix A.2 for the details of the proof.

2. *Symmetric NPDO expressed as divergence of two distributions.* We then suggest an alternative way to view the new (ϵ, δ)-symmetric-NPDO notion. In particular, the symmetric (ϵ, δ)-matching on the bipartite graph can be viewed as two different assignments of weights to the edges of the bipartite graph, such that if we view each assignment as a probability distribution, then the two distributions are (ϵ, δ)-close. This allows us to express (ϵ, δ)-NPDO as a divergence notion on distributions. With this new view, we can now rely on the aforementioned mathematical tool to reason about the divergence of distributions, which leads to our NPDO advanced composition theorem.

Regarding our symmetric NPDO notion. We envision that our symmetric NPDO notion can be of independent interest in future work, such that the equivalence of NPDO and symmetric NPDO provides a new flexibility in designing algorithms for differential obliviousness. On the one hand, the original NPDO notion of Zhou et al. [ZSCM23] is more intuitive and sometimes easier for algorithm designers to use. On the other hand, our new symmetric NPDO notion is more operational in theoretical proofs. In particular, many tools developed in traditional DP rely on reasoning about divergence notions (e.g. variants of composition frameworks [Mir17, BS16, DDR20, VZ23], privacy amplification [BBG18, FMT23], privacy accounting [FZ21, ZDW22] and

privacy auditing [TTS+22, NHS+23, SNJ23]). By showing the equivalence of NPDO and symmetric NPDO, we are able to recast NPDO notions in the form of divergence between two probability distributions — this allows us to reuse the mathematical tools from classical DP to reason about differential obliviousness. For the same reason, when we have the symmetric NPDO formulation, it is also easier to generalize to other notions of divergence — see our subsequent Section 1.2.

1.2 Main Result 2: Composition for Generalized Notions of NPDO

As mentioned, we want to generalize the divergence notion in NPDO to a more general one, e.g., the tradeoff function notion in f-DP [DDR20, VZ23] which generalizes a line of earlier works [Mir17, BS16, DRS22, DDR20, VZ23].

A strawman attempt is to start with the basic (ϵ, δ)-DO notion by Chan et al. [CCMS19], which is formulated as a statement on the divergence of two distributions. While we can easily replace the divergence notion in basic (ϵ, δ)-DO with a more general one, the resulting notions would not be composable for the same reason why (ϵ, δ)-DO is not composable.

To get composability, we want to start with the (ϵ, δ)-NPDO notion proposed by Zhou et al. [ZSCM23]. However, as mentioned, their (ϵ, δ)-NPDO is not formulated as a statement of divergence over two distributions due to the $\mathcal{N}(\cdot)$ operator. Fortunately, recall that as a key stepping stone in proving the advanced composition theorem for (ϵ, δ)-NPDO, we formulated an equivalent notion, that is, (ϵ, δ)-symmetric-NPDO, which is indeed stated in terms of the divergence over two distributions (defined as two ways to assign weights to a particular bipartite graph that capture randomized executions on neighboring inputs). This alternative formulation of NPDO allows us to easily replace the divergence notion with more general ones, resulting in Rényi-NPDO, zero concentrated-NPDO, Gaussian-NPDO, and g-NPDO, which are analogous to Rényi-DP [Mir17], zero concentrated-DP [BS16], Gaussian-DP [DRS22], and f-DP [DRS22], respectively.

Using the same proof framework we developed for proving the (ϵ, δ)-NPDO composition, we then prove the corresponding composition theorems for Rényi-NPDO, zero concentrated-NPDO, Gaussian-NPDO, and g-NPDO.

1.3 Additional Related Work

Earlier works have shown various applications of DO, and scenarios in which it can asymptotically outperform any fully oblivious algorithm. Beimel, Nissim, and Zaheri [BNZ19] apply DO to property testing, and they show that DO achieves an almost linear factor improvement over any fully oblivious algorithm in the dense graph model and at most quadratic improvement in the bounded degree model. Chu et al. [CZSC21] showed that DO algorithms can achieve an almost linear speedup over any fully oblivious algorithm for performing database joins. Gordon et al. [GKLX22] showed that for the privacy amplification theorem in the shuffle model [CSU+19, EFM+19, GDDa, BBGN19, GDD+b, GGK+, Che21], one can replace a fully secure shuffler with a differentially oblivious shuffler, and retain almost the same privacy amplification guarantees. Gordon et al. [GKLX22]'s proof only works for the randomized response mechanism; more recently, Zhou et al. [ZSCM23] showed how to generalize Gordon et al. [GKLX22]'s proof to any local DP mechanism, by directly applying their NPDO composition theorem (Theorem 1.3).

A line of work [WCM18, PY19, PY23, KS21] focused on differentially oblivious RAM [WCM18, PY19], differentially oblivious Turing Machines [KS21], or differentially oblivious data structures [PY23, CCMS19, BKK+21]. In this line of work, the neighboring relation is defined over the sequence of operations. This line of work has not considered composition; however, if these techniques need to be composed to design new algorithms, and the second algorithm's input operations come from

the first algorithm's output, the same compositional issue will arise which we address in our work. Wagh et al. [WCM18] explored differentially private Oblivious RAM (ORAM) as a relaxation of standard ORAM. With this relaxation, they were able to achieve constant-factor improvements over known ORAM constructions [SvDS+13, SvDS+18]. In an elegant work, Persiano and Yeo [PY19] proved that a generic compiler that can compile *any* program to a "differentially oblivious" counterpart must suffer from logarithmic slowdown. In a subsequent work, the same authors proved more general lower bounds for differentially oblivious RAM and differentially oblivious data structures [PY23]. Chan et al. [CCMS19] showed that for range query data structures, using DO can achieve asymptotic gains over any fully oblivious algorithm (even when we allow the fully oblivious algorithm to leak the true length of the answer); and not only so, their DO algorithm achieves non-interactivity which is not known for any fully oblivious data structures (with statistical security). Kellaris et al. [BKK+21] applied a computational notion of differential obliviousness to an outsourced database application, and their construction relied on ORAM as a blackbox. The subsequent work of Chan et al. [CCMS19] showed how to achieve the same goals without having to rely on ORAM, and in a way that asymptotically outperforms applying a blackbox ORAM scheme. The work of Komargodski and Shi [KS21] showed upper bounds and lower bounds for the performance of differentially oblivious Turing Machines.

2 Preliminaries

We will use the following notations:

- For any $x \in \mathbb{R}$, denote $[x]_+ := \max(x, 0)$.
- We say that some set is *discrete* iff it is finite or countably infinite.
- If a distribution D is defined over some space Ω, we often use the notation $D : \Omega$.

2.1 Execution Model

Consider an *algorithm* (also called a *mechanism*) $\mathsf{M} : \mathcal{X}_0 \to \mathcal{X}_1$, where $\mathcal{X}_0$ is its input space and $\mathcal{X}_1$ is its output space. For each data space $\mathcal{X}_i$ where $i \in \{0, 1\}$, there is some corresponding binary neighboring relation $\sim_i$ that indicates that two data points are "close" in their respective space $\mathcal{X}_i$. Unless otherwise stated, the neighboring relation is required to be reflexive and symmetric (and usually not transitive). For example, $x \sim_i x'$ may mean that the two input x and x' differ in at most c positions; it may also mean that x and x' have edit distance at most c.

Unlike in the standard DP setting, the *adversary* cannot directly observe the algorithm's output, but can observe some behavioral patterns of the algorithm during the course of execution, henceforth called the *view*. We model the *view* of the adversary as an element in some view space $\mathcal{V}_1$.

Just like the work of Zhou et al. [ZSCM23], our DO formulation is not tied to any particular execution model. For example,

- M can be a RAM or PRAM program whose access patterns are observable by an adversary — in this case, the access patterns of the program form the view; or
- M can be a protocol whose message communication patterns are observable by the adversary — in this case, the communication patterns of the protocol form the view.

We use the following notation throughout the paper. Given some input $x \in \mathcal{X}_0$ to M, we define the following random variables:

- The output $\mathsf{Out}^{\mathsf{M}}(x)$ is a distribution on $\mathcal{X}_1$.
- The view $\mathsf{View}^{\mathsf{M}}(x)$ is a distribution on $\mathcal{V}_1$.
- The execution $\mathsf{Exec}^{\mathsf{M}}(x)$ outputs the joint distribution $(\mathsf{View}^{\mathsf{M}}(x), \mathsf{Out}^{\mathsf{M}}(x))$ on the execution space $\mathcal{V}_1 \times \mathcal{X}_1$.

Later in our technical sections, it is helpful to have a neighboring relation for the execution space $\mathcal{V}_1 \times \mathcal{X}_1$. In particular, we say that (v, x) is neighboring to (v', x) iff $v = v'$ and $x \sim_1 x'$. For this reason, without risk of ambiguity, we extend the notation $\sim_1$ to denote a neighboring relation for the execution space too.

Definition 2.1 (Extension of neighboring relation to execution space.)**.** *Given a binary neighboring relation $\sim_1$ on the output space $\mathcal{X}_1$, we extend it naturally to the execution space $\mathcal{V}_1 \times \mathcal{X}_1$ by the rule: $(v, x) \sim_1 (v', x')$ iff $v = v'$ and $x \sim_1 x'$.*

Sequential composition of algorithms. Given k mechanisms denoted $\mathsf{M}_1, \ldots, \mathsf{M}_k$, where for $i \in [k]$, $\mathsf{M} : \mathcal{X}_{i-1} \to \mathcal{X}_i$, the sequentially composed algorithm $\mathsf{M}_k \circ \mathsf{M}_{k-1} \circ \ldots \circ \mathsf{M}_1 : \mathcal{X}_0 \to \mathcal{X}_k$ is the one that runs M_1 first on some input $x \in \mathcal{X}_0$, then runs M_2 on the output of M_1, and so on.

We assume that act of passing the output of M_{i-1} to M_i does not generate any observable event for the adversary. For example, as explained by Zhou et al. [ZSCM23], if we consider $\mathsf{M}_1, \ldots, \mathsf{M}_k$ to be RAM algorithms, we can assume that M_{i-1} writes its output on some output tape, and then the next algorithm M_i will simply treat M_{i-1}'s output tape as its input tape. In natural DO or NPDO algorithms [ZSCM23], to hide output and input lengths, M_{i-1} may actually write some extra filler symbols on its output tape beyond the actual output; and similarly, M_i may make some fake reads from its input tape. The access patterns incurred by reading or writing input/output tapes are treated as part of the view of each individual algorithm concerned.

2.2 Divergence Notions

We define some useful divergence notions that measure the closeness of two distributions.

Max divergence and (ϵ, δ)-closeness. First, we review the max-divergence notion that is used by (ϵ, δ)-DP and (ϵ, δ)-DO.

Definition 2.2 (δ-max divergence)**.** *Suppose $\delta \in [0, 1]$. Given distributions A and B on the same sample space Ω, the δ-max divergence is defined as follows:*

$$\mathsf{D}^{\delta}(A \| B) = \sup_{S \subseteq \Omega : \Pr[A \in S] \geq \delta} \left[\log \frac{\Pr[A \in S] - \delta}{\Pr[B \in S]} \right]_+ . \tag{3}$$

In fact, $\mathsf{D}^{\delta}(A \| B) \leq \epsilon$ is equivalent to the following:

$$\forall S \subseteq \Omega, \Pr[A \in S] \leq e^{\epsilon} \cdot \Pr[B \in S] + \delta. \tag{4}$$

Definition 2.3 ((ϵ, δ)-close)**.** *Given two distributions A and B on the same sample space Ω, we say A and B are (ϵ, δ)-close if*

$$\max(\mathsf{D}^{\delta}(A \| B), \mathsf{D}^{\delta}(B \| A)) \leq \epsilon.$$

Max divergence w.r.t. some neighboring relation. As mentioned in Section 1.1, the NPDO notion by Zhou et al. [ZSCM23] cannot be expressed as a standard divergence notion between two distributions, since the notion is defined with respect to some neighboring relation $\sim$.

For convenience, we introduce a new max-divergence notion that is defined w.r.t. to some neighboring relation $\sim$.

Definition 2.4 (δ-max divergence w.r.t. neighboring relation $\sim$)**.** *Given two distributions A and B defined over some space Ω and a neighboring relation $\sim$, for $0 \le \delta \le 1$, the divergence $\mathsf{D}^{\delta}_{\sim}(A\|B)$ is the infimum over $\epsilon \ge 0$ such that for every $S \subseteq \Omega$,*

$$\Pr[A \in S] \le e^{\epsilon} \cdot \Pr[B \in \mathcal{N}(S)] + \delta,$$

where the neighbor set $\mathcal{N}(S)$ is defined as $\mathcal{N}(S) := \{b \in \Omega \mid \exists a \in S, a \sim b\}$.

Using Definition 2.4, we can restate the NPDO notion (Definition 1.2) as follows:

Definition 2.5 (NPDO: restatement of Definition 1.2)**.** *An algorithm $\mathsf{M} : \mathcal{X}_0 \to \mathcal{X}_1$ with the view space $\mathcal{V}$, input neighboring relation $\sim_0$, and output neighboring relation $\sim_1$ is said (ϵ, δ)-NPDO iff for any for all neighboring inputs $x \sim_0 x'$ from $\mathcal{X}_0$, $\mathsf{D}^{\delta}_{\sim_1}(\mathsf{Exec}^{\mathsf{M}}(x)\|\mathsf{Exec}^{\mathsf{M}}(x')) \le \epsilon$.*

2.3 (ϵ, δ)-NPDO and (ϵ, δ)-Matching

Zhou et al. [ZSCM23] showed that an algorithm M satisfies (ϵ, δ)-NPDO iff there exists an (ϵ, δ)-matching in a specific bipartite graph that encodes the distribution of two neighboring executions $\mathsf{Exec}(x)$ and $\mathsf{Exec}(x')$ where x and x' are neighboring. Consider an algorithm $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ with the view space $\mathcal{V}$. Recall that a randomized execution $\mathsf{Exec}(x)$ over some input $x \in \mathcal{X}$ gives a (view, output) pair henceforth denoted $(v, y) \in \mathcal{V} \times \mathcal{Y}$. Henceforth we use $\sim_0$ to denote input-neighboring, and use and $\sim_1$ to denote output-neighboring. By Definition 2.1, we also use the same notation $\sim_1$ to denote an extended neighboring relation on the execution space $\mathcal{V} \times \mathcal{Y}$.

Bipartite graph induced by two distributions. Let $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ be an algorithm with the view space $\mathcal{V}$. Consider two randomized executions $\mathsf{Exec}^{\mathsf{M}}(x)$ and $\mathsf{Exec}^{\mathsf{M}}(x')$ on two neighboring inputs $x \sim x'$, and create the following bipartite graph:

- Each vertex on the left corresponds to a pair $(v, y) \in \mathcal{V} \times \mathcal{Y}$, and the weight on the vertex is its probability density measure in $\mathsf{Exec}^{\mathsf{M}}(x)$.
- Similarly, each vertex on the right corresponds to a pair $(v', y') \in \mathcal{V} \times \mathcal{Y}$, and the weight on the vertex is its probability measure in $\mathsf{Exec}^{\mathsf{M}}(x')$.
- Draw an edge between (v, y) and (v', y') iff $(v, y) \sim_1 (v', y')$, i.e., iff $v' = v$ and $y' \sim_1 y$.

In the above, the bipartite graph is induced by the two distributions $\mathsf{Exec}^{\mathsf{M}}(x)$ and $\mathsf{Exec}^{\mathsf{M}}(x')$ and the neighboring relation $\sim_1$. More generally, we can also define a bipartite graph induced by any two distributions and w.r.t. some neighboring relation $\sim$.

(ϵ, δ)-matching. Given the vertex weights (corresponding to probability density) on the aforementioned bipartite graph, an (ϵ, δ)-matching is a weight assignment to the **edges**, such that the following properties are satisfied:

- For every vertex on the left, the sum of weights on all incident edges is upper bounded by the vertex's weight;

- For every vertex on the right, the sum of weights on all incident edges is at most e^ϵ times the vertex's weight;
- The sum of all edge weights is at least $1 - \delta$.

More intuitively, one can imagine that each left vertex is a factory which has produced some percentage of the supplies, and the total amount of supplies produced is 1. Each right vertex is some consumer who is requesting some percentage of the supplies, and the total amount requested is also 1. Now, the factories want to route their supplies to the consumers, such that each consumer does not receive more than e^ϵ times their requested amount; and moreover, all but δ fraction of the supplies must be successfully distributed.

Formally, we define (ϵ, δ)-matching for any two distributions A and B defined over the same space Ω, and w.r.t. to some neighboring relation $\sim$.

Definition 2.6 ((ϵ, δ)-matching)**.** *Given two distributions A and B defined over Ω , and a neighboring relation $\sim$, an (ϵ, δ)-matching from A to B denoted $w : \Omega \times \Omega \to [0, 1]$ is a mapping that satisfies the following conditions:*

1. *For all $a \in \Omega$ and $b \in \Omega$, $w(a, b) > 0$ only if $a \sim b$;*
2. *For all $a \in \Omega$, $\sum_{b \in \Omega} w(a, b) \leq \Pr[A = a]$;*
3. *For all $b \in \Omega$, $\sum_{a \in \Omega} w(a, b) \leq e^\epsilon \cdot \Pr[B = b]$;*
4. $\sum_{a \in \Omega} \sum_{b \in \Omega} w(a, b) \geq 1 - \delta$.

As mentioned, the best way to understand the above definition is to think of the bipartite graph induced by the distributions A and B, and redistributing weights from left vertices to incident edges.

(ϵ, δ)-NPDO and (ϵ, δ)-matching. Zhou et al. [ZSCM23] showed the following equivalence between (ϵ, δ)-NPDO and the existence of an (ϵ, δ)-matching in the bipartite graph induced by two randomized executions $\mathsf{Exec}^{\mathsf{M}}(x)$ and $\mathsf{Exec}^{\mathsf{M}}(x')$ on neighboring inputs x and x'.

Lemma 2.7 (Equivalence of $\mathsf{D}^\delta_\sim \leq \epsilon$ and existence of an (ϵ, δ)-matching)**.** *Suppose A and B are distributions over some discrete sample space Ω, and let $\sim$ be a neighboring relation on Ω. Under the Axiom of Choice, the following statements are equivalent:*

1. $\mathsf{D}^\delta_\sim(A \| B) \leq \epsilon$.
2. *There exists an (ϵ, δ)-matching from A to B w.r.t. $\sim$.*

As a direct corollary of Lemma 2.7, we have the following:

Corollary 2.8 (Equivalence of (ϵ, δ)-NPDO and existence of an (ϵ, δ)-matching)**.** *Let $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ be an algorithm with the view space $\mathcal{V}$, input relation $\sim_0$, and output relation $\sim_1$. Suppose that $\mathcal{V}$ and $\mathcal{Y}$ are discrete spaces, then, the following statements are equivalent under the Axiom of Choice:*

1. M *satisfies (ϵ, δ)-NPDO;*
2. *For any $x \sim_0 x'$ from $\mathcal{X}$, there exists an (ϵ, δ)-matching from $\mathsf{Exec}^{\mathsf{M}}(x)$ to $\mathsf{Exec}^{\mathsf{M}}(x')$ w.r.t. the extended neighboring relation $\sim_1$ for the execution space $\mathcal{V} \times \mathcal{Y}$.*

3 Symmetric NPDO

Recall that (ϵ, δ)-NPDO is expressed in the form of the δ-max divergence *w.r.t. some neighboring relation* $\sim$ (Definition 2.4). To prove an advanced composition theorem for NPDO, we want to leverage mathematical tools for reasoning about the divergence between composed distributions — however, existing tools work only for standard notions of divergence that is not defined w.r.t. to some relation $\sim$.

In this section, we introduce an equivalent formulation of (ϵ, δ)-NPDO called (ϵ, δ)-symmetric-NPDO. The new notion (ϵ, δ)-symmetric-NPDO is indeed expressed in the form of standard δ-max divergence over two distributions (Definition 2.2) — the challenge is how to define these two distributions such that the resulting definition is equivalent to the original (ϵ, δ)-NPDO.

3.1 Neighbor-Respecting Refined Distribution

In Section 2.3, we introduced the notion of an induced bipartite graph given two distributions A and B and some neighboring relation $\sim$. We also defined an (ϵ, δ)-matching from A to B and w.r.t. $\sim$. Our first idea is to interpret the edge weights (henceforth denoted W) in the (ϵ, δ)-matching as a distribution. Unfortunately, W may not form a valid distribution because the edge weights may not sum to 1 due to the δ probability mass remaining.

Neighbor-respecting refined distribution. We will instead consider a distribution of weights from the left vertices to its incident edges such that all weights must be completely distributed without any remainder. With this intuition in mind, we can define a notion called a neighbor-respecting refined distribution. Intuitively, given a distribution A on some space Ω with the neighboring relation $\sim$, imagine a bipartite graph where the left vertex set Ω_1 and the right vertex set Ω_2 are both equal to Ω, and each left vertex has a weight equal to its probability measure in A. Further, $(a, a') \in \Omega_1 \times \Omega_2$ is an edge in the graph iff $a \sim a'$. Now, imagine that we want to completely redistribute the left vertices' weights onto its incident edges, such that there is no weight remaining. The resulting assignment of weights on the edges $W : (\Omega_1 \times \Omega_2) \to [0, 1]$ can be viewed as a distribution, and we call it the neighbor-respecting refined distribution of A w.r.t. $\sim$.

More formally, we define neighbor-respecting refined distribution as below.

Definition 3.1 (Neighbor-respecting refined distribution)**.** *Let Ω_1 and Ω_2 be two discrete sets both equal to Ω, and let $\sim$ be a neighboring relation on Ω. Let A be a distribution on Ω_1 and let W be a distribution on $\Omega_1 \times \Omega_2$. We say that the distribution W is a neighbor-respect refined distribution of A w.r.t. $\sim$ iff*

- *The marginal distribution of W on Ω_1 henceforth denoted $W_{|\Omega_1}$ is equal to A.*
- *For any (a, a') in the support of W, it must be that $a \sim a'$.*

Symmetrically, let B be a distribution on Ω_2 and let U be a distribution on $\Omega_1 \times \Omega_2$. We say that the distribution U is a neighbor-respect refined distribution of B w.r.t. $\sim$ iff

- *The marginal distribution of U on Ω_2 henceforth denoted $W_{|\Omega_2}$ is equal to B.*
- *For any (a, a') in the support of U, it must be that $a \sim a'$.*

Using the terminology of neighbor-respecting refined distribution, the existence of an (ϵ, δ)-matching from A to B w.r.t. $\sim$ is in fact equivalent to the existence of a neighbor-respecting refined distribution W on the space $\Omega_1 \times \Omega_2$ of A w.r.t. $\sim$ such that $\mathsf{D}^{\delta}(W_{|\Omega_2}||B) \leq \epsilon$.

Lemma 3.2. *Let $\Omega_1 = \Omega_2 = \Omega$ denote a discrete set with the neighboring relation $\sim$, and let A and B be distributions on Ω_1 and Ω_2, respectively. Under the Axiom of Choice, the following statements are equivalent:*

- *There exists an (ϵ, δ)-matching from A to B w.r.t. $\sim$.*
- *There exists a neighbor-respecting refined distribution W over $\Omega_1 \times \Omega_2$ of A w.r.t. $\sim$, such that $\mathsf{D}^\delta(W_{|\Omega_2}||B) \leq \epsilon$.*

Proof. Deferred to Appendix A.1. □

Just like the (ϵ, δ)-matching notion, the distribution W in Lemma 3.2 can be viewed as a way to re-distribute the weights on the sources (defined by the distribution A), in a way that approximately respects the capacities of the destinations (defined by the distribution B). The difference between an (ϵ, δ)-matching and W is the following. An (ϵ, δ)-matching allows δ mass to remain at the sources, and requires that each destination receive no more than e^ϵ times its capacity (henceforth called the e^ϵ-relaxed capacity). By contrast, the neighbor-respecting refined distribution W insists that all mass be routed away from the sources; however, the condition $\mathsf{D}^\delta(W_{|\Omega_2}||B) \leq \epsilon$ says that we can tolerate up to δ total excess over all destinations, relative to their e^ϵ-relaxed capacities.

3.2 Symmetric NPDO

Our symmetric NPDO notion is defined in a way similar to Lemma 3.2. However, Lemma 3.2 is asymmetric and considers only one neighbor-respecting distributions W that can be viewed as routing probability mass from left to right. In symmetric NPDO, we make the definition symmetric by having two neighbor-respecting distributions W and U — one can be viewed as routing weights from the left to the right, and the other routing weights from the right to the left. Further, the two distributions W and U must be (ϵ, δ)-close.

Definition 3.3 ((ϵ, δ)-symmetric NPDO)**.** *An algorithm $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ with the view space $\mathcal{V}$ is said to satisfy (ϵ, δ)-symmetric NPDO w.r.t. input relation $\sim_0$ and output relation $\sim_1$, iff for any neighboring inputs $x \sim_0 x'$ from $\mathcal{X}$, the following hold for the two distributions $\mathsf{Exec}^{\mathsf{M}}(x) : \mathcal{V}_1 \times \mathcal{Y}_1$ and $\mathsf{Exec}^{\mathsf{M}}(x') : \mathcal{V}_2 \times \mathcal{Y}_2$ where $\mathcal{V}_1 = \mathcal{V}_2 = \mathcal{V}$ and $\mathcal{Y}_1 = \mathcal{Y}_2 = \mathcal{Y}$:*

- *There exist two neighbor-respecting refined distributions W and U, of $\mathsf{Exec}^{\mathsf{M}}(x)$ and $\mathsf{Exec}^{\mathsf{M}}(x')$ respectively, both defined over $\mathcal{V}_1 \times \mathcal{Y}_1 \times \mathcal{V}_2 \times \mathcal{Y}_2$ and w.r.t. $\sim_1$;*
- *W and U are (ϵ, δ)-close.*

We prove the following theorem that shows the equivalence of NPDO and symmetric NPDO — this is the most non-trivial step in proving the NPDO advanced composition theorem.

Theorem 3.4 (Equivalence of NPDO and symmetric NPDO)**.** *Suppose that an algorithm $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ has finite view and output spaces. Then, M satisfies (ϵ, δ)-NPDO w.r.t. input relation $\sim_0$ and output relation $\sim_1$ iff it satisfies (ϵ, δ)-symmetric-NPDO w.r.t. $\sim_0$ and $\sim_1$.*

Proof. We provide an informal proof roadmap below and the full proof is deferred to Appendix A.2. □

Proof roadmap. The direction that symmetric NPDO implies NPDO is easy. The other direction, i.e., NPDO implies symmetric NPDO, is much more involved to prove. We explain the high-level intuition of our proof (Appendix A.2) below.

Two arbitrary neighbor-respecting refined distributions W and U may not be (ϵ, δ)-close. Instead, the idea is to consider the pair (ϵ, δ)-matchings w and u induced by W and U and gradually adjust the weights of w and u, until for every "edge" $a \sim b$, the weights are close to each other, i.e., $e^{-\epsilon}u(a,b) \leq w(a,b) \leq e^{\epsilon}u(a,b)$. Then, we convert the final pair of matchings to a pair of refined distributions W' and U' and then W' and U' are (ϵ, δ)-close.

The adjustment algorithm goes as follows. For each step, we first find all the "violating" edge in the bipartite graph, such that if $w(a,b) > e^{\epsilon}u(a,b)$, we assign a direction $(b \to a)$ to the edge, and vice versa. The idea is that we start with any violating edge, say $b \to a$. It means that the weight of $u(a,b)$ is too small and we try to increase it. When we gradually increase $u(a,b)$, either the violating edge will be removed, or the constraint of u on b will become tight. We prove that in the latter case, we can always find a new violating edge $b \to a'$ to be adjusted. We now try to adjust the weight $u(a,b), u(a',b)$ simultaneously (we want to increase $u(a,b)$ and decrease $u(a',b)$). We again adjust them until one violating edge is removed or the constraint on a' is tight (the constraint on b will be preserved because we are increasing and decreasing the same amount on $u(a,b), u(a',b)$). We again prove that there exists some violating edge $(a' \to b')$ in the latter case. The algorithm continues that before we find a loop, each time we will adjust all the visited violated edge simultaneously, until we can remove one violating edge, or all the constraints over the vertices of those edges are tight. In the latter case, we can always find a new violating edge and repeat the process. Finally, if we find a loop, we can always find the minimum adjustment required on the loop to remove at least one violating edge and adjust all edges on the loop correspondingly, which guarantees to remove one violating edge.

For this proof to work, we need the view and output spaces to be finite[1] We can see that the adjustment algorithm goes in steps. To ensure the algorithm stops in finite steps, we require the mechanism's space to be finite, such that there's no path consisting of violating edges of infinite length. Also, since each step only guarantees to remove one violating edge, we may need infinite step to remove all the violating edges if there are infinitely many of them.

4 Advanced Composition for (ϵ, δ)-NPDO

Given Theorem 3.4, to prove the advanced composition theorem for NPDO (Theorem 1.4), we can instead prove it for symmetric NPDO. Since symmetric NPDO is expressed in the form of max divergence over two distributions, we will be able to rely on standard tools to reason about the distance between composed distributions. Specifically, we will make use of the advanced composition theorem for standard DP which we define below.

4.1 Additional Preliminary

Theorem 4.1 (Advanced k-fold adaptive composition theorem for (ϵ, δ)-DP [DRV10])**.** *Suppose mechanisms $\mathsf{M}_1, \ldots, \mathsf{M}_k$ are (ϵ, δ)-DP and we consider the k-fold adaptive composition $\mathsf{M} = \mathsf{M}_k \circ \mathsf{M}_{k-1} \circ \cdots \circ \mathsf{M}_1$. Then, for all $\delta' > 0$, M is $(\epsilon', k\delta + \delta')$-DP, where $\epsilon' = \epsilon\sqrt{2k \ln \frac{1}{\delta'}} + 2k\epsilon^2$.*

[1] In fact, if we assume the axiom of choice, the bipartite graph can be infinite as long as each connected component of the bipartite graph is finite. We call this case locally finite.

4.2 Proof of Theorem 1.4

With all the tools prepared, we now present the proof to the advanced composition theorem for (ϵ, δ)-NPDO. Below, we set up the appropriate random experiments such that we can eventually leverage the standard advanced composition of DP to complete the proof.

Theorem 4.2 (Advanced composition theorem for NPDO: restatement of Theorem 1.4)**.** *Let $\epsilon \geq 0$, $\delta, \delta' \in [0,1]$ and $k \geq 2$. Suppose for $i \in [k]$, the algorithm $\mathsf{M}_i : \mathcal{Y}_{i-1} \to \mathcal{Y}_i$ with view space $\mathcal{V}_i$ is (ϵ, δ)-NPDO with respect to the neighboring relations in its input and output spaces. Suppose that all algorithms have finite output and view spaces. Then, the composition $\mathsf{M} = \mathsf{M}_k \circ \mathsf{M}_{k-1} \circ \cdots \circ \mathsf{M}_1$ is $(\epsilon', \delta' + k\delta)$-NPDO, where $\epsilon' = \epsilon\sqrt{2k \ln \frac{1}{\delta'}} + 2k\epsilon^2$.*

Proof. To avoid excessive notation uses, we consider a simpler case where all mechanisms $\mathsf{M}_1, \ldots, \mathsf{M}_k$ shares the same view space $\mathcal{V}$, same input/output space $\mathcal{Y}$ and same input/output relation $\sim$. The following proof can be easily generalized to the case in the original theorem statement.

Fix a pair of input y_0, y_0' for the k-fold composition mechanism M. The experiment $\mathsf{Exec}^{\mathsf{M}}(y_0)$ is described below, and $\mathsf{Exec}^{\mathsf{M}}(y_0')$ is defined similarly.

- For $i = 1, \ldots, k$, $(v_i, y_i) \leftarrow \mathsf{Exec}^{\mathsf{M}_i}(y_{i-1})$, where v_i is the view and y_i is the i-th intermediate output.
- The experiment outputs $(v_1, \ldots, v_k, y_k)$, where $v_1, \ldots, v_k$ are the views observed by the adversary and y_k is the final output.

Recall that in the symmetric NPDO definition 3.3, we need to find two statistically close refined distributions W and U for $\mathsf{Exec}^{\mathsf{M}}(y_0)$ and $\mathsf{Exec}^{\mathsf{M}}(y_0')$, correspondingly. We now construct W as follows.

- For $i = 1, \ldots, k$:
  - Given a neighboring pair $y_{i-1} \sim y_{i-1}'$, there's a pair of (ϵ, δ)-close refined distribution, $W_i^{y_{i-1}, y_{i-1}'}, U_i^{y_{i-1}, y_{i-1}'}$, such that $W_i^{y_{i-1}, y_{i-1}'}$ are refined distributions for $\mathsf{Exec}^{\mathsf{M}_i}(y_{i-1})$ and $\mathsf{Exec}^{\mathsf{M}_i}(y_{i-1}')$, correspondingly.
  - Sample $((v_i, y_i), (v_i, y_i'))$ from $W_i^{y_{i-1}, y_{i-1}'}$.
- Output $((v_1, \ldots, v_k, y_k), (v_1, \ldots, v_k, y_k'))$.

We can construct U similarly, where for each step $i \in [k]$, we sample $((v_i, y_i), (v_i, y_i'))$ from $U_i^{y_{i-1}, y_{i-1}'}$ instead. Notice to make the definitions consistent, given any pair of neighboring inputs $y_{i-1} \sim y_{i-1}'$, the pair of refined distributions $W_i^{y_{i-1}, y_{i-1}'}, U_i^{y_{i-1}, y_{i-1}'}$ will be a unique pair of refined distributions.

Now let's check W's marginal distribution over the variables $(v_1, \ldots, v_k, y_k)$ is exactly $\mathsf{Exec}^{\mathsf{M}}(y_0)$: for $i \in [k]$, $W_i^{y_{i-1}, y_{i-1}'}$'s marginal distribution over the variables (v_i, y_i) is exactly $\mathsf{Exec}^{\mathsf{M}_i}(y_{i-1})$, which is the same as the original random experiment of $\mathsf{Exec}^{\mathsf{M}}(y_0)$. Similarly, U's marginal distribution over the variables $(v_1', \ldots, v_k', y_k')$ is exactly $\mathsf{Exec}^{\mathsf{M}}(y_0')$.

The final step is to prove that W and U are $(\epsilon', \delta' + k\delta)$-close. In fact, W can be considered as a post-processed distribution of the joint distribution of $\left(W_1^{y_0, y_0'}, W_2^{y_1, y_1'}, \ldots, W_k^{y_{k-1}, y_{k-1}'}\right)$. Here, for $i \in \{2, \ldots, k\}$, each $W_i^{y_{i-1}, y_{i-1}'}$ is adaptively dependent on the sampling result of the previous

step $W_{i-1}^{y_{i-2},y'_{i-2}}$. Similarly, U can be considered as a post-processed distribution of the joint distribution of $\left(U_1^{y_0,y'_0}, U_2^{y_1,y'_1}, \ldots, U_k^{y_{k-1},y'_{k-1}}\right)$. Now, for each $i \in [k]$, we know $W_i^{y_{i-1},y'_{i-1}}$ and $U_{i-1}^{y_{i-2},y'_{i-2}}$ are (ϵ, δ)-close. Therefore, applying the k-fold composition theorem from standard (ϵ, δ)-DP (Theorem 4.1) and we get that W and U are $(\epsilon', \delta' + k\delta)$-close for $\epsilon' = \epsilon\sqrt{2k \ln \frac{1}{\delta'}} + 2k\epsilon^2$ and any $\delta' > 0$. □

Remark 4.3. Since our proof provides a reduction to the standard k-fold composition theorem of (ϵ, δ)-DP, one can have tighter bound on the privacy parameters based on tighter analyses of k-fold DP composition (e.g. Kairouz et al. [KOV15] and Murtagh et al. [MV16]).

5 Defining NPDO with a General Divergence Notion

In Section 3, we saw that (ϵ, δ)-NPDO can be defined in terms of D^δ divergence over neighbor-respecting refined distributions. In the literature, other notions of divergence have been considered, such as Rényi divergence [Mir17, BS16], trade-off function [DDR20, VZ23], and others [DRS22].

In this section, we will generalize differential obliviousness to other notions of divergence.

Divergence notion. Suppose G is some divergence notion for comparing two distributions on the same sample space. Suppose ϵ is some parameter that indicates divergence, where smaller means the two distributions are closer. Then, we use the notation

$$\mathsf{G}(P \| Q) \leq \epsilon$$

to mean two distributions P and Q are ϵ-close under G. In general, G does not have to be symmetric and it may return either a single value or a function. For example, δ-max divergence $\mathsf{D}^\delta(P||Q)$ and Rényi divergence (see Definition 6.1) return a single value, whereas the $\mathsf{Pow}(\cdot||\cdot)$ function (see Definition 6.6 and Equation (6)) returns a function. In the case $\mathsf{G}(P\|Q)$ and ϵ are functions, then $\leq$ means that less than or equal to at every point.

For a divergence notion G to be used in a privacy definition, we want it to be well-formed in the sense that the following data processing inequality should be satisfied.

Definition 5.1 (Data processing inequality)**.** *A divergence notion G satisfies the data processing inequality if given any two joint distributions (X, Y) and (X', Y') on the same space, the corresponding marginal distributions satisfy the monotone property:*

$$\mathsf{G}(X \| X') \leq \mathsf{G}((X, Y) \| (X', Y')).$$

One way to generalize DO to general divergence notions is to extend the original DO notion of Chan et al. [CCMS19], resulting in the following definition.

Definition 5.2 (Generic DO)**.** *Let $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ be an algorithm with input relation $\sim_0$ and output relation $\sim_1$, let G be a divergence notion, and let ϵ be some corresponding divergence parameter. We say that M is ϵ-DO(G) if for all neighboring inputs $x \sim_0 x'$ from $\mathcal{X}$, the distributions $\mathsf{View}^{\mathsf{M}}(x)$ and $\mathsf{View}^{\mathsf{M}}(x')$ are close in the following sense:*

$$\mathsf{G}(\mathsf{View}^{\mathsf{M}}(x) \| \mathsf{View}^{\mathsf{M}}(x')) \leq \epsilon.$$

The above generic DO notion, however, suffers from the same limitation as the original DO notion [CCMS19] — it is not amenable to composition.

Since we care about composition, we will instead generalize the (ϵ, δ)-NPDO notion to more general notions of divergence. Generalizing with the original NPDO definition of Zhou et al. [ZSCM23] is unnatural since the NPDO notion is not defined as a standard divergence notion over two distributions due to the $\mathcal{N}(\cdot)$ operator in Equation (2). Fortunately, to prove the advanced theorem for NPDO, we defined an equivalent notion called symmetric NPDO, and symmetric NPDO is indeed defined as the divergence over a pair of neighbor-respecting refined distributions. Therefore, we can extend the symmetric NPDO notion when generalizing to other divergence notions. The resulting notion is called generic NPDO as defined below.

Definition 5.3 (Generic NPDO)**.** *Let $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ be a randomized algorithm with view space $\mathcal{V}$, input relation $\sim_0$ and output relation $\sim_1$ (which can be extended to the execution space $\mathcal{V} \times \mathcal{Y}$). Let G be a divergence notion, and ϵ be some corresponding divergence parameter.*

Then, we say that M is ϵ-NPDO(G) if for all neighboring inputs $x \sim_0 x'$, there exists a pair of neighbor-respecting refined distributions W and U of $\mathsf{Exec}^{\mathsf{M}}(x)$ and $\mathsf{Exec}^{\mathsf{M}}(x')$, respectively, such that the following holds:

$$\mathsf{G}(W \| U) \leq \epsilon. \tag{5}$$

Additional notations: neighbor-respecting refinement pairing. Since we always use a pair of neighbor-respecting refines distributions together (e.g., Definitions 3.3 and 5.3), it helps to introduce some new notations.

Henceforth, we will use the notation $\varphi = (W, U)$ to denote a pair of neighbor-respecting refined distributions of $\mathsf{Exec}^{\mathsf{M}}(x)$ and $\mathsf{Exec}^{\mathsf{M}}(x')$, respectively, and we say that $\varphi = (W, U)$ is a neighbor-respecting refinement pairing between $\mathsf{Exec}^{\mathsf{M}}(x)$ and $\mathsf{Exec}^{\mathsf{M}}(x')$. Given such a neighbor-respecting refinement pairing, we use the notation $\varphi(\mathsf{Exec}^{\mathsf{M}}(x)) = W$ and $\varphi(\mathsf{Exec}^{\mathsf{M}}(x')) = U$.

We also introduce the following notation to denote the divergence of a neighbor-respecting refinement pairing φ:

$$\mathsf{G}^{\varphi}(\mathsf{Exec}^{\mathsf{M}}(x) \| \mathsf{Exec}^{\mathsf{M}}(x')) = \mathsf{G}(\varphi(\mathsf{Exec}^{\mathsf{M}}(x)) \| \varphi(\mathsf{Exec}^{\mathsf{M}}(x')))$$

Using this notation, we can rephrase Equation (5) as:

$$\mathsf{G}^{\varphi}(\mathsf{Exec}^{\mathsf{M}}(x) \| \mathsf{Exec}^{\mathsf{M}}(x')) \leq \epsilon.$$

6 NPDO Composition for Various Divergence Notions

6.1 Rényi NPDO and Zero Concentrated NPDO

Rényi NPDO. Rényi DP [Mir17] of order α is based on the Rényi divergence of order α defined as:

$$\mathsf{D}_\alpha(P \| Q) = \frac{1}{\alpha - 1} \log \left(E_P \left(\frac{p}{q} \right)^{\alpha - 1} \right) = \frac{1}{\alpha - 1} \log \int p^\alpha q^{1-\alpha}.$$

A mechanism M is (α, ϵ)-RDP if for all neighboring input pairs x and x', $\mathsf{D}_\alpha(\mathsf{M}(x) \| \mathsf{M}(x')) \leq \epsilon$. We can analogously define (α, ϵ)-Rényi NPDO by instantiate the generic NPDO notion as ϵ-NPDO(D_α), i.e., plugging the D_α divergence into the generic NPDO definition in 5.3:

Definition 6.1 ((α,ϵ)-Rényi NPDO)**.** *An algorithm $\mathsf{M}:\mathcal{X}\to\mathcal{Y}$ is (α,ϵ)-Rényi NPDO w.r.t input neighboring relation $\sim_0$ and output neighboring relation $\sim_1$ iff for any neighboring input pair $x\sim_0 x'$ from $\mathcal{X}$, there exists a refinement pair $\varphi(x,x')$ w.r.t. $\sim_1$ such that*

$$\mathsf{D}_\alpha^{\varphi(x,x')}(\mathsf{Exec}^{\mathsf{M}}(x)\|(\mathsf{Exec}^{\mathsf{M}}(x'))\le\epsilon.$$

Rényi divergence has the following composition theorem:

Lemma 6.2 ([Mir17])**.** *Suppose distributions X and X' on sample space $\mathcal{X}$ satisfy that $\mathsf{D}_\alpha(X\|X')\le\epsilon_1$. Also, for any $x\in\mathcal{X}$, distributions $Y(x)$ and $Y'(x)$ on sample space $\mathcal{Y}$ satisfy that $\mathsf{D}_\alpha(Y(x)\|Y'(x))\le\epsilon_2$. Then,*

$$\mathsf{D}_\alpha\left((X,Y(X))\|(X',Y(X'))\right)\le\epsilon_1+\epsilon_2.$$

Based on this lemma, we can prove the composition theorem for (α,ϵ)-Rényi NPDO.

Theorem 6.3 (Composition theorem for (α,ϵ)-Rényi NPDO)**.** *Suppose $\mathsf{M}_1:\mathcal{X}\to\mathcal{Y}$ is (α,ϵ_1)-Rényi NPDO with respect to the neighboring relations $\sim_0$ in its input and $\sim_1$ in output spaces. Suppose $\mathsf{M}_2:\mathcal{Y}\to\mathcal{Z}$ is (α,ϵ_2)-Rényi NPDO with respect to the neighboring relations $\sim_1$ in its input and $\sim_2$ in output spaces. Then, the composition $\mathsf{M}=\mathsf{M}_2\circ\mathsf{M}_1$ is $(\alpha,\epsilon_1+\epsilon_2)$-Rényi NPDO w.r.t $\sim_0$ and $\sim_2$.*

Proof. Let $\mathcal{V}_1$ and $\mathcal{V}_2$ be the view space for M_1 and M_2, respectively. Fix a pair of neighboring intpus $x\sim_0 x'\in\mathcal{X}$ for M. Our final goal is define a neighbor-respecting refinement pairing φ (depending on the ordered pair (x,x')) between the two execution distributions $\mathsf{Exec}^{\mathsf{M}_2\circ\mathsf{M}_1}(\cdot)$ on x and x' satisfying Definition 5.3.

Observe that both execution distributions are on the sample space $\mathcal{V}_1\times\mathcal{V}_2\times\mathcal{Z}$. Note that, in our extended relation, neighboring tuples must have the same element in the view space; this let us slightly simplify the notation when we consider neighbor-respecting refinements.

Because M_1 is (α,ϵ_1)-Rényi NPDO, there exists a neighbor-respecting refinement pairing $\mu(\mathsf{Exec}^{\mathsf{M}_1}(x))$ and $\mu(\mathsf{Exec}^{\mathsf{M}_1}(x'))$, which are two distributions on $(\mathcal{V}_1\times\mathcal{Y})\times(\mathcal{V}_1'\times\mathcal{Y}')$, such that

$$\mathsf{D}_\alpha^{\mu}(\mathsf{Exec}^{\mathsf{M}_1}(x)\|\mathsf{Exec}^{\mathsf{M}_1}(x'))\le\epsilon_1.$$

For any $y\sim_1 y'$, because M_2 is (α,ϵ_2)-Rényi NPDO, there exists a neighbor-respecting refinement pairing $\sigma^{(y,y')}(\mathsf{Exec}^{\mathsf{M}_2}(y))$ and $\sigma^{(y,y')}(\mathsf{Exec}^{\mathsf{M}_2}(y'))$ which are two distributions on $(\mathcal{V}_2\times\mathcal{Z})\times(\mathcal{V}_2'\times\mathcal{Z}')$, such that

$$\mathsf{D}_\alpha^{\sigma^{(y,y')}}(\mathsf{Exec}^{\mathsf{M}_2}(y)\|\mathsf{Exec}^{\mathsf{M}_2}(y'))\le\epsilon_2.$$

Now let's define the refinement pair $\varphi(\mathsf{Exec}^{\mathsf{M}_2\circ\mathsf{M}_1}(x))$ and $\varphi(\mathsf{Exec}^{\mathsf{M}_2\circ\mathsf{M}_1}(x'))$, which are two distributions over $(\mathcal{V}_1\times\mathcal{V}_2\times\mathcal{Z})\times(\mathcal{V}_1'\times\mathcal{V}_2'\times\mathcal{Z}')$. We highlight the difference of them with underlining.

- $\varphi(\mathsf{Exec}^{\mathsf{M}_2\circ\mathsf{M}_1}(x))$:
  - Sample $((v_1,y),(v_1,y'))$ from $\underline{\mu(\mathsf{Exec}^{\mathsf{M}_1}(x))}$.
  - Based on y,y', sample $((v_2,z),(v_2,z'))$ from $\underline{\sigma^{(y,y')}(\mathsf{Exec}^{\mathsf{M}_2}(y))}$.
  - Output $((v_1,v_2,z),(v_1,v_2,z'))$.
- $\varphi(\mathsf{Exec}^{\mathsf{M}_2\circ\mathsf{M}_1}(x'))$:
  - Sample $((v_1,y),(v_1,y'))$ from $\underline{\mu(\mathsf{Exec}^{\mathsf{M}_1}(x'))}$.

– Based on y, y', sample $((v_2, z), (v_2, z'))$ from $\underline{\sigma^{(y,y')}(\mathsf{Exec}^{\mathsf{M}_2}(y'))}$.
– Output $((v_1, v_2, z), (v_1, v_2, z'))$.

We verify that the marginal distribution of $\varphi(\mathsf{Exec}^{\mathsf{M}_2 \circ \mathsf{M}_1}(x))$ over $\mathcal{V}_1 \times \mathcal{V}_2 \times \mathcal{Z}$ is exactly $\mathsf{Exec}^{\mathsf{M}_2 \circ \mathsf{M}_1}(x)$:

1. The marginal distribution of $\mu(\mathsf{Exec}^{\mathsf{M}_1}(x))$ over the variables (v_1, y) is exactly $\mathsf{Exec}^{\mathsf{M}_1}(x)$;
2. Conditioned on all pair of y, y', the marginal distribution of $\sigma^{(y,y')}(\mathsf{Exec}^{\mathsf{M}_2}(y))$ over the variables (v_2, z) is exactly $\mathsf{Exec}^{\mathsf{M}_2}(y)$.

Similarly, the marginal distribution of $\varphi(\mathsf{Exec}^{\mathsf{M}_2 \circ \mathsf{M}_1}(x'))$ over $\mathcal{V}_1' \times \mathcal{V}_2' \times \mathcal{Z}'$ is exactly $\mathsf{Exec}^{\mathsf{M}_2 \circ \mathsf{M}_1}(x')$. Finally, we have

$$\begin{aligned}
&\mathsf{D}_\alpha(\varphi(\mathsf{Exec}^{\mathsf{M}_2 \circ \mathsf{M}_1}(x)) \| \varphi(\mathsf{Exec}^{\mathsf{M}_2 \circ \mathsf{M}_1}(x'))) \\
\le &\mathsf{D}_\alpha\left(\left(\mu(\mathsf{Exec}^{\mathsf{M}_1}(x)), \sigma^{(y,y')}(\mathsf{Exec}^{\mathsf{M}_2}(y))\right) \| \left(\mu(\mathsf{Exec}^{\mathsf{M}_1}(x')), \sigma^{(y,y')}(\mathsf{Exec}^{\mathsf{M}_2}(y'))\right)\right) \\
& \hfill \text{(data processing inequality)} \\
\le &\epsilon_1 + \epsilon_2. \hfill \text{(Lemma 6.2)}
\end{aligned}$$

□

Zero Concentrated NPDO. Zero concentrated DP (zCDP) [BS16] is defined based on the following divergence:

$$\mathsf{D}_z(P \| Q) := \sup_{\alpha > 1} \frac{1}{\alpha} \cdot \mathsf{D}_\alpha(P \| Q).$$

A mechanism M is ϵ-zCDP if for all neighboring inputs x and x', $\mathsf{D}_z(\mathsf{M}(x) \| \mathsf{M}(x')) \le \epsilon$. We can analogously define ϵ-zNPDO by plugging D_z in the generic definition 5.3 as follows.

Definition 6.4 (ϵ-zNPDO)**.** *A mechanism $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ with view space V is ϵ-zNPDO w.r.t input neighboring relation $\sim_0$ and output neighboring relation $\sim_1$ if for any neighboring input pair $x \sim_0 x'$ from $\mathcal{X}$, there exists a refinement pair $\varphi(x, x')$ w.r.t $\sim_1$ such that*

$$\mathsf{D}_z^{\varphi(x,x')}(\mathsf{Exec}^{\mathsf{M}}(x) \| (\mathsf{Exec}^{\mathsf{M}}(x')) \le \epsilon.$$

Based on nearly the same proof as Theorem 6.3, we can have the following composition theorem:

Theorem 6.5 (Composition theorem for ϵ-zNPDO)**.** *Suppose $\mathsf{M}_1 : \mathcal{X} \to \mathcal{Y}$ is ϵ_1-zNPDO with respect to the neighboring relations $\sim_0$ in its input and $\sim_1$ in output spaces. Suppose $\mathsf{M}_2 : \mathcal{Y} \to \mathcal{Z}$ is ϵ_2-zNPDO with respect to the neighboring relations $\sim_1$ in its input and $\sim_2$ in output spaces. Then, the composition $\mathsf{M} = \mathsf{M}_2 \circ \mathsf{M}_1$ is $(\epsilon_1 + \epsilon_2)$-zNPDO w.r.t $\sim_0$ and $\sim_2$.*

6.2 g-NPDO

Dong et al. [DRS22] defined f-DP which uses tradeoff functions to characterize the divergence between distributions. We will define an analogous notion for differential obliviousness calld g-NPDO. As mentioned later in Section 6.3 and in Section 6.4, we can view composition of Gaussian-NPDO and (ϵ, δ)-NPDO as special cases of g-NPDO.

6.2.1 Background: From Tradeoff Functions to Power Functions

Our g-NPDO notion will use power functions to characterize divergence. To understand the power function, we first review the notion of a tradeoff function.

Tradeoff function. Given two distributions H_0 and H_1, the tradeoff function $\mathsf{T}(H_0 \| H_1)$ outputs a function that captures the tradeoff curve between two types of errors.

Since our divergence measure will be a function, it helps to define a partial order for two functions. Let f_1 and f_2 be two functions, then $f_1 \leq f_2$ means that for all $x \in [0,1]$, $f_1(x) \leq f_2(x)$.

More formally, tradeoff functions are inspired by hypothesis testing.

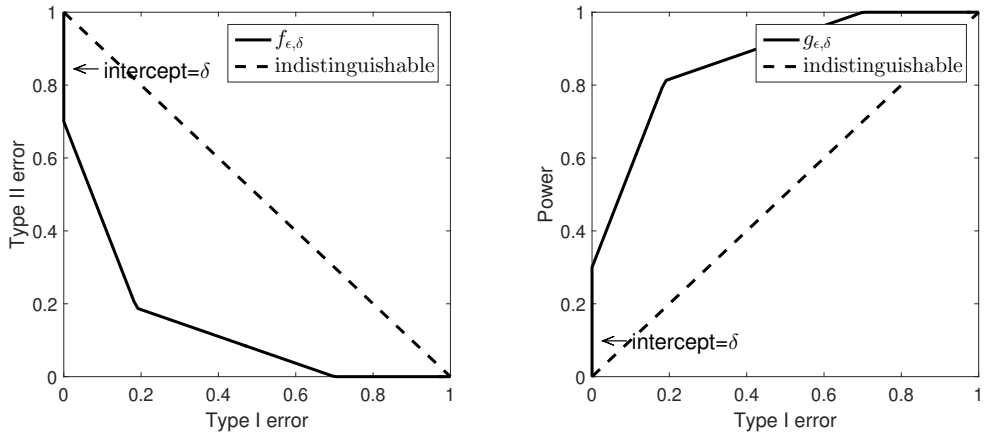
Definition 6.6 (Tradeoff Function)**.** *Suppose H_0 and H_1 are distributions on the same sample space, where H_0 is interpreted as the null hypothesis and H_1 as the alternate hypothesis. Define the tradeoff function $\mathsf{T}(H_0 \| H_1) : [0,1] \to [0,1]$ such that given a value $x \in [0,1]$, the function returns the probability of Type-2 error (accepting H_1) for the most powerful (randomized) test that has the probability of Type-1 error (rejecting H_0) being exactly x.*

A tradeoff function $\mathsf{T}(H_0 || H_1)$ measures the *similarity* between two distributions. A larger function function means the two distributions are closer. The class of tradeoff functions has a maximal element $x \mapsto 1 - x$, which corresponds to two identical distributions (corresponding to the diagonal line in Figure 1(a)). Dong et al. [DRS22] provide more detailed explanation about the intuition behind Definition 6.6.

Example: tradeoff functions for (ϵ, δ)-closeness. Figure 1(a) shows the tradeoff function:

$$f_{\epsilon,\delta}(x) := \max\{0, e^{-\epsilon} \cdot (1 - \delta - x), -e^{\epsilon} \cdot x + 1 - \delta\}.$$

Wasserman and Zhou [WZ10] showed that if two distributions H_0 and H_1 are (ϵ, δ)-close, then the tradeoff function $\mathsf{T}(H_0 || H_1)$ has to be above $f_{\epsilon,\delta}$, but below $y = 1 - x$.



(a) $f_{\epsilon,\delta}$ (b) $g_{\epsilon,\delta}$

Figure 1: Examples: tradeoff functions and power functions for (ϵ, δ)-closeness. Here $\epsilon = 1, \delta = 0.3$. (a) is the curve for $f_{\epsilon,\delta}$ and (b) is the curve for $g_{\epsilon,\delta}$.

From tradeoff functions to power functions. We want to use $\mathsf{T}(\cdot||\cdot)$ which outputs a tradeoff function as a divergence notion, but directly using it is unnatural because a larger tradeoff function means closer in distance. We therefore flip the sign and define a related notion called a power function denoted $\mathsf{Pow}(H_0\|H_1)$:

$$\mathsf{Pow}(H_0\|H_1) := 1 - \mathsf{T}(H_0\|H_1) \tag{6}$$

We can use $\mathsf{Pow}(\cdot||\cdot)$ as the corresponding the divergence notion which outputs a power function when given two distributions.

With power functions, a smaller function means closer in distance which is more intuitive. The identity function is the minimal element over all power functions (see the diagonal line in Figure 1(b)).

The power function also has a natural interpretation in the context of hypothesis testing. Specifically, the *power* of a test refers to one minus the Type-2 error.

Example: power functions for (ϵ, δ)-closeness. In Figure 1(b), we see that the corresponding power function for $f_{\epsilon,\delta}$ is:

$$g_{\epsilon,\delta}(x) := \max\{1, 1 - e^{-\epsilon}(1 - \delta - x), e^{\epsilon} \cdot x + \delta\}. \tag{7}$$

Hence, if two distributions H_0 and H_1 are (ϵ, δ)-close, then the power function $\mathsf{Pow}(H_0||H_1)$ has to be below $g_{\epsilon,\delta}$, but above the diagonal line $y = x$.

Another way to interpret the power function is to think of it as the fractional knapsack problem — see the remark below.

Remark 6.7 (Intuition: power function as the fractional knapsack problem [Kad68])**.** Given distributions H_0 and H_1 on some sample space Ω and a value $\alpha \in [0, 1]$, the power function $\mathsf{Pow}(H_0\|H_1)(x)$ can be interpreted as an instance of the *fractional knapsack problem* as follows:

- Each element $\omega \in \Omega$ has weight $H_0(\omega)$ and reward $H_1(\omega)$.
- The parameter x gives the capacity constraint on the weight.
- The value $\mathsf{Pow}(H_0\|H_1)(x)$ is the maximum achievable reward subject to the capacity constraint, where an item may be selected fractionally.

  (The fractional feature is only relevant when the sample space Ω is discrete.)

In fact, the Neyman-Pearson lemma [NP33] can be interpreted as the classical observation that the the maximum reward-to-weight ratio heuristic can solve the fractional knapsack problem optimally.

With the above fractional knapsack interpretation, we can think of the divergence notion as follows. When two distributions H_0 and H_1 are the same, it is clear that given any capacity constraint x, the maximum reward is also x; this means the power function is exactly the identity function. However, if the two distributions are very different, this means that there are items whose reward-to-weight ratios are large; hence, in this case, the power function can initially grow faster than the identity function.

6.2.2 Properties of Power Functions

Naturally, all natural properties of tradeoff functions [DRS22] can be expressed equivalently in terms of power functions, albeit with some slight modifications. Unless otherwise stated, we will mostly work with power functions, and quote counterparts from [DRS22] where appropriate.

Fact 6.8 (Valid power functions [DRS22, Proposition 2.2])**.** *Suppose $g : [0,1] \to [0,1]$ is a function. Then, there exist distributions X and Y such that $\mathsf{Pow}(X\|Y) = g$* iff *g is continuous, concave and $g(x) \geq x$ for all $x \in [0,1]$.*

Proof. (Sketch) One can take X to be the uniform distribution on $[0,1]$ and Y to be the distribution on $[0,1]$ with g as the CDF. □

Ubiquity of power functions. Intuitively, the ubiquity of the power function comes from the following fact: the power function summarizes the "difference" between two distributions, and it contains enough information such that any well-formed divergence between the distributions can be computed solely based on the power function. Here, well-formed divergence means the divergence satisfy the data processing inequality in Definition 5.1, which should be a basic property for all divergence notions used to define privacy.

Fact 6.9 (Ubiquity of power functions [DRS22, Proposition B.1])**.** *Suppose a divergence notion $\mathsf{G}(\cdot\|\cdot)$ satisfies the data processing inequality, then there exists a functional ℓ_{G} such that $\mathsf{G}(X\|Y) = \ell_{\mathsf{G}}(\mathsf{Pow}(X\|Y))$.*

Tensor product and composition. As remarked in [DRS22], the tensor product gives a complete characterization of many known DP compositions, which eventually allows us to transfer abundant composition result for normal DP to NPDO.

Definition 6.10 (Tensor product)**.** *Given any two power functions g_1, g_2 such that there exist some distributions X, X', Y, Y' where $g_1 = \mathsf{Pow}(X\|X')$ and $g_2 = \mathsf{Pow}(Y\|Y')$, the tensor product of g_1, g_2 is defined as*

$$g_1 \otimes g_2 := \mathsf{Pow}(X \times Y \| X' \times Y'),$$

where the product distribution $X \times Y$ means that sampling from X and Y independently.

Moreover, the tensor product $g_1 \otimes g_2$ is well-defined that $g_1 \otimes g_2$ is the same for any such distributions X, X', Y, Y'.

Theorem 6.11 (Adaptive composition for power functions [DRS22])**.** *Let g_1 and g_2 be two power functions. Suppose distributions X and X' on sample space $\mathcal{X}$ satisfy that $\mathsf{Pow}(X\|X') \leq g_1$. Also, for any $x \in \mathcal{X}$, distributions $Y(x)$ and $Y'(x)$ on sample space $\mathcal{Y}$ satisfy that $\mathsf{Pow}(Y(x)\|Y'(x)) \leq g_2$. Then,*

$$\mathsf{Pow}\left((X, Y(X))\|(X', Y(X'))\right) \leq g_1 \otimes g_2.$$

6.2.3 g-NPDO and Composition

Definition 6.12 (g-NPDO)**.** *Given a power function g, an algorithm $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ is said to be g-NPDO w.r.t. its input and output relations iff it is g-NPDO(Pow) by Definition 5.3 w.r.t. its input and output relations.*

Theorem 6.13 (Composition theorem for g-NPDO)**.** *Let $\mathsf{M}_1 : \mathcal{X} \to \mathcal{Y}$ and $\mathsf{M}_2 : \mathcal{Y} \to \mathcal{Z}$ be two randomized algorithms. Suppose that for $i \in \{1,2\}$, M_i is g_i-NPDO w.r.t. its corresponding input and output neighboring relations. Then, the composed algorithm $\mathsf{M}_2 \circ \mathsf{M}_1 : \mathcal{X} \to \mathcal{Z}$ is $g_1 \otimes g_2$-NPDO.*

Proof. Fix a pair of inputs $x \sim_0 x'$. The refinement pair φ is defined exactly the same as the proof of Theorem 6.3. The remaining proof is nearly the same as in Theorem 6.3 except that in the final step, we use Theorem 6.11:

$$\begin{aligned}
&\mathsf{Pow}(\varphi(\mathsf{Exec}^{\mathsf{M}_2\circ\mathsf{M}_1}(x))\|\varphi(\mathsf{Exec}^{\mathsf{M}_2\circ\mathsf{M}_1}(x')))\\
\le&\mathsf{Pow}\left(\left(\mu(\mathsf{Exec}^{\mathsf{M}_1}(x)),\sigma^{(y,y')}(\mathsf{Exec}^{\mathsf{M}_2}(y))\right)\parallel\left(\mu(\mathsf{Exec}^{\mathsf{M}_1}(x')),\sigma^{(y,y')}(\mathsf{Exec}^{\mathsf{M}_2}(y'))\right)\right)\\
&\hfill\text{(data processing inequality)}\\
\le&g_1\otimes g_2. \hfill\text{(Theorem 6.11)}
\end{aligned}$$

□

6.3 (ϵ, δ)-NPDO

Our g-NPDO composition theorem (Theorem 6.13) gives an alternative way to prove the advanced theorem for (ϵ, δ)-NPDO (Theorem 6.16). As mentioned, if two distributions A and B are (ϵ, δ)-close, then $\mathsf{Pow}(A||B) \le g_{\epsilon,\delta}$ where $g_{\epsilon,\delta}$ is defined in Equation (7).

Therefore, Theorem 6.16 follows directly from Theorem 6.13 and the following lemma (Lemma 6.14) which was described by Dong et al. [DRS22].

Lemma 6.14 ([DRS22])**.** *For all $\epsilon \ge 0$, $\delta, \delta' \in [0, 1]$ and $k \ge 2$,*

$$g_{\epsilon,\delta}^{\otimes k} \le g_{\epsilon',\delta'+k\delta}$$

where $\epsilon' = \epsilon\sqrt{2k\ln\frac{1}{\delta'}} + 2k\epsilon^2$.

6.4 Gaussian NPDO

In this section, we define Gaussian NPDO and prove a composition theorem for Gaussian NPDO. It turns out Gaussian NPDO composition can be viewed as a special case of g-NPDO composition.

Analogous to μ-Gaussian-DP [DRS22], we can define μ-Gaussian-NPDO as follows:

Definition 6.15 (μ-Gaussian NPDO)**.** *Let $N(\mu, \sigma^2)$ denote the Gaussian distribution with mean μ and variance σ^2. An algorithm $\mathsf{M} : \mathcal{X} \to \mathcal{Y}$ is μ-Gaussian NPDO w.r.t. its input and output relations iff it is g_μ-NPDO by Definition 6.12 where $g_\mu := \mathsf{Pow}(N(0, 1)\|N(\mu, 1))$.*

Example. In Figure 2, we depict the Gaussian power functions g_μ for different parameters μ and compare them with the power function representing (ϵ, δ)-closeness.

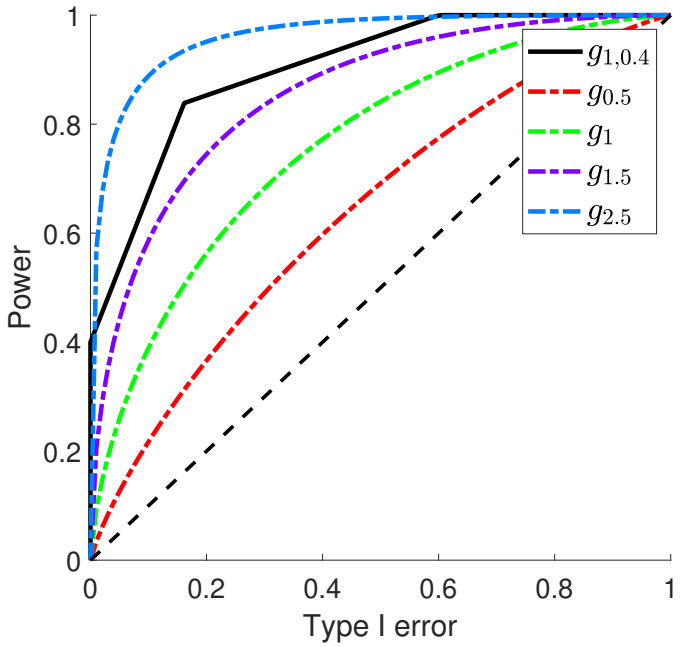


Figure 2: Comparing the power functions of Gaussian g_μ and (ϵ, δ)-closeness. Here, $\epsilon = 1, \delta = 0.4$. For smaller $\mu = 0.5, 1, 1.5$, g_μ is (ϵ, δ)-close; for larger $\mu = 2.5$, g_μ is not (ϵ, δ)-close. The line $y = x$ represents the power function for two identical distributions.

Theorem 6.16 (Composition theorem for μ-Gaussian NPDO)**.** *Suppose for $i \in [k]$, the algorithm $\mathsf{M}_i : \mathcal{Y}_{i-1} \to \mathcal{Y}_i$ with view space $\mathcal{V}_i$ is μ_i-Gaussian NPDO with respect to the neighboring relations in its input and output spaces. Then, the composition $\mathsf{M} = \mathsf{M}_k \circ \mathsf{M}_{k-1} \circ \cdots \circ \mathsf{M}_1$ is μ-Gaussian NPDO, where*

$$\mu = \sqrt{\mu_1^2 + \mu_2^2 + \cdots + \mu_k^2}.$$

Proof. It is a direct corollary of the g-NPDO composition theorem Theorem 6.13 and the composition theorem of the Gaussian power functions [DRS22]: for any $\mu_1, \ldots, \mu_k \geq 0$, $\otimes_{i \in [k]} g_{\mu_i} = g_\mu$ where $\mu = \sqrt{\mu_1^2 + \mu_2^2 + \cdots + \mu_k^2}$ □

Acknowledgments

This work is in part supported by a grant from ONR, a gift from Cisco, NSF awards under grant numbers CIF-1705007, 2128519, 2044679 and 1704788, and a Packard Fellowship. The work is also supported by DARPA SIEVE research program. T-H. Hubert Chan was partially supported by the Hong Kong RGC under the grants 17201220, 17202121 and 17203122.

References

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

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

[BCP15] Elette Boyle, Kai-Min Chung, and Rafael Pass. Oblivious parallel ram. In *Theory of Cryptography Conference (TCC)*, 2015.

[BGHP16] Gilles Barthe, Marco Gaboardi, Justin Hsu, and Benjamin Pierce. Programming language techniques for differential privacy. *ACM SIGLOG News*, 3(1):34–53, feb 2016.

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

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

[BS16] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *TCC (B1)*, volume 9985 of *Lecture Notes in Computer Science*, pages 635–658, 2016.

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

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

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

[CZJ+17] Ethan Cecchetti, Fan Zhang, Yan Ji, Ahmed E. Kosba, Ari Juels, and Elaine Shi. Solidus: Confidential distributed ledger transactions via PVORM. In *ACM CCS*, pages 701–717. ACM, 2017.

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

[DDR20] Jinshuo Dong, David Durfee, and Ryan Rogers. Optimal differential privacy composition for exponential mechanisms. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 2597–2606. PMLR, 2020.

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

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

[DRS22] Jinshuo Dong, Aaron Roth, and Weijie J Su. Gaussian differential privacy. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(1):3–37, 2022.

[DRV10] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *FOCS*, pages 51–60. IEEE Computer Society, 2010.

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

[FMT23] Vitaly Feldman, Audra McMillan, and Kunal Talwar. Stronger privacy amplification by shuffling for rényi and approximate differential privacy. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4966–4981. SIAM, 2023.

[FZ21] Vitaly Feldman and Tijana Zrnic. Individual privacy accounting via a renyi filter. *Advances in Neural Information Processing Systems*, 34:28080–28091, 2021.

[GDDa] Antonious M Girgis, Deepesh Data, and Suhas Diggavi. Shuffled model of differential privacy in federated learning. page 11.

[GDD+b] Antonious M. Girgis, Deepesh Data, Suhas Diggavi, Peter Kairouz, and Ananda Theertha Suresh. Shuffled model of federated learning: Privacy, communication and accuracy trade-offs.

[GGK+] Badih Ghazi, Noah Golowich, Ravi Kumar, Pasin Manurangsi, Rasmus Pagh, and Ameya Velingker. Pure differentially private summation from anonymous messages.

[GHL+14] Craig Gentry, Shai Halevi, Steve Lu, Rafail Ostrovsky, Mariana Raykova, and Daniel Wichs. Garbled ram revisited. In *EUROCRYPT*, pages 405–422, 2014.

[GKK+12] S. Dov Gordon, Jonathan Katz, Vladimir Kolesnikov, Fernando Krell, Tal Malkin, Mariana Raykova, and Yevgeniy Vahlis. Secure two-party computation in sublinear (amortized) time. In *ACM CCS*, pages 513–524. ACM, 2012.

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

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

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

[Kad68] Joseph B Kadane. Discrete search and the neyman-pearson lemma. *Journal of Mathematical Analysis and Applications*, 22(1):156–171, 1968.

[KOV15] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1376–1385. JMLR.org, 2015.

[KS21] Ilan Komargodski and Elaine Shi. Differentially oblivious turing machines. In *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 68:1–68:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[KTH+19] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. Privatesql: A differentially private sql query engine. *Proc. VLDB Endow.*, 12(11):1371–1384, jul 2019.

[LHH+15] Chang Liu, Michael Hicks, Austin Harris, Mohit Tiwari, Martin Maas, and Elaine Shi. Ghostrider: A hardware-software system for memory trace oblivious computation. In *ASPLOS*, 2015.

[LN18] Kasper Green Larsen and Jesper Buus Nielsen. Yes, there is an oblivious ram lower bound! In *CRYPTO*, 2018.

[LWN+15] Chang Liu, Xiao Shaun Wang, Kartik Nayak, Yan Huang, and Elaine Shi. ObliVM: A programming framework for secure computation. In *IEEE S & P*, 2015.

[McS10] Frank McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. *Commun. ACM*, 53:89–97, September 2010.

[Mir17] Ilya Mironov. Renyi differential privacy. In *Computer Security Foundations Symposium (CSF), 2017 IEEE 30th*, pages 263–275. IEEE, 2017.

[MLS+13] Martin Maas, Eric Love, Emil Stefanov, Mohit Tiwari, Elaine Shi, Kriste Asanovic, John Kubiatowicz, and Dawn Song. Phantom: Practical oblivious computation in a secure processor. In *ACM Conference on Computer and Communications Security (CCS)*, 2013.

[MTS+12] Prashanth Mohan, Abhradeep Guha Thakurta, Elaine Shi, Dawn Song, and David Culler. GUPT: Privacy preserving data analysis made easy. In *ACM SIGMOD*, 2012.

[MV16] Jack Murtagh and Salil P. Vadhan. The complexity of computing the optimal composition of differential privacy. In *TCC (A1)*, volume 9562 of *Lecture Notes in Computer Science*, pages 157–175. Springer, 2016.

[NH12] Arjun Narayan and Andreas Haeberlen. Djoin: Differentially private join queries over distributed databases. In *10th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2012, Hollywood, CA, USA, October 8-10, 2012*, pages 149–162. USENIX Association, 2012.

[NHS+23] Milad Nasr, Jamie Hayes, Thomas Steinke, Borja Balle, Florian Tramèr, Matthew Jagielski, Nicholas Carlini, and Andreas Terzis. Tight auditing of differentially private machine learning. *arXiv preprint arXiv:2302.07956*, 2023.

[NP33] J. Neyman and E. S. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231:289–337, 1933.

[PY19] Giuseppe Persiano and Kevin Yeo. Lower bounds for differentially private rams. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 404–434. Springer, 2019.

[PY23] Giuseppe Persiano and Kevin Yeo. Lower bound framework for differentially private and oblivious data structures. In *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part I*, volume 14004 of *Lecture Notes in Computer Science*, pages 487–517. Springer, 2023.

[QJS+22] Lianke Qin, Rajesh Jayaram, Elaine Shi, Zhao Song, Danyang Zhuo, and Shumo Chu. Differentially oblivious relational database operators. *Proc. VLDB Endow.*, 16(4):842–855, 2022.

[RS21] Vijaya Ramachandran and Elaine Shi. Data oblivious algorithms for multicores. In *SPAA*, 2021.

[RYF+13] Ling Ren, Xiangyao Yu, Christopher W. Fletcher, Marten van Dijk, and Srinivas Devadas. Design space exploration and optimization of path oblivious RAM in secure processors. In *ISCA*, pages 571–582, 2013.

[SBTL18] Ali Shafiee, Rajeev Balasubramonian, Mohit Tiwari, and Feifei Li. Secure DIMM: moving ORAM primitives closer to memory. In *IEEE International Symposium on High Performance Computer Architecture, HPCA 2018, Vienna, Austria, February 24-28, 2018*, pages 428–440. IEEE Computer Society, 2018.

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

[sig22] Technology deep dive: Building a faster oram layer for enclaves. `https://signal.org/blog/building-faster-oram/`, 2022.

[SNJ23] Thomas Steinke, Milad Nasr, and Matthew Jagielski. Privacy auditing with one (1) training run. *arXiv preprint arXiv:2305.08846*, 2023.

[SS13] Emil Stefanov and Elaine Shi. Oblivistore: High performance oblivious cloud storage. In *S & P*, 2013.

[SvDS+13] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path oram: An extremely simple oblivious ram protocol. In *ACM CCS*, page 299–310, 2013.

[SvDS+18] Emil Stefanov, Marten van Dijk, Elaine Shi, T.-H. Hubert Chan, Christopher W. Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: an extremely simple oblivious RAM protocol. *J. ACM*, 65(4):18:1–18:26, 2018.

[TTS+22] Florian Tramer, Andreas Terzis, Thomas Steinke, Shuang Song, Matthew Jagielski, and Nicholas Carlini. Debugging differential privacy: A case study for privacy auditing. *arXiv preprint arXiv:2202.12219*, 2022.

[Vad17] Salil Vadhan. The complexity of differential privacy. 2017.

[VRG20] Elisabet Lobo Vesga, Alejandro Russo, and Marco Gaboardi. A programming framework for differential privacy with accuracy concentration bounds. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 411–428. IEEE, 2020.

[VZ23] Salil Vadhan and Wanrong Zhang. Concurrent composition theorems for differential privacy. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, page 507–519, 2023.

[WBNM22] Chenghong Wang, Johes Bater, Kartik Nayak, and Ashwin Machanavajjhala. Incshrink: Architecting efficient outsourced databases using incremental MPC and differential privacy. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 818–832. ACM, 2022.

[WCM18] Sameer Wagh, Paul Cuff, and Prateek Mittal. Differentially private oblivious RAM. *PoPETs*, 2018(4):64–84, 2018.

[WCS15] Xiao Shaun Wang, T-H. Hubert Chan, and Elaine Shi. Circuit ORAM: On Tightness of the Goldreich-Ostrovsky Lower Bound. In *CCS*, 2015.

[WNL+14] Xiao Shaun Wang, Kartik Nayak, Chang Liu, T-H. Hubert Chan, Elaine Shi, Emil Stefanov, and Yan Huang. Oblivious Data Structures. In *CCS*, 2014.

[WRRW23] Justin Whitehouse, Aaditya Ramdas, Ryan Rogers, and Steven Wu. Fully-adaptive composition in differential privacy. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 36990–37007. PMLR, 2023.

[WST12] Peter Williams, Radu Sion, and Alin Tomescu. Privatefs: A parallel oblivious file system. In *CCS*, 2012.

[WZ10] Larry Wasserman and Shuheng Zhou. A statistical framework for differential privacy. *Journal of the American Statistical Association*, 105(489):375–389, 2010.

[ZDW22] Yuqing Zhu, Jinshuo Dong, and Yu-Xiang Wang. Optimal accounting of differential privacy via characteristic function. In *International Conference on Artificial Intelligence and Statistics*, pages 4782–4817. PMLR, 2022.

[ZK17] Danfeng Zhang and Daniel Kifer. Lightdp: towards automating differential privacy proofs. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, pages 888–901. ACM, 2017.

[ZSCM23] Mingxun Zhou, Elaine Shi, T.-H. Hubert Chan, and Shir Maimon. A theory of composition for differential obliviousness. In *EUROCRYPT (3)*, volume 14006 of *Lecture Notes in Computer Science*, pages 3–34. Springer, 2023.

A Deferred Proofs from Section 3

A.1 Proof of Lemma 3.2

First, we show that given and (ϵ, δ)-matching, we can create a suitable neighbor-respecting refined distribution W. Suppose W is induced from the (ϵ, δ)-matching, and moreover, the remaining δ weight at the sources will be assigned to self edges of the form (a, a). More specifically, if some source in the left vertex set a has some weight remaining in the (ϵ, δ)-matching, we simply assign the remaining weight on the edge (a, a). By construction, the marginal distribution of W over Ω_1 is indeed A. Therefore, we just need to prove that for any subset S, $\Pr[W_{|\Omega_2} \in S] \leq e^\epsilon \Pr[B \in S] + \delta$. Notice that since W is valid distribution, $\sum_{a \sim b} W(a, b) = 1$. Also, by the definition of (ϵ, δ)-matching $\sum_{a \sim b} w(a, b) \geq 1 - \delta$. Also, for all $a \sim b$, $W(a, b) \geq w(a, b)$. Thus, for any subset T, we have that $\sum_{(a,b) \in T} W(a, b) \leq \sum_{(a,b) \in T} w(a, b) + \delta$. Then, $\sum_{b \in S} \Pr[W_{|\Omega_2} \in S] = \sum_{b \in S} \sum_{a \sim b} W(a, b) \leq \sum_{b \in S} \sum_{a \sim b} w(a, b) + \delta$. By the definition of (ϵ, δ)-matching, $\sum_{b \in S} \sum_{a \sim b} w(a, b) \leq e^\epsilon \sum_{b \in S} \Pr[B = b]$, which concludes the first part of the proof.

Next, we show that given a satisfying neighbor-respecting refined distribution W, we can construct an (ϵ, δ)-matching $w(\cdot, \cdot)$. In fact, it is easy to construct such w. We first let $w(a, b) = W(a, b)$ for all $a \sim b$. For any "overflowing" vertex $b \in \Omega_2$, such that $\Pr[W_{|\Omega_2} = b] = \sum_{a \sim b} W(a, b) >$

$e^\epsilon \Pr[B = b]$, we continuously reduce some weight of w on some adjacent edges (a, b), until $\sum_{a\sim b} w(a, b) = \Pr[B = b]$. We do the adjustment simultaneously for all "overflowing" vertices because the adjustments will not interfere with each other. We only need to prove that the total "reduced weight" is no more than δ, so the matching satisfies that $\sum_{a,b} w(a, b) \geq 1 - \delta$. Since $\mathsf{D}^\delta(W_{|\Omega_2} \| B) \leq \epsilon$, we have that

$$\sum_{b\in\Omega_2} \left[\Pr[W_{|\Omega_2} = b] - e^\epsilon \Pr[B = b]\right]_+ \leq \delta,$$

which is exactly the amount of the reduced weight of w.

A.2 Equivalence of NPDO and Symmetric NPDO

The following lemma allows us to show that the weaker notion of NPDO in [ZSCM23] can achieve the symmetric property for the case of finite sample spaces.

Lemma A.1 (Bridging the Symmetry Gap)**.** *Suppose A and B are distributions on* finite *sample spaces Ω_1 and Ω_2, respectively, where $\sim$ is a neighboring relation on (Ω_1, Ω_2). Let $\epsilon \geq 0$ and $0 \leq \delta \leq 1$. Suppose $\mathsf{D}^\delta_\sim(A\|B) \leq \epsilon$ and $\mathsf{D}^\delta_\sim(B\|A) \leq \epsilon$, i.e., A and B are (ϵ, δ)-close w.r.t the neighboring relation $\sim$. Then, there exists a refinement pairing W and U for A and B such that* $\max\{\mathsf{D}^\delta(W\|U), \mathsf{D}^\delta(U\|W)\} \leq \epsilon$.

Remark A.2. The converse of Lemma A.1 is trivial.

Proof. In this proof, we visualize the relation $\sim$ as a bipartite graph on (Ω_1, Ω_2). For $i \in \Omega_1$ and $j \in \Omega_2$, we use $A(i)$ and $B(j)$ to denote the corresponding probability masses. Sometimes, we denote $\rho := e^\epsilon$ for convenience.

Since $\mathsf{D}^\delta_\sim(A\|B) \leq \epsilon$, we have an (ϵ, δ)-matching w w.r.t $\sim$ from A to B. Also, $\mathsf{D}^\delta_\sim(B\|A) \leq \epsilon$, so we have an (ϵ, δ)-matching u w.r.t $\sim$ from B to A. For each edge $(i, j) \in \Omega_1 \times \Omega_2$ that $i \sim j$, we have the edge weights $w(i, j)$ and $u(i, j)$.

Constraints on w and u. We have the following constraints on w:

- Conservation constraint. For any $i \in \Omega_1$, $\sum_{j\in\Omega_2: j\sim i} w(i, j) \leq A(i)$.
- Receiving constraint. For any $j \in \Omega_2$, $\sum_{i\in\Omega_1: i\sim j} w(i, j) \leq \rho B(j)$.
- Total weight constraint. $\sum_{(i,j): i\sim j} w(i, j) \geq 1 - \delta$.

We have the following constraints on u:

- Receiving constraint. For any $i \in \Omega_1$, $\sum_{j\in\Omega_2: j\sim i} u(i, j) \leq \rho A(i)$.
- Conservation constraint. For any $j \in \Omega_2$, $\sum_{i\in\Omega_1: i\sim j} u(i, j) \leq B(j)$.
- Total weight constraint. $\sum_{(i,j): i\sim j} u(i, j) \geq 1 - \delta$.

Algorithm 1: OneStepAdjust

If any violating edge becomes legal during the execution, the algorithm halts.
Find a violating edge i_1, j_1 such that $w(i_1, j_1) > \rho u(i_1, j_1)$;
if *Conservation constraint at* j_1 *is not tight* **then**
    Let Δ be the minimum of the following:
    - $(w(i_1, j_1) - \rho u(i_1, j_1))/\rho$; `// The min amount of increasing` $u(i_1, j_1)$ `to fix` $(j_1 \to i_1)$
    - $B(j_1) - \sum_{i:i\sim j_1} u(i, j_1)$; `// The slack of the conservation constraint at` j_1
    - $\rho A(i_1) - \sum_{j:j\sim i_1} u(i_1, j)$; `// The slack of the receiving constraint at` i_1

    Increase $u(i_1, j_1)$ by Δ ;
end
while *Receving constraint at* i_1 *is not tight* **do**
    Find an edge (i', j_1) such that $w(i', j_1) < \rho u(i', j_1)$;
    Let Δ be the minimum of the following:
    - $(w(i_1, j_1) - \rho u(i_1, j_1))/\rho$; `// The min amount of increasing` $u(i_1, j_1)$ `to fix` $(j_1 \to i_1)$
    - $(\rho u(i', j_1) - w(i', j_1))/\rho$; `// The max amount of legally decreasing` $u(i', j_1)$
    - $\rho A(i_1) - \sum_{j:j\sim i_1} u(i_1, j)$; `// The slack of the receiving constraint at` i_1

    Increase $u(i_1, j_1)$ and decrease $u(i', j_1)$ by Δ;
end
Let the visited path $P = (j_1, i_1)$;
while *true* **do**
    Let the last vertex in P be $i_t \in \Omega_1$;
    `//` When the last vertex in P is some $j_t \in \Omega_2$, the adjustment will be made symmetically on the weights of u.
    Find a violating edge $(i_t \to j_{t+1})$ that $u(i_t, j_{t+1}) > \rho w(i_t, j_{t+1})$;
    if j_{t+1} *is not visited* **then**
        Append j_{t+1} to P ;
        for $\ell = t, \ldots, 1$ **do**
            if $j_{\ell+1}$ *is tight* **then** break;
            Let Δ_ℓ be the minimum of the following:
            - $\rho B(j_{\ell+1}) - \sum_{i:i\sim j_{\ell+1}} w(i, j_{\ell+1})$; `// The slack of the receiving constraint at` $j_{\ell+1}$
            - $(u(i_\ell, j_{\ell+1}) - \rho w(i_\ell, j_{\ell+1}))/\rho$; `// The min amount of increasing` $w(i_\ell, j_{\ell+1})$ `to fix` $(i_\ell \to j_{\ell+1})$
            - $w(i_\ell, j_\ell) - \rho u(i_\ell, j_\ell)$; `// The min amount of decreasing` $w(i_\ell, j_\ell)$ `to fix` $(i_\ell \leftarrow j_\ell)$

            Increase $w(i_\ell, j_{\ell+1})$ and decrease $w(i_\ell, j_\ell)$ by Δ ;
        end
    else
        Let the loop be $(j_{c_1} \to i_{c_1} \to \cdots \to j_{c_m} \to i_{c_m})$ and denote $j_{c_{m+1}} = j_{c_1}$;
        Let Δ be the minimal adjustment for w to make one edge legal along the loop ;
        For all $t \in [m]$, decrease $w(i_{c_t}, j_{c_t})$ and increase $w(i_{c_t}, j_{c_{t+1}})$ by Δ ; `// At least one violating edge becomes legal.`
    end
end

Violating Edge. An edge (i, j) is called a *legal* edge if the following condition is satisfied:

$$e^{-\epsilon} \cdot u(i,j) \le w(i,j) \le e^{\epsilon} \cdot u(i,j).$$

Otherwise, it is called a *violating* edge. If there is no violating edge, based on the matchings w and u, we can define induce two distributions W and U as a neighbor-respecting refinement pairing φ between A and B using the idea in the proof of Lemma 2.7. We first check that $\mathsf{D}^{\delta}(W\|U) \le \epsilon$. For any $S \subseteq \Omega_1 \times \Omega_2$, we have

$$\Pr[W \in S] \le \sum_{(i,j)\in\Omega_1\times\Omega_2} w(i,j) + \delta \le \sum_{(i,j)\in\Omega_1\times\Omega_2} e^{\epsilon} \cdot u(i,j) + \delta \le e^{\epsilon} \Pr[U \in S] + \delta,$$

where the first inequality follows from the total weight constraint on w and the second inequality follows because there is no violating edge. Similarly, we can show that $\mathsf{D}^{\delta}(U\|W) \le \epsilon$. Hence, W and U are indeed the required refinement pair.

Edge Weights Adjustment. Our goal is to adjust the edge weights w and u such that all edges eventually become legal, while maintaining all weight constraints.

The easy case is $\rho = 1$, because we simply set the $w'(i,j) = u'(i,j) = \frac{w(i,j)+u(i,j)}{2}$. It is easy to verify that w' and u' satisfy all the requirements. Hence, we assume $\rho > 1$ for the rest of the proof.

We consider a subgraph of the bipartite graph consisting of violation edges. We assign a direction to each violating edge, depending on how the condition is violated. For $i \in \Omega_1$ and $j \in \Omega_2$:

- If $w(i,j) > \rho u(i,j)$, we assign a direction $(j \to i)$ to the edge.
- If $u(i,j) > \rho w(i,j)$, we assign a direction $(i \to j)$ to the edge.

Our strategy is to adjust w and u iteratively, until all edges are legal. We will maintain the receiving and conservation constraints during the adjustment. Also, our adjustment will only maintain or increase the total weight of w or u, so they still satisfy the total weight constraint. In each step, we will make at least one violating edge legal without any legal edge becoming violating again. This guarantees that the adjustment process will terminate in a finite number of steps.

One Step Adjustment. We start from any violating edge. Without loss of generality, say $i_1 \in \Omega_1$ and $j_1 \in \Omega_2$ form a violating edge such that the direction is $j_1 \to i_1$, i.e., $w(i_1, j_1) > \rho u(i_1, j_1)$, which means $u(i_1, j_1)$ is too small and we will try to increase it.

Case 1: The conservation constraint at j_1 and the receiving constraint at i_1 are both not tight. We can directly increase $u(i_1, j_1)$ until at least one constraint becomes tight.

Case 2: The conservation constraint at j_1 is tight but the receiving constraint at i_1 is not tight. We have that $\rho \sum_{i\sim j_1} u(i, j_1) = \rho B(j) \ge \sum_{i\sim j_1} w(i, j_1)$. But as long as $\rho u(i_1, j_1) < w(i_1, j_1)$, there must be an i', such that $\rho u(i', j_1) > w(i', j_1)$. We can increase $u(i_1, j_1)$ and decrease $u(i', j_1)$ until the receiving constraint at i_1 is tight or $\rho u(i', j_1) = w(i', j_1)$. If we cannot decrease $u(i', j_1)$ anymore, we can always find another $i'' \in \Omega_1$ and continue. Notice that since we will simultaneously increase an edge and decrease an edge by the same amount, the total weight constraint is maintained.

Case 3: The tricky case is that the receiving constraint at $i_1 \in \Omega_1$ becomes tight before $j_1 \to i_1$ becomes legal. We next argue that there must exist $j_2 \in \Omega_2$ such that the edge $(i_1 \to j_2)$ is violating. The reason is that we know the receiving constraint at i_1 is tight: $\sum_{j\sim i_1} u(i_1, j) = \rho A(i_1) \ge \rho \sum_{j\sim i_1} w(i_1, j)$. Moreover, i_1 has an incoming violating edge (i_1, j_1): $u(i_1, j_1) \le \rho u(i_1, j) < w(i_1, j_1) \le \rho w(i_1, j_1)$. Hence, there must exists a violating edge $(i_1 \to j_2)$ that $u(i_1, j_2) > \rho w(i_1, j_2)$. Now, suppose the receiving constraint at j_2 is not tight. We can now similarly increase $w(i_1, j_2)$

and decrease $w(i_1, j_1)$ simultaneously, until one edge becomes legal or the receving constraints at j_2 becomes tight.

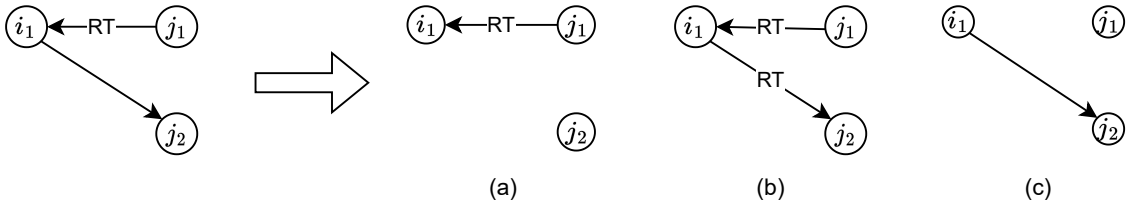


Figure 3: Illustration of Case 3. When the receiving constraint at i_1 becomes tight, there will be a j_2 such that $i_1 \to j_2$. Adjusting the weight of $w(i_1, j_1), w(i_1, j_2)$ will lead to three possible case: (a) $(i_1 \to j_2)$becomes legal; (b) Both receiving constraint at i_1 and j_2 become tight; (c) $(j_1 \to i_1)$becomes legal.

We can now see a repeating pattern. Whenver a vertex has tight receiving constraint and at least one incoming violating edge, it has an outgoing violating edge. We call such a vertex as a tight vertex. It applies to j_2 and there must exist a violating edge $(j_2 \to i_2)$ such that $w(i_2, j_2) > \rho u(i_2, j_2)$. Again, if i_2 is not tight, then we are able to increase $u(i_2, j_2)$ and decrease $u(i_1, j_2)$. Notice that now i_1 is not tight because of the decreasing of $u(i_1, j_2)$. We can now adjust $u(i_1, j_1)$ and $u(i', j_1)$ until one edge becomes legal or i_1 is tight.

The full description of our strategy is described as following and also presented in Algorithm 1.

Suppose the current visited path P contains vertices $j_1, i_1, j_2, i_2, \ldots, j_t, i_t$. By construction, all the vertices along the path except j_1 have tight receiving constraints. By the aforementioned argument, there exists a violation edge $(i_t \to j_{t+1})$. Now, assume j_{t+1} is not visited before. If j_{t+1}'s receiving constraint is not tight, we are able to increase $w(i_t, j_{t+1})$ and decrease $w(i_t, j_t)$, until j_{t+1} becomes tight. We now know j_t is not tight. If the previous adjustment does not fix any edges, we can continue to do the adjustment "backward" along the path: increase $w(i_{t-1}, j_t)$ and decrease $w(i_{t-1}, j_{t-1})$. We keep doing the "back adjustment propagation" until one edge becomes legal or all the vertices along the path except j_1 become tight again. We then add j_{t+1} to the path and repeat the process if no edge is fixed. The case when the last vertex in P is some $j_t \in \Omega_2$ is symmetrical, except that we are adjusting the weight of u instead of w.

Now, what if j_{t+1} is visited before? It means we find a loop consisting of violating edges. Say the loop has $2m$ vertices: $(j_{c_1}, i_{c_1}, \ldots, j_{c_m}, i_{c_m})$. Take Δ be the minimal amount of adjustment for w to make at least one edge along the loop legal. We can now adjust all the w weights along the loop: decrease all $w(i_{c_t}, j_{c_t})$ by Δ and increase $w(i_{c_t}, j_{c_{t+1}})$ by Δ. We will also decrease $w(i_{c_m}, j_{c_1})$ by Δ. This ensures that at least one of the edge along the loop will become legal and it will not violate any receiving nor any preserving constraints because we are always adjusting a pair of adjacent edge to any vertex simultaneously. Finally, this also maintains the total weight constraint because there are exactly m edges increased by Δ and m edges decreased by Δ!

This process will terminate in a finite number of steps because we consider finite sample spaces Ω_1 and Ω_2. □

B Group Privacy Theorem for NPDO

In this section, we provide the characterization of group privacy theorem for NPDO, described in the language of power function.

Triangle Inequality and Group Privacy. Given the privacy guarantee for neighboring inputs, *group privacy* refers to the privacy guarantees that can be deduced for multi-hop neighboring inputs. When we define differential privacy or NPDO with power functions, the group privacy theorem is consequence of the triangle ineuqality.

Fact B.1 (Triangle Inequality for Power Functions [DRS22, Theorem 2.14])**.** *Suppose X, Y, Z are distributions on the same sample space such that $\mathsf{Pow}(X\|Y) \le g_1$ and $\mathsf{Pow}(Y\|Z) \le g_2$. Then, $\mathsf{Pow}(X\|Z) \le g_2 \circ g_1$, where the composition is defined as $(g_2 \circ g_1)(x) := g_2(g_1(x))$.*

In other words, $\mathsf{Pow}(X\|Z) \le \mathsf{Pow}(Y\|Z) \circ \mathsf{Pow}(X\|Y)$.

Proof. (Sketch) Let S be any measurable subset in the sample space. From $\mathsf{Pow}(X\|Y) \le g_1$, we have $\Pr[Y \in S] \le g_1(\Pr[X \in S])$. From $\mathsf{Pow}(Y\|Z) \le g_2$, we have $\Pr[Z \in S] \le g_2(\Pr[Y \in S]) \le (g_2 \circ g_1)(\Pr[X \in S])$, as required. □

To achieve a similar result to Fact B.1 for NPDO, we need a corresponding lemma that considers the neighbor-respecting refinement pairing that is involved in the definition of NPDO(Pow) as in Definition 5.3.

Lemma B.2 (Triangle Inequality for Refinement Pairs)**.** *Suppose X_1, X_2 and X_3 are distributions on sample spaces Ω_1, Ω_2 and Ω_3, respectively. Furthermore, suppose for each $i \in \{1, 2\}$, there is some neighbor-respecting refinement pairing μ_i between X_i and X_{i+1} with respect to some neighboring relation $\sim_i$ on (Ω_i, Ω_{i+1}) such that $\mathsf{Pow}^{\mu_i}(X_i\|X_{i+1}) \le g_i$.*

Then, by defining a neighboring relation $\sim_3$ on (Ω_1, Ω_3) as having a common neighbor in Ω_2, there exists a neighbor-respecting refinement pairing φ between X_1 and X_3 (w.r.t. $\sim_3$) such that $\mathsf{Pow}^{\varphi}(X_1\|X_3) \le g_2 \circ g_1$.

Proof. We first define refinements for all three distributions X_1, X_2 and X_3 on the common sample space $\Omega_1 \times \Omega_2 \times \Omega_3$.

From the given pairing μ_1 between X_1 and X_2, we can write $\mu_1(X_1) = (X_1, Y_2^{(1)}(X_1))$, where $Y_2^{(1)}(u)$ is the conditional distribution on Ω_2 given $X_1 = u$. (From the neighboring-respecting property, the support of $Y_2^{(1)}(u)$ may contain only neighbors of u.) Similarly, we can write $\mu_1(X_2) = (Y_1^{(2)}(X_2), X_2)$, where $Y_1^{(2)}(v)$ is the conditional distribution on Ω_1 given $X_2 = v$.

For the pairing μ_2 between X_2 and X_3, we define the following similarly for distributions in $\Omega_2 \times \Omega_3$: $\mu_2(X_2) = (X_2, Y_3^{(2)}(X_2))$ and $\mu_2(X_3) = (Y_2^{(3)}(X_3), X_3)$.

Next, we can define the refinement triplet $\widehat{\rho}$ for all three distributions on $\Omega_1 \times \Omega_2 \times \Omega_3$:

- $\widehat{\rho}(X_1) = (X_1, Y_2^{(1)}(X_1), Y_3^{(2)}(Y_2^{(1)}(X_1)))$:
  - Sample $u \in \Omega_1$ according to X_1;
  - Sample $v \in \Omega_2$ from $Y_2^{(1)}(u)$;
  - Sample $w \in \Omega_3$ from $Y_3^{(2)}(v)$;
  - Output (u, v, w), which ensures that $u \sim_1 v \sim_2 w$.
- $\widehat{\rho}(X_2) = (Y_1^{(2)}(X_2), X_2, Y_3^{(2)}(X_2))$:
  - Sample $v \in \Omega_2$ from X_2;
  - Sample u from $Y_1^{(2)}(v)$;
  - Sample w from $Y_3^{(2)}(v)$;

– Output (u, v, w), which ensures that $u \sim_1 v \sim_2 w$.

- $\widehat{\rho}(X_3) = (Y_1^{(2)}(Y_2^{(3)}(X_3)), Y_2^{(3)}(X_3), X_3)$
  - Sample $w \in \Omega_3$ from X_3;
  - Sample $v \in \Omega_2$ from $Y_2^{(3)}(w)$;
  - Sample $u \in \Omega_1$ from $Y_1^{(2)}(v)$;
  - Output (u, v, w), which ensures that $u \sim_1 v \sim_2 w$.

Now we have

$$\mathsf{Pow}(\widehat{\rho}(X_1)\|\widehat{\rho}(X_2)) = \mathsf{Pow}(\mu_1(X_1)\|\mu_1(X_2)) \leq g_1.$$

This is a consequence of the post-processing theorem of power function, because starting from the distribution pair μ_1 on $\Omega_1 \times \Omega_2$, one applies the same conditional distribution $Y_3^{(2)}(\cdot)$ on Ω_3 given the generated value in Ω_2 to produce the distribution pair $\widehat{\rho}$. By a similar argument, we have

$$\mathsf{Pow}(\widehat{\rho}(X_2)\|\widehat{\rho}(X_3)) = \mathsf{Pow}(\mu_2(X_2)\|\mu_1(X_3)) \leq g_2.$$

Finally, we can define the neighbor-respecting refinement pairing ρ between X_1 and X_3 by the marginal distribution of $\widehat{\rho}(X_1)$ and $\widehat{\rho}(X_3)$ over $\Omega_1 \times \Omega_3$. Then, we have: $\mathsf{Pow}(\rho(X_1)\|\rho(X_3)) \leq \mathsf{Pow}(\widehat{\rho}(X_1)\|\widehat{\rho}(X_3)) \leq g_2 \circ g_1$, where the first inequality is due to data processing and the second inequality follows from Fact B.1. Moreover, from the construction, if (u, w) is in the support of $\rho(X_1)$ or $\rho(X_3)$, then there exists $v \in \Omega_2$ such that $u \sim_1 v \sim_2 w$. This completes the proof. □

Corollary B.3. *Suppose a mechanism is g-NPDO(Pow) with respect to its input and output neighboring relations. Then, the mechanism is $g^{\circ k}$-NPDO(Pow) with respect to its k-hop input and output neighboring relations.*

Proof. This can be proved by induction on k, where the inductive step is a straightforward application of Lemma B.2. □