Concurrent Composition for Interactive Differential Privacy with Adaptive Privacy-Loss Parameters

Samuel Haney
sam.haney@tmlt.io
Tumult Labs
Raleigh, USA

Michael Shoemate
shoematem@seas.harvard.edu
Harvard University
Cambridge, USA

Grace Tian
gracetian6@gmail.com
Harvard University
Cambridge, USA

Salil Vadhan
salil_vadhan@harvard.edu
Harvard University
Cambridge, USA

Andrew Vyrros
andrew_vyrros@g.harvard.edu
Harvard University
Cambridge, USA

Vicki Xu
vickixu@college.harvard.edu
Harvard University
Cambridge, USA

Wanrong Zhang
wanrongzhang@fas.harvard.edu
Harvard University
Cambridge, USA

ABSTRACT

In this paper, we study the concurrent composition of interactive mechanisms with adaptively chosen privacy-loss parameters. In this setting, the adversary can interleave queries to existing interactive mechanisms, as well as create new ones. We prove that every valid privacy filter and odometer for noninteractive mechanisms extends to the concurrent composition of interactive mechanisms if privacy loss is measured using (ϵ, δ)-DP, f-DP, or Rényi DP of fixed order. Our results offer strong theoretical foundations for enabling full adaptivity in composing differentially private interactive mechanisms, showing that concurrency does not affect the privacy guarantees. We also provide an implementation for users to deploy in practice.

CCS CONCEPTS

• **Theory of computation** → **Interactive computation**; **Concurrent algorithms**; • **Security and privacy** → **Privacy-preserving protocols**; **Privacy protections**.

KEYWORDS

Differential Privacy, Interactive Mechanisms, Concurrent Composition, Adaptivity

ACM Reference Format:
Samuel Haney, Michael Shoemate, Grace Tian, Salil Vadhan, Andrew Vyrros, Vicki Xu, and Wanrong Zhang. 2023. Concurrent Composition for Interactive Differential Privacy with Adaptive Privacy-Loss Parameters. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3576915.3623128

CCS '23, November 26–30, 2023, Copenhagen, Denmark

1 INTRODUCTION

1.1 Differential Privacy

Differential privacy is a framework for protecting the privacy of individuals when analyzing data. It is a mathematical definition of privacy that that ensures that the results of an analysis do not reveal too much information about any individual in the dataset. Because of its powerful worst-case guarantee, differential privacy has become a leading approach in privacy-preserving data analysis, where it is used to enable the analysis of sensitive data while preserving the privacy of individuals.

Differential privacy can be defined in terms of a general database space $\mathcal{X}$ and a binary neighboring relation on $\mathcal{X}$. For example, if databases contain an ordered and known number n of real-valued entries, then $\mathcal{X} = \mathbb{R}^n$. The binary relation on $\mathcal{X}$ specifies which datasets are *neighboring*, meaning that they differ on one individual's data. For example, if $x = \mathbb{R}^n$, then $x, x' \in \mathcal{X}$ are neighboring if they differ on one coordinate. Differential privacy requires that the output distributions should be roughly the same on the two neighboring datasets.

Definition 1.1 (Differential Privacy). *A randomized mechanism $\mathcal{M} : \mathcal{X} \to \mathcal{R}$ is* (ϵ, δ)-differentially private *if for every pair of neighboring datasets $x, x' \in \mathcal{X}$, and for every subset of possible outputs $\mathcal{S} \subseteq \mathcal{R}$,*

$$Pr[\mathcal{M}(x) \in \mathcal{S}] \leq \exp(\epsilon) \cdot Pr[\mathcal{M}(x') \in \mathcal{S}] + \delta.$$

By requiring that an analysis be robust to changes in neighboring datasets, differential privacy provides a guarantee that the privacy of individuals in the dataset is protected, regardless of what other data might be included or excluded.

In recent years, other forms of differential privacy have enjoyed use to address various shortcomings of (ϵ, δ)-DP (also known as approximate-DP) regarding composition. Some standard variants include Rényi DP (RDP) [17], f-DP [3] (the formal definition is given in Section 3), and zero-concentrated differential privacy (zCDP) [2, 8].

Definition 1.2 (Rényi divergence [19]). *For two probability distributions P and Q, the* Rényi divergence *of order $\alpha > 1$ is*

$$D_\alpha(P||Q) = \frac{1}{\alpha - 1} \log \mathrm{E}_{x \sim Q} \left[\frac{P(x)}{Q(x)} \right]^\alpha .$$

Definition 1.3 (Rényi DP [17]). *A randomized mechanism $\mathcal{M}$ is (α, ϵ)-*Rényi differentially private *(ϵ-RDP$_\alpha$) if for all every two neighboring datasets x and x',*

$$D_\alpha(\mathcal{M}(x)||\mathcal{M}(x')) \leq \epsilon$$

1.2 Composition of Differentially Private Mechanisms

A fundamental question in differential privacy concerns how privacy degrades under multiple mechanisms run on the same database.

1.2.1 Composition Theorems for Noninteractive Mechanisms. Definition 1.1 defines differential privacy for *noninteractive mechanisms* $\mathcal{M}$. Composition for such noninteractive mechanisms has been extensively studied in the literature. We will denote the (noninteractive, non-adaptive) *composition* of k noninteractive mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_k$ on a dataset x as $\mathcal{M} := Comp(\mathcal{M}_1, \ldots, \mathcal{M}_k)$, where $\mathcal{M}(x)$ returns $\mathcal{M}_1(x), \ldots, \mathcal{M}_k(x)$, with each $\mathcal{M}_j$ executed independently on its own random coins.

Basic composition of (ϵ, δ)-differential privacy [4] finds that the privacy-loss parameters scale at most linearly with the number of mechanisms composed. Advanced composition [9] provides a tighter bound where the parameters scale sublinearly with the number of mechanisms, and optimal composition [13, 18] provides an exact guarantee of this composition. DP variants such as Rényi DP, f-DP, and z-CDP can provide tighter composition bounds by capturing more information about the mechanisms $\mathcal{M}_i$ being composed than just the two parameters ϵ_i and δ_i.

1.2.2 Composition Theorems for Interactive Mechanisms. While many differential privacy mechanisms are noninteractive, some mechanisms are expressly desired to be interactive, receiving and responding to adaptive queries from analysts. Examples include adaptive composition procedures [9], the Sparse Vector Technique [5–7], and Private Multiplicative Weights [12]. Thus, interactive mechanisms have been used as the basic abstraction in the programming frameworks of the open-source software project OpenDP [11] as well as the Tumult Analytics platform [14] [1]. An interactive mechanism $\mathcal{M}$ is a party interacting with an analyst adversary in an interactive protocol, wherein each party has its *random coin*, which captures the randomness used by the mechanism.

Definition 1.4 (Interactive Algorithms). *An* interactive algorithm*, also known as a* randomized state machine*, consists of a randomized algorithm $\mathcal{M} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ that takes the current state $s \in \{0,1\}^*$, a query $q \in \{0,1\}^*$, and returns a new state $s' \in \{0,1\}^*$ and an answer $a \in \{0,1\}^*$, written $(s', a) = \mathcal{M}(s, q)$. When we wish to make the randomness r of $\mathcal{M}$ more explicit, we write $(s', a) = \mathcal{M}(s, q; r)$.*

An interactive algorithm interacts with an analyst or adversary $\mathcal{A}$ as follows.

Definition 1.5 (Interaction between two mechanisms). *For two interactive algorithms $\mathcal{M}$ and $\mathcal{A}$, the* interaction *between $\mathcal{M}$ and $\mathcal{A}$ on an input $x \in \{0,1\}^*$ to $\mathcal{M}$ is the following random process (denoted $(\mathcal{A} \leftrightarrow \mathcal{M}(x))$):*

(1) Initialize $s_1^{\mathcal{M}} := x$, $m_0 = \lambda$, $s_1^{\mathcal{A}} = \lambda$, where λ is the empty string.
(2) Repeat the following for $i = 1, 2, \ldots$.
  (a) If i is odd, let $(s_{i+1}^{\mathcal{M}}, m_i) = \mathcal{M}(s_i^{\mathcal{M}}, m_{i-1})$ and $s_{i+1}^{\mathcal{A}} = s_i^{\mathcal{A}}$.
  (b) If i is even, let $(s_{i+1}^{\mathcal{A}}, m_i) = \mathcal{A}(s_i^{\mathcal{A}}, m_{i-1})$ and $s_{i+1}^{\mathcal{M}} = s_i^{\mathcal{M}}$.
  (c) if $m_i = \texttt{halt}$, then exit loop.

In the context of DP, think of the input x as a sensitive dataset that needs to be protected. In this context, an *interactive mechanism* is a randomized state machine whose input space upon initialization is the space of datasets $\mathcal{X}$, and takes in a database $x \in \mathcal{X}$. An *adversary* is a randomized state machine that takes in an empty string. The view of an adversary captures everything the adversary receives during the execution.

Definition 1.6 (View of the adversary in an interactive mechanism). *Let $\mathcal{M}$ be an interactive mechanism and $\mathcal{A}$ be an adversary interacting with the mechanism. $\mathcal{A}$'s* view *of $(\mathcal{A}, \mathcal{M}(x))$ is the tuple $\texttt{View}_{\mathcal{A}}(\mathcal{A} \leftrightarrow \mathcal{M}(x)) = (r_0, m_1, r_2, m_3, r_4, \ldots)$ consisting of all the messages m_i received by $\mathcal{A}$ from $\mathcal{M}$ together with random coins r_i that $\mathcal{A}$ tosses when computing m_i (i.e. for even i, $(s_{i+1}, m_i) = \mathcal{A}(s_i^{\mathcal{A}}, m_{i-1}; r_i)$).*

For shorthand, we will drop the subscript $\mathcal{A}$ when referring to `View` for the rest of this paper, because future references to `View` in the rest of this paper always concern the adversary's view.

We can define interactive differential privacy based on measuring the same (ϵ, δ)-closeness, as in noninteractive differential privacy, between the views of the adversary on two neighboring datasets.

Definition 1.7 ((ϵ, δ)-DP interactive mechanisms). *An interactive mechanism $\mathcal{M}$ is an* (ϵ, δ)-differentially private interactive mechanism*, or* I.M. *for short, if for every pair of neighboring datasets $x, x' \in \mathcal{X}$, every interactive adversary algorithm $\mathcal{A}$, and every subset of possible views $\mathcal{S} \subseteq Range(\texttt{View})$ we have*

$$Pr[\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{M}(x)] \in \mathcal{S}) \leq e^\epsilon Pr[\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{M}(x')) \in \mathcal{S}] + \delta.$$

To capture variants such as Rényi DP and f-DP, we will define a broader version of I.M.s based on a generalized notion of differential privacy in Section 2.

When we consider the composition of interactive mechanisms, it is straightforward to extend the composition theorems for noninteractive mechanisms to the *sequential composition* of interactive mechanisms. A subtler case is the *concurrent composition*, in which an adversary can interleave queries to multiple mechanisms concurrently, first studied by Vadhan and Wang [21]. The queries can therefore depend on the answers received from other mechanisms. This is useful in many practical settings. For example, an analyst may run data analyses using multiple interactive analyses on the same dataset simultaneously and the queries in multiple analyses might be correlated. Concurrent composition, formally defined in Section 2.1, maintains k interactive mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_k$. It is itself an interactive mechanism and the query it received from an adversary is of the form of (j, q), meaning it issues a standard query q to $\mathcal{M}_j$. Concurrent composition theorems allow us to understand

the privacy guarantee of the overall analysis given the privacy guarantees for each interactive analysis when they are executed independently.

Previous work provides concurrent composition theorems for several different types of differential privacy. Vadhan and Wang [21] shows that every composition theorem for noninteractive ϵ-DP mechanisms extends to concurrent composition for interactive ϵ-DP mechanisms. Lyu [15] and Vadhan and Zhang [22] generalize this result to (ϵ, δ)-DP when $\delta > 0$.

Theorem 1.8 ([15, 22]). *Suppose that for all noninteractive mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_k$ such that $\mathcal{M}_i$ is (ϵ_i, δ_i)-DP for $i = 1, \ldots, k$, their composition $Comp(\mathcal{M}_1, \ldots, \mathcal{M}_k)$ is (ϵ, δ)-DP. Then for all interactive mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_k$ with finite communication[1] such that $\mathcal{M}_i$ is (ϵ_i, δ_i)-DP for $i = 1, \ldots, k$, their concurrent composition $ConComp(\mathcal{M}_1, \ldots, \mathcal{M}_k)$ is (ϵ, δ)-DP.*

This composition theorem also extends to f-DP [22]. Moreover, Lyu [15] shows that the privacy adds up under concurrent composition for any fixed order of $\alpha > 1$ for Rényi DP (RDP).

Theorem 1.9 ([15]). *For all $\alpha > 1$, $k \in \mathbb{N}$, $\epsilon_1, \ldots, \epsilon_k > 0$, and all interactive mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_k$ such that $\mathcal{M}_i$ is (α, ϵ_i)-RDP for $i = 1, 2 \ldots, k$, their concurrent composition $ConComp(\mathcal{M}_1, \ldots, \mathcal{M}_k)$ is $(\alpha, \sum_{i=1}^{k} \epsilon_i)$-RDP.*

In all of the above cases, privacy-loss parameters are set upfront prior to data analysis and are fixed for all queries or computations performed on the data. However, in practice, data analysts may not know in advance what they want to do with the data and may want to adaptively choose the privacy-loss parameters of subsequent mechanisms as they go along. This can lead to more efficient use of privacy resources. For example, we may wish to spend more privacy-loss budget on fine-grained analysis, and less privacy-loss budget on exploratory analysis. Therefore, allowing *full adaptivity* where not only the mechanisms, but also the privacy-loss parameters themselves and the length of the composition can be chosen adaptively as a function of intermediate analyses, is important in practice. This consideration motivates the study of privacy *filters* and *odometers*, which we discuss in the following section.

1.3 Odometers and Filters

Rogers, Roth, Ullman, and Vadhan [20] define two primitives for the adaptive composition of DP mechanisms, privacy *filters* and privacy *odometers*, to allow for the ability to track privacy loss during an interaction. Their definitions were given specifically for (ϵ, δ)-DP; here we follow Lécuyer [16] and work with a more general formalism that applies to arbitrary privacy measures, including f-DP and Rényi-DP.

A privacy *filter* is a mechanism that halts computation on a dataset once a preset privacy-loss budget is exceeded. It takes a global privacy-loss budget as an input and is equipped with a continuation rule that halts computation whenever the budget is exceeded. At each round, the continuation rule takes all privacy-loss parameters up to the current round, and outputs either `continue` or `halt`. If `continue`, the mechanism answers the query with the current privacy parameter. Once the mechanism outputs `halt`, no further computation is allowed. (An equivalent alternative is to refuse to answer the correct query, but allow the analyst to try again with new queries. However, the halt formulation is more convenient for presentation in this paper. For notational convenience in this paper, our algorithm pseudocode will have the filter go into a looping state that always returns the same state and a dummy message once `halt` is reached.) The filter guarantees that the interaction is differentially private according to the desired privacy-loss budget.

Definition 1.10 ($\mathcal{F}$-filtered composition of noninteractive (ϵ, δ)-DP mechanisms ($\mathcal{F}$-Filt(NIM))). *Let $\mathcal{F}$ be a continuation rule that takes in a sequence of privacy-loss parameters $(\epsilon_1, \delta_1), \ldots,$ and a target privacy-loss budget (ϵ, δ), and maps to a binary decision: $\mathcal{F} : \mathbb{R}^*_{\geq 0} \times \mathbb{R}^*_{\geq 0} \times \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \{0, 1\}$, where 1 means* `continue` *and 0 means* `halt`*, and $\mathbb{R}^*_{\geq 0} = \cup_{k=0}^{\infty} \mathbb{R}^k$. The $\mathcal{F}$-*filtered composition of noninteractive mechanisms, *denoted as $\mathcal{F}$-Filt(NIM), is an interactive mechanism. At the $(k+1)$th round, $\mathcal{F}(\cdot; (\epsilon, \delta))$-Filt(NIM)$(s, m)$ is executed as follows:*

1. *If $m = \lambda$, initialize $s = (s, [])$, where $[]$ is an empty list. Return (s, λ), where λ is an empty string.*
2. *Parse $s = (x, [(\mathcal{M}_1, (\epsilon_1, \delta_1)), \ldots, (\mathcal{M}_{k-1}, (\epsilon_k, \delta_k))])$*
3. *If $\mathcal{M}_{k+1}$ is $(\epsilon_{k+1}, \delta_{k+1})$-DP and $\mathcal{F}((\epsilon_1, \delta_1), \ldots, (\epsilon_{k+1}, \delta_{k+1}); (\epsilon, \delta)) = 1$:*
   (a) *Let $s' = (x, [(\mathcal{M}_1, (\epsilon_1, \delta_1)), \ldots, (\mathcal{M}_{k+1}, (\epsilon_{k+1}, \delta_{k+1}))])$*
   (b) *Let $m' = \mathcal{M}_{k+1}(x)$*
4. *Else, let $s' = s$, $m' =$* `halt`

Note that at each round k of computation, whether the mechanism $\mathcal{M}_k$ is (ϵ_k, δ_k)-DP is something the implementation needs to be able to verify, e.g. by having verified privacy-loss parameters attached to every mechanism.

Definition 1.11 (Valid (ϵ, δ)-DP filter for noninteractive mechanisms). *We say a continuation rule $\mathcal{F}$ is a valid (ϵ, δ)-DP NIM-filter for noninteractive mechanisms if for every pair of (ϵ, δ), $\mathcal{F}(\cdot; (\epsilon, \delta))$-Filt(NIM)$(\cdot)$ is an (ϵ, δ)-DP interactive mechanism.*

A privacy *odometer* is a mechanism that allows the analyst to keep track of the privacy loss at each step of computation. It is equipped with a privacy-loss accumulator $\mathcal{G}$, which gives an upper bound on the accumulated privacy loss at each step. We note that our terminology is slightly different from previous work by Rogers et al. [20], where they refer to the privacy-loss accumulator as the odometer, whereas our odometer is a mechanism that has a specially-designated privacy loss query. When posing this query, the mechanism outputs the current privacy loss up to that point.

`privacy_loss` queries are deterministic and do not change the state of the odometer. We will stipulate that they are encoded as binary strings in a fixed and recognizable way, such as using all strings that begin with a 1 to be `privacy_loss` queries, and all strings that begin with a 0 to be ordinary queries that can then be further parsed.

Definition 1.12 ($\mathcal{G}$-odometer for composition of noninteractive mechanisms ($\mathcal{G}$-Odom(NIM))). *Let $\mathcal{G}$ be a privacy-loss accumulator that takes privacy-loss parameters $(\epsilon_1, \delta_1), \ldots, (\epsilon_k, \delta_k)$ in a sequence, and maps to a global privacy loss (ϵ, δ), so $\mathcal{G} : \mathbb{R}^k_{\geq 0} \times$*

[1] Their proof relies on an induction argument on the number of messages exchanged, which requires an assumption of *finite communication*.

$\mathbb{R}^k_{\geq 0} \to \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$, *for every* $k = 1, 2, \ldots$. *A* $\mathcal{G}$*-odometer for composition of noninteractive mechanisms is denoted as* $\mathcal{G}$*-Odom(NIM).* $\mathcal{G}$*-Odom*$(NIM)(s, m)$ *is executed as follows:*

1. *If* $m = \lambda$, *initialize* $s = (x, [])$, *where* $[]$ *is an empty list, and return* (s, λ)
2. *Parse* $s = (x, [(\mathcal{M}_1, (\epsilon_1, \delta_1), \ldots, (\mathcal{M}_k, (\epsilon_k, \delta_k))])$
3. *If* $m = (\mathcal{M}_{k+1}, (\epsilon_{k+1}, \delta_{k+1}))$ *and* $\mathcal{M}_{k+1}$ *is* $(\epsilon_{k+1}, \delta_{k+1})$*-DP:*
   (a) *Let* $s' = (x, [(\mathcal{M}_1, (\epsilon_1, \delta_1), \ldots, (\mathcal{M}_{k+1}, (\epsilon_{k+1}, \delta_{k+1}))])$
   (b) *Let* $m' = \mathcal{M}_{k+1}(x)$
4. *If* $m =$ `privacy_loss`, *then return* $\mathcal{G}((\epsilon_1, \delta_1), \ldots, (\epsilon_k, \delta_k))$.

To measure the privacy loss at each round, we use *truncated view* of an adversary $\mathcal{A}$ interacting with $\mathcal{G}$*-Odom(NIM)* as defined in Definition 1.13.

Definition 1.13 (View between adversary $\mathcal{A}$ with approximate-DP odometer O truncated at (ϵ, δ)). *Given a privacy-loss parameter* (ϵ, δ), *an adversary* $\mathcal{A}$, *and approximate-DP odometer* O, $\texttt{Trunc}_{(\epsilon,\delta)}(\texttt{View}(\mathcal{A} \leftrightarrow O(x)))$ *is constructed as follows:*

1. *Initialize* $s_1^{\mathcal{M}} := x$, $m_0 = \lambda$, $s_1^{\mathcal{A}} = \lambda$, *where* λ *is the empty string.*
2. *Repeat the following for* $i = 1, 2, \ldots$.
   (a) *If* i *is odd, let* $(s_{i+1}^{\mathcal{M}}, a_i) := O(s_i^{\mathcal{M}}, q_{i-1})$, *and let* $(s_i^{\mathcal{M}}, (\epsilon_i, \delta_i)) = O(s_i^{\mathcal{M}}, \texttt{privacy-loss})$, $s_{i+1}^{\mathcal{A}} = s_i^{\mathcal{A}}$.
   (b) *If* i *is even, let* $(s_{i+1}^{\mathcal{A}}, q_i) := \mathcal{A}(s_i^{\mathcal{A}}, a_{i-1}; r_i^{\mathcal{A}})$ *and* $s_{i+1}^{\mathcal{M}} = s_i^{\mathcal{M}}$.
   (c) *If* $\epsilon_i \geq \epsilon$ *or* $\delta_i \geq \delta$, *then exit loop.*
   (d) *If* $a_i =$ `halt`, *then exit loop.*
3. *Return* $\texttt{Trunc}_{(\epsilon,\delta)}(\texttt{View}(\mathcal{A} \leftrightarrow O(x)) = (r_0^{\mathcal{A}}, a_1, \ldots, r_{i-2}^{\mathcal{A}}, a_{i-1})$.

Using truncated view for analyzing odometers is first introduced by Lécuyer [16] for RDP. We extend their definition to a general notion of truncated view, which allows us to quantify privacy loss with all other DP variants, where we can simply replace (ϵ, δ) with other privacy-loss parameters for other DP variants. In contrast, the previous definition of odometer in Rogers et al. [20] only holds for (ϵ, δ)-DP.

Then the privacy guarantee of an odometer is defined as measuring the closeness between the distributions of the truncated views of an adversary.

Definition 1.14 (Valid approximate-DP privacy-loss accumulator for noninteractive mechanisms). *We say* $\mathcal{G}$ *is a* valid approximate-DP NIM-privacy-loss accumulator *if* $\mathcal{G}$*-Odom*(NIM) *is a valid approximate-DP odometer.*

Definition 1.15 (Valid approximate-DP odometer for noninteractive mechanisms). *We say* O *is a valid approximate-DP* odometer *if for every pair of* (ϵ, δ), *every adversary* $\mathcal{A}$, *and every pair of adjacent datasets* x, x',

$$D(\texttt{Trunc}_{(\epsilon,\delta)}(\texttt{View}(\mathcal{A} \leftrightarrow O(x)))\,||\,\texttt{Trunc}_{(\epsilon,\delta)}(\texttt{View}(\mathcal{A} \leftrightarrow O(x')))) \preceq (\epsilon, \delta),$$

where $\preceq$ *is a partial order and* $(\epsilon_1, \delta_1) \preceq (\epsilon_1, \delta_1)$ *iff* $\epsilon_1 \leq \epsilon_2$ *and* $\delta_1 \leq \delta_2$.

A number of previous works construct privacy filters and odometers for noninteractive mechanisms. The original odometer definition proposed by Rogers et al. [20] was defined specifically for (ϵ, δ)-DP and requires a simultaneous guarantee about the privacy loss being bounded at all points *in time* when composing with adaptive privacy-loss parameters. Specifically, the privacy loss is defined as $\texttt{Loss}(v) = \log\left(\frac{\Pr[\texttt{View}(\mathcal{A} \leftrightarrow O(x)) = v]}{\Pr[\texttt{View}(\mathcal{A} \leftrightarrow O(x')) = v]}\right)$, and the odometer is defined as follows.

Definition 1.16 (Odometer in [20]). *We say* $\mathcal{G}$ *is a valid approximate DP* odometer *if for every adversary* $\mathcal{A}$ *in* k*-fold composition with adaptive privacy-loss parameters, and every pair of adjacent datasets* x, x', *the following holds with probability at most* δ_g *over* $v \leftarrow \texttt{View}(\mathcal{A} \leftrightarrow O(x))$

$$|\texttt{Loss}(v)| > \mathcal{G}(\epsilon_1, \delta_1, \ldots, \epsilon_k, \delta_k).$$

Note that this definition yields a form of approximate DP in its final guarantee, because there is always a δ_g probability of failure.

Whitehouse, Ramdas, Rogers, and Wu [24] gave constructions of odometers and filters that quantitatively improve on the results presented in Rogers et al [20], including when the composed mechanisms satisfy zCDP or Rényi DP (but their odometer only guarantees the same, (ϵ, δ) property of Definition 1.16). For providing Rényi DP guarantees when composing Rényi DP mechanisms, Feldman and Zrnic [10] constructed privacy filters. Definitions and constructions of privacy odometers for Rényi DP were given by Lécuyer [16]. Our general definition of odometers follows Lécuyer's, which we find more natural and general, since it is applicable to all forms of DP. This definition can be thought of as being in analogy to the definition of a p-value in statistics. A p-value has an interpretation if we set a significance level α before we observe our random events, and then reject the null hypothesis if the p-value is larger than α. Similarly the privacy-loss reported by our odometer definition has an interpretation if we set a privacy-loss threshold (ϵ, δ) before the mechanism is executed and halt when the odometer's privacy-loss query is not going to be smaller than (ϵ, δ).

1.4 Our Results on Concurrent Filters and Odometers

In this paper, we consider privacy filters and odometers for composing *interactive* mechanisms. We analogously define $\mathcal{F}$-filtered sequential composition of interactive mechanisms, or $\mathcal{F}$*-FiltSeq(IM)* for short, and $\mathcal{F}$-filtered concurrent composition of interactive mechanisms, or $\mathcal{F}$*-FiltCon(IM)* for short. The continuation rule $\mathcal{F}$ here determines whether the adversary can create new interactive mechanisms with adaptively-chosen privacy-loss parameters. Similarly, we define $\mathcal{G}$-odometer for the sequential composition of interactive mechanisms, or $\mathcal{G}$*-OdomSeq(IM)* for short, and $\mathcal{G}$-odometer for the concurrent composition of interactive mechanisms, or $\mathcal{G}$*-OdomCon(IM)* for short. It should be noted that in both our filters and odometers, the "privacy-loss budget" is paid at the launch of each additional interactive mechanism. In particular, any `privacy_loss` queries asked to the odometer between mechanism launches will return the same value.

As composition theorems for noninteractive mechanisms typically can be easily extended to the sequential composition of interactive mechanisms, we focus on the case of concurrent composition, as defined formally in Definition 2.4. In this setting, the adversary can interleave queries to existing interactive mechanisms. In the filter and odometer case, the adversary can also create new interactive mechanisms with adaptively-chosen privacy-loss parameters

(for filters, with the added stiuplation that the privacy loss with the new interactive mechanism added does not exceed the privacy budget). The complex dependence between the adversary's interacting with the different mechanisms makes it non-trivial to extend filters and odometers to this case. Nevertheless, we prove that every valid privacy filter and odometer for noninteractive mechanisms extends to a privacy filter and odometer for interactive mechanisms for (ϵ, δ)-DP, f-DP, and RDP.

Definition 1.17 (Valid (ϵ, δ)-DP filter for interactive mechanisms). *We say a continuation rule $\mathcal{F}$ is a* valid (ϵ, δ)-DP IM-filter *for interactive mechanisms if $\mathcal{F}(\cdot; (\epsilon, \delta))$-FiltCon(IM) is an (ϵ, δ)-DP interactive mechanism.*

Definition 1.18 (Valid (ϵ, δ)-DP privacy-loss accumulator for interactive mechanisms). *We say $\mathcal{G}$ is a valid* approx-DP IM-privacy-loss accumulator *if $\mathcal{G}$-OdomCon(IM) is a valid approx-DP odometer.*

Theorem 1.19 ((ϵ, δ)-DP filters and privacy-loss accumulators).

(1) $\mathcal{F}$ is a valid (ϵ, δ)-DP NIM-filter if and only if $\mathcal{F}$ is a valid (ϵ, δ)-DP IM-filter.
(2) $\mathcal{G}$ is a valid (ϵ, δ)-DP NIM-privacy-loss accumulator if and only if $\mathcal{G}$ is a valid (ϵ, δ)-DP IM-privacy-loss accumulator.

Theorem 1.20 (f-DP filters and privacy-loss accumulators).

(1) $\mathcal{F}$ is a valid f-DP NIM-filter if and only if $\mathcal{F}$ is a valid f-DP IM-filter.
(2) $\mathcal{G}$ is a valid f-DP NIM-privacy-loss accumulator if and only if $\mathcal{G}$ is a valid f-DP IM-privacy-loss accumulator.

Theorem 1.21 (RDP filters and privacy-loss accumulators).

(1) $\mathcal{F}(\epsilon_1, \epsilon_2, \ldots; \epsilon) = \mathbb{I}(\sum_i \epsilon_i \le \epsilon)$ is a valid (α, ϵ)-RDP IM-filter for every fixed order of $\alpha > 1$.
(2) $\mathcal{G}(\epsilon_1, \ldots, \epsilon_i) = \sum_{i=1}^{i} \epsilon_i$ is a valid (α, ϵ)-RDP IM-privacy loss accumulator for every fixed order of $\alpha > 1$.

These theorems offer strong theoretical foundations for enabling *full adaptivity* in composing differentially private interactive mechanisms. Firstly, they allow data analysts to adaptively choose privacy loss parameters as well as the subsequent interactive analyses. By adjusting the strength of the privacy guarantee to reflect the actual needs of the analysis as they go along, this important feature allows us to optimize the trade-off between privacy and utility. Second, our results are particularly useful for DP libraries. For example, OpenDP[2] and Tumult Labs[3] have interactive mechanisms as a core abstraction. Prior to our work, supporting adaptive selection of privacy-loss parameters has necessitated enforcing sequentiality on the use of interactive mechanisms. This restriction makes the libraries less user-friendly, as we cannot allow analysts or even DP programs interact with multiple interactive mechanisms simultaneously. Therefore, there is a gap between the existing and desired functionality. It also increases complexity in the libraries, necessitating a control system to prevent interleaving queries. Given the results in our paper, we can now remove this interleaving restriction, while maintaining the *full adaptivity* feature. We provide further discussion and our implementation for privacy filters and odometers for interactive mechanisms in Section 5.

[2] https://opendp.org
[3] https://www.tmlt.io

1.5 Proof Strategy

In this section, we explain our proof strategy, and we defer the detailed proof to Section 3 and Section 4.

To prove the above theorems, our strategy is to leverage interactive postprocessing from interactive mechanisms to interactive mechanisms. We note that previous work [22] only considers interactive postprocessing from non-interactive mechanisms to interactive mechanisms. Here, we provide a general formulation called "person-in-the-middle", which is formally defined in Definition 2.15.

A *person-in-the-middle (PIM) mechanism* is a randomized mechanism that acts as an interlocutor between two interacting mechanisms $\mathcal{A}$ and $\mathcal{M}$. When $\mathcal{A}$ attempts to send a message to $\mathcal{M}$, the PIM mechanism can undergo an interaction with $\mathcal{A}$ until finally passing on the message at the end of this interaction to $\mathcal{M}$. Similarly, when $\mathcal{M}$ attempts to send a message to $\mathcal{A}$, the PIM mechanism can interact with $\mathcal{M}$ to modify the message before passing it to $\mathcal{A}$.

This construction of the postprocessing by PIM mechanism allows us the following theorem.

Theorem 1.22 (Privacy preserved under postprocessing PIM mechanism, informally stated). *If $\mathcal{P}$ is an interactive postprocessing mechanism and $\mathcal{M}$ is a differentially-private interactive mechanism with respect to any privacy measure (such as (ϵ, δ)-DP, f-DP, and Rényi-DP), then $\mathcal{P} \circ \mathcal{M}$ is also differentially-private with the same privacy-loss parameters.*

A useful corollary follows from Theorem 1.22.

Corollary 1.23. *Suppose $\mathcal{N}$ is an interactive mechanism over database space $\mathcal{X}$ such that for every pair of neighboring datasets x, x' and every deterministic adversary $\mathcal{A}$, there exists an (ϵ, δ)-DP I.M. $\mathcal{M}$ and an interactive postprocessing $\mathcal{P}$ such that*

$$\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{N}(x)) \equiv \texttt{View}(\mathcal{A} \leftrightarrow \mathcal{P}(\mathcal{M}(x)))$$
$$\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{N}(x')) \equiv \texttt{View}(\mathcal{A} \leftrightarrow \mathcal{P}(\mathcal{M}(x')))$$

Then $\mathcal{N}$ is (ϵ, δ)-DP.

We will use this corollary to prove Theorem 1.20. Our proof also relies on the following reduction theorem [22] showing that every interactive f-DP mechanism can be simulated by an interactive postprocessing of a noninteractive mechanism, for a fixed pair of neighboring datasets x, x'.

Theorem 1.24 ([22]). *For every trade-off function f, every interactive f-DP mechanism $\mathcal{M}$ with finite communication, and every pair of neighboring datasets x, x', there exists a noninteractive f-DP mechanism $\mathcal{N}$ and a randomized interactive postprocessing mechanism $\mathcal{P}$ such that for every adversary $\mathcal{A}$, we have*

$$\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{M}(x)) \equiv \texttt{View}(\mathcal{A} \leftrightarrow \mathcal{P}(\mathcal{N}(x)))$$
$$\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{M}(x')) \equiv \texttt{View}(\mathcal{A} \leftrightarrow \mathcal{P}(\mathcal{N}(x')))$$

Theorem 1.24 allows us to reduce the composition of interactive mechanisms $\mathcal{M}$ to the composition of noninteractive mechanisms $\mathcal{N}$, as needed to prove Theorem 1.20. Fixing adjacent datasets x, x',

we can then define an interactive postprocessing $\mathcal{P}$ such that for every deterministic adversary $\mathcal{A}$, $\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{P} \circ \mathcal{F}\text{-}Filt(NIM)) \equiv \texttt{View}(\mathcal{A} \leftrightarrow \mathcal{F}\text{-}FiltCon(IM))$ on any database x. Therefore we can use composition theorems for noninteractive f-DP mechanisms to construct concurrent f-DP filters. Theorem 1.19 follows directly from Theorem 1.20, as f-DP captures (ϵ, δ)-DP as a special case. In fact, the special case of of Theorem 1.24 for (ϵ, δ)-DP was shown independently by Lyu [15].

Our proof of Theorem 1.21 for RDP follows a different approach. Our proof relies on the concurrent composition theorem in Theorem 1.9 for two interactive RDP mechanisms, and our strategy is to apply induction on the number of mechanisms being composed.

When composing two mechanisms (i.e. $K = 2$ in Theorem 1.21), by the concurrent composition theorem in Theorem 1.9, an adversary can only start a second mechanism $\mathcal{M}_2$ after $\mathcal{M}_1$ if and only if its privacy-loss parameter ϵ_2 is at most $\epsilon - \epsilon_1$, where ϵ is the target privacy-loss budget and ϵ_1 is the privacy-loss parameter for $\mathcal{M}_1$. We define the following filter

$$\mathcal{F}_2(\epsilon_1, \ldots, \epsilon_k; \epsilon) = \begin{cases} \mathbb{I}(\sum_{i=1}^{k} \epsilon_i \leq \epsilon) & \text{if } k \leq 2 \\ 0 & \text{otherwise.} \end{cases}$$

$\mathcal{F}_2$ is a valid (α, ϵ)-RDP IM-filter for composing two mechanisms. This is because if we fix a deterministic adversary $\mathcal{A}$ as Lemma 2.3 allows us to do, then $\mathcal{M}_1, \epsilon_1$ are predetermined adaptively, which means that the adversary interacting with $\mathcal{F}_2$-*FiltCon(IM)* is equivalent to the adversary interacting with $\text{ConComp}(\mathcal{M}_1, \mathcal{U}_2)$, where $\mathcal{U}_2$ is a universal $(\alpha, \epsilon - \epsilon_1)$-*RDP* mechanism, and the adversary's first query to $\mathcal{U}_2$ is an (α, ϵ_2)-*RDP* mechanism $\mathcal{M}_2$. Upon verifying $\epsilon_2 \leq \epsilon - \epsilon_1$, $\mathcal{U}_2$ interacts with the adversary just as $\mathcal{M}_2(x)$ does. We then apply an induction argument on the number of mechanisms being composed. For $K > 2$, we consider $\mathcal{F}_K\text{-}FiltCon(IM)$, which is a filter for composing up to K interactive mechanisms. Assuming the privacy of $\mathcal{F}_{K-1}\text{-}FiltCon(IM)$, we argue that we can construct a postprocessing $\mathcal{P}$ interacting with $\mathcal{F}_2\text{-}FiltCon(IM)$ such that an adversary's interaction with $\mathcal{P} \circ \mathcal{F}_2\text{-}FiltCon(IM)$ is equivalent to its interaction with with $\mathcal{F}_K\text{-}FiltCon(IM)$.

Crucially, the proof for Rényi DP uses the fact that Rényi DP of a fixed order α is measured by a single real number ϵ, and the optimal composition theorem is additive, i.e. $\epsilon = \epsilon_1 + \cdots + \epsilon_K$. This is not satisfied in (ϵ, δ)-DP or f-DP, hence we use a different strategy in Theorem 1.20, as discussed above.

Feldman and Zrnic [10] prove that $\mathcal{F}(\epsilon_1, \epsilon_2, \ldots; \epsilon) = \mathbb{I}(\sum_i \epsilon_i \leq \epsilon)$ is a valid RDP NIM-filter for every fixed order of $\alpha > 1$. They construct a supermartingale for the privacy loss, and then apply the optional stopping theorem to bound the overall privacy loss when the algorithm halts. The main idea of our proof strategy is to leverage interactivity, which allows us to proceed by induction, describing $\mathcal{F}_k$ in terms of $\mathcal{F}_{k-1}$. This strategy can significantly simplify the proof in [10] of the privacy filter for even noninteractive RDP mechanisms, by similarly doing induction that uses the sequential composition of two interactive RDP mechanisms.

We then can convert this privacy filter to a valid RDP IM-privacy loss accumulator based on the following lemma.

Lemma 1.25. *A function $\mathcal{G}$ is a valid RDP IM-privacy loss accumulator if and only if $\mathcal{F}(\cdot; \epsilon) = I(\mathcal{G}(\cdot) \leq \epsilon)$ is a valid RDP IM-filter.*

We give a generalized version of Lemma 1.25 in Section 2, meaning that an analogue of Lemma 1.25 also holds for all other variants of DP.

2 CONCURRENT FILTER AND ODOMETER

2.1 Preliminaries

Vadhan and Zhang [22] define the generalized notion of privacy, termed $\mathcal{D}$ DP, based on the following generalized probability distance. The distance is defined on a partially ordered set, which allows us to compare privacy guarantees at different protection levels. It should enjoy the postprocessing and joint convexity properties.

Definition 2.1 (Generalized probability distance [22]). *A* generalized probability distance measure *is a tuple $(\mathcal{D}, \preceq, D)$ such that*

(1) $(\mathcal{D}, \preceq)$ is a partially ordered set (poset).
(2) D is a mapping that takes any two random variables X, X' over the same measurable space to an element $D(X, X')$ of $\mathcal{D}$.
(3) (Postprocessing.) The generalized distance mapping D is closed under postprocessing, meaning that for every measurable function g, $D(g(X), g(X')) \preceq D(X, X')$.
(4) (Joint Convexity.) Suppose we have a collection of random variables $(X_i, X_i')_{i \in \mathcal{I}}$ and a random variable I distributed on $\mathcal{I}$. If $D(X_i, X_i') \preceq d$ for all $i \in \mathcal{I}$, then $D(X_I, X_I') \preceq d$.

With this definition, the difficulty of distinguishing two neighboring datasets is measured by the generalized distance between the distributions of an adversary's views.

Definition 2.2 ($\mathcal{D}$ DP [22]). *Let $(\mathcal{D}, \preceq, D)$ be a generalized probability distance. For $d \in \mathcal{D}$, we call an interactive mechanism $\mathcal{M}$ d-$\mathcal{D}$* DP *if for every interactive algorithm $\mathcal{A}$ and every pair of neighboring datasets x, x', we have*

$$D(\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{M}(x)), \texttt{View}(\mathcal{A} \leftrightarrow \mathcal{M}(x'))) \preceq d.$$

A convenient property of $\mathcal{D}$ DP is that it suffices to consider deterministic adversaries. Since the filters and odometers we define will be $\mathcal{D}$ DP, we will only consider deterministic adversaries in this paper.

Lemma 2.3 (Deterministic adversaries for d-$\mathcal{D}$ DP [21][22]). *An interactive mechanism M is d-$\mathcal{D}$ DP, if and only if for every pair of neighboring datasets x, x', for every deterministic adversary $\mathcal{A}$,*

$$D(\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{M}(x)) || \texttt{View}(\mathcal{A} \leftrightarrow \mathcal{M}(x'))) \preceq d.$$

For the rest of this paper, we will notate $\mathcal{D}^* = \bigcup_{k=0}^{\infty} \mathcal{D}^k$.

Finally, we define concurrent composition formally. In this setting, the adversary can arbitrarily and adaptively interleave queries between several differentially-private mechanisms, meaning that the queries can be dependent.

Definition 2.4 (Concurrent composition of interactive mechanisms [21]). *Let $\mathcal{M}_1, \ldots, \mathcal{M}_k$ be interactive mechanisms taking private datasets $x_1, \ldots, x_k$ respectively. The concurrent composition of $\mathcal{M}_1, \ldots, \mathcal{M}_k$, denoted $\mathcal{M} = \text{ConComp}(\mathcal{M}_1, \ldots, \mathcal{M}_k)$, is defined as in Algorithm 1.*

Algorithm 1 Concurrent composition of interactive mechanisms

procedure $\mathcal{M}(s, m)$:
  if $m = \lambda$ **then**
    $s \leftarrow (x, [(\mathcal{M}_1, s_1), \ldots, (\mathcal{M}_k, s_k)])$ ▷ initialize k mechanisms
    return (s, λ)
  end if
  Parse $s = (x, [(\mathcal{M}_1, s_1), \ldots, (\mathcal{M}_k, s_k)])$
  if $m = (j, q)$ where $j = 1, \ldots, k$ and q is a query to $\mathcal{M}_j$ **then**
    $(s'_j, m') \leftarrow \mathcal{M}_j(s_j, q)$
    $s' \leftarrow (x, [(\mathcal{M}_1, s_1), \ldots, (\mathcal{M}_j, s'_j), \ldots, (\mathcal{M}_k, s_k)])$
  else if m cannot be parsed correctly **then**
    $s' \leftarrow s$, $m' \leftarrow$ `invalid query`
  end if
  return (s', m')
end procedure

2.2 Filters

We can now define a generalization of the privacy filter for the composition of noninteractive mechanisms as introduced in Section 1.3. Recall that filters are based on continuation rules. In line with the fully-adaptive setting, the number of interactive mechanisms need not be specified beforehand.

Definition 2.5 ($\mathcal{F}$-filtered composition of $\mathcal{D}$ DP noninteractive mechanisms ($\mathcal{F}$-*Filt*(NIM))). *Let $\mathcal{F}(\cdot; d) : \mathcal{D}^* \times \mathcal{D} \to \{1, 0\}$. The $\mathcal{F}$-filtered composition of $\mathcal{D}$ DP noninteractive mechanisms, denoted as $\mathcal{F}$-Filt(NIM), is an interactive mechanism executed in Algorithm 2.*

If there is a finite maximum K of noninteractive privacy-loss parameters $d_1, d_2, \ldots, d_K$ to take in, we denote the continuation rule $\mathcal{F}_K(\cdot; d) : \mathcal{D}^{\leq K} \times \mathcal{D} \to \{1, 0\}$, where d is the target privacy budget.

Definition 2.6 (Valid $\mathcal{D}$ DP filter for noninteractive mechanisms). *Let $\mathcal{F}$ be a continuation rule $\mathcal{F} : \mathcal{D}^* \times \mathcal{D} \to \{0, 1\}$. We say $\mathcal{F}$ is a valid $\mathcal{D}$ DP* NIM-filter *for noninteractive mechanisms if for every $d \in \mathcal{D}$, $\mathcal{F}(\cdot; d)$-Filt(NIM) is a d-$\mathcal{D}$ DP interactive mechanism.*

We can now similarly define the privacy filter for concurrent composition of interactive mechanisms.

Definition 2.7 ($\mathcal{F}$-filtered concurrent composition of $\mathcal{D}$ DP interactive mechanisms). *The $\mathcal{F}$-filtered concurrent composition of $\mathcal{D}$ DP interactive mechanisms, denoted as $\mathcal{F}$-FiltCon(IM), is an interactive mechanism as executed in Algorithm 3.*

Algorithm 2 $\mathcal{F}$-filtered composition of $\mathcal{D}$ DP noninteractive mechanisms ($\mathcal{F}$-*Filt*(NIM)), for $\mathcal{F} : \mathcal{D}^{\leq K} \to \mathcal{D}$, where K could be ∞.

procedure $\mathcal{F}(\cdot; d)$-*Filt*(NIM)(s, m):
  if $m = \lambda$ **then** ▷ initialize filter
    $s' \leftarrow (s, [])$, where [] is an empty list
    return (s', λ)
  end if
  Parse s as $s = (x, [(\mathcal{M}_1, d_1), (\mathcal{M}_2, d_2), \ldots, (\mathcal{M}_k, d_k)])$
  if $m = (\mathcal{M}', d')$ **then**
    if $\mathcal{M}'$ is d'-$\mathcal{D}$ DP **then**
      $\mathcal{M}_{k+1} \leftarrow \mathcal{M}'$
      $d_{k+1} \leftarrow d'$
      if $\mathcal{F}(d_1, \ldots, d_{k+1}) \neq 1$ **then**
        $m' \leftarrow$ `insufficient budget`, $s' \leftarrow$ `Halt`
      else
        Random sample $r_{k+1} :=$ coin toss for $\mathcal{M}_{k+1}$
        $s' \leftarrow (x, [(\mathcal{M}_1, d_1), \ldots, (\mathcal{M}_{k+1}, d_{k+1})])$
        $m' \leftarrow \mathcal{M}_{k+1}(x; r_{k+1})$
      end if
    end if
  else if m cannot be parsed correctly **then**
    $s' \leftarrow s$, $m' \leftarrow$ `invalid query`
  end if
  return (s', m')
end procedure

Algorithm 3 $\mathcal{F}$-filtered concurrent composition of $\mathcal{D}$ DP interactive mechanisms ($\mathcal{F}$-*FiltCon*(IM)), for $\mathcal{F} : \mathcal{D}^{\leq K} \to \mathcal{D}$, where K could be ∞.

procedure $\mathcal{F}(\cdot; d)$-*FiltCon*(IM)(s, m):
  if $m = \lambda$ **then** ▷ initialize filter
    $s' \leftarrow (s, [])$, where [] is an empty list
    return (s', λ)
  end if
  Parse s as $s = (x, [(\mathcal{M}_1, d_1, s_1), \ldots, (\mathcal{M}_k, d_k, s_k)])$
  if $m = (\mathcal{M}', d')$ **then**
    if $\mathcal{M}'$ is d'-$\mathcal{D}$ DP **then**
      $\mathcal{M}_{k+1} \leftarrow \mathcal{M}'$
      $d_{k+1} \leftarrow d'$
      if $\mathcal{F}(d_1, \ldots, d_{k+1}) \neq 1$ **then**
        $s' \leftarrow$ `Halt`, $m' \leftarrow$ `insufficient budget`
      else
        $(s_{k+1}, m) \leftarrow \mathcal{M}_{k+1}(x, \lambda)$
        $s' \leftarrow (x, d, [(\mathcal{M}_1, d_1, s_1), \ldots, (\mathcal{M}_{k+1}, d_{k+1}, s_{k+1})])$
        $m' \leftarrow$ `yes`
      end if
    end if
  else if $m = (j, q)$ where $j = 1, \ldots, k$ and q is a query to $\mathcal{M}_j$ **then**
    $(s'_j, m_{i+1}) \leftarrow \mathcal{M}_j(s_j, q)$
    $s' \leftarrow (x, d, [((\mathcal{M}_1, d_1, s'_1), \ldots, (\mathcal{M}_j, d_j, s'_j), \ldots), \ldots, (\mathcal{M}_k, d_k, s'_k)])$
  else if m_i cannot be parsed correctly **then**
    $s' \leftarrow s$, $m' \leftarrow$ `invalid query`
  end if
  return (s', m')
end procedure

A privacy filter is one in which Definition 2.7 holds for all distances $d \in \mathcal{D}$.

Definition 2.8 (Valid $\mathcal{D}$ DP filter for interactive mechanisms). *Let $\mathcal{F}$ be a continuation rule $\mathcal{F} : \mathcal{D}^* \times \mathcal{D} \to \{0, 1\}$, where $\mathcal{D}^* = \bigcup_{k=0}^{\infty} \mathcal{D}^k$. We say $\mathcal{F}$ is a valid $\mathcal{D}$ DP concurrent* IM-filter *if for every $d \in \mathcal{D}$, $\mathcal{F}(\cdot; d)$-FiltCon(IM) is a d-$\mathcal{D}$ DP interactive mechanism.*

2.3 Odometer

We can similarly generalize our definition for privacy odometers to $\mathcal{D}$ DP.

Definition 2.9 ($\mathcal{G}$-odometer for composition of noninteractive mechanisms ($\mathcal{G}$-Odom(NIM))). *A $\mathcal{G}$-odometer for the composition of noninteractive $\mathcal{D}$ DP mechanisms, denoted as $\mathcal{G}$-Odom(NIM), is executed as in Algorithm 4.*

Algorithm 4 $\mathcal{G}$-odometer for composition of noninteractive mechanisms ($\mathcal{G}$-Odom(NIM)) .

procedure $\mathcal{G}$-Odom$(NIM)(s, m)$:
  if $m = \lambda$ **then**
    $s' \leftarrow (s, [])$, where [] is an empty list
    return (s', λ)
  end if
  Parse s as $s = (x, [(\mathcal{M}_1, d_1), (\mathcal{M}_2, d_2), \ldots, (\mathcal{M}_k, d_k)])$
  if $m = (\mathcal{M}', d')$ **then**
    if $\mathcal{M}'$ is d'-$\mathcal{D}$ DP **then**
      $\mathcal{M}_{k+1} \leftarrow \mathcal{M}'$, $d_{k+1} \leftarrow d'$
      $s' \leftarrow (x, [(\mathcal{M}_1, d_1), \ldots, (\mathcal{M}_{k+1}, d_{k+1})])$
      Random sample $r_{k+1} :=$ coin toss for $\mathcal{M}_{k+1}$
      $m' \leftarrow \mathcal{M}_{k+1}(x; r_{k+1})$
    else
      $s' \leftarrow s_i$, $m' \leftarrow$ `Divergence cannot be parsed`
    end if
  else if $m =$ `privacy_loss` **then**
    $s' \leftarrow s, m' \leftarrow \mathcal{G}(d_1, \ldots, d_k)$
  else if m cannot be parsed correctly **then**
    $s' \leftarrow s, m' \leftarrow$ `invalid query`
  end if
  return (s', m')
end procedure

Odometers come with a per-mechanism privacy guarantee, meaning that the divergence between the views after each new mechanism is started should be at most some set distance $d \in \mathcal{D}$ apart. Because the odometer can receive a `privacy_loss` query at any point in computation, a notion that captures the during-computation guarantee is necessary. The generalized *truncated view* of the adversary up to the n-th round of interaction to an odometer equipped with continuation rule $\mathcal{G}$ is defined as in Algorithm 5.

Definition 2.10 (Truncated view for odometers [16]). *For a deterministic adversary $\mathcal{A}$ and odometer O on dataset x, define the truncated view*

$$\texttt{Trunc}_d(\texttt{View}(\mathcal{A} \leftrightarrow O(x)))$$

as the return value including the randomness and query answers of Algorithm 5.

Algorithm 5 Truncated view of adversary interacting with odometer O given input d

Initialize $s_1^{\mathcal{M}} \leftarrow x$, $m_0 \leftarrow \lambda$, $s_1^{\mathcal{A}} \leftarrow \lambda$
for $i = 1, 2, \ldots$ **do**
  if i is odd **then**
    $(s_{i+1}^{\mathcal{M}}, a_i) \leftarrow O(s_i^{\mathcal{M}}, q_{i-1})$
    $(s_i^{\mathcal{M}}, d_i) = O(s_i^{\mathcal{M}},$ `privacy_loss`$)$
    $s_{i+1}^{\mathcal{A}} \leftarrow s_i^{\mathcal{A}}$
    if $d_i \not\preceq d$ **then**
      Exit loop
    end if
  else if i is even **then**
    $(s_{i+1}^{\mathcal{A}}, q_i) \leftarrow \mathcal{A}(s_i^{\mathcal{A}}, a_{i-1}; r_i^{\mathcal{A}})$, $s_{i+1}^{\mathcal{M}} \leftarrow s_i^{\mathcal{M}}$
  end if
  if $a_i =$ `halt` **then**
    Exit loop
  end if
end for
return $\texttt{Trunc}_d(\texttt{View}(\mathcal{A} \leftrightarrow O(x))) = (r_0^{\mathcal{A}}, a_1, \ldots, r_{i-2}^{\mathcal{A}}, a_{i-1})$

Having generalized truncated view to $\mathcal{D}$ DP, we can now define a *valid* $\mathcal{D}$ DP odometer, and a valid privacy-loss accumulator for an odometer of noninteractive mechanisms.

Definition 2.11 (Valid $\mathcal{D}$-DP odometer). *We say O is a valid $\mathcal{D}$ DP odometer if for every $d \in \mathcal{D}$, adversary $\mathcal{A}$, and every pair of adjacent datasets x, x',*

$$D(\texttt{Trunc}_d(\texttt{View}(\mathcal{A} \leftrightarrow O(x)))||\texttt{Trunc}_d(\texttt{View}(\mathcal{A} \leftrightarrow O(x')))) \preceq d$$

Definition 2.12 (Valid $\mathcal{D}$-DP privacy-loss accumulator for noninteractive mechanisms). *Let $\mathcal{G} : \mathcal{D}^* \to \mathcal{D}$. We say $\mathcal{G}$ is a valid $\mathcal{D}$ DP NIM-privacy-loss accumulator if for every $d \in \mathcal{D}$, $\mathcal{G}(\cdot; d)$-Filt(NIM) is a valid d-$\mathcal{D}$ DP odometer.*

We now define the privacy odometer that keeps track of the total privacy loss across multiple interactive mechanisms, where the adversary can interleave queries to existing interactive mechanisms, as well as create new interactive mechanisms with privacy-loss budgets chosen adaptively. Crucially, our construction of odometers have a specially-designated privacy-loss query, which outputs a conservative upper bound on the privacy loss for the interaction up to that point. It should be noted that this odometer model for I.M.s does not define a per-query privacy loss as in the odometer for noninteractive mechanisms. Instead, the "privacy-loss budget" is paid at the launch of each additional interactive mechanism. Any `privacy_loss` queries asked between mechanism launches will return the same value.

Definition 2.13 ($\mathcal{G}$-odometer for concurrent composition of interactive mechanisms ($\mathcal{G}$-OdomCon(IM))). *A $\mathcal{G}$-odometer for the concurrent composition of interactive mechanisms, denoted as $\mathcal{G}$-OdomCon(IM), is executed as in Algorithm 6.*

We are now ready to define the notion of a *valid* $\mathcal{D}$ DP privacy-loss accumulator.

Algorithm 6 $\mathcal{G}$-odometer for concurrent composition of interactive mechanisms ($\mathcal{G}$-*OdomCon*(IM)).

procedure $\mathcal{G}$-*OdomCon*(IM)(s, m):
  if $m = \lambda$ **then**
    $s' \leftarrow (s, [])$, where [] is an empty list
    return (s', λ)
  end if
  Parse $s = (x, (\mathcal{M}_1, d_1, s_1), (\mathcal{M}_2, d_2, s_2), \ldots, (\mathcal{M}_k, d_k, s_k)))$
  if $m = (\mathcal{M}', d')$ **then**
    if $\mathcal{M}'$ is d'-$\mathcal{D}$ DP **then**
      $\mathcal{M}_{k+1} \leftarrow \mathcal{M}'$
      $d_{k+1} \leftarrow d'$
      $(s_{k+1}, m) \leftarrow \mathcal{M}_{k+1}(x, \lambda)$
      $s' \leftarrow (x, [(\mathcal{M}_1, d_1, s_1), \ldots, (\mathcal{M}_{k+1}, d_{k+1}, s_{k+1})])$
      $m' \leftarrow$ `yes`
    else
      $s' \leftarrow s, m' \leftarrow$ "`Divergence cannot be parsed`"
    end if
  else if $m = (j, q)$ where $j = 1, \ldots, k$ and q is a query to $\mathcal{M}_j$ **then**
    $(s'_j, m') \leftarrow \mathcal{M}_j(s_j, q)$
    $s' \leftarrow (x, [(\ldots, (\mathcal{M}_j, d_j, s'_j), \ldots)])$
  else if $m_i =$ `privacy_loss` **then**
    $s' \leftarrow s, m' \leftarrow \mathcal{G}(d_1, \ldots, d_k)$
  else if m cannot be parsed correctly **then**
    $s' \leftarrow s, m' \leftarrow$ `invalid query`
  end if
  return (s', m')
end procedure

Definition 2.14 (Valid $\mathcal{D}$ DP privacy-loss accumulator for interactive mechanisms). *We say $\mathcal{G}$ is a valid $\mathcal{D}$ DP* concurrent IM-privacy-loss accumulator *if for every $d \in \mathcal{D}$, $\mathcal{G}(\cdot; d)$-OdomCon(IM) is a valid d-$\mathcal{D}$ DP odometer.*

2.4 I.M.-to-I.M. Interactive Postprocessing

Before we move on to defining concurrent filters and odometers, we will formulate interactive postprocessing in terms of state machines, which will become a helpful tool for us to prove important theorems related to concurrent filters and odometers for f-DP and Rényi-DP mechanisms. We will first give a formal definition of person-in-the-middle mechanisms.

Definition 2.15 (Person-in-the-middle mechanism). *An* interactive postprocessing mechanism *(or 'PIM mechanism' for 'person-in-the-middle') is a randomized mechanism $\mathcal{P} : \{0,1\}^* \times \{Q, A\} \times \{0,1\}^* \to \{0,1\}^* \times \{Q, A\} \times \{0,1\}^*$. $\mathcal{P}$ takes its current state $s \in \{0,1\}^*$, a value $v \in \{Q, A\}$ to indicate whether it is receiving a query (from an analyst or adversary) or an answer (from the mechanism it is postprocessing), and a message m (which is either a query or an answer) and returns a new state $s' \in \{0,1\}^*$, a value $v' \in \{Q, A\}$ to indicate whether it is asking a query (of the mechanism it is postprocessing) or providing an answer (if it is answering a query) and message m' (which is either a query or an answer), denoted $(s', v', m') = \mathcal{P}(s, v, m)$.*

With the PIM primitive, we can now more precisely define a postprocessed interactive mechanism.

Definition 2.16 (Postprocessed interactive mechanism). *Let $\mathcal{M}$ be an interactive mechanism and $\mathcal{P}$ be an interactive postprocessing. Then the postprocessing of $\mathcal{M}$ by a PIM algorithm $\mathcal{P}$, denoted $\mathcal{P} \circ \mathcal{M}$, is the interactive mechanism defined in Algorithm 7.*

Algorithm 7 Postprocessing of $\mathcal{M}$ by PIM algorithm $\mathcal{P}$, denoted $\mathcal{P} \circ \mathcal{M}$.

procedure $\mathcal{P} \circ \mathcal{M}(s, m)$:
  if $m = \lambda$ **then**
    $s^{\mathcal{M}} \leftarrow x, s^{\mathcal{P}} \leftarrow \lambda, m \leftarrow \lambda, v \leftarrow A$ ▷ initialize $\mathcal{M}$ and $\mathcal{P}$
  else
    Parse s as $s = (s^{\mathcal{M}}, s^{\mathcal{P}})$, and let $v = Q$.
  end if
  while $v = Q$ **do** ▷ while $\mathcal{P}$ is equipped to interact with $\mathcal{M}$, conduct interaction
    $(s^{\mathcal{M}}, m) \leftarrow \mathcal{M}(s^{\mathcal{M}}, m)$
    $(s^{\mathcal{P}}, v, m) = \mathcal{P}(s^{\mathcal{P}}, A, m)$
  end while
  return $((s^{\mathcal{M}}, s^{\mathcal{P}}), m)$
end procedure

Theorem 2.17 (Privacy holds under I.M.-to-I.M. postprocessing). *If $\mathcal{P} : \{0,1\}^* \times \{Q, A\} \times \{0,1\}^* \to \{0,1\}^* \times \{Q, A\} \times \{0,1\}^*$ is an interactive postprocessing algorithm and $\mathcal{M}$ is a d-$\mathcal{D}$ DP interactive mechanism, then $\mathcal{P} \circ \mathcal{M}$ is d-$\mathcal{D}$ DP.*

Proof. Let $\mathcal{A}$ be the deterministic adversary against $\mathcal{P} \circ \mathcal{M}$. By Definition 1.6, $\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{P} \circ \mathcal{M}(x)) = (r_0, m_1, r_2, m_3, r_4, \ldots)$, where r_i are the random coins that $\mathcal{A}$ tosses and m_i are the messages received by $\mathcal{A}$. Define an adversary $\mathcal{A}'$ that maintains the states of $\mathcal{A}$ and $\mathcal{P}$ as submachines, constructed as in Algorithm 8.

Informally, Algorithm 8 says that when $\mathcal{A}'$ is to generate a query (determined by $v = Q$), it will run $\mathcal{A}$ with $\mathcal{P}$ to generate a query. When $\mathcal{A}'$ is to receive an answer (determined by $v = A$), it will run $\mathcal{P}$ to first check if $\mathcal{P}$ wants to interactively process it with $\mathcal{M}$ first. If so, the interaction between $\mathcal{P}$ and $\mathcal{M}$ would proceed until $\mathcal{P}$ determines that the answer is in a state to be passed to $\mathcal{A}$, done by checking the output value v_{out} of $\mathcal{P}$ given the state, the value A, and the message as inputs at that round.

Because $\mathcal{M}$ is d-$\mathcal{D}$ DP, the interaction between $\mathcal{A}'$ and $\mathcal{M}$ is d-$\mathcal{D}$ DP. Define a noninteractive postprocessing function g that transforms $\texttt{View}(\mathcal{A}' \leftrightarrow \mathcal{M}(x)) = (r'_0, m'_1, r'_2, m'_3, \ldots)$ on any dataset x as in Algorithm 9. Informally, the new view object generated by Algorithm 9 makes visible the submachine interaction between $\mathcal{A}$ and $\mathcal{P}$ in the operation of $\mathcal{A}'$, and snips the interaction between $\mathcal{A}'$ and $\mathcal{M}$ such that if there is an extended sequence between $\mathcal{M}$ and submachine $\mathcal{P}$ of $\mathcal{A}$, only the only the final answer from $\mathcal{M}$ before $\mathcal{P}$ and $\mathcal{A}$ begin interacting will be included. Since noninteractive postprocessing preserves privacy properties by Definition 2.1 (3), we know that $D(g(\texttt{View}(\mathcal{A}' \leftrightarrow \mathcal{M})(x))||g(\texttt{View}(\mathcal{A}' \leftrightarrow \mathcal{M})(x'))) \le D(\texttt{View}(\mathcal{A}' \leftrightarrow \mathcal{M})(x)||\texttt{View}(\mathcal{A}' \leftrightarrow \mathcal{M})(x')) \le d$.

The view of $\mathcal{A}$ against $\mathcal{P} \circ \mathcal{M}$ on any dataset x can be computed by postprocessing the view of $\mathcal{A}'$ against $\mathcal{M}$ on x using g; in other

Algorithm 8 Adversary $\mathcal{A}'$ that maintains the states of $\mathcal{A}$ and $\mathcal{P}$ as submachines.

procedure $\mathcal{A}'(s, m)$:
  if $s = \lambda$ **then** ▷ need to initialize $\mathcal{A}'$
    $(s^{\mathcal{A}}, m_0) \leftarrow \mathcal{A}(\lambda, \lambda)$ ▷ initialize adversary submachine
    $(s^{\mathcal{P}}, v, m') \leftarrow \mathcal{P}(\lambda, A, m_0)$ ▷ initialize PIM submachine
    return $((s^{\mathcal{A}}, s^{\mathcal{P}}), m')$
  end if
  Parse $s = (s^{\mathcal{A}}, s^{\mathcal{P}})$
  $v_{out} \leftarrow A$ ▷ initialize v_{out}
  if $v = Q$ **then**
    $j \leftarrow 1$ ▷ j indexes the internal communication $\mathcal{A} \leftrightarrow \mathcal{P}$
    while $v_{out} = A$ **do**
      Let $(s^{\mathcal{A}}_{j+1}, m_j) = \mathcal{A}(s^{\mathcal{A}}_j, m)$
      Let $(s^{\mathcal{P}}_{j+1}, v_{out}, m'_j) = \mathcal{P}(s^{\mathcal{P}}_{j+1}, v, m_j)$
      $j \leftarrow j + 1$
    end while
    $, s' \leftarrow (s^{\mathcal{A}}_j, s^{\mathcal{P}}_j), v \leftarrow A, m' \leftarrow m'_j$
  else if $v = A$ **then**
    $(s^{\mathcal{P}}, v_{out}, m') \leftarrow \mathcal{P}(s^{\mathcal{P}}, v, m)$
    $s' \leftarrow (s^{\mathcal{A}}, s^{\mathcal{P}})$
    if $v_{out} = A$ **then** ▷ $\mathcal{P}$ will provide an answer to $\mathcal{A}$
      $v \leftarrow Q$ ▷ prime $\mathcal{A}$ to ask a query
    end if
  end if
  return (s', m')
end procedure

words, $\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{P} \circ \mathcal{M}) \equiv g(\texttt{View}(\mathcal{A}' \leftrightarrow \mathcal{M}))$ on any dataset x. Therefore, $D(\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{P} \circ \mathcal{M})(x) || \texttt{View}(\mathcal{A} \leftrightarrow \mathcal{P} \circ \mathcal{M})(x')) \le d$. This means $\mathcal{P} \circ \mathcal{M}$ is also d-$\mathcal{D}$ DP. □

A useful corollary follows directly from Theorem 2.17.

Corollary 2.18. *Suppose $\mathcal{N}$ is an interactive mechanism over database space $\mathcal{X}$ such that for every pair of neighboring datasets x, x' and every deterministic adversary $\mathcal{A}$, there exists a d-$\mathcal{D}$ DP $\mathcal{M}$ and an interactive postprocessing $\mathcal{P}$ such that*

$$\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{N}(x)) \equiv \texttt{View}(\mathcal{A} \leftrightarrow \mathcal{P} \circ \mathcal{M}(x))$$
$$\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{N}(x')) \equiv \texttt{View}(\mathcal{A} \leftrightarrow \mathcal{P} \circ \mathcal{M}(x')).$$

Then $\mathcal{N}$ is d-$\mathcal{D}$ DP.

Proof. By Theorem 2.17, $\mathcal{P} \circ \mathcal{M}$ is d-$\mathcal{D}$ DP, which means for every pairs of datasets x, x', every adversary $\mathcal{A}$, we have $D(\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{N}(x)) || \texttt{View}(\mathcal{A} \leftrightarrow \mathcal{N}(x'))) = D(\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{P} \circ \mathcal{M}(x)) || \texttt{View}(\mathcal{A} \leftrightarrow \mathcal{P} \circ \mathcal{M}(x'))) \le d$. Therefore, $\mathcal{N}$ is also d-$\mathcal{D}$ DP. □

A particularly convenient property of privacy-loss accumulators and privacy filters is that a valid privacy-loss accumulator can be converted into a valid privacy filter, and vice versa. This bijective property allows us to use a filter to build an odometer, and an odometer to build a filter of the form $\mathbb{I}(\mathcal{G}(\cdot) \preceq d)$. Any nice properties of one would hold for the other.

Lemma 2.19. *(1) A function $\mathcal{G} : \mathcal{D}^* \to \mathcal{D}$, where $\mathcal{D}^* = \bigcup_{k=0}^{\infty} \mathcal{D}^k$, is a valid NIM-privacy-loss accumulator if and only if $\mathcal{F} : \mathcal{D}^* \times \mathcal{D} \to \{0, 1\}$ constructed from $\mathcal{G}(\cdot)$ such that $\mathcal{F}(\cdot; d) = \mathbb{I}(\mathcal{G}(\cdot) \preceq d)$ is a valid $\mathcal{D}$-DP NIM-filter.*

(2) A function $\mathcal{G} : \mathcal{D}^* \to \mathcal{D}$, where $\mathcal{D}^* = \bigcup_{k=0}^{\infty} \mathcal{D}^k$, is a valid concurrent IM-privacy-loss accumulator if and only if $\mathcal{F} : \mathcal{D}^* \times \mathcal{D} \to \{0, 1\}$ constructed from $\mathcal{G}(\cdot)$ such that $\mathcal{F}(\cdot; d) = \mathbb{I}(\mathcal{G}(\cdot) \preceq d)$ is a valid $\mathcal{D}$-DP concurrent IM-filter.

Algorithm 9 Noninteractive postprocessing g that transforms $\texttt{View}(\mathcal{A}' \leftrightarrow \mathcal{M}(x))$ to $\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{P} \circ \mathcal{M}(x))$

procedure $g(\texttt{View}(\mathcal{A}' \leftrightarrow \mathcal{M}(x)))$
  $i \leftarrow 1$ ▷ i indexes input
  $j \leftarrow 0$ ▷ j indexes output
  Parse $\texttt{View}(\mathcal{A}' \leftrightarrow \mathcal{M}(x))$ as $(r'_0, m'_1, r'_2, m'_3, \dots)$
  $v, \leftarrow Q, v_{out} \leftarrow A$
  while m_i **do**
    if $v = Q$ **then** ▷ adding interaction $\mathcal{A} \leftrightarrow \mathcal{P}$
      Store $(s^{\mathcal{A}}, m^{\mathcal{A}}) \leftarrow (s^{\mathcal{A}}_i, m'_i)$ as temporary variable
      $(s^{\mathcal{P}}, v_{out}, m^{\mathcal{P}}) \leftarrow (s^{\mathcal{P}}_i, v, \lambda)$
      while $v_{out} \neq Q$ **do** ▷ simulate $\mathcal{A} \leftrightarrow \mathcal{P}$ on m
        Let $(s^{\mathcal{A}}, m^{\mathcal{A}}) = \mathcal{A}(s^{\mathcal{A}}, m^{\mathcal{A}}; r^{\mathcal{A}}_j)$
        Let $(s^{\mathcal{P}}, v_{out}, m^{\mathcal{P}}) = \mathcal{P}(s^{\mathcal{P}}, v, m^{\mathcal{A}}; r^{\mathcal{P}}_j)$
        $m_{j+1} \leftarrow m^{\mathcal{P}}$ ▷ record message $\mathcal{A}$ receives
        $j \leftarrow j + 2$
      end while
      $v \leftarrow A$
    else if $v = A$ **then** ▷ abbreviating interaction $\mathcal{P} \leftrightarrow \mathcal{M}$
      $(s^{\mathcal{P}}, v_{out}, m) \leftarrow \mathcal{P}(s^{\mathcal{P}}_i, v, m'_i; r^{\mathcal{P}}_j)$
      if $v_{out} = Q$ **then**
        $v \leftarrow A$
      else if $v_{out} = A$ **then** ▷ record the history
        $m_{j+1} \leftarrow m, v \leftarrow Q$
        $j \leftarrow j + 2$
      end if
    end if
    $i \leftarrow i + 2$
  end while
  return $(r^{\mathcal{A}}_0, m_1, r^{\mathcal{A}}_2, m_3, \dots, r^{\mathcal{A}}_j, m_{j+1}, \dots)$
end procedure

The full proof of this theorem is enclosed in the appendix in the full version of this paper. The approach is to first define an I.M. $\mathcal{M}$ that such that for any adversary $\mathcal{A}$ and dataset x, $\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{M}(x))$ is exactly the same as $\texttt{Trunc}_d(\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{G}\text{-}OdomCon(IM)(x, \cdot)))$. Then, using Theorem 2.17, we can show that $\mathcal{M}$ is an *interactive post-processing* of $\mathcal{F}(\cdot; d)$-$FiltCon(IM)$.

3 CONCURRENT FILTER & ODOMETER FOR f-DP

f-DP is based on a hypothesis testing interpretation of differential privacy. Consider a hypothesis testing problem that attempts to quantify the difficulty of measuring the output of a mechanism on two adjacent datasets:

H_0: the dataset is x versus H_1: the dataset is x'

Let $Y := \mathcal{M}(x)$ and $Y' := \mathcal{M}(x')$ be the output distributions of mechanism $\mathcal{M}$ on neighboring datasets x, x'. For a given rejection rule ϕ, the type I error $\alpha_\phi = \mathbb{E}[\phi(Y)]$ is the probability of rejecting H_0 when H_0 is true, while the type II error $\beta_\phi = 1 - \mathbb{E}[\phi(Y')]$ is the probability of failing to reject H_0 when H_1 is true.

The trade-off function characterizes the optimal boundary between achievable type I and type II errors, which in our case is calculated by finding the minimal achievable type II error after fixing the type I error at any level.

Definition 3.1 (Trade-off function [3]). *For any two probability distributions Y and Y' on the same space, the* trade-off function $T(Y, Y') : [0,1] \to [0,1]$ *is defined as*

$$T(Y, Y')(\alpha) = \inf\{\beta_\phi : \alpha_\phi < \alpha\},$$

where the infimum is taken over all measurable rejection rules ϕ.

Definition 3.2 (f-DP [3]). *Let f be a tradeoff function. A mechanism $\mathcal{M} : \mathcal{X} \to \mathbb{R}$ is f-differentially private if for every pair of neighboring datasets $x, x' \in \mathcal{X}$, we have*

$$T(\mathcal{M}(x), \mathcal{M}(x')) \geq f.$$

In our generalized DP framework in Definition 2.1, the probability distance measure for f-DP is $(S, \preceq, T)$. The distance mapping is the trade-off function T in Definition 3.2. The partially ordered set $(S, \preceq)$ consists of the set S of all trade-off functions $g : [0,1] \to [0,1]$ such that g is convex, continuous, non-increasing, and $g(x) \leq 1 - x$ for $x \in [0,1]$. The partial ordering is defined as $f_1 \preceq f_2$ if $f_1(\alpha) \geq f_2(\alpha)$ holds for all $\alpha \in [0,1]$. A larger trade-off function means less privacy loss.

The recent result of Vadhan and Zhang [22] shows that every interactive f-DP mechanism can be simulated by an interactive postprocessing of a noninteractive f-DP mechanism.

Theorem 3.3 (Reduction of f-DP to noninteractive mechanism [22]). *For every $f \in \mathcal{F}$ and every interactive f-DP mechanism $\mathcal{M}$ with finite communication complexity, and every pair of neighboring datasets x and x', there exists a pair of random variables Y, Y' and a randomized interactive mechanism $\mathcal{P}$, which we call an* interactive postprocessing, *such that $D(Y, Y') \preceq f$, and for every deterministic adversary $\mathcal{A}$, we have*

$$\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{M}(x)) \equiv \texttt{View}(\mathcal{A} \leftrightarrow \mathcal{P}(Y)) \tag{1}$$

$$\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{M}(x')) \equiv \texttt{View}(\mathcal{A} \leftrightarrow \mathcal{P}(Y')). \tag{2}$$

Essentially, the interactive postprocessing function in Theorem 3.3 takes the output of the non-interactive mechanism as the input, and interacts with the adversary just as the interactive mechanism does. It will use the output of the non-interactive mechanism to simulate all the answers to the adversary. This result implies that for any fixed pair of datasets x, x', to analyze the concurrent composition of interactive mechanisms $\mathcal{M}_i$, it suffices to consider the composition of their noninteractive versions $\mathcal{N}_i$. As a result, composition theorems for noninteractive f-DP mechanisms extend to the concurrent composition of interactive f-DP mechanisms.

Since (ϵ, δ)-DP is a specific case of f-DP [3], where $f_{\epsilon,\delta} = \max\{0, 1 - \delta - \exp(\epsilon)\alpha, \exp(-\epsilon)(1 - \delta - \alpha)\}$, a corollary of this result is that an interactive (ϵ, δ)-DP mechanism can be postprocessed to a noninteractive (ϵ, δ)-DP mechanism, a result shown independently by Lyu [15].

3.1 Concurrent Filter for f-DP

Using Theorem 3.3, we derive a concurrent f-DP filter from an f-DP filter for noninteractive mechanisms.

Theorem 3.4 (Concurrent f-DP Filter). *Suppose that $\mathcal{F} : \mathcal{D}^* \to \{1, 0\}$ is a valid f-DP continuation rule for noninteractive mechanisms. Then $\mathcal{F}$ is a valid f-DP continuation rule for interactive mechanisms with finite communication.*

Proof. Fix a pair of adjacent datasets x, x'. For any such pair, we can define the I.M.-to-I.M. postprocessing $\mathcal{P}$ that takes $\mathcal{F}\text{-}Filt(NIM)$ to $\mathcal{F}\text{-}FiltCon(IM)$. For simplicity, we will assume that $\mathcal{A}$ is deterministic, as Lemma 2.3 allows us to do. By the postprocessing property of f-DP IMs given in Theorem 3.3, we know that an f_j-DP mechanism $\mathcal{M}_i$ can be simulated by an interactive postprocessing $\mathcal{P}_j$ of a noninteractive f_j-DP mechanism $\mathcal{N}_j$.

$\mathcal{P}$ is constructed as follows. $\mathcal{P}$ depends on whether it is primed to receive a query $v = Q$ from the adversary or an answer $v = A$ from the mechanism, to pass to the other party. If the adversary asks to start a new interactive mechanism, $\mathcal{P}$ will make note of the corresponding noninteractive mechanism and the noninteractive-to-interactive postprocessing and pass the request on $\mathcal{F}\text{-}FiltSeq(NIM)$. Otherwise, if the adversary queries an existing interactive mechanism $\mathcal{M}_j$, $\mathcal{P}$ will use the corresponding interactive postprocessing $\mathcal{P}_j$ to answer the query instead of passing the message forward to $\mathcal{M}$. The only time $\mathcal{P}$ will interact with $\mathcal{M}$ is to start a new mechanism and to pass back the confirmation that a new mechanism has begun to $\mathcal{A}$. The algorithmic pseudocode of $\mathcal{P}$ is included in the appendix in the full version of this paper.

Note that this algorithm only interacts with $\mathcal{F}\text{-}FiltSeq(NIM)$ in starting a new mechanism. For the rest of the interactive queries, the algorithm directly uses the interactive postprocessing of the corresponding noninteractive mechanism in question to answer the queries. By Theorem 3.3, we know that for every deterministic adversary $\mathcal{A}$ and any database x, there exists an $\mathcal{F}\text{-}Filt(NIM)$ that is f-DP and an interactive postprocessing $\mathcal{P}$ such that $\texttt{View}(\mathcal{A} \leftrightarrow \mathcal{P} \circ \mathcal{F}\text{-}Filt(NIM)(x)) \equiv \texttt{View}(\mathcal{A} \leftrightarrow \mathcal{F}\text{-}FiltCon(IM)(x))$. Because $\mathcal{F}\text{-}Filt(NIM)$ is f-DP, by Theorem 2.17, $\mathcal{F}\text{-}FiltCon(IM)$ is f-DP. □

An analogue of Theorem 3.4 holds for (ϵ, δ)-DP.

Corollary 3.5 (Concurrent (ϵ, δ) Filter). *Every filter for noninteractive (ϵ, δ)-DP mechanisms is also a concurrent filter of interactive (ϵ, δ)-DP mechanisms.*

Concurrent f-DP odometers can be defined similarly from f-DP odometers for noninteractive mechanisms.

Theorem 3.6 (Concurrent f-DP odometer). *Suppose that $\mathcal{G} : \mathcal{D}^* \to \mathcal{D}$ is a valid f-DP privacy-loss accumulator for noninteractive mechanisms. Then $\mathcal{G}$ is a valid f-DP privacy-loss accumulator for interactive mechanisms with finite communication.*

Proof. By Lemma 2.19, if $\mathcal{G}$ is a valid f-DP privacy-loss accumulator, then $\mathcal{F}(\cdot; f) = I(\mathcal{G}(\cdot) \preceq f)$ is a valid f-DP filter for noninteractive mechanisms. By Theorem 3.4, if $\mathcal{F}$ is an f-DP filter for noninteractive mechanisms, then $\mathcal{F}$ will be an f-DP concurrent filter for interactive mechanisms. Applying Lemma 2.19 again, $\mathcal{G}$ will be a valid f-DP privacy-loss accumulator for interactive

mechanisms, and $\mathcal{G}$-*OdomCon*(IM) is an f-DP interactive mechanism. □

Now we give an example of how to construct a valid (ϵ, δ)-DP privacy filter and odometer.

Theorem 3.7 ((ϵ, δ)-DP privacy filter [24]). *For every $\delta' > 0$,*

$$\mathcal{F}((\epsilon_1, \delta_1), \ldots, (\epsilon_k, \delta_k); (\epsilon, \delta))$$
$$= \mathbb{I}\left(\epsilon \leq \sqrt{2\log\left(\frac{1}{\delta'}\right)\sum_{m\leq k}\epsilon_m^2} + \frac{1}{2}\sum_{m\leq k}\epsilon_m^2\right) \cdot \mathbb{I}\left(\delta' + \sum_{m\leq k}\delta_k \leq \delta\right)$$

is a valid approx-DP continuation rule for noninteractive mechanisms and interactive mechanisms.

Theorem 3.8 ((ϵ, δ)-DP privacy odometer). *Let $\delta = \delta' + \delta''$ be a target approximation parameter such that $\delta' > 0$, $\delta'' \geq 0$. Then*

$$\mathcal{G}((\epsilon_1, \delta_1), \ldots, (\epsilon_k, \delta_k))$$
$$= \begin{cases} \left(\sqrt{2\log\left(\frac{1}{\delta'}\right)\sum_{m\leq k}\epsilon_m^2} + \frac{1}{2}\sum_{m\leq k}\epsilon_m^2, \delta\right) & \text{if } \delta' + \sum_{m\leq k}\delta_m \leq \delta \\ (\infty, \infty) & \text{otherwise} \end{cases}$$

is a valid approx-DP privacy-loss accumulator for noninteractive and interactive mechanisms.

4 CONCURRENT FILTER & ODOMETER FOR RÉNYI-DP

Rényi DP is a relaxation of DP based on Rényi divergences. In this section, we show that privacy loss adds up for Rényi DP, in the setting of concurrent composition with adaptive privacy-loss parameters. This bound matches naturally with the previous concurrent composition result where all privacy-loss parameters are fixed upfront.

Definition 4.1 (Rényi divergence [19]). *For two probability distributions P and Q, the* Rényi divergence *of order $\alpha > 1$ is*

$$D_\alpha(P||Q) = \frac{1}{\alpha - 1}\log \mathrm{E}_{x\sim Q}\left[\frac{P(x)}{Q(x)}\right]^\alpha.$$

Definition 4.2 (Rényi DP [17]). *A randomized mechanism $\mathcal{M}$ is (α, ϵ)-*Rényi differentially private *(ϵ-RDP$_\alpha$) if for all every two neighboring datasets x and x',*

$$D_\alpha(\mathcal{M}(x)||\mathcal{M}(x')) \leq \epsilon$$

In our generalized DP framework in Definition 2.1, for Rényi DP of order α, the partially ordered set $\mathcal{D}$ is $((\mathbb{R}^{\geq 0}) \cup \{\infty\}, \leq)$. The distance mapping is α-Rényi divergence for $\alpha \in (1, \infty)$. Rényi divergence is also closed under postprocessing due to the data-processing inequality, and it satisfies the joint convexity. We will notate RDP$_\alpha$ as the family of distance measures corresponding to RDP, where $\mathcal{D} = (\alpha, \cdot)$-RDP = RDP$_\alpha$.

Our proof leverages the $K = 2$ case of the concurrent composition theorem proved by Lyu [15], where the privacy-loss parameters are specified unfront.

Theorem 4.3 (Concurrent composition of Rényi-DP mechanisms [15]). *For all $\alpha > 1$, $k \in \mathbb{N}$, $\epsilon_1, \ldots, \epsilon_k > 0$, and all interactive mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_k$ such that $\mathcal{M}_i$ is ϵ_i-RDP$_\alpha$, the concurrent composition ConComp$(\mathcal{M}_1, \ldots, \mathcal{M}_k)$ is $\sum_{i=1}^k \epsilon_i$-RDP$_\alpha$.*

4.1 Concurrent Filter for Rényi-DP

Theorem 4.4 (Theorem 1.21 restated). *An RDP filter with the continuation rule $\mathcal{F}(\epsilon_1, \epsilon_2, \ldots; \epsilon) = \mathbb{I}(\sum_i \epsilon_i \leq \epsilon)$ is a valid RDP IM-filter for every fixed order of $\alpha > 1$.*

Though generalized probability distance also covers RDP, the reduction in Theorem 3.3 holds only if the probability distance also satisfies properties of coupling and chain rule [22], which RDP does not. Therefore, we use a different strategy to prove 4.4. We prove this theorem by inducting on a finite number of mechanisms being maintained by the $\mathcal{F}$-filtered concurrent composition of interactive mechanisms $\mathcal{F}$-*FiltCon*(IM), and then taking the limit of those mechanisms. To do so, we first consider $\mathcal{F}$-*FiltCon*(IM) maintains up to K mechanisms.

Lemma 4.5. *Let $\alpha > 1$ and define*

$$\mathcal{F}_K(\epsilon_1, \epsilon_2, \ldots, \epsilon_k; \epsilon) = \begin{cases} \mathbb{I}(\sum_{i=1}^k \epsilon_i \leq \epsilon) & \text{if } k \leq K \\ 0 & \text{otherwise} \end{cases}$$

Then $\mathcal{F}_K$ is a valid RDP$_\alpha$ IM-filter.

Proof. We prove this by inducting on the maximum number K of mechanisms that $\mathcal{F}_K$-*FiltCon*(IM) maintains. Fix a pair of datasets x, x'. By Lemma 2.3, it suffices to consider a deterministic adversary to prove this theorem.

Base case: $K = 2$. Intuitively, with two mechanisms, the budget partition is determined before any mechanisms are queried, so we can reduce the problem to concurrent RDP composition. Suppose the adversary starts a ϵ_1-RDP$_\alpha$ mechanism $\mathcal{M}_1$ through $\mathcal{F}_2$-*FiltCon*(IM). By the constraints of $\mathcal{F}_2$-*FiltCon*(IM), the second mechanism $\mathcal{M}_2$ it is able to start must have privacy parameter $\epsilon_2 \leq \epsilon - \epsilon_1$. The adversary can adaptively interleave queries between $\mathcal{M}_1$ and $\mathcal{M}_2$.

We will first define the notion of a *universal* mechanism to help us with the reduction. Informally, a "universal" mechanism is any mechanism with the privacy loss bounded by the preset privacy loss budget. The universal mechanism will first verify if a $\mathcal{M}$ attempted by an adversary to start satisfies the given privacy loss parameter guarantee as well as if the given privacy loss parameter does not exceed the preset privacy loss budget. Upon verifying this, it will interact with the adversary as $\mathcal{M}$. Given a preset privacy budget ϵ, an RDP$_\alpha$ universal mechanism $\mathcal{U}(s, m)$ would operate as follows:

(1) If $m = \lambda$, initialize $s = (x, ())$ and return (s, λ)
(2) If $m = (\mathcal{M}, \epsilon_1)$, where $\mathcal{M}$ is an RDP$_\alpha$ mechanism:
  (a) If $s = (x, ())$ and $\epsilon_1 \leq \epsilon$, let $s' = (x, (\mathcal{M}, \epsilon_1, s_\mathcal{M}))$ and $m' =$ `yes`
  (b) Else let $s' = s$ and $m' =$ `invalid query`
(3) If $m = q$, where q is a query:
  (a) If $s = (x, ())$, let $s' = s$ and $m' =$ `no mechanism to query`
  (b) Else parse $s = (x, (\mathcal{M}, \epsilon_1, s_\mathcal{M}))$:
    (i) Let $(s'_\mathcal{M}, m') = \mathcal{M}(s_\mathcal{M}, q)$
    (ii) Let $s' = (x, (\mathcal{M}, \epsilon_1, s'_\mathcal{M}))$
(4) Return (s', m')

Given an adversary $\mathcal{A}$ and an interaction between $\mathcal{A}$ and $\mathcal{F}_2$-*FiltCon*(IM), we will construct an adversary $\mathcal{A}'$ that interacts with ConComp$(\mathcal{M}_1, \mathcal{U}_2)$, where $\mathcal{U}_2$ is a universal $(\epsilon - \epsilon_1)$-RDP$_\alpha$ mechanism, and the adversary's first query to $\mathcal{U}_2$ is an ϵ_2-RDP$_\alpha$

mechanism $\mathcal{M}_2$. For each query $\mathcal{A}$ makes to $\mathcal{F}_2$-$FiltCon(IM)$ prior to starting $\mathcal{M}_2$, $\mathcal{A}'$ will make a query to $\mathcal{M}_1$. When $\mathcal{A}$ starts $\mathcal{M}_2$, $\mathcal{A}'$ will make its first query to $\mathcal{U}_2$. Upon verifying $\epsilon_2 \le \epsilon - \epsilon_1$, $\mathcal{U}_2$ interacts with the adversary just as $\mathcal{M}_2$ does. Afterward, $\mathcal{A}'$ can interleave queries at will between $\mathcal{M}_1$ and $\mathcal{M}_2$, interacting with $\mathrm{ConComp}(\mathcal{M}_1, \mathcal{U}_2)$ just as $\mathcal{A}$ does with $\mathcal{F}_2$-$FiltCon(IM)$.

Notice that in this case, $\mathtt{View}(\mathcal{A}' \leftrightarrow \mathrm{ConComp}(\mathcal{M}_1, \mathcal{U}_2)(x)) \equiv \mathtt{View}(\mathcal{A}' \leftrightarrow \mathrm{ConComp}(\mathcal{M}_1, \mathcal{M}_2)(x))$, and the same holds for the computation on x'. By Theorem 4.3, $\mathrm{ConComp}(\mathcal{M}_1, \mathcal{M}_2)$ is ϵ-RDP_α, which means $\mathcal{F}_2$-$FiltCon(IM)$ an ϵ-RDP_α I.M. By Definition 2.8, $\mathcal{F}_2(\cdot;\epsilon)$ is a valid RDP_α IM-filter.

Induction on K. We assume that $\mathcal{F}_{K-1}$-$FiltCon(IM)$ is differentially private. We define the postprocessing $\mathcal{P}$ interacting with $\mathcal{F}_2$-$FiltCon(IM)$ so that for any $K > 1$ and dataset x, $\mathtt{View}(\mathcal{A} \leftrightarrow \mathcal{P} \circ \mathcal{F}_2\text{-}FiltCon(IM)(x)) \equiv \mathtt{View}(\mathcal{A} \leftrightarrow \mathcal{F}_K\text{-}FiltCon(IM)(x))$. The algorithmic pseudocode of $\mathcal{P}$ is in the appendix in the full version of this paper.

Given a privacy-loss budget ϵ, suppose the adversary starts a ϵ_1-RDP_α mechanism $\mathcal{M}_1$ with $\epsilon_1 \le \epsilon$. Inductively assume $\mathcal{F}_{K-1}$-$FiltCon(IM)_{\alpha,\epsilon-\epsilon_1}(x,\cdot)$ is an $(\epsilon - \epsilon_1)$-RDP_α I.M. As soon as $\mathcal{M}_1$ gets created, $\mathcal{P}$ will feed in $\mathcal{M}_1$ to $\mathcal{F}_2$-$FiltCon(IM)$. Upon receiving confirmation that $\mathcal{M}_1$ has properly started, $\mathcal{P}$ will then feed in F_{k-1} to $\mathcal{F}_2$-$FiltCon(IM)$ with an $\epsilon - \epsilon_1$ budget. We don't need to handle the parsing in the $v = Q$ section because $\mathcal{P}$ will always start two mechanisms at once. We can view $\mathcal{F}_{K-1}$-$FiltCon(IM)$ as the first query to $\mathcal{U}_2$.

If the adversary attempts to query mechanism $\mathcal{M}_j$ where $j > 1$, then $\mathcal{P}$ will parse the query as a query to the $(j-1)$st mechanism of $\mathcal{F}_{K-1}$-$FiltCon(IM)$, which is the second mechanism maintained by $\mathcal{F}_2$-$FiltCon(IM)$. (Concretely, if the adversary thinks it's interacting concurrently with $\mathcal{F}_4$-$FiltCon(IM)$ maintaining $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4$, then $\mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4$ would be under $\mathcal{F}_3$-$FiltCon(IM)$ as $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ respectively, and we've assumed by induction that $\mathcal{F}_3$-$FiltCon(IM)$ is Rényi-differentially private.)

By construction, for any deterministic adversary $\mathcal{A}$ and dataset x, $\mathtt{View}(\mathcal{A} \leftrightarrow \mathcal{P} \circ \mathcal{F}_2\text{-}FiltCon(IM)(x))$ is identically distributed as $\mathtt{View}(\mathcal{A} \leftrightarrow \mathcal{F}_K\text{-}FiltCon(IM)(x))$. By Theorem 2.17, $\mathcal{P} \circ \mathcal{F}_2$-$FiltCon(IM)$ is an ϵ-RDP_α I.M., which means $\mathcal{F}_K$-$FiltCon(IM)$ is ϵ-RDP_α I.M. By Definition 2.8, $\mathcal{F}_K$ is a valid RDP_α IM-filter. □

To show that the filter works with an unbounded number of mechanisms, we examine the behavior as $K \to \infty$. We define a sequence of measurable spaces that are well-behaved in that we can marginalize out distributions even when the sequence is not finite. Suppose $(\Omega^\infty, \Gamma^\infty)$ is the direct product of an infinite sequence of measurable spaces $(\Omega_1, \Gamma_1), (\Omega_2, \Gamma_2), \ldots$, i.e. $\Omega^\infty = \Omega_1 \times \Omega_2 \times \ldots$ and Γ^∞ is the smallest σ-algebra containing all cylinder sets $S_n(A) = \{x^\infty \in \Omega^\infty \mid x_1, x_2, \ldots, x_n \in A\}$, $A \in \Gamma^n$, for $n = 1, 2, \ldots$ and $\Gamma^n = \Gamma_1 \otimes \Gamma_2 \otimes \cdots \otimes \Gamma_n$. We call a sequence of probability distributions $P^1, P^2, \ldots$ *consistent* if P^n is a distribution on $\Omega^n = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_n$ and $P^{i+1}(A \times \Omega_{i+1}) = P^i(A)$ for $A \in \Gamma^i$, $i = 1, \ldots, n$.

For any such sequence there exists a distribution P^∞ on $(\Omega^\infty, \Gamma^\infty)$ such that its marginal distribution on Ω^n is P^n, in the sense that $P^\infty(S_n(A)) = P^n(A)$, $A \in \Gamma^n$.

Lemma 4.6 ([23]). *Let $P^1, P^2, \ldots$ and $Q^1, Q^2, \ldots$ be consistent sequences of probability distributions on $(\Omega^1, \Gamma^1), (\Omega^2, \Gamma^2), \ldots$, where, for $n = 1, 2, \ldots, \infty$, (Ω^n, Γ^n) is the direct product of the first n measurable spaces in the infinite sequence $(\Omega_1, \Gamma_1), (\Omega_2, \Gamma_2), \ldots$. Then for any $a \in (0, \infty]$*

$$\lim_{n\to\infty} D_\alpha(P^n || Q^n) = D_\alpha(P^\infty || Q^\infty).$$

Proof of Theorem 4.4. Fix a pair of datasets x, x' and a divergence parameter $\alpha > 1$. Consider a $\mathcal{F}_k(\cdot;\epsilon)$-$FiltCon(IM)$ with exactly k queries in its View. (Note that this implies the adversary interacts with k queries at maximum, keeping within the constraints of the concurrent filter.) Let q_i denote the adversary's ith query and a_i denote the answer to the adversary's ith query.

Define sequences of random variables $X_1, \ldots, X_k$ and $Y_1, \ldots, Y_k$ where $X_i := (q_1, a_1, \ldots, q_i, a_i)$ on x and $Y_i := (q_1, a_1, \ldots, q_i, a_i)$ on x'. The corresponding distributions for X_i, Y_i are X^i, Y^i respectively. By construction, both the sequences of X_i and Y_i are consistent.

By Theorem 4.3, for every $k = 1, 2, \ldots$ and $\alpha > 1$, we have $D_\alpha(X_k || Y_k) \le \sum_{i=1}^k \epsilon_i$, where equality holds if each query starts a new mechanism. By construction, $\sum_{i=1}^k \epsilon_i \le \epsilon$. Then, invoking Lemma 4.6,

$$D_\alpha(X_\infty || Y_\infty) = \lim_{k\to\infty} D_\alpha(X_k || Y_k) \le \epsilon,$$

where the last inequality is because the sequence of $D_\alpha(X_k || Y_k)$ monotonically increases as $k \to \infty$, and is bounded above by ϵ. □

Our approach can be used to simplify the proof in [10] for the bounds on privacy filters for sequentially-composed noninteractive DP mechanisms. They identify the sequence of losses incurred for each query – in this case the invocation of a noninteractive mechanism and the reception of its answers – as a supermartingale. They then apply the optional stopping theorem for martingales to bound the divergence between adjacent datasets and verify the validity of the filter.

The basis for our proof is that concurrent composition of two interactive RDP mechanisms implies a fully adaptive $\mathcal{F}_2$-$FiltCon(IM)$. Having interactivity allows us to proceed by induction in the proof of Lemma 4.5, describing $\mathcal{F}_k$ in terms of $\mathcal{F}_{k-1}$. We could do the same for the sequential composition since induction preserves sequentiality. Then by Lemma 4.5, $\mathcal{F}_k$-$FiltSeq(IM)$ is an ϵ-RDP_α I.M. Thus, the sequential composition of two interactive RDP mechanisms of order α implies a fully adaptive filter for sequential composition of RDP mechanisms of order α.

4.2 Concurrent Odometer for Rényi-DP

Because of the bijective relationship between valid continuation rules and privacy-loss accumulators, we are able to define an additive relationship for concurrent odometers of Rényi mechanisms as well.

Theorem 4.7 (Valid ϵ-RDP_α odometer for concurrently-composed interactive mechanisms). *Let $\alpha > 1$. Let $\mathcal{G} : \mathcal{D}^* \to \mathcal{D}$ be a relation. If $\mathcal{G}(\epsilon_1, \ldots, \epsilon_K) = \sum_{i=1}^K \epsilon_i$, then $\mathcal{G}$ is a valid privacy-loss accumulator for the concurrent composition of a sequence of mechanisms $\mathcal{M}_1, \mathcal{M}_2, \ldots$, where $\mathcal{M}_i$ is an ϵ_i-RDP_α interactive mechanism.*

Proof. Define a relation $\mathcal{F}(\epsilon_1, \ldots, \epsilon_K; \epsilon) : \mathcal{D}^* \times \mathcal{D} \to \mathcal{D}$ such that $\mathcal{F}(\epsilon_1, \ldots, \epsilon_K; \epsilon) = \mathbb{I}(\mathcal{G}(\epsilon_1, \ldots, \epsilon_K \le \epsilon)$. By Theorem 4.4, $\mathcal{F}$

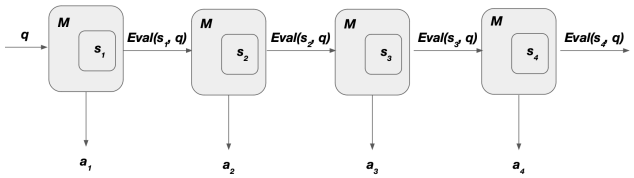

Figure 1: Diagram of a queryable.

is a valid RDP$_\alpha$ filter. By Lemma 2.19, $\mathcal{G}$ is a valid privacy-loss accumulator for Rényi mechanisms of order α. □

5 IMPLEMENTATION

In this section, we provide an overview of the implementation of differentially private mechanisms in the open-source software project OpenDP and the Tumult Analytics platform. We then discuss the implications of our results on the two platforms.

5.1 Practical Application in the OpenDP Library

The OpenDP Library [11] is a modular collection of statistical algorithms used to build differentially private computations. The OpenDP Library represents differentially private computations with measurements and odometers. Measurements and odometers fully characterize the privacy guarantees they give in terms of an input domain, input metric, and privacy measure. They both contain a function to make a differentially private release and a privacy map to reason about the privacy spend of a release. A privacy map is a function that takes in a bound on the distance between adjacent inputs/datasets (d_{in}), and translates it to a bound on the distance between respective outputs (d_{out}, a privacy parameter). From the perspective of the library, an *interactive measurement* is a measurement for which the function emits a *queryable* (Figure 1), OpenDP terminology for an object that implements an interactive mechanism. A queryable is modeled as an opaque state machine, consisting of an internal state value and a transition function. When an analyst passes a query into the queryable, the state is updated, and an answer is returned. All interactivity in the OpenDP Library uses queryables, and the privacy guarantee of an interactive measurement is defined exactly as in Definitions 1.7 and 2.2.

Interactive measurements are well-suited to capture interactive composition when the desired privacy parameters for each query are known up-front. For example, the internal state of a compositor queryable consists of the dataset and per-query privacy parameters. Each time a query q_i (which may itself be a measurement) is passed to the compositor queryable, a privacy parameter (like ϵ_i) is popped off of its internal state. Unlike measurements, odometers always emit a queryable from their function, and the privacy map is maintained *inside* the queryable. OpenDP's implementation of odometers differs from the pseudocode in this paper, in that the input distance is not fixed up-front: the `privacy_loss` query also holds a d_{in}. Thus, queryables spawned by odometers can be queried with a bound on the distance between adjacent input datasets d_{in}, which returns the current corresponding privacy spend d_{out}.

The state of odometer queryables in OpenDP consist of the dataset and a vector of privacy maps (one from each query). When the odometer queryable is passed a privacy-loss query, the queryable passes d_{in} from the query into each of the child privacy maps stored in its state, which emit the d_i used in Algorithms 4 and 6. The privacy loss returned is the composition of these parameters.

A filter is an instance of an interactive measurement. Any odometer can be converted into a filter IM by fixing an upper bound on the privacy parameters, as in Lemma 2.19. That is, any queryable spawned by the filter IM will refuse to answer any queries that would exceed the configured privacy parameters.

The OpenDP Library uses "wrapping" to handle situations where a queryable must influence the behavior of queryables it spawns in order to uphold its privacy guarantees. For example, a sequential composition queryable must wrap any queryables it spawns in a logic that first asks the sequential compositor queryable for permission to execute (which allows the sequential compositor to maintain sequentiality), executes the query, and then recursively wraps any queryable in the answer with the same logic.

Concurrent composition improves upon sequential composition in that concurrent composition does not need to influence the behavior of child queryables. Thus, concurrent compositors, odometers, and filters benefit from a simple implementation, in that they do not need to implement complicated sequentiality constraints via "wrapping". More specifically, implementing *any* composition primitive for interactive mechanisms is more complex than for noninteractive mechanisms, as the composition primitive needs to provide the analyst/adversary access to many interactive mechanisms (in such a way that the only access to the sensitive data is through queries, and in particular ensuring that the private internal state of the mechanisms cannot be inspected directly), in contrast to simply sending the analyst the results of noninteractive mechanisms. In the case of sequential composition, the compositor implementation had to be even more complex in order to enforce sequentiality; our work removes the need for this additional complexity.

Concurrent composition also allows the analyst to work in a more exploratory fashion. By allowing non-sequential access to mechanisms, the analyst may sequence their queries arbitrarily. Analysts may also return to previous (un-exhausted) mechanisms without incurring additional privacy penalties. This will also allow analysts to interact with multiple mechanisms simultaneously in settings where computational concurrency may make sequencing ambiguous. Our filter and odometer theorems also allow analysts to choose privacy parameters adaptively, and partition their privacy budget across different mechanisms, providing added flexibility.

We provide our implementation of concurrent composition based on the OpenDP Library on Github at https://github.com/concurrent-composition/concurrent-composition.

5.2 Practical Application in the Tumult Framework

Tumult Analytics [14] [1] is an open-source framework for releasing aggregate information from sensitive datasets with differential privacy. Analytics queries are evaluated using differentially private mechanisms in the context of a Tumult Analytics *session*, which is essentially a privacy filter. Analytics is written on top of Tumult

Core, a framework for constructing differentially private mechanisms with automated privacy guarantees. Interactive mechanisms are the basic abstraction for interactivity in Tumult Core (an interactive mechanism is called a *queryable* in Tumult Core). Tumult Core currently restricts interleaving queries to the composition of queryables to prevent concurrent composition. For example, suppose we start with a privacy filter queryable, and our first query spawns a new queryable using some portion of the budget. Tumult Core requires that the user finish interacting with this spawned queryable before asking a new query to the parent privacy filter, otherwise we may (adaptively) spawn a second queryable and concurrently ask queries of both spawned queryables. Note that while certain concurrent composition results are known from prior work, Tumult Core currently does not allow concurrent composition under any circumstances. The results in this paper allows for removing this restriction.

One application of spawning queryables within queryables in Tumult Analytics is with parallel composition. Parallel composition in differential privacy says that if we partition a dataset and run DP queries on each subset in the partition, the overall privacy loss is the maximum privacy loss across the mechanisms. In Tumult Analytics, the mechanisms the user runs on each subset of the partitioned data can be interactive. Under the hood, this is implemented as follows: the top level privacy filter spawns a new interactive mechanism, consisting of multiple privacy filters (one for each subset in the partition). To prevent concurrent composition where it is not allowed, we require that the user finish interacting with this parallel composition interactive mechanism before they can ask any more queries of the top-level privacy filter, or spawn a new parallel composition mechanism from the top-level privacy filter.

The results in this paper allow for removing the restrictions on concurrent composition, and thereby improve the existing system in two ways. First, it reduces code complexity and improves auditablility. The privacy guarantee of an interactive mechanism is contingent on the promise that there will be no concurrent composition in scenarios where it is not allowed. This property is more challenging to verify than other properties of the system that contribute to the privacy guarantee required, which tend to be localized and easy to verify in isolation.

Second, removing restrictions on concurrent composition improves the UX of Tumult Analytics. As mentioned previously, users of Analytics must finish interacting with one queryable before they start interacting with a subsequent queryable, and this multi-queryable scenario happens frequently when using parallel composition. However, a user may want to interleave queries to two or more queryables. Also, even if interleaving queries isn't required, the user may structure their code such that queries to multiple queryables are interleaved. This would cause the user to see an unfamiliar error, and require them to restructure their code.

Both of these issues would be solved if restrictions on concurrent composition for privacy filters were removed, which the results of our paper can enable.

ACKNOWLEDGMENTS

S.V. is supported by a gift from Apple, NSF grant BCS-2218803, a grant from the Sloan Foundation and a Simons Investigator Award. V.X. is supported in part by NSF grant BCS-2218803. W.Z. is supported in part by a Computing Innovation Fellowship from the Computing Research Association (CRA) and the Computing Community Consortium (CCC), and the BU Census grant.

REFERENCES

[1] Skye Berghel, Philip Bohannon, Damien Desfontaines, Charles Estes, Sam Haney, Luke Hartman, Michael Hay, Ashwin Machanavajjhala, Tom Magerlein, Gerome Miklau, Amritha Pai, William Sexton, and Ruchit Shrestha. 2022. Tumult Analytics: a robust, easy-to-use, scalable, and expressive framework for differential privacy. *arXiv preprint arXiv:2212.04133* (Dec. 2022).

[2] Mark Bun and Thomas Steinke. 2016. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*. Springer, 635–658.

[3] Jinshuo Dong, Aaron Roth, and Weijie J Su. 2019. Gaussian differential privacy. *arXiv preprint arXiv:1905.02383* (2019).

[4] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 486–503.

[5] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. 2010. Differential privacy under continual observation. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC '10)*. 715–724.

[6] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. 2009. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the 41st ACM Symposium on Theory of Computing (STOC '09)*. 381–390.

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

[8] Cynthia Dwork and Guy N Rothblum. 2016. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887* (2016).

[9] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. 2010. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 51–60.

[10] Vitaly Feldman and Tijana Zrnic. 2022. Individual Privacy Accounting via a Renyi Filter. *arXiv preprint arXiv:2008.11193* (2022).

[11] Marco Gaboardi, Michael Hay, and Salil Vadhan. 2020. A programming framework for opendp. *Manuscript* (2020).

[12] Moritz Hardt and Guy N Rothblum. 2010. A multiplicative weights mechanism for privacy-preserving data analysis. In *2010 IEEE 51st annual symposium on foundations of computer science*. IEEE, 61–70.

[13] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. 2015. The composition theorem for differential privacy. In *International conference on machine learning*. PMLR, 1376–1385.

[14] Tumult Labs. 2022. *Tumult Analytics*. https://tmlt.dev

[15] Xin Lyu. 2022. Composition Theorems for Interactive Differential Privacy. In *Thirty-sixth Conference on Neural Information Processing Systems*.

[16] Mathias Lécuyer. 2021. Practical Privacy Filters and Odometers with Rényi Differential Privacy and Applications to Differentially Private Deep Learning. arXiv:2103.01379 [stat.ML]

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

[18] Jack Murtagh and Salil Vadhan. 2016. The complexity of computing the optimal composition of differential privacy. In *Theory of Cryptography Conference*. Springer, 157–175.

[19] Alfréd Rényi. 1961. On measures of entropy and information. In *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability*, Vol. 1. Berkeley, California, USA.

[20] Ryan Rogers, Aaron Roth, Jonathan Ullman, and Salil Vadhan. 2021. Privacy Odometers and Filters: Pay-as-you-Go Composition. arXiv:1605.08294 [cs.CR]

[21] Salil Vadhan and Tianhao Wang. 2021. Concurrent Composition of Differential Privacy. In *Theory of Cryptography Conference*. Springer, 582–604.

[22] Salil Vadhan and Wanrong Zhang. 2023. Concurrent Composition Theorems for Differential Privacy. In *55th Annual ACM Symposium on Theory of Computing*.

[23] Tim Van Erven and Peter Harremos. 2014. Rényi divergence and Kullback-Leibler divergence. *IEEE Transactions on Information Theory* 60, 7 (2014), 3797–3820.

[24] Justin Whitehouse, Aaditya Ramdas, Ryan Rogers, and Zhiwei Steven Wu. 2023. Fully Adaptive Composition in Differential Privacy. arXiv:2203.05481 [cs.LG]

A APPENDIX

A full version is posted on arXiv at https://arxiv.org/abs/2309.05901.