

Valentin Hartmann*, Vincent Bindschaedler, Alexander Bentkamp, and Robert West

Privacy accounting ε conomics: Improving differential privacy composition via a posteriori bounds

Abstract: Differential privacy (DP) is a widely used notion for reasoning about privacy when publishing aggregate data. In this paper, we observe that certain DP mechanisms are amenable to a *posteriori* privacy analysis that exploits the fact that some outputs leak less information about the input database than others. To exploit this phenomenon, we introduce output differential privacy (ODP) and a new composition experiment, and leverage these new constructs to obtain significant privacy budget savings and improved privacy–utility trade-offs under composition. All of this comes at no cost in terms of privacy; we do not weaken the privacy guarantee.

To demonstrate the applicability of our *a posteriori* privacy analysis techniques, we analyze two well-known mechanisms: the Sparse Vector Technique and the Propose-Test-Release framework. We then show how our techniques can be used to save privacy budget in more general contexts: when a differentially private iterative mechanism terminates before its maximal number of iterations is reached, and when the output of a DP mechanism provides unsatisfactory utility. Examples of the former include iterative optimization algorithms, whereas examples of the latter include training a machine learning model with a large generalization error. Our techniques can be applied beyond the current paper to refine the analysis of existing DP mechanisms or guide the design of future mechanisms.

Keywords: differential privacy, differential privacy composition

DOI 10.56553/popets-2022-0070

Received 2021-11-30; revised 2022-03-15; accepted 2022-03-16.

*Corresponding Author: Valentin Hartmann: EPFL,
E-mail: valentin.hartmann@epfl.ch

Vincent Bindschaedler: University of Florida, E-mail:
vbindsch@cise.ufl.edu

Alexander Bentkamp: State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences & Vrije Universiteit Amsterdam, E-mail: bentkamp@ios.ac.cn

Robert West: EPFL, E-mail: robert.west@epfl.ch

1 Introduction

Differential privacy (DP) is a formal notion of privacy for aggregate data releases from databases. Its definition characterizes the extent of what the output of a randomized aggregation mechanism M that is invoked on a database reveals about individual database records. The guarantee is given in terms of the indistinguishability of neighboring databases, that is, databases \mathbf{x} and \mathbf{x}' where \mathbf{x} can be obtained from \mathbf{x}' by either adding/removing one record or by changing the values of one record:

Definition 1 (Differential Privacy [16, 18]). *A randomized algorithm M is (ε, δ) -differentially private if for all pairs of neighboring databases $\mathbf{x}, \mathbf{x}' \in \mathcal{D}$ and for all $S \subseteq \text{Range}(M)$,*

$$\Pr(M(\mathbf{x}) \in S) \leq e^\varepsilon \Pr(M(\mathbf{x}') \in S) + \delta.$$

In the definition, \mathcal{D} denotes the space of databases. If $\delta = 0$, we call M an ε -differentially private mechanism and say that M fulfills *pure differential privacy*.

The DP guarantee holds over all possible sets of outputs. There is a good reason for this: we do not want to end up in a situation where we get unlucky with the database or the randomness of the mechanism M and leak more information about a database record than we intended to when releasing the output of M . However, in this paper we show that when composing mechanisms, i.e., invoking a sequence of mechanisms instead of a single one, one can exploit DP guarantees that only hold w.r.t. proper subsets of $\text{Range}(M)$. We capture the collection of these subset-specific guarantees in the *output differential privacy* (ODP) guarantee of M , which consists of a partition of $\text{Range}(M)$ and privacy guarantees associated with each set in the partition. By adapting mechanisms later in the sequence to the privacy guarantees associated with the outputs of previously invoked mechanisms, one can improve utility in two ways: (1) by reducing the amount of noise that is required for guaranteeing privacy, or (2) by increasing the number of mechanisms that are invoked. All of this is achieved

while retaining the same standard DP guarantee for the sequence of mechanisms.

We emphasize and expand on this crucial last point: in this paper we do *not* weaken the DP guarantee, we do *not* use or propose a relaxed definition of privacy, and we do *not* leak any additional private information by applying our techniques. In fact, all the mechanisms we consider satisfy the traditional definition of DP (Def. 1). Instead, we simply observe that for some DP mechanisms, some outputs happen to leak less private information than other outputs. We show how to exploit this fact to improve the privacy–utility tradeoff offered by these mechanisms in practice.

To make the concept of ODP more concrete, we start with an example. Let $f : \mathcal{D} \rightarrow \mathbb{R}$ be a function that maps databases to values in \mathbb{R} and has sensitivity 1, i.e., $|f(\mathbf{x}) - f(\mathbf{x}')| \leq 1$ for all neighboring databases $\mathbf{x}, \mathbf{x}' \in \mathcal{D}$. Then the Laplace mechanism that releases $f(\mathbf{x}) + \text{Lap}(1/\varepsilon)$, the value of f plus noise drawn from the Laplace distribution with parameter $1/\varepsilon$, fulfills ε -DP [18]. With the Laplace mechanism as a building block we define the toy mechanism M_{toy} that takes as input a database \mathbf{x} as follows:

1. Flip an unbiased coin c .
2. If c came up heads, return $f(\mathbf{x}) + \text{Lap}(1/\varepsilon)$.
3. If c came up tails, return \perp .

Here \perp is a symbol that is independent of \mathbf{x} . If c comes up heads, the ε -differentially private Laplace mechanism is invoked, whose output depends on \mathbf{x} and might contain (a limited amount of) information about individual database records. If c comes up tails, however, the output is independent of \mathbf{x} and thus does not contain any information about individual records in \mathbf{x} . The fact that c comes up tails also does not reveal any information about \mathbf{x} since the coin flip is independent of \mathbf{x} as well. Thus, an adversary learns nothing about \mathbf{x} if they receive \perp as the output of M_{toy} . This means that in the case of a \perp -output the adversary should be allowed to receive the result of a second ε -differentially private mechanism if the overall privacy budget is ε . In the case where an output is produced via the Laplace mechanism, however, the adversary should not receive a second output.

While this is a toy example where the output \perp serves no practical purpose, we show examples of well-known mechanisms that exhibit the same behavior — some outputs leak more private information than others — notably the Sparse Vector Technique [19, 29] and the

mechanisms from the Propose-Test-Release framework [17].

1.1 Our contributions

In this paper we introduce the concept of *output differential privacy* (ODP), which can be used to more accurately describe the leakage of private information of mechanisms whose different outputs reveal different amounts of information about the database they are invoked on. Since ODP is an extension of DP, there exists a trivial ODP guarantee for every DP mechanism. However, our framework only yields improvements for mechanisms with non-trivial ODP guarantees. This class of mechanisms includes the well-known Sparse Vector Technique (SVT) and the mechanisms from the Propose-Test-Release (PTR) framework, but also mechanisms that can be derived from DP mechanisms with only trivial ODP guarantees (Sec. 5 and 6), even in a black-box fashion (Sec. 5). When composing mechanisms with non-trivial ODP guarantees with other DP mechanisms, the more fine-grained ODP guarantees can be used to improve the utility of the composition over using the coarse DP guarantees. Utility here is measured in terms of the noise required to be added or the maximal number of allowed mechanism invocations to not exceed a given DP guarantee. This is achieved via a novel composition protocol that keeps track of the actual leakage of the mechanism’s outputs instead of using the leakage of the worst-case output, in a way that preserves standard DP.

How to benefit from ODP. For simplicity assume that we only compose one ODP mechanism M_1 with one other DP mechanism M_2 . After having produced an output s_1 via M_1 , we check how much s_1 would reveal about the worst-case database that M_1 could have been invoked on. For a worst-case output this bound will not be better than the regular DP bound. For a non-worst-case output such as the \perp from the example of M_{toy} , however, we have a better bound on the leakage than the DP bound and need to subtract less from the remaining privacy budget. This means that there is more privacy budget left to spend on M_2 , and hence M_2 can produce a less noisy and more accurate result. We give some concrete examples for this in Sec. 4. Alternatively, we could decide to spend the saved privacy budget on invoking a third mechanism M_3 . In some cases it might not be of interest to invoke more than one mechanism on the database, e.g., when the database serves

only as training data for one particular machine learning (ML) model. In such cases saved privacy budget can be spent on mechanism invocations on other databases to which individuals from the first database might have contributed as well. Examples for this are user databases for different products of the same company or the results of different surveys from the same city.

Useful for small to medium length compositions. There has been a lot of research on better asymptotic composition bounds (advanced composition bounds) when the number of composed mechanisms is large (see Sec. 2), whereas our framework yields improvements for as little as two composed mechanisms, up to a small to medium number of mechanisms. For a more thorough discussion, see Appendix A. There we also discuss the potential for an advanced composition theorem within our framework.

A formally verified composition theorem. Motivated by mistakes in previous composition theorems [29, 32], we have formally verified the proof of our ODP composition theorem in the proof assistant Lean [13]. Proof assistants are software tools to develop and check formal mathematical proofs. They are used in various areas of computer science — e.g., to verify algorithms and data structures, programming language semantics, security protocols, or hardware specifications. We make the formal proof available online and hope that it can help future formalization endeavors of DP mechanisms or theorems. In this paper we are not concerned with measurability and assume that all sets that we deal with are measurable. However, in the formal proof of the composition theorem we also show measurability.

1.2 Organization of the paper

We start by summarizing prior work and describing how it relates to ODP in Sec. 2. We then formally introduce our ODP framework, consisting of definitions and a novel composition protocol, in Sec. 3. The privacy proof of the composition protocol is deferred to the appendix. As already mentioned, the ODP framework can be applied to PTR and the SVT, which we describe in detail in Sec. 4. In Sec. 5 we demonstrate how ODP can be used to save privacy budget in iterative mechanisms with a non-fixed number of iterations. ODP also allows to recover already spent privacy budget in case the output of a DP mechanism is unsatisfactory, as we show in Sec. 6. We conclude in Sec. 7. The appendix contains

the discussion of a possible extension to advanced composition (Appendix A) and most of the proofs.

2 Related work

One of the two major components of the ODP framework is a new composition experiment that improves utility in certain cases. Extending privacy guarantees from a single mechanism to sequences of mechanisms has been an important subject of study from the early days of DP research. The first result for the composition of mechanisms is the simple composition theorem [16], which states that a mechanism that invokes k (ϵ, δ) -DP mechanisms fulfills $(k\epsilon, k\delta)$ -DP. This statement also holds for pure DP with $\delta = 0$ and cannot be improved upon if pure DP is also required for the composition. However, Dwork et al. [22] later proved the advanced composition theorem, which shows that for the cost of an increase in the δ -part of the composition guarantee, the ϵ -part can be decreased to $\mathcal{O}(\epsilon^2 k + \epsilon\sqrt{k})$. Since then, optimal composition theorems both for the case of homogeneous composition [25] (where all composed mechanisms have the same (ϵ, δ) -guarantee) and heterogeneous composition [31] (where they may have different (ϵ, δ) -guarantees) have been found. The optimality of these composition theorems holds w.r.t. general (ϵ, δ) -DP mechanisms, that is, if the DP guarantees of the mechanisms are fixed, but the data analyst is free to choose any mechanisms that fulfill these DP guarantees. By restricting the choice of mechanisms, tighter composition bounds can be given. To this end, relaxations of DP such as concentrated DP [21], the Rényi-divergence based zero-concentrated DP [10] and Rényi DP [30], and f -DP [14] have been introduced. These definitions can capture the composition behavior of specific mechanisms more precisely. What these improvements have in common with advanced composition is that they only give an improved level of privacy over simple composition if the number of composed mechanisms is large enough (see Table 1).

The most flexible setting that classic composition theorems consider is one where the number of mechanisms to be invoked and their DP guarantees are fixed ahead of time, but where under these constraints the data analyst may adaptively choose the mechanism to be invoked in each step and the database to invoke it on based on the outputs of the previous mechanisms. This is formalized in a so-called composition experiment [22]. The ODP composition experiment that we propose

gives the data analyst a level of freedom that goes beyond what is possible in the classic setting: the data analyst may adaptively choose not only mechanisms and databases, but also neither the number of mechanisms nor their DP guarantees need to be fixed ahead of time. This setting has been previously studied by Rogers et al. [32]. They introduce privacy filters, which are functions that can be used as stopping rules to prevent a given privacy budget from being exceeded. This is essentially the same as how we prevent the choice of mechanisms whose invocation would exceed the privacy budget in our composition experiment. In fact, we can even almost equivalently reformulate our composition experiment using a new variant of privacy filters (see Appendix A). As opposed to us, Rogers et al. give advanced composition results that are asymptotically (in the number of queries) better than our simple composition results. These results have been improved upon by showing that composition results obtained via Rényi DP also hold for the privacy filter setting [23, 26]. However, our composition results are still superior whenever the number of queries is small to medium.

The observation that some outputs of DP mechanisms leak less private information than others and that in these cases one only has to account for the leakage of the actual output has been exploited before, though only in the design of quite specific types of mechanisms and not for developing a general framework as we do. Propose-Test-Release (PTR) [17] is a method for designing DP mechanisms based on robust statistics. PTR mechanisms consist of chains of mechanisms of a particular type whose different outputs leak different amounts of private information, and the authors exploit the fact that only certain sequences of outputs are possible to give better DP guarantees. The Sparse Vector Technique (SVT) [19, 29] is a technique for releasing the (binary) results of a sequence of threshold queries with DP, in a way that each positive result contributes a certain amount to the private leakage, but all negative results together only contribute a fixed amount, which can be used to output arbitrarily many negative results with a fixed privacy budget. The DP mechanism for top- k selection by Durfee and Rogers [15] can be seen as a combination of the ideas of PTR and the SVT. The goal of their mechanism is to return the top- k elements from a database. However, the mechanism might return less than k elements. In that case, it can be invoked again multiple times until k elements have been returned, with an additional cost in δ but without any additional cost in ϵ . We dedicate Sec. 4 to PTR and the SVT, where we show how with ODP we can reduce the privacy budget

that these mechanisms use up when composing them with other mechanisms.

Dwork and Rothblum [21] formalize the distribution of the amount of leakage of private information of mechanisms over their outputs via the so-called privacy loss random variable. They — and later Sommer et al. [33] — use the fact that it is unlikely that a mechanism will, over many iterations, always produce an output with a high privacy loss to show improved composition bounds. These are, however, *a priori* bounds that do not take into account the privacy loss of the actually produced outputs.

Ligett et al. [28] do compute the privacy loss of the actually produced outputs when computing noisy expected risk minimization (ERM) models. They consider the setting where a model does not need to fulfill a predetermined privacy requirement, but instead its loss should not exceed a predetermined value. They propose algorithms to compute the most private model that still fulfills the loss requirement. The authors introduce *ex-post* DP to measure the privacy of a model, which is the special case of our ODP definition when $\delta = 0$. This is why all algorithms proposed in their paper are compatible with our new composition theorem. As opposed to us, Ligett et al. do not provide a way to go from ex-post DP to standard DP. Our ODP framework thus widens the applicability of their mechanisms.

We are not the first to employ automated reasoning to verify differential privacy. While we restrict ourselves to verifying our abstract composition theorem, others have gone a step further and developed tools to verify differential privacy of concrete programs. Barthe et al. [4] developed a specialized Hoare logic, later extended by Barthe and other colleagues [3], and implemented this logic in the toolbox CertiPriv, based on the Coq proof assistant [5]. An alternative approach by Barthe et al. [2] transforms probabilistic programs into nonprobabilistic programs such that proving the transformed program to fulfill a certain specification establishes differential privacy of the original program. Later approaches [6, 35, 36, 38, 40] rely on the SMT solver Z3 [12], the MaxSMT solver ν Z [8], or the probabilistic analysis tool PSI [24] to minimize the manual effort necessary to prove or disprove differential privacy. Wang et al.’s tool DPGen [37] can even transform programs violating differential privacy into differentially private ones. Recent work by Bichsel et al. [7] uses machine learning to detect differential privacy violations.

3 Output differential privacy

In this section we introduce our ODP framework. It is an extension of DP: it contains DP as a special case, but allows for more precise, output-specific privacy accounting.

Definition 2 (Output Differential Privacy (ODP)).

Let \mathcal{O} be a set, and let $\mathcal{P} = \{P_k\}_{k \in \mathcal{K}}$ be a partition of \mathcal{O} , where \mathcal{K} is a countable index set. Let $\mathcal{E} : \mathcal{P} \mapsto \mathbb{R}_{\geq 0}$ be a function that assigns to each set in the partition a non-negative value, and let $\delta \geq 0$. A randomized mechanism M with output set \mathcal{O} is called $(\mathcal{P}, \mathcal{E}, \delta)$ -output differentially private $((\mathcal{P}, \mathcal{E}, \delta)$ -ODP) if for all $S \subseteq \mathcal{O}$ and for all neighboring databases \mathbf{x}, \mathbf{x}' :

$$\Pr(M(\mathbf{x}) \in S) \leq \delta + \sum_{k \in \mathcal{K}} e^{\mathcal{E}(P_k)} \Pr(M(\mathbf{x}') \in S \cap P_k),$$

where the probability space is over the coin flips of the mechanism M . If M is $(\mathcal{P}, \mathcal{E}, \delta)$ -output differentially private for some \mathcal{E} and δ , we call \mathcal{P} an ODP partition for M .

As an example, consider the mechanism M_{toy} from the introduction. An ODP partition for M_{toy} would be $P_1 = \mathbb{R}$, $P_2 = \{\perp\}$, with $\mathcal{E}(P_1) = \varepsilon$, $\mathcal{E}(P_2) = 0$ and $\delta = 0$.

Note that the assumption of the countability of \mathcal{P} is a technical one that is required in the proof of our composition theorem (Thm. 7), but not a restriction in practice due to the finiteness (and thus countability) of computer representations.

The following results allow us to convert a DP guarantee to an ODP guarantee (Lemma 3) and an ODP guarantee to a DP guarantee (Lemma 4):

Lemma 3. Let M be an (ε, δ) -differentially private mechanism. Then M is $(\mathcal{P}, \mathcal{E}, \delta)$ -output differentially private for any partition \mathcal{P} of $\text{Range}(M)$ and the constant function $\mathcal{E} \equiv \varepsilon$.

Proof. Follows directly from the definitions of DP and ODP. \square

Lemma 4. Let M be a $(\mathcal{P}, \mathcal{E}, \delta)$ -output differentially private mechanism. Then M is $(\sup_{P \in \mathcal{P}} \mathcal{E}(P), \delta)$ -differentially private.

Proof. Let $\varepsilon^* = \sup_{P \in \mathcal{P}} \mathcal{E}(P)$. Let \mathbf{x}, \mathbf{x}' be neighboring databases and let $S \subseteq \text{Range}(M)$. Then

$$\begin{aligned} \Pr(M(\mathbf{x}) \in S) &\leq \delta + \sum_{k \in \mathcal{K}} e^{\mathcal{E}(P_k)} \Pr(M(\mathbf{x}') \in S \cap P_k) \\ &\leq \delta + \sum_{k \in \mathcal{K}} e^{\varepsilon^*} \Pr(M(\mathbf{x}') \in S \cap P_k) \\ &= \delta + e^{\varepsilon^*} \Pr(M(\mathbf{x}') \in S). \end{aligned}$$

\square

We sometimes want to build up an ODP guarantee from privacy guarantees that only hold w.r.t. subsets of the output set of a mechanism. We call such guarantees *subset differential privacy* guarantees, and show how they can be combined into an ODP guarantee (Lemma 6). However, as we show in Sec. 5.2, this does not always result in an optimal ODP guarantee.

Definition 5 (Subset Differential Privacy). Let \mathcal{O} be a set and $R \subseteq \mathcal{O}$ a subset of \mathcal{O} . Let $\varepsilon \geq 0$ and $\delta \geq 0$. A randomized mechanism M with output set \mathcal{O} is called (R, ε, δ) -subset differentially private if for all $S \subseteq R$ and for all neighboring databases \mathbf{x}, \mathbf{x}' :

$$\Pr(M(\mathbf{x}) \in S) \leq e^\varepsilon \Pr(M(\mathbf{x}') \in S) + \delta,$$

where the probability space is over the coin flips of the mechanism M .

Lemma 6. Let \mathcal{O} be a set, and let $\mathcal{P} = \{P_k\}_{k \in \mathcal{K}}$ be a partition of \mathcal{O} , where \mathcal{K} is a countable index set. Let M be a randomized mechanism with output set \mathcal{O} and let $\mathcal{E} : \mathcal{P} \mapsto \mathbb{R}_{\geq 0}$ and $\Delta : \mathcal{P} \mapsto \mathbb{R}_{\geq 0}$ be functions such that M is $(P_k, \mathcal{E}(P_k), \Delta(P_k))$ -subset differentially private for all $k \in \mathcal{K}$. Then M is $(\mathcal{P}, \mathcal{E}, \sum_{k \in \mathcal{K}} \Delta(P_k))$ -output differentially private.

Proof. Let $S \subseteq \mathcal{O}$ and let \mathbf{x}, \mathbf{x}' be neighboring databases. Then

$$\begin{aligned} \Pr(M(\mathbf{x}) \in S) &= \sum_{k \in \mathcal{K}} \Pr(M(\mathbf{x}) \in S \cap P_k) \\ &\leq \sum_{k \in \mathcal{K}} \left[e^{\mathcal{E}(P_k)} \Pr(M(\mathbf{x}') \in S \cap P_k) + \Delta(P_k) \right] \\ &= \sum_{k \in \mathcal{K}} \Delta(P_k) + \sum_{k \in \mathcal{K}} e^{\mathcal{E}(P_k)} \Pr(M(\mathbf{x}') \in S \cap P_k). \end{aligned}$$

\square

3.1 Composition of ODP mechanisms

As mentioned in the introduction, ODP can be used to give better utility when composing mechanisms. As

in classical adaptive composition [22], we model composition as a game between a data curator (in the real world this would be the entity with access to the private databases) and an adversary (in the real world a data analyst) that is allowed to spend a total privacy budget of $(\varepsilon_t, \delta_t)$ (in terms of DP) on mechanism invocations (Alg. 1). In each round, the adversary chooses a mechanism M_i and a pair of neighboring databases $\mathbf{x}^{i,0}, \mathbf{x}^{i,1}$. Based on a bit b that is only known to the data curator, the data curator returns $M_i(\mathbf{x}^{i,b})$. The adversary may base their choices in an iteration on the mechanism outputs it has seen in previous iterations. The goal of the adversary is to infer b from the mechanism outputs. Our goal is to ensure that this is not possible with high confidence, by bounding how much the output distribution under $b = 0$ can differ from the output distribution under $b = 1$. Note that this is a hypothetical game that is required for the privacy analysis. In the real world there is no bit b , and only one private database in each round, on which the mechanism in that round is invoked.

As opposed to previous composition experiments, we require the adversary to not only choose each mechanism M_i such that its DP guarantee (as computed via Lemma 4) does not exceed the remaining privacy budget, but to also return an ODP partition \mathcal{P}_i for M_i . Due to Lemma 3, each DP mechanism has a trivial associated ODP partition, thus this requirement does not exclude any DP mechanisms. However, if the ODP partition is non-trivial, such as for the mechanisms in Sec. 4, 5 and 6, the partition can be used to save privacy budget: let $k \in \mathcal{K}_i$ be such that the output of M_i falls into the set $P_{i,k}$ of M_i 's ODP partition. If $\mathcal{E}_i(P_{i,k}) < \sup_{P \in \mathcal{P}_i} \mathcal{E}_i(P)$, then the incurred ε -cost is smaller than the ε of the mechanism's DP guarantee.

Something that sets Alg. 1 apart from the classic composition experiment, but has been used in combination with the privacy filters and odometers of Rogers et al. [32], is that the privacy guarantees of the mechanisms do not have to be fixed ahead of time. Instead, the adversary can adaptively, i.e., based on the outputs of previous mechanisms, choose the privacy parameters of the next mechanism that they want to invoke. This also means that the adversary can adaptively choose the number of iterations: if they want to spend all of the privacy budget in the first $I' < I$ iterations, they can choose a mechanism that always produces the same output independently of the database and thus is $(0, 0)$ -DP for the remaining $I - I'$ iterations. Like Rogers et al., we limit the maximal number of iterations by a fixed number I . This is purely for technical reasons and not a limitation in practice, since I can be chosen arbitrarily

Algorithm 1 ODPComposition($\mathcal{A}, \varepsilon_t, \delta_t, b$)

- 1: Select coin tosses $R_{\mathcal{A}}^b$ for \mathcal{A} uniformly at random.
 - 2: Set remaining budget $\varepsilon_r = \varepsilon_t, \delta_r = \delta_t$.
 - 3: **for** $i = 1, \dots, I$ **do**
 - 4: $\mathcal{A} = \mathcal{A}(R_{\mathcal{A}}^b, \{A_j^b\}_{j=1}^i)$ chooses
 - neighboring databases $\mathbf{x}^{i,0}, \mathbf{x}^{i,1}$,
 - a triple $(\mathcal{P}_i = \{P_{i,k}\}_{k \in \mathcal{K}_i}, \mathcal{E}_i, \delta_i)$, and
 - a mechanism M_i
 such that
 - $\sup_{P \in \mathcal{P}_i} \mathcal{E}_i(P) \leq \varepsilon_r$,
 - $\delta_i \leq \delta_r$, and
 - M_i is $(\mathcal{P}_i, \mathcal{E}_i, \delta_i)$ -ODP
 - 5: Sample $A_i^b = M_i(\mathbf{x}^{i,b})$
 - 6: Let k be such that $A_i^b \in P_{i,k}$
 - 7: $\varepsilon_r \leftarrow \mathcal{E}_i(P_{i,k})$
 - 8: $\delta_r \leftarrow \delta_i$
 - 9: \mathcal{A} receives A_i^b
 - 10: **return** view $V^b = (R_{\mathcal{A}}^b, A_1^b, \dots, A_I^b)$
-

ily large. Note that our composition experiment can almost equivalently be formulated using a new variant of privacy filters (see Appendix A). We choose the formulation in Alg. 1 throughout most of the paper for a more easily accessible presentation.

We show that our composition scheme provides DP:

Theorem 7. *For every adversary \mathcal{A} and for every set of views \mathcal{V} of \mathcal{A} returned by Alg. 1 we have that*

$$\Pr(V^0 \in \mathcal{V}) \leq e^{\varepsilon_t} \Pr(V^1 \in \mathcal{V}) + \delta_t.$$

We defer the proof to Appendix B, where we first show the theorem statement for a composition length of $I = 2$. We define sets of views \mathcal{V}^k where the outputs of M_1 come from the same set P_k of M_1 's ODP partition. Then we apply an extension of a proof of the simple composition theorem for (ε, δ) -mechanisms by Dwork and Lei [17, Lemma 28] to such sets \mathcal{V}_k . By taking unions over \mathcal{V}_k 's, we can analyze arbitrary sets of views. For this we make use of the countability of ODP partitions. The general theorem statement finally follows by induction.

3.2 Formal verification of the composition theorem

We have formally verified the proof of Thm. 7 in the proof assistant Lean [13]. The formal proof is available

online¹. The effort of formalizing our proof has paid off: During the process, we discovered an error in a previous version of Thm. 7 that all authors and reviewers had previously missed. At first glance, in Alg. 1, it might seem as if the δ_i that is subtracted from δ_r on line 8 could also depend on $P_{i,k}$. A counterexample shows that this is a fallacy, which was subtly hidden in a previous version of our proof.

Unlike our pen-and-paper proof in Appendix B, the mechanized version discusses all questions of measurability. The measurability tactic of Lean’s mathematical library could resolve many measurability proofs automatically, but some of them had to be carried out manually. For example, showing that Alg. 1 is measurable (as a function from the sample space to the resulting view) requires an induction over the number of iterations, which is out of reach of automation. Apart from this hurdle, Lean’s mathematical library [34] was surprisingly mature for our purposes, given that it is still relatively young.

4 ODP analysis of existing mechanisms

In this section, we analyze two well-known DP mechanisms using our ODP framework.

4.1 Sparse Vector Technique

The Sparse Vector Technique (SVT) [19, 29] is a method for releasing the results of a sequence of threshold com-

parisons with DP. There are multiple variants of SVT. We work with the improved variant of the mechanism due to Lyu et al. [29, Alg. 1]; see Alg. 2 (M_{SVT}). In this variant, the data analyst sends a stream Q of adaptively chosen \mathbb{R} -valued queries q_i and thresholds T_i to the data curator, who adds the same noise ρ to the thresholds, and different noise values ν_i to the query results. For each query i they then return the result of the comparison $q_i(\mathbf{x}) + \nu_i \geq T_i + \rho$: if the inequality holds, \top is returned, otherwise \perp is returned. The data curator then moves on to the next query until the stream ends — which, as opposed to Lyu et al., we make explicit by letting the data analyst send a STOP query —, or until a prespecified number c of \top -outputs has been produced. What sets the SVT apart from other DP mechanisms is that for a fixed privacy budget an arbitrary number of \perp outputs can be produced; however, only a limited number of \top outputs. Lyu et al. show that M_{SVT} fulfills $(\varepsilon_1 + \varepsilon_2)$ -DP. From their proof it can be seen that all \perp -outputs together contribute ε_1 to the privacy guarantee and each of the at most c \top -outputs contributes ε_2/c . Thus, intuitively, we should be able to save privacy budget if less than c \top -outputs are produced. By slightly modifying the proof by Lyu et al., we can show the following lemma:

Lemma 8. *For each integer $c' \in [0, c]$, let $S_{c'}^{SVT}$ be the set of outputs of M_{SVT} with c' \top -entries. Let further $\mathcal{P} = \{S_{c'}^{SVT}\}_{c'=0}^c$ and let, for $0 \leq c' \leq c$,*

$$\mathcal{E}(S_{c'}^{SVT}) = \varepsilon_1 + \frac{c'}{c}\varepsilon_2$$

Then M_{SVT} is $(\mathcal{P}, \mathcal{E}, 0)$ -ODP.

Proof. Our proof follows closely the one of Lyu et al. [29, Thm. 1]. The only differences are that we explicitly consider the query at which the stream Q stops and that we do not bound c' by c , but work with the exact value. Let \mathbf{x}, \mathbf{x}' be neighboring databases and assume that the sensitivity of all queries q_i is bounded by Δ . Let

$$\begin{aligned} f_i(\mathbf{x}, z) &= \Pr(q_i(\mathbf{x}) + \nu_i < T_i + z), \\ g_i(\mathbf{x}, z) &= \Pr(q_i(\mathbf{x}) + \nu_i \geq T_i + z). \end{aligned}$$

We have that

$$\begin{aligned} g_i(\mathbf{x}, z - \Delta) &= \Pr(q_i(\mathbf{x}) + \nu_i \geq T_i + z - \Delta) \\ &\leq \Pr(q_i(\mathbf{x}') + \Delta + \nu_i \geq T_i + z - \Delta) \\ &= \Pr(q_i(\mathbf{x}') + \nu_i \geq T_i + z - 2\Delta) \\ &\leq e^{\varepsilon_2/c} \Pr(q_i(\mathbf{x}') + \nu_i \geq T_i + z). \end{aligned} \quad (1)$$

Let $l > 0$, $c' \leq c$ and let $a \in \{\perp, \top\}^l \cap S_{c'}^{SVT}$ be an output of M_{SVT} of length l that contains c' \top ’s. Let

Algorithm 2 $M_{SVT}(\mathbf{x}, Q, \Delta, T_1, T_2, \dots, c)$

```

1:  $\rho \sim \text{Lap}(\Delta/\varepsilon_1)$ 
2: count = 0
3: for  $q_i \in Q$  do
4:   if  $q_i = \text{STOP}$  or count  $\geq c$  then
5:     BREAK
6:    $\nu_i \sim \text{Lap}(2c\Delta/\varepsilon_2)$ 
7:   if  $q_i(\mathbf{x}) + \nu_i \geq T_i + \rho$  then
8:     Output  $a_i = \top$ 
9:     count += 1
10:  else
11:    Output  $a_i = \perp$ 
```

¹ <https://doi.org/10.6084/m9.figshare.19330649>

$I_{\perp} = \{i \mid a_i = \perp\}$, $I_{\top} = \{i \mid a_i = \top\}$. Then

$$|I_{\top}| = c'. \quad (2)$$

By $\Pr(q_{l+1} = \text{ST} \mid a)$ denote the probability that the data analyst chooses the STOP query as the next query after having received the l outputs in a , and let $z_- = z - \Delta$. We have:

$$\begin{aligned} & \frac{\Pr(M_{\text{SVT}}(\mathbf{x}) = a)}{\Pr(M_{\text{SVT}}(\mathbf{x}') = a)} \\ &= \frac{\Pr(M_{\text{SVT}}(\mathbf{x}) = a \mid q_{l+1} = \text{ST}) \Pr(q_{l+1} = \text{ST} \mid a)}{\Pr(M_{\text{SVT}}(\mathbf{x}') = a \mid q_{l+1} = \text{ST}) \Pr(q_{l+1} = \text{ST} \mid a)} \\ &= \frac{\Pr(M_{\text{SVT}}(\mathbf{x}) = a \mid q_{l+1} = \text{ST})}{\Pr(M_{\text{SVT}}(\mathbf{x}') = a \mid q_{l+1} = \text{ST})} \\ &= \frac{\int_{-\infty}^{\infty} \Pr(\rho = z) \prod_{i \in I_{\perp}} f_i(\mathbf{x}, z) \prod_{i \in I_{\top}} g_i(\mathbf{x}, z) dz}{\int_{-\infty}^{\infty} \Pr(\rho = z) \prod_{i \in I_{\perp}} f_i(\mathbf{x}', z) \prod_{i \in I_{\top}} g_i(\mathbf{x}', z) dz} \\ &= \frac{\int_{-\infty}^{\infty} \Pr(\rho = z_-) \prod_{i \in I_{\perp}} f_i(\mathbf{x}, z_-) \prod_{i \in I_{\top}} g_i(\mathbf{x}, z_-) dz}{\int_{-\infty}^{\infty} \Pr(\rho = z) \prod_{i \in I_{\perp}} f_i(\mathbf{x}', z) \prod_{i \in I_{\top}} g_i(\mathbf{x}', z) dz} \\ &\leq \frac{\int_{-\infty}^{\infty} e^{\varepsilon_1} \Pr(\rho = z) \prod_{i \in I_{\perp}} f_i(\mathbf{x}', z) \prod_{i \in I_{\top}} g_i(\mathbf{x}, z_-) dz}{\int_{-\infty}^{\infty} \Pr(\rho = z) \prod_{i \in I_{\perp}} f_i(\mathbf{x}', z) \prod_{i \in I_{\top}} g_i(\mathbf{x}', z) dz} \\ &\stackrel{(1)}{\leq} \frac{\int_{-\infty}^{\infty} e^{\varepsilon_1} \Pr(\rho = z) \prod_{i \in I_{\perp}} f_i(\mathbf{x}, z) \prod_{i \in I_{\top}} e^{\varepsilon_2/c} g_i(\mathbf{x}, z) dz}{\int_{-\infty}^{\infty} \Pr(\rho = z) \prod_{i \in I_{\perp}} f_i(\mathbf{x}', z) \prod_{i \in I_{\top}} g_i(\mathbf{x}', z) dz} \\ &\stackrel{(2)}{=} e^{\varepsilon_1} (e^{\varepsilon_2/c})^{c'} = e^{\varepsilon_1 + \frac{c'}{c} \varepsilon_2}. \end{aligned}$$

Since this bound hold for every element $a \in S_{c'}^{\text{SVT}}$, it also holds for all subsets of $S_{c'}^{\text{SVT}}$. Hence, M_{SVT} is $(S_{c'}^{\text{SVT}}, e^{\varepsilon_1 + \frac{c'}{c} \varepsilon_2}, 0)$ -subset differentially private for every $c' \leq c$. The ODP bound then follows from Lemma 6. \square

Application. A common use case for SVT is the differentially private release of only those entries of a vector with large magnitude, instead of the entire vector [27, 39]. This can be desirable for multiple reasons: to be able to release the entries with less noise, since the privacy budget needs to be divided between fewer entries; to release only those values of a histogram that are large enough in magnitude so that they will not be dominated by the added noise; or to reduce communication costs in a distributed setting. Assume that f is a function that takes as input the private database and returns the vector of interest. The queries would then

be $q_i(\mathbf{x}) = |f_i(\mathbf{x})|$, i.e., the absolute value of the i -th entry of the vector $f(\mathbf{x})$. Only if the corresponding output a_i is \top , a differentially private version of $f_i(\mathbf{x})$ is released. When analyzing SVT with DP [29, Sec. 4.1], the available privacy budget $\varepsilon_t = \varepsilon_1 + \varepsilon_2 + \varepsilon_3$ is divided into a budget $\varepsilon_1 + \varepsilon_2$ for SVT itself, and a budget ε_3 for the release of the vector entries. When using the Laplace mechanisms with the same variance to perturb the vector entries, a privacy budget of ε_3/c is available for each of the at most c entries that get released. However, it is not guaranteed that c entries will surpass the threshold and thus get released. ODP allows us to do the following: first invoke SVT. Let $c' \leq c$ be the number of entries that surpassed the threshold. We have now used up a privacy budget of $\varepsilon_1 + \frac{c'}{c} \varepsilon_2$ and thus have a privacy budget of $\varepsilon'_3 = \varepsilon_3 + \frac{c-c'}{c} \varepsilon_2$ available for the release of the c' vector entries. Assume that $c' < c$. Then we have $\varepsilon'_3 > \varepsilon_3$. Further, we only need to split this budget between c' instead of between c vector entries and thus have a budget of ε'_3/c' per entry. This is not possible with a pure DP analysis, where we always have to assume the worst-case of c released vector entries. These two sources of budget saving — from SVT itself via Lemma 8, and from making use of the knowledge of the actual number of released entries — lead to less noise in the released entries. Adapting the noise per entry to the actual number of released entries would also be possible with privacy filters [32], but saving budget from SVT itself is only possible with ODP.

In Fig. 1 we numerically demonstrate the advantage of using ODP when releasing a sparse vector via SVT². The vector in our example has 100 entries that each have a value of either 0 or 1000. We assume that at most 20 of the entries lie above the threshold $T = 500$, i.e., $c = 20$. The total privacy budget is $\varepsilon = 1$, split into a budget $\varepsilon_1 + \varepsilon_2 = 1/2$ for determining the indices of the large vector entries and a budget $\varepsilon_3 = 1/2$ for releasing the corresponding values. For splitting budget between ε_1 and ε_2 we choose the optimal ratio $\varepsilon_1/\varepsilon_2 = 1/(2c)^{2/3}$ according to Lyu et al. [29]. We further assume that the queries $q_i(\mathbf{x}) = |f_i(\mathbf{x})|$ have sensitivity $\Delta = 1$. Since c is only an upper bound on the number of large entries, the actual number of large entries can be lower. The ODP analysis can exploit cases with less than c large entries for adding less noise to the values of the released entries, while with the standard analysis of SVT always the same amount of noise has to be added. For

² The code for generating the plots in this paper can be found at <https://doi.org/10.6084/m9.figshare.19330649>.

different numbers of entries with value 1000 we compare the amount of noise added to the released vector entries with and without an ODP analysis. Without an ODP analysis this is a fixed amount, whereas with an ODP analysis the amount of noise depends on the number of released vector entries, which itself depends on the outcome of a random comparison. Fig. 1 therefore shows an upper bound on the expected amount of noise for the ODP analysis based on Chebyshev's inequality, where the expectation is taken over the randomness of ρ and the ν_i in Alg. 2.

4.2 Propose-Test-Release

Propose-Test-Release (PTR) by Dwork and Lei [17] is a framework for designing differentially private mechanisms based on the idea of connecting DP and robust statistics. Using PTR, the authors derive mechanisms for differentially privately estimating the median, the trimmed mean, the interquartile range and the coefficients of a linear regression model. The structure of PTR mechanisms is that they first propose a bound on the local sensitivity of the query function and then differentially privately test whether the local sensitivity lies below this bound. If the local sensitivity lies below the proposed bound, the result of the query with noise adapted to the bound is returned. Otherwise \perp (no response) is returned. In the case of a \perp -output, the local sensitivity was high, meaning that the query result likely would have not been robust to changes in individual data points, and thus not very useful anyway. The basic building blocks of mechanisms based on PTR are (ε, δ) -PTR functions, or short, PTR functions. They get as input a database \mathbf{x} and a value s that may be used for computing a proposed bound on the local sensitivity. For example, Dwork and Lei propose a PTR function for computing the α -trimmed mean, i.e., the mean of an empirical distribution when discarding the upper and lower $\alpha/2$ -quantile. The local sensitivity, that is, the sensitivity w.r.t. a concrete database, of the α -trimmed mean depends on the distance between the $\alpha/2$ - and the $(1 - \alpha/2)$ -quantile. The authors provide a mechanism for computing this distance (a generalization of the mechanism M_{IQR} below), and use the output of this mechanism as the second input s to the PTR function for the α -trimmed mean. Formally, PTR functions are defined as follows:

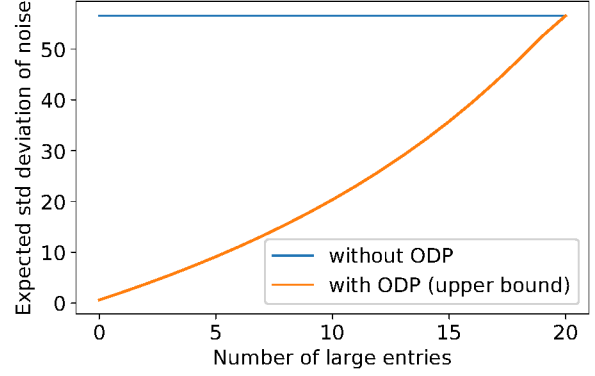


Fig. 1. Noise required with and without an ODP analysis when releasing the values of at most 20 large entries of a 100-dimensional sparse vector with a total privacy budget of $\varepsilon = 1$. We vary the number of large entries in the vector. Large entries have value 1000, small entries have value 0, and the threshold for releasing an entry is 500.

Definition 9 ((ε, δ) -PTR function). *A function $T(\mathbf{x}, s) : \mathcal{D} \times (\mathcal{C} \cup \{\perp\}) \rightarrow \mathcal{C}' \cup \{\perp\}$ is (ε, δ) -PTR if*

1. $\Pr(T(\mathbf{x}, \perp) = \perp) = 1$ for all $\mathbf{x} \in \mathcal{D}$.

2. For all $s \in \mathcal{C}$, $\mathbf{x}, \mathbf{x}' \in \mathcal{D}$ neighboring,

$$\Pr(T(\mathbf{x}, s) = \perp) \leq e^\varepsilon \Pr(T(\mathbf{x}', s) = \perp),$$

$$\Pr(T(\mathbf{x}, s) \neq \perp) \leq e^\varepsilon \Pr(T(\mathbf{x}', s) \neq \perp).$$

3. There exists $G(T, \mathbf{x}) \subseteq \mathcal{C}$ such that if $s \in G(T, \mathbf{x})$, then for all \mathbf{x}' neighboring \mathbf{x} and all $\mathcal{C}' \subseteq \mathcal{C}$,

$$\Pr(T(\mathbf{x}, s) \in \mathcal{C}') \leq e^{2\varepsilon} \Pr(T(\mathbf{x}', s) \in \mathcal{C}');$$

4. for all $\mathbf{x} \in \mathcal{D}$, if $s \notin G(T, \mathbf{x})$: $\Pr(T(\mathbf{x}, s) \neq \perp) \leq \delta$.

Let T be a PTR function. If $s = \perp$ and thus no bound on the local sensitivity can be computed from s , T returns \perp . The conditions in 2 allow for improved privacy bounds when composing PTR functions where the next function is only invoked if the previous one did not return \perp (see the example of M_{IQR} below). In that case, all PTR function invocations except from the last one only increase the privacy budget spending by ε instead of 2ε . Not every proposed sensitivity bound is large enough for every dataset to ensure privacy when adding noise according to this bound. The bounds that are admissible for a dataset \mathbf{x} are described by the set $G(T, \mathbf{x})$. If a proposed bound is too small, then with probability at least $1 - \delta$ the \perp -symbol will be returned to prevent revealing too much information about \mathbf{x} .

As can easily be seen, a PTR function T fulfills $(2\varepsilon, \delta)$ -DP. Letting $P_1 = \mathcal{C}'$, $P_2 = \{\perp\}$ and $\mathcal{E}(P_1) =$

2ε , $\mathcal{E}(P_2) = \varepsilon$, T also fulfills $(\{P_1, P_2\}, \mathcal{E}, \delta)$ -ODP. Hence we can save privacy budget in the case of a \perp -output. We exemplify this for the mechanism proposed by Dwork and Lei for approximating the interquartile range (IQR) of an empirical distribution on \mathbb{R} , that is, the difference between the 75th and the 25th percentile, which serves as a measure of scale. We use M_{IQR} to denote this mechanism. M_{IQR} works by discretizing \mathbb{R} into buckets and proposing a local sensitivity of the discretized IQR. If the IQR has at most the proposed sensitivity, a noisy version of the IQR is released, otherwise \perp is released. To avoid an unlucky choice of the discretization, M_{IQR} uses two discretizations, where the second one is a shifted version of the first one. If the output produced by the mechanism resulting from the first discretization (denoted by M_1^D) is not \perp , this output is returned and the computation ends. Otherwise the mechanism resulting from the second discretization (denoted by M_2^D) is invoked and its output is returned. The authors show that, for $j = 1, 2$, M_j^D is a PTR function. Without having formalized the concept of ODP yet, they use the fact that a \perp -output of M_1^D contains less information about the database than a non- \perp -output, combined with the fact that M_2^D is only invoked if M_1^D 's output is \perp to show that M fulfills $(3\varepsilon, \delta)$ -DP (instead of the naive $((2+2)\varepsilon, (1+1)\delta)$ -DP), because one never has to account for a non- \perp -output of both M_1^D and M_2^D . The authors show that the same reasoning can be applied to more general compositions of PTR functions to save privacy budget also in the computation of, e.g., the median or of regression parameters.

ODP analysis: Treating M_1^D and M_2^D as separate mechanisms. By treating each PTR function that makes up one of their mechanisms as a separate mechanism and composing them via Alg. 1, we can save ε -budget beyond the improved analysis by Dwork and Lei, but require additional δ -budget. In the example of M_{IQR} we either only invoke M_1^D (if the output is in \mathbb{R}) or invoke both M_1^D and M_2^D . Since M_1^D and M_2^D are PTR functions, they fulfill $(\{\mathbb{R}, \{\perp\}\}, \mathcal{E}, \delta)$ -ODP with $\mathcal{E}(\mathbb{R}) = 2\varepsilon$ and $\mathcal{E}(\perp) = \varepsilon$. One of the following three cases will occur:

1. M_1^D returns r for some $r \in \mathbb{R}$ (and M_2^D never gets invoked);
2. M_1^D returns \perp and M_2^D returns r for some $r \in \mathbb{R}$;
3. both M_1^D and M_2^D return \perp .

In case 1 we spend a privacy budget of $(2\varepsilon, \delta)$, in case 2 we spend a budget of $(3\varepsilon, 2\delta)$, and in case 3 we spend a

budget of $(2\varepsilon, 2\delta)$. Compared with the DP analysis, in cases 1 & 3 we save a budget of $(\varepsilon, 0)$, but we spend an additional budget of $(0, \delta)$ in cases 2 & 3. By choosing the order of the two discretizations used for M_1^D and M_2^D uniformly at random, we can ensure that with probability at least $1/2$ we will be in case 1 if at least one of the two mechanisms returns a value in \mathbb{R} .

ODP analysis: Treating M_{IQR} as a single mechanism. When treating M_{IQR} as a single mechanism, we can even strictly improve upon the original analysis in terms of ODP. We have the following lemma, which shows that we can save privacy budget if both discretizations result in a \perp -output:

Lemma 10. *Let $P_1 = \mathbb{R}$, $P_2 = \{\perp\}$ and $\mathcal{E}(P_1) = 3\varepsilon$, $\mathcal{E}(P_2) = 2\varepsilon$. Then M_{IQR} is $(\{P_1, P_2\}, \mathcal{E}, \delta)$ -ODP.*

Proof. Since M_{IQR} is $(3\varepsilon, \delta)$ -DP, it is in particular $(\mathbb{R}, 3\varepsilon, \delta)$ -subset differentially private. Let \mathbf{x} and \mathbf{x}' be neighboring databases. Since (1) the randomnesses of M_1^D and M_2^D are independent and (2) M_1^D and M_2^D are (ε, δ) -PTR functions, it holds that

$$\begin{aligned} \Pr(M(\mathbf{x}) = \perp) &= \Pr(M_1^D(\mathbf{x}) = \perp, M_2^D(\mathbf{x}) = \perp) \\ &\stackrel{(1)}{=} \Pr(M_1^D(\mathbf{x}) = \perp) \Pr(M_2^D(\mathbf{x}) = \perp) \\ &\stackrel{(2)}{\leq} e^\varepsilon \Pr(M_1^D(\mathbf{x}') = \perp) e^\varepsilon \Pr(M_2^D(\mathbf{x}') = \perp). \end{aligned}$$

Thus M_{IQR} is $(\{\perp\}, 2\varepsilon, 0)$ -subset differentially private. The statement of the lemma then follows from Lemma 6. \square

Application. There are multiple statistics that can be computed via PTR mechanisms. Typically one is not interested in only a single differentially private statistic of a dataset, but in multiple statistics. E.g., one might want to release a measure of the location of the data and a measure of its spread. For this task one could first invoke M_{IQR} to compute the IQR of the data and then another DP mechanism M_{median} to compute its median [17]. The total privacy budget $(\varepsilon_t, \delta_t)$ would be divided into a budget $(3\varepsilon_1, \delta_1)$ for M_{IQR} and a budget $(\varepsilon_2, \delta_2)$ for M_{median} , where $3\varepsilon_1 + \varepsilon_2 = \varepsilon_t$ and $\delta_1 + \delta_2 = \delta_t$. With a DP analysis, M_{IQR} would always use up $(3\varepsilon_1, \delta_1)$ from the budget. However, with ODP composition and when using our second ODP analysis (Lemma 10), M_{IQR} only uses up $(2\varepsilon_1, \delta_1)$ in the case of a \perp -output, and thus the budget remaining for M_{median} increases to $(\varepsilon_1 + \varepsilon_2, \delta_2)$ in that case. The increased budget can be used to reduce the amount of noise added in the computation of the differentially private median, which makes the released estimate more accurate.

5 ODP for mechanisms with variable numbers of iterations

There are many iterative algorithms for which the required number of iterations is not known beforehand. Instead, they are executed until a certain criterion is reached, e.g., the norm of the gradient in an optimization problem falls below a prespecified threshold or the validation loss starts increasing in the training of a machine learning (ML) model. A common technique to make the final output of iterative algorithms differentially private is by making the intermediate values in each iteration differentially private. In stochastic gradient descent, for instance, which is often used for training neural networks, this is typically achieved by adding Gaussian noise to each gradient [1, 9]. The DP guarantees for the intermediate values are then combined using a composition theorem to get a DP guarantee for the final output. For this it is necessary to know beforehand — i.e., before executing the iterative algorithm — how many iterations will be performed. If the total privacy budget is fixed, a larger number of iterations means that less privacy budget can be used on each iteration, whereas with a smaller number of iterations more privacy budget is available for each iteration. If, as in many cases, the optimal number of iterations is not known *a priori*, the number will often be either overestimated or underestimated. If the number of iterations is overestimated, privacy budget is wasted on iterations that are not required. If it is underestimated, the iterative algorithm will halt before it has reached an optimal solution, e.g., an ML model might have a larger error than it could have with more iterations.

ODP allows us to escape this dilemma. Consider a data analyst that chooses a number k of iterations for the iterative algorithm. With standard DP analysis, the algorithm always runs for k iterations, even if it converges at an earlier iteration $k' < k$. The privacy budget for the remaining $k - k'$ iterations is thus wasted. With ODP, however, the algorithm can be stopped at iteration k' and the budget that was reserved for the remaining $k - k'$ iterations can be used via ODP composition for other queries on the same database or on other databases that might share individuals who contributed data with the original database. This solves the problem of overestimating the number of iterations. To solve the problem of underestimating the number of iterations, the data analyst can purposely choose a large number for k that is likely to be an overestimate. This is not problematic anymore, since in the case where it was

indeed an overestimate, the algorithm can again halt earlier, and the remaining privacy budget can be used for other tasks.

Algorithm 3 $M_{\text{iter}} [\{k_i\}_{i=1}^n, \{M_k\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}](\mathbf{x})$

```

1: for  $k = 1, \dots, k_n$  do
2:    $s_k = M_k((s_1, \dots, s_{k-1}), \mathbf{x})$ 
3:   if  $k \in \{k_i\}_{i=1}^{n-1}$  then
4:     Let  $i$  such that  $k = k_i$ 
5:     if  $h^{k_i}(s_1, \dots, s_{k_i}) = 1$  then
6:       return  $(s_1, \dots, s_{k_i})$ 
7: return  $(s_1, \dots, s_{k_n})$ 

```

We consider an iterative mechanism M as defined in Alg. 3. Let $k_1 < k_2 < \dots < k_n$ be numbers of iterations after which M might stop. For $k = 1, \dots, k_n$, let M_k be a differentially private mechanism that takes as input the output of previous mechanisms and a database. M_k is the mechanism executed in the k -th iteration of M . Define a set of binary functions

$$h^{k_i} : \prod_{k=1}^{k_i} \text{Range}(M_k) \mapsto \{0, 1\},$$

$i = 1, \dots, n-1$, that act as stopping criteria. After k_i iterations, M evaluates h^{k_i} on the outputs s_1, \dots, s_{k_i} produced so far: if $h^{k_i}(s_1, \dots, s_{k_i}) = 1$, M halts and returns (s_1, \dots, s_{k_i}) ; otherwise it continues. If $h^{k_i}(s_1, \dots, s_{k_i}) = 0$ for all $i < n$, M halts and returns (s_1, \dots, s_{k_n}) after iteration k_n . Note that we assume for simplicity that h^{k_i} can be evaluated with only the information that has already been computed in a differentially private way, i.e., it only requires access to \mathbf{x} via M_1, \dots, M_{k_i} (this is, e.g., the case if h^{k_i} is based on the gradient norm in differentially private SGD or the validation score of an ML model on a public validation set). If this is not the case and the computation of h^{k_i} requires access to the database \mathbf{x} , we can simply add an additional mechanism \tilde{M}_{k_i} after M_{k_i} that computes h^{k_i} in a differentially private way, and reindex the sequence of mechanisms such that $M_{k_i} = \tilde{M}_{k_i}$.

For $i = 1, \dots, n$, let

$$P_i = \{s \mid s \in \text{Range}(M), |s| = k_i\},$$

where $|s|$ denotes the length of the vector s , and let $\mathcal{P} = \{P_i\}_{i=1}^n$. Throughout this section we always mean this partition \mathcal{P} when referring to the partition of the output space of an iterative mechanism. When we want to make the dependency on an iterative mechanism M'

explicit, we write $\mathcal{P}^{M'} = \{P_i^{M'}\}_{i=1}^n$. In Sec. 5.1 we show a generic ODP bound for M with the ODP partition \mathcal{P} based on Lemma 6 that is compatible with any DP composition theorem. In Sec. 5.2 we show how this generic bound can be improved upon via a direct derivation that depends on the specific composition setting. Thus, Sec. 5.2 also acts as a demonstration of how Lemma 6 does not always yield an optimal ODP bound.

The iterative mechanism in Alg. 3 is defined by a sequence $\{k_i\}_{i=1}^n$ of potential stopping points, a sequence $\{M_k\}_{k=1}^{k_n}$ of mechanisms that are invoked in the different iterations, and a sequence $\{h^{k_i}\}_{i=1}^{n-1}$ of stopping criteria. We denote an iterative mechanism, given by such a set of parameters, via $M_{\text{iter}}[\{k_i\}_{i=1}^n, \{M_k\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}]$.

5.1 ODP bound based on Lemma 6

We derive an ODP bound for iterative mechanisms via Lemma 6 and any composition theorem COMP that is compatible with M_1, \dots, M_{k_n} , i.e., that can be applied to the sequence of mechanisms M_1, \dots, M_{k_n} . This could, e.g., be the optimal composition theorem for adaptive, heterogeneous composition [31] (see Sec. 5.2), for the most general class of mechanisms.

Lemma 11. *Let*

$$M = M_{\text{iter}}[\{k_i\}_{i=1}^n, \{M_k\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}]$$

be an iterative mechanism. Let COMP be a DP composition theorem that is compatible with M_1, \dots, M_{k_n} . COMP takes as input a sequence of mechanisms and a desired value ε , and returns a value δ such that the sequence fulfills (ε, δ) -DP. For $i = 1, \dots, n$, let $\varepsilon^{k_i} \geq 0$ be a desired ε -value, and let

$$\delta^{k_i} = \text{COMP}(M_1, \dots, M_{k_i}; \varepsilon^{k_i})$$

be the δ returned by the composition theorem. Then M fulfills $(\mathcal{P}, \mathcal{E}, \delta)$ -ODP with

$$\mathcal{E}(P_i) = \varepsilon^{k_i}$$

and

$$\delta = \sum_{i=1}^n \delta^{k_i}$$

for all $i = 1, \dots, n$.

Proof. We can assume w.l.o.g. that the stopping criteria h^{k_i} are deterministic. If they are not deterministic, we can let M_{k_i} , in addition to its original output, return a sample from the distribution of h^{k_i} 's randomness, which

h^{k_i} can then access. This does not change the DP guarantee of M_{k_i} , since the distribution of h^{k_i} 's randomness is independent of the database.

Let $i \in \{1, \dots, n\}$. Define $M^{k_i} = (M_1, \dots, M_{k_i})$. Because h^{k_i} is deterministic, it holds for any database \mathbf{x} and any $S \subseteq P_i$ that

$$\Pr(M(\mathbf{x}) \in S) = \Pr(M^{k_i}(\mathbf{x}) \in S).$$

Since M^{k_i} is $(\varepsilon^{k_i}, \delta^{k_i})$ -differentially private, we have, for any neighboring databases \mathbf{x}, \mathbf{x}' :

$$\begin{aligned} \Pr(M(\mathbf{x}) \in S) &= \Pr(M^{k_i}(\mathbf{x}) \in S) \\ &\leq e^{\varepsilon^{k_i}} \Pr(M^{k_i}(\mathbf{x}') \in S) + \delta^{k_i} \\ &= e^{\varepsilon^{k_i}} \Pr(M(\mathbf{x}') \in S) + \delta^{k_i}. \end{aligned}$$

Hence M is $(P_{k_i}, \varepsilon^{k_i}, \delta^{k_i})$ -subset differentially private. The statement of the lemma then follows from Lemma 6. \square

5.2 ODP bound via direct derivation

In this subsection we give an example that shows that we can get better ODP bounds for iterative mechanisms than the ones obtained by applying Lemma 11. This comes at the cost of losing generality, since we cannot plug in any existing DP composition theorem anymore, but have to do a derivation from scratch. We consider the adaptive, heterogeneous composition of arbitrary differentially private mechanisms. Our proof extends the one by Murtagh and Vadhan [31] for a composition setting without stopping rules to one with stopping rules. Adaptive, heterogeneous composition means that, for $k = 1, 2, \dots$, we assume that M_k fulfills $(\varepsilon_k, \delta_k)$ -DP for some fixed ε_k and δ_k , and that M_k may depend on the outputs of M_1, \dots, M_{k-1} , but we assume nothing beyond that.

Let

$$M = M_{\text{iter}}[\{k_i\}_{i=1}^n, \{M_k\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}]$$

be an iterative mechanism, based on mechanisms $\{M_k\}_{k=1}^{k_n}$ as described above. For a function $\mathcal{E} : \mathcal{P} \mapsto \mathbb{R}_{\geq 0}$ we define the smallest δ such that M fulfills $(\mathcal{P}, \mathcal{E}, \delta)$ -ODP as

$$\begin{aligned} &\text{OptDel}(\{k_i\}_{i=1}^n, \{M_k\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}, \mathcal{E}) \\ &= \inf\{\delta \mid M_{\text{iter}}[\{k_i\}_{i=1}^n, \{M_k\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}] \\ &\quad \text{is } (\mathcal{P}, \mathcal{E}, \delta)\text{-ODP}\}. \end{aligned}$$

Given \mathcal{E} , $\{k_i\}_{i=1}^n$ and a fixed list of DP parameters $\{(\varepsilon_k, \delta_k)\}_{k=1}^{k_n}$, we want to find the minimal δ such that $M_{\text{iter}}[\{k_i\}_{i=1}^n, \{M_k\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}]$ fulfills $(\mathcal{P}, \mathcal{E}, \delta)$ -ODP for any sequence $\{M_k\}_{k=1}^{k_n}$ of mechanisms that fulfill $(\varepsilon_k, \delta_k)$ -DP, $k = 1, \dots, k_n$, and any sequence of stopping criteria. We thus define

$$\begin{aligned} & \text{OptDel}(\{k_i\}_{i=1}^n, \{(\varepsilon_k, \delta_k)\}_{k=1}^{k_n}, \mathcal{E}) \\ &= \sup_{\substack{\{M_k\}_{k=1}^{k_n} \\ \{h^{k_i}\}_{i=1}^{n-1}}} \{ \text{OptDel}(\{k_i\}_{i=1}^n, \{M_k\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}, \mathcal{E}) \\ & \quad | M_k \text{ is } (\varepsilon_k, \delta_k)\text{-DP, } k = 1, \dots, k_n \}. \end{aligned}$$

The remainder of this subsection is devoted to deriving the expression for $\text{OptDel}(\{k_i\}_{i=1}^n, \{(\varepsilon_k, \delta_k)\}_{k=1}^{k_n}, \mathcal{E})$ in Thm. 15. Further, following Thm. 15, we compare this optimal value with the one that can be obtained via Lemma 11. All proofs from this subsection are deferred to Appendix C.

Like Kairouz et al. [25] and Murtagh and Vadhan [31], we derive an expression for $\text{OptDel}(\{k_i\}_{i=1}^n, \{(\varepsilon_k, \delta_k)\}_{k=1}^{k_n}, \mathcal{E})$ by showing that it suffices to analyze a class of randomized response mechanisms and to then compute the optimal ODP bound for these randomized response mechanisms. Kairouz et al. show the following lemma:

Lemma 12 ([25]). *For $\varepsilon, \delta \geq 0$, let the randomized response mechanism $\tilde{M}_{(\varepsilon, \delta)} : \{0, 1\} \rightarrow \{0, 1, 2, 3\}$ be defined as (dropping the dependency on (ε, δ) for simplicity)*

$$\begin{aligned} \Pr(\tilde{M}(0) = 0) &= \delta & \Pr(\tilde{M}(1) = 0) &= 0 \\ \Pr(\tilde{M}(0) = 1) &= \frac{(1 - \delta)e^\varepsilon}{1 + e^\varepsilon} & \Pr(\tilde{M}(1) = 1) &= \frac{(1 - \delta)}{1 + e^\varepsilon} \\ \Pr(\tilde{M}(0) = 2) &= \frac{(1 - \delta)}{1 + e^\varepsilon} & \Pr(\tilde{M}(1) = 2) &= \frac{(1 - \delta)e^\varepsilon}{1 + e^\varepsilon} \\ \Pr(\tilde{M}(0) = 3) &= 0 & \Pr(\tilde{M}(1) = 3) &= \delta. \end{aligned}$$

Then for any mechanism M that is (ε, δ) -DP and any pair of neighboring databases $\mathbf{x}^0, \mathbf{x}^1$ there exists a function T such that $T(\tilde{M}_{(\varepsilon, \delta)}(b))$ is identically distributed to $M(\mathbf{x}^b)$ for $b = 0, 1$.

Based on this result, we show that it suffices to compute the ODP guarantee of a mechanism whose iterations consist of invocations of randomized response mechanisms, in order to compute $\text{OptDel}(\{k_i\}_{i=1}^n, \{(\varepsilon_k, \delta_k)\}_{k=1}^{k_n}, \mathcal{E})$:

Lemma 13. *For any $\{k_i\}_{i=1}^n$, $\{(\varepsilon_k, \delta_k)\}_{k=1}^{k_n}$ and any \mathcal{E} we have that*

$$\begin{aligned} & \text{OptDel}(\{k_i\}_{i=1}^n, \{(\varepsilon_k, \delta_k)\}_{k=1}^{k_n}, \mathcal{E}) \\ &= \sup_{\{h^{k_i}\}_{i=1}^{n-1}} \text{OptDel}(\{k_i\}_{i=1}^n, \{\tilde{M}_{(\varepsilon_k, \delta_k)}\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}, \mathcal{E}). \end{aligned}$$

For the proof we need the following post-processing lemma, whose proof follows that of the post-processing lemma of DP [20, Prop. 2.1]:

Lemma 14. *Let $f = (f_1, \dots, f_{k_n})$ be a randomized function, where f_k may depend on the output of f_1, \dots, f_{k-1} , and let*

$$M = M_{\text{iter}}[\{k_i\}_{i=1}^n, \{M_k\}_{k=1}^{k_n}, \{h^{k_i} \circ (f_1, \dots, f_{k_i})\}_{i=1}^{n-1}]$$

be an iterative mechanism that fulfills $(\mathcal{P}^M = \{P_i^M\}_{i=1}^n, \mathcal{E}, \delta)$ -ODP. Assume that the output of each mechanism M_k , $k = 2, \dots, k_n$, only depends on the database but not on the outputs of M_1, \dots, M_{k-1} . Then the iterative mechanism

$$M' = M_{\text{iter}}[\{k_i\}_{i=1}^n, \{f_k \circ M_k\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}]$$

fulfills $(\mathcal{P}^{M'} = \{P_i^{M'}\}_{i=1}^n, \mathcal{E}', \delta)$ -ODP, where

$$\mathcal{E}'(P_i^{M'}) = \mathcal{E}(P_i^M),$$

for $i = 1, \dots, n$.

Lemma 14 allows us to prove Lemma 13, and with Lemma 13 we can prove the main result of this subsection. In the theorem and its proof we write q_1, \dots, q_k for the first k elements of a vector q .

Theorem 15. *Let $\{k_i\}_{i=1}^n$ be numbers of iterations, let $\{(\varepsilon_k, \delta_k)\}_{k=1}^{k_n}$ be DP parameters and let \mathcal{E} be a function that assigns ε -values to outputs of different lengths. For $i = 1, \dots, n$, define, for every $Q \in \{0, 1, 2, 3\}^{k_i}$ and for $b = 0, 1$,*

$$\Pr_{b_i}^{k_i}(Q) = \Pr((\tilde{M}_{(\varepsilon_1, \delta_1)}(b), \dots, \tilde{M}_{(\varepsilon_{k_i}, \delta_{k_i})}(b)) \in Q).$$

For sets $Q_{k_i} \in \{0, 1, 2, 3\}^{k_i}$ and $Q_{k_j} \in \{0, 1, 2, 3\}^{k_j}$ with $i < j$, write, with a slight abuse of notation, $Q_{k_i} \cap Q_{k_j} = \emptyset$ if $q_1, \dots, q_{k_i} \neq q'_1, \dots, q'_{k_i}$ for all $q \in Q_{k_i}$, $q' \in Q_{k_j}$. Then

$$\begin{aligned} & \text{OptDel}(\{k_i\}_{i=1}^n, \{(\varepsilon_k, \delta_k)\}_{k=1}^{k_n}, \mathcal{E}) \\ &= \max_{\substack{Q_{k_i} \in \{0, 1, 2, 3\}^{k_i} \\ i=1, \dots, n \\ Q_{k_j} \cap Q_{k_i} = \emptyset \text{ for all } j < i}} \sum_{i=1}^n \left[\Pr_0^{k_i}(Q_{k_i}) - e^{\mathcal{E}(P_i)} \Pr_1^{k_i}(Q_{k_i}) \right]. \end{aligned} \tag{3}$$

This quantity can be computed by iterating over the exponentially (in k_n) many possible sets Q_{k_i} . Murtagh and Vadhan [31] analyze the special case $n = 1$ and show that already that case is $\#P$ -complete. Thus, there is no hope for an efficient exact algorithm, but there might exist efficient approximation algorithms.

The main goal of this subsection is to show that one can get better ODP guarantees for iterative mechanisms than the ones from applying Lemma 11. With Lemma 11, we would get the following ODP- δ :

$$\begin{aligned} \delta &= \sum_{i=1}^n \max_{Q_{k_i} \in \{0,1,2,3\}^{k_i}} \left[\Pr_0^{k_i}(Q_{k_i}) - e^{\mathcal{E}(P_i)} \Pr_1^{k_i}(Q_{k_i}) \right] \\ &= \max_{\substack{Q_{k_i} \in \{0,1,2,3\}^{k_i} \\ i=1,\dots,n}} \sum_{i=1}^n \left[\Pr_0^{k_i}(Q_{k_i}) - e^{\mathcal{E}(P_i)} \Pr_1^{k_i}(Q_{k_i}) \right]. \quad (4) \end{aligned}$$

This is a direct application of Lemma 11 and the optimal composition theorem for adaptive, heterogeneous mechanisms by Murtagh and Vadhan [31] (without the simplifications of the expression performed by the authors). Intuitively, Thm. 15 gives us an exact characterization of the sets of outputs that are possible, whereas in Eq. 4 we also need to take the maximum over impossible sets of outputs: let q_1, \dots, q_{k_i} be an output sequence such that the iterative mechanism consisting of a sequence of randomized response mechanisms terminates in iteration k_i . Then it is impossible for the mechanism to output a sequence of length $> k_i$ with the prefix q_1, \dots, q_{k_i} .

This can lead to larger values for δ in Eq. 4. For example, assume that $\delta_k > 0$ for all $k = 1, \dots, k_n$. We then have $\Pr(\tilde{M}_{(\varepsilon_k, \delta_k)}(0) = 0) = \delta_k > 0$ and $\Pr(\tilde{M}_{(\varepsilon_k, \delta_k)}(1) = 0) = 0$ for all $k = 1, \dots, k_n$. Thus, each Q_{k_i} of the maximizer in Eq. 4 contains an output vector consisting of k_i δ 's, whereas for the sets Q_{k_i} of the maximizer in Eq. 3 it must hold that at most one of them contains an output vector that only consists of δ 's, leading to a strictly smaller maximum.

We leave a quantification of the size of the gap between Lemma 11 and Thm. 15 for future work, since this requires an efficient approximation algorithm for the maximization problem in Thm. 15.

5.3 Comparison with privacy filters

What we propose — stopping iterative mechanisms if they do not need more iterations, and thereby saving privacy budget that can be used on other queries — can also be done with the privacy filters introduced by Rogers et al. [32]. In fact, a similar method for saving pri-

vacuity budget when stopping early has recently been proposed in this context, though in combination with the related privacy odometers, which track privacy spending [26]. The disadvantage of privacy filters is that they cannot simply use any existing DP composition result, but require their own composition theorems. It has been shown that Rényi DP (RDP) composition can be used for privacy filters [23, 26], which is the currently best composition result for privacy filters. While RDP composition is a powerful tool, it does not always yield the best composition bounds [9]. With ODP, on the other hand, one can make use of any DP composition result via Lemma 11, though at an additional cost in δ that depends on the number n of potential stopping points and their positions.

6 Recovering already spent privacy budget

Assume that a data analyst invokes a DP mechanism M on a database \mathbf{x} , which produces an output $A \sim M(\mathbf{x})$. For some reason the data analyst is not satisfied with the output. E.g., A could be a neural network that does not perform much better than random guessing, it could be a logistic regression model whose coefficients are not statistically significant, it could be some statistic with too large a confidence interval, etc. In all of these cases, A would be essentially useless and the data analyst would not publish it. Thus, it would be desirable if the analyst could get back the privacy budget that they spent on computing A . For this, we could define a new mechanism \tilde{M} that first computes $A \sim M(\mathbf{x})$ and then invokes a test M_{test} on A that checks whether A should be released or not. If A should be released, \tilde{M} returns A , otherwise \tilde{M} returns the symbol \perp . In the first case, the data analyst would have to pay for the privacy cost of A and (potentially) the privacy cost of the test M_{test} , in the second case only for the privacy cost of M_{test} . M_{test} should thus be designed such that it uses up much less privacy budget than M .

Formally, we want \tilde{M} to be $(\{\text{Range}(M), \{\perp\}\}, \mathcal{E}, \delta)$ -ODP with $\mathcal{E}(\{\perp\}) < \mathcal{E}(\text{Range}(M))$. The test M_{test} is a function of the differentially private output A of M , and in some cases also of a private database on which to evaluate this output. This second input is not necessary if the test can be performed on A directly. In the ML setting, there might be a private database \mathbf{x} that is split up into a training set $\mathbf{x}^{\text{train}}$ that is used to train a differentially private model, and a test set \mathbf{x}^{test} on which M_{test}

Algorithm 4 $\text{ERMOuputPert}(\mathbf{x}^{\text{train}}, \varepsilon, l, N, \Lambda)$

-
- 1: Sample random vector q with element-wise density proportional to $\exp\left(-\frac{n^{\text{train}}\Lambda\varepsilon}{2}\|q\|_2\right)$
 - 2: Compute $p_{\min} = p_{\min}(\mathbf{x}^{\text{train}})$ according to Eq. 5
 - 3: **return** $\tilde{p}_{\min} = p_{\min} + q$
-

Algorithm 5 $\text{LogRegWithTest}(\mathbf{x}^{\text{train}}, \mathbf{x}^{\text{test}}, \varepsilon_1, \varepsilon_2, \Lambda, t)$

-
- 1: $\tilde{p}_{\min} = \text{ERMOuputPert}(\mathbf{x}^{\text{train}}, \varepsilon_1, l_{\text{LogReg}}, N_{L_2}, \Lambda)$
 - 2: $a = \max\left(\frac{2}{n^{\text{test}}}, 2\left(\exp\left(\frac{2}{n^{\text{train}}\Lambda}\right) - 1\right)\right)$
 - 3: Sample $r \sim \text{Lap}(a/\varepsilon_2)$
 - 4: **if** $s(\tilde{p}_{\min}, \mathbf{x}^{\text{test}}) + r \leq t$ **then**
 - 5: **return** \tilde{p}_{\min}
 - 6: **return** \perp
-

evaluates the model. In the case of a non- \perp -output, we only have to account for the privacy of M_{test} w.r.t. its database input \mathbf{x}^{test} but not w.r.t. $\mathbf{x}^{\text{train}}$, since its first input, which is based on $\mathbf{x}^{\text{train}}$, is the already differentially private output of M . Since $\mathbf{x}^{\text{train}}$ and \mathbf{x}^{test} are disjoint, we can compose the guarantees of M (w.r.t. $\mathbf{x}^{\text{train}}$) and of M_{test} (w.r.t. \mathbf{x}^{test}) via parallel composition. For the case of a \perp -output, however, we want to exploit the fact that the output of M does not get revealed to obtain a better privacy guarantee. We thus need to compute a privacy guarantee of $M_{\text{test}}(M(\mathbf{x}^{\text{train}}), \mathbf{x}^{\text{test}})$ w.r.t. both $\mathbf{x}^{\text{train}}$ and \mathbf{x}^{test} .

6.1 Example: ERM

As an example we consider the output perturbation mechanism by Chaudhuri et al. [11] for releasing linear ML models with differential privacy. The class of models that their mechanism applies to includes, among others, logistic regression and (an approximation to) support vector machines (SVMs). Chaudhuri et al. assume covariates $x \in \mathbb{R}^d$, $\|x\|_2 \leq 1$, and labels $y \in [-1, 1]$. For a set $\mathbf{x}^{\text{train}}$ of n^{train} training records they consider the empirical risk minimization problem

$$p_{\min}(\mathbf{x}^{\text{train}}) = \arg \min_p \frac{1}{n^{\text{train}}} \sum_{(x,y) \in \mathbf{x}^{\text{train}}} l(y p^T x) + \Lambda N(p), \quad (5)$$

where the minimization is over all parameter vectors p ; l is a loss function; and N a regularizer with regularization strength Λ .

Chaudhuri et al. design Alg. 4 for privately releasing the optimal parameter vector p and show that the algorithm fulfills DP:

Theorem 16. *If N is differentiable and 1-strongly convex, and l is convex and differentiable with $|l'(z)| \leq 1$ for all z , then the L_2 -sensitivity of p_{\min} is at most $\frac{2}{n^{\text{train}}\Lambda}$, and Alg. 4 is ε -DP.*

We extend Alg. 4 with a differentially private test that checks whether the differentially private model returned by Alg. 4 performs well enough for the respective application; only in that case will we release the model. For this we exemplarily look at the case of logistic regression. The loss function of logistic regression is defined as

$$l_{\text{LogReg}}(z) = \ln(1 + e^{-z}),$$

often regularized with L_2 -regularization $N_{L_2}(p) = \frac{1}{2}\|p\|_2^2$. A logistic regression model h_p with parameter vector p predicts the probability that the label of a record x is 1 as

$$h'_p(x) = \frac{1}{1 + \exp(-p^T x)},$$

and the probability that the label is -1 as $1 - h'_p(x)$. To map this output to the interval $[-1, 1]$ we define

$$h_p(x) = 2h'_p(x) - 1.$$

The prediction for the label of a record x is then typically $\text{sign}(h_p(x))$. Based on these label predictions it would be natural to use the accuracy of the model, i.e., the fraction of correct predictions, as the measure of model performance. However, if $h_p(x)$ is close to 0 on all test records, then a small change in p might flip all label predictions, which implies that the sensitivity of the accuracy function is large. Since we want to perform a differentially private test on the model performance, we need a measure with smaller sensitivity. The error function s as defined below has this property.

Let \mathbf{x}^{test} be a private test set of size n^{test} that is disjoint from the training set. For our test we use the error function

$$s(p, \mathbf{x}^{\text{test}}) = \frac{1}{n^{\text{test}}} \sum_{(x,y) \in \mathbf{x}^{\text{test}}} |h_p(x) - y|$$

— i.e., the mean absolute error made by the logistic regression model —, which can take values in $[0, 2]$. With this error function we define Alg. 5, which computes a differentially private logistic regression parameter vector, but only releases this vector if the model error on the test set is not larger than a threshold t . In Appendix D we prove the following theorem about the privacy of Alg. 5:

Theorem 17. Let $\varepsilon_1, \varepsilon_2 > 0$ and let

$$a = \max \left(\frac{2}{n^{\text{test}}}, 2 \left(\exp \left(\frac{2}{n^{\text{train}} \Lambda} \right) - 1 \right) \right).$$

Then Alg. 5 is $(\{\mathbb{R}^d, \{\perp\}\}, \mathcal{E}, 0)$ -ODP with

$$\mathcal{E}(\mathbb{R}^d) = \max \left(\varepsilon_1, \frac{2/n^{\text{test}}}{a} \varepsilon_2 \right)$$

and

$$\mathcal{E}(\{\perp\}) = \min(\mathcal{E}(\mathbb{R}^d), \varepsilon_2)$$

w.r.t. a change in either $\mathbf{x}^{\text{train}}$ or \mathbf{x}^{test} .

In Alg. 5 we have a budget ε_1 for the computation of the noisy parameter vector and a budget ε_2 for the test. Since we want to save budget in the case of a \perp -output, we will choose $\varepsilon_2 < \varepsilon_1$. This results in the following corollary:

Corollary 18. Let $\varepsilon_1 \geq \varepsilon_2 > 0$ and let

$$a = \max \left(\frac{2}{n^{\text{test}}}, 2 \left(\exp \left(\frac{2}{n^{\text{train}} \Lambda} \right) - 1 \right) \right).$$

Then Alg. 5 is $(\{\mathbb{R}^d, \{\perp\}\}, \mathcal{E}, 0)$ -ODP with

$$\mathcal{E}(\mathbb{R}^d) = \varepsilon_1$$

and

$$\mathcal{E}(\{\perp\}) = \varepsilon_2$$

w.r.t. a change in either $\mathbf{x}^{\text{train}}$ or \mathbf{x}^{test} .

Proof. The corollary follows from Thm. 17 and from the fact that

$$\frac{2/n^{\text{test}}}{a} \leq 1.$$

□

Thus, in the case where $\varepsilon_1 \geq \varepsilon_2$, we have the entire DP budget available for computing the noisy parameter vector when the test is passed. However, we pay indirectly for the test since we do not use the entire dataset for training, but reserve part of the records for testing.

Influence of noise on the test. Since noise gets added to the value of the error function, Alg. 5 will not always return \perp and save privacy budget when the model's error exceeds the threshold t . In Fig. 2 we plot the influence of this noise for $\Lambda = 1$, different values of ε_2 and different sizes of the database. We assume that 70% of the records are used for training and 30% are used for testing. The y -axis shows the 95th percentile of the distribution of the noise that is added to the error. This can be interpreted as follows: if the error exceeds t plus

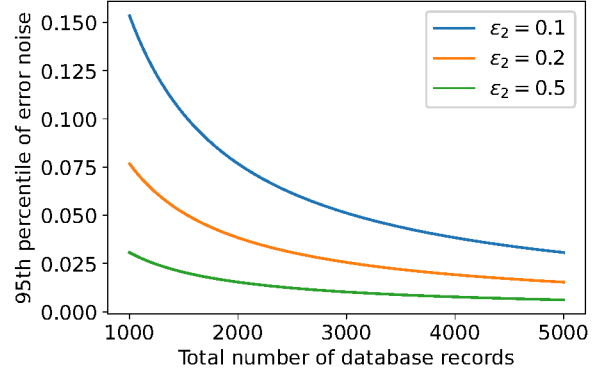


Fig. 2. 95th percentile of the noise added to the model's error for $\Lambda = 1$, different values of ε_2 and different sizes of the database (of which 70% train, 30% test data). With probability at least 95% Alg. 5 will return \perp if the error exceeds t plus the 95th percentile.

the value of the 95th percentile, then with probability at least 95% Alg. 5 will return \perp , and thus an amount of $\varepsilon_1 - \varepsilon_2$ privacy budget will be saved as compared to releasing the noisy model parameters.

7 Conclusions

In this paper, we showed how to exploit the fact that certain DP mechanisms can produce outputs that leak less private information than other outputs. Specifically, we introduced a new composition experiment and a *posteriori* analysis techniques that lead to privacy budget savings under composition. We demonstrated the power of our techniques with three examples: (1) an improved analysis of the Sparse Vector Technique and the Propose-Test-Release framework; (2) accounting for the actual number of iterations performed by an iterative mechanism; (3) recovery of already spent privacy budget when a mechanism's output is unsatisfactory.

A data analyst can profit from our privacy budget savings by reducing the amount of noise that they need to add within mechanisms to provide DP, or by invoking more mechanisms.

Future research can utilize our techniques to improve the analysis of existing DP mechanisms under composition or to guide the design of new mechanisms that provide improved privacy-utility tradeoffs. Another promising direction is the combination of our *a posteriori* analyses with advanced composition analyses.

Acknowledgments

We would like to thank the anonymous reviewers and Borja Balle for their comments, which in particular helped improve the application examples for ODP.

Bentkamp's research has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 713999, Matryoshka). It has also been funded by a Chinese Academy of Sciences President's International Fellowship for Postdoctoral Researchers (grant No. 2021PT0015). Bindschaedler's work is in part supported by the National Science Foundation under CNS-1933208. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

- [1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.
- [2] G. Barthe, M. Gaboardi, E. J. G. Arias, J. Hsu, C. Kunz, and P. Strub. Proving differential privacy in Hoare logic. In *IEEE 27th Computer Security Foundations Symposium*, pages 411–424, 2014.
- [3] G. Barthe, M. Gaboardi, B. Grégoire, J. Hsu, and P. Strub. Proving differential privacy via probabilistic couplings. In *31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 749–758, 2016.
- [4] G. Barthe, B. Köpf, F. Olmedo, and S. Z. Béguelin. Probabilistic relational reasoning for differential privacy. *ACM Transactions on Programming Languages and Systems*, 35(3):9:1–9:49, 2013.
- [5] Y. Bertot and P. Castéran. *Interactive theorem proving and program development: Coq'Art: The calculus of inductive constructions*. Springer Science & Business Media, 2013.
- [6] B. Bichsel, T. Gehr, D. Drachsler-Cohen, P. Tsankov, and M. T. Vechev. DP-Finder: Finding differential privacy violations by sampling and optimization. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 508–524, 2018.
- [7] B. Bichsel, S. Steffen, I. Bogunovic, and M. T. Vechev. DP-Sniper: Black-box discovery of differential privacy violations using classifiers. In *IEEE Symposium on Security and Privacy*, pages 391–409, 2021.
- [8] N. Bjørner, A. Phan, and L. Fleckenstein. ν Z—An optimizing SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 9035 of *Lecture Notes in Computer Science*, pages 194–199, 2015.
- [9] Z. Bu, J. Dong, Q. Long, and W. Su. Deep learning with Gaussian differential privacy. *Harvard Data Science Review*, 2(3), 2020.
- [10] M. Bun and T. Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658, 2016.
- [11] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(29):1069–1109, 2011.
- [12] L. M. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340, 2008.
- [13] L. M. de Moura, S. Kong, J. Avigad, F. van Doorn, and J. von Raumer. The Lean theorem prover (system description). In *International Conference on Automated Deduction*, volume 9195 of *Lecture Notes in Computer Science*, pages 378–388, 2015.
- [14] J. Dong, A. Roth, and W. J. Su. Gaussian differential privacy. *arXiv preprint arXiv:1905.02383*, 2019.
- [15] D. Durfee and R. M. Rogers. Practical differentially private top- k selection with pay-what-you-get composition. *Advances in Neural Information Processing Systems*, 32, 2019.
- [16] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503, 2006.
- [17] C. Dwork and J. Lei. Differential privacy and robust statistics. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 371–380, 2009.
- [18] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pages 265–284, 2006.
- [19] C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 381–390, 2009.
- [20] C. Dwork and A. Roth. *The Algorithmic Foundations of Differential Privacy*. now, 2014.
- [21] C. Dwork and G. N. Rothblum. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*, 2016.
- [22] C. Dwork, G. N. Rothblum, and S. Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2010.
- [23] V. Feldman and T. Zrnic. Individual privacy accounting via a renyi filter. *Advances in Neural Information Processing Systems*, 34, 2021.
- [24] T. Gehr, S. Misailovic, and M. T. Vechev. PSI: Exact symbolic inference for probabilistic programs. In *Computer Aided Verification*, volume 9779 of *Lecture Notes in Computer Science*, pages 62–83, 2016.
- [25] P. Kairouz, S. Oh, and P. Viswanath. The composition theorem for differential privacy. *IEEE Transactions on Information Theory*, 63(6):4037–4049, 2017.
- [26] M. Léculuyer. Practical privacy filters and odometers with Rényi differential privacy and applications to differentially private deep learning. *arXiv preprint arXiv:2103.01379*, 2021.

- [27] W. Li, F. Milletari, D. Xu, N. Rieke, J. Hancox, W. Zhu, M. Baust, Y. Cheng, S. Ourselin, M. J. Cardoso, et al. Privacy-preserving federated brain tumour segmentation. In *International workshop on machine learning in medical imaging*, pages 133–141, 2019.
- [28] K. Ligett, S. Neel, A. Roth, B. Waggoner, and Z. S. Wu. Accuracy first: Selecting a differential privacy level for accuracy-constrained ERM. *Journal of Privacy and Confidentiality*, 9:2, 2019.
- [29] M. Lyu, D. Su, and N. Li. Understanding the sparse vector technique for differential privacy. *Proceedings of the VLDB Endowment*, 10(6):637–648, 2017.
- [30] I. Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275, 2017.
- [31] J. Murtagh and S. Vadhan. The complexity of computing the optimal composition of differential privacy. In *Theory of Cryptography Conference*, pages 157–175, 2016.
- [32] R. Rogers, A. Roth, J. Ullman, and S. Vadhan. Privacy odometers and filters: Pay-as-you-go composition. *arXiv preprint arXiv:1605.08294*, 2021.
- [33] D. M. Sommer, S. Meiser, and E. Mohammadi. Privacy loss classes: The central limit theorem in differential privacy. *Proceedings on Privacy Enhancing Technologies*, 2019(2):245–269, 2019.
- [34] The mathlib community. The Lean mathematical library. In *International Conference on Certified Programs and Proofs*, pages 367–381, 2020.
- [35] Y. Wang, Z. Ding, D. Kifer, and D. Zhang. CheckDP: An automated and integrated approach for proving differential privacy or finding precise counterexamples. In *2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 919–938, 2020.
- [36] Y. Wang, Z. Ding, G. Wang, D. Kifer, and D. Zhang. Proving differential privacy with shadow execution. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, page 655–669, 2019.
- [37] Y. Wang, Z. Ding, Y. Xiao, D. Kifer, and D. Zhang. DPGen: Automated program synthesis for differential privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, page 393–411, 2021.
- [38] D. Zhang and D. Kifer. LightDP: Towards automating differential privacy proofs. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, pages 888–901, 2017.
- [39] H. Zhang, I. Mironov, and M. Hejazi. Wide network learning with differential privacy. *arXiv preprint arXiv:2103.01294*, 2021.
- [40] H. Zhang, E. Roth, A. Haeberlen, B. C. Pierce, and A. Roth. Testing differential privacy with dual interpreters. *Proceedings of the ACM on Programming Languages*, 4(OOPSLA):165:1–165:26, 2020.

A Towards advanced composition

Simple versus advanced composition. The current ODP framework is suited for improving composition when a small to medium number of mechanisms is composed. When the number of mechanisms is large, though, it is more beneficial to use one of the advanced composition theorems (see Sec. 2). Advanced composition decreases the ε in the DP guarantee when compared to simple composition, at the cost of increasing the δ -term. This increase in δ is unacceptably high for short to medium-length compositions, but asymptotically advanced composition is superior to simple composition. To provide sufficient privacy, a common recommendation is to choose $\delta \ll 1/n$, where n is the size of the database.

We consider the setting where the privacy guarantees — and thereby the utilities — of the mechanisms that are composed are fixed and the data curator has a requirement on how large the value of δ of their composition may at most be. In this setting the goal is to choose the composition theorem that yields the smallest ε while respecting the requirement on δ . For advanced composition theorems such as the optimal advanced composition theorem for the setting of composing arbitrary mechanisms with the same DP guarantees [25] by Kairouz et al., δ cannot be made arbitrarily small if a non-zero improvement in ε over simple composition is required (i in their Thm. 3.3 needs to be at least 1 in that case). The shorter the length of the composition, the larger the minimal δ . Thus, the requirement on δ can only be fulfilled if the composition is long enough; otherwise this composition theorem cannot be used.

Table 1 shows the minimal length of the composition that is required to achieve a δ not greater than a given value and an improvement in ε over simple composition when composing 0.1-DP mechanisms via the optimal advanced composition theorem by Kairouz et al. If we require fewer compositions, then advanced composition does not at the same time yield an improvement in ε over simple composition — or our composition protocol, which is based on simple composition — and fulfill the requirement on δ . Since ODP composition improves upon simple composition, it is the best choice in terms of ε in this range of composition lengths whenever at least one of the composed mechanisms has a non-trivial ODP guarantee. When the number of compositions is larger, then it depends on the concrete mechanisms whether ODP composition or advanced composition is better. Fixing the mechanisms and letting the number

of compositions go to infinity, advanced composition is superior.

Advanced composition for ODP. While in simple composition, on which our composition theorem is based, privacy in terms of ε degrades at a rate of $\mathcal{O}(\varepsilon k)$, where k is the number of invoked mechanisms and ε the ε -part of the DP guarantees of the composed mechanisms — which we, for simplicity, assume is the same for all mechanisms —, advanced composition theorems have a more gentle degradation rate of $\mathcal{O}(\varepsilon^2 k + \varepsilon \sqrt{k})$ (but only yield better results for large enough k). Thus, it would be desirable to derive an advanced composition theorem in the framework of ODP. Privacy accounting in Alg. 1 is done by subtracting after each mechanism invocation the amount of privacy budget that this mechanism has used up with its output. However, when we want privacy to degrade at a rate of $\mathcal{O}(\varepsilon \sqrt{k})$ and want to keep the freedom for the adversary to freely choose mechanisms in each iteration whose privacy guarantees are not fixed beforehand, the amount of privacy degradation that was due to one particular mechanism is not known before Alg. 1 has terminated.

For this reason, it makes sense to reformulate Alg. 1 using a *privacy filter*, as introduced by Rogers et al. [32], which newly evaluates the entire sequence of mechanisms once a new mechanism gets invoked. In Alg. 1 the adversary is only allowed to invoke mechanisms that fit within the remaining privacy budget. If the budget is used up, they may only invoke $(0,0)$ -DP mechanisms. When using a privacy filter, we do not constrain the adversary in their choice of mechanisms, but instead stop the composition if invoking the mechanism chosen by the adversary would exceed the privacy budget. We define the new composition experiment in Alg. 6, and the privacy filter for ODP $\mathcal{F}_{\varepsilon_t, \delta_t}$, which we term *ODP filter*, as follows:

Definition 19 (ODP Filter). *Fix $\varepsilon_t, \delta_t \geq 0$. A function $\mathcal{F}_{\varepsilon_t, \delta_t} : \mathbb{R}_{\geq 0}^{2I} \rightarrow \{\text{HALT}, \text{CONT}\}$ is a valid ODP filter if, for all adversaries \mathcal{A} and for all sets of views \mathcal{V} of \mathcal{A} returned by Alg. 6, we have that*

$$\Pr(V^0 \in \mathcal{V}) \leq e^{\varepsilon_t} \Pr(V^1 \in \mathcal{V}) + \delta_t.$$

Alg. 1 is equivalent to a Alg. 6 with the ODP filter that receives as input a sequence $(\varepsilon^1, \delta_1, \dots, \varepsilon^I, \delta_I)$, and returns HALT if $\sum_{i=1}^I \varepsilon^i > \varepsilon_t$ or $\sum_{i=1}^I \delta_i > \delta_t$, and CONT otherwise. Due to Thm. 7, this is a valid ODP filter. In future work, it would be interesting to explore whether there exist valid ODP filters that have an asymptotically better than linear privacy degradation, as it is

the case for privacy filters in the non-ODP setting. As an argument for why this might be the case, consider the mechanism M_{toy} from the introduction. Recall that M_{toy} flips a coin, and based on the result either invokes the Laplace mechanism on a function of the database, or returns the symbol \perp , i.e., does not reveal any information about the database. If M_{toy} is invoked k times, and in k' of the cases the output is \perp , then intuitively not more information about the database should have been revealed than when invoking the Laplace mechanism $k - k'$ times without returning \perp between some of the invocations. An ODP filter that applies an advanced composition theorem to the composition of the $k - k'$ Laplace mechanisms and only returns HALT if the resulting composition bound exceeds $(\varepsilon_t, \delta_t)$ should thus be valid.

Algorithm 6 ODPFilterComposition($\mathcal{A}, \mathcal{F}_{\varepsilon_t, \delta_t}, b$)

- 1: Select coin tosses $R_{\mathcal{A}}^b$ for \mathcal{A} uniformly at random.
 - 2: **for** $i = 1, \dots, I$ **do**
 - 3: $\mathcal{A} = \mathcal{A}(R_{\mathcal{A}}^b, \{A_j^b\}_{j=1}^i)$ chooses
 - neighboring databases $\mathbf{x}^{i,0}, \mathbf{x}^{i,1}$,
 - a triple $(\mathcal{P}_i = \{P_{i,k}\}_{k \in \mathcal{K}_i}, \mathcal{E}_i, \delta_i)$, and
 - a mechanism M_i that is $(\mathcal{P}_i, \mathcal{E}_i, \delta_i)$ -ODP
 - 4: **if** $\mathcal{F}_{\varepsilon_t, \delta_t}(\varepsilon^1, \delta_1, \dots, \varepsilon^{i-1}, \delta_{i-1}, \sup_{P \in \mathcal{P}_i} \mathcal{E}_i(P), \delta_i, 0, 0, \dots, 0, 0) = \text{HALT}$ **then**
 - 5: $A_i^b, \dots, A_I^b = \perp$
 - 6: **BREAK**
 - 7: Sample $A_i^b = M_i(\mathbf{x}^{i,b})$
 - 8: Let k such that $A_i^b \in P_{i,k}$
 - 9: Let $\varepsilon^i = \mathcal{E}_i(P_{i,k})$
 - 10: \mathcal{A} receives A_i^b
 - 11: **return** view $V^b = (R_{\mathcal{A}}^b, A_1^b, \dots, A_I^b)$
-

B Proof of Thm. 7

Lemma 20. *Let \mathbf{x}^0 and \mathbf{x}^1 be neighboring databases. Let $\varepsilon_t \geq 0$ and $\delta_t \geq 0$. Let M_1 be a $(\{P_k\}_{k \in \mathcal{K}}, \mathcal{E}, \delta)$ -output differentially private mechanism with output set \mathcal{O}_1 such that $\mathcal{E}(P_k) \leq \varepsilon_t$ for all k and $\delta \leq \delta_t$. For each $v \in \mathcal{O}_1$, let $M_2^0(v)$ and $M_2^1(v)$ be random variables with values in a set \mathcal{O}_2 such that for all subsets $\mathcal{V} \subseteq \mathcal{O}_2$ and for $v \in P_k$,*

$$\Pr(M_2^0(v) \in \mathcal{V}) \leq e^{\varepsilon_t - \mathcal{E}(P_k)} \Pr(M_2^1(v) \in \mathcal{V}) + \delta_t - \delta.$$

δ	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}	10^{-11}	10^{-12}
I	17	20	24	27	31	35	38	42

Table 1. The minimum number I of 0.1-DP mechanisms that need to be composed so that a δ -value of at most δ and a smaller ε -value than with simple composition is achieved when using the optimal composition theorem for homogeneous composition [25]. Below that number of composed mechanisms, advanced composition does not offer an advantage in terms of ε over simple composition, while ODP composition can offer an advantage over simple composition. A common recommendation is to choose $\delta \ll 1/n$, where n is the size of the database, or even cryptographically small (corresponding to roughly 10^{-12}).

Then, for all $\mathcal{V} \subseteq \mathcal{O}_1 \times \mathcal{O}_2$, we have

$$\begin{aligned} & \Pr((M_1(\mathbf{x}^0), M_2^0(M_1(\mathbf{x}^0))) \in \mathcal{V}) \\ & \leq e^{\varepsilon_t} \Pr((M_1(\mathbf{x}^1), M_2^1(M_1(\mathbf{x}^1))) \in \mathcal{V}) + \delta_t. \end{aligned}$$

Note that the two occurrences of $M_1(\mathbf{x}^0)$ in the equation refer to the same random variable, and likewise for the two occurrences of $M_1(\mathbf{x}^1)$.

Proof. Let $\mathcal{V} \subseteq \mathcal{O}_1 \times \mathcal{O}_2$. We partition \mathcal{V} into slices $\mathcal{V}_k = \mathcal{V} \cap (P_k \times \mathcal{O}_2)$ for $k \in \mathcal{K}$. In the following, we roughly follow a proof of the simple composition theorem for (ε, δ) -mechanisms by Dwork and Lei [17, Lemma 28].

We define the signed measure

$$\mu(S) = \Pr(M_1(\mathbf{x}^0) \in S) - \sum_{k \in \mathcal{K}} e^{\mathcal{E}(P_k)} \Pr(M_1(\mathbf{x}^1) \in S \cap P_k).$$

We denote the positive measure resulting from its Hahn decomposition by μ_+ . This implies that for all $S \subseteq \mathcal{O}_1$,

$$\begin{aligned} & \Pr(M_1(\mathbf{x}^0) \in S) \\ & \leq \sum_{k \in \mathcal{K}} e^{\mathcal{E}(P_k)} \Pr(M_1(\mathbf{x}^1) \in S \cap P_k) + \mu_+(S). \end{aligned}$$

For $k \in \mathcal{K}$ and $S \subseteq P_k$, it follows that

$$\Pr(M_1(\mathbf{x}^0) \in S) \leq e^{\mathcal{E}(P_k)} \Pr(M_1(\mathbf{x}^1) \in S) + \mu_+(S). \quad (6)$$

By our assumption on M_2^0 and M_2^1 , for every $v \in P_k$ and every $S \subseteq \mathcal{O}_1 \times \mathcal{O}_2$,

$$\begin{aligned} & \Pr((v, M_2^0(v)) \in S) = \Pr((v, M_2^0(v)) \in S) \wedge 1 \\ & \leq [e^{\varepsilon_t - \mathcal{E}(P_k)} \Pr((v, M_2^1(v)) \in S) + \delta_t - \delta] \wedge 1 \\ & \leq [e^{\varepsilon_t - \mathcal{E}(P_k)} \Pr((v, M_2^1(v)) \in S)] \wedge 1 + \delta_t - \delta, \end{aligned} \quad (7)$$

where \wedge denotes the minimum operator.

First, we consider a single partition P_k and its corresponding slice \mathcal{V}_k . We write V^0 for $(M_1(\mathbf{x}^0), M_2^0(M_1(\mathbf{x}^0)))$ and V^1 for $(M_1(\mathbf{x}^1), M_2^1(M_1(\mathbf{x}^1)))$. We have

$$\begin{aligned} & \Pr(V^0 \in \mathcal{V}_k) \\ & = \Pr((M_1(\mathbf{x}^0), M_2^0(M_1(\mathbf{x}^0))) \in \mathcal{V}_k) \end{aligned}$$

$$\begin{aligned} & = \int_{P_k} \Pr((v, M_2^0(v)) \in \mathcal{V}_k) \Pr(M_1(\mathbf{x}^0) \in dv) \\ & \stackrel{(7)}{\leq} \int_{P_k} ([e^{\varepsilon_t - \mathcal{E}(P_k)} \Pr((v, M_2^1(v)) \in \mathcal{V}_k)] \wedge 1 + \delta_t - \delta) \\ & \quad \times \Pr(M_1(\mathbf{x}^0) \in dv) \\ & = \int_{P_k} ([e^{\varepsilon_t - \mathcal{E}(P_k)} \Pr((v, M_2^1(v)) \in \mathcal{V}_k)] \wedge 1) \\ & \quad \times \Pr(M_1(\mathbf{x}^0) \in dv) + (\delta_t - \delta) \Pr(M_1(\mathbf{x}^0) \in P_k) \\ & \stackrel{(6)}{\leq} \int_{P_k} ([e^{\varepsilon_t - \mathcal{E}(P_k)} \Pr((v, M_2^1(v)) \in \mathcal{V}_k)] \wedge 1) \\ & \quad \times (e^{\mathcal{E}(P_k)} \Pr(M_1(\mathbf{x}^1) \in dv) + \mu_+(dv)) \\ & \quad + (\delta_t - \delta) \Pr(M_1(\mathbf{x}^0) \in P_k) \\ & \leq e^{\varepsilon_t} \int_{P_k} \Pr((v, M_2^1(v)) \in \mathcal{V}_k) \Pr(M_1(\mathbf{x}^1) \in dv) \\ & \quad + \mu_+(P_k) + (\delta_t - \delta) \Pr(M_1(\mathbf{x}^0) \in P_k) \\ & \leq e^{\varepsilon_t} \Pr((M_1(\mathbf{x}^1), M_2^1(M_1(\mathbf{x}^1))) \in \mathcal{V}_k) \\ & \quad + \mu_+(P_k) + (\delta_t - \delta) \Pr(M_1(\mathbf{x}^0) \in P_k) \\ & = e^{\varepsilon_t} \Pr(V^1 \in \mathcal{V}_k) + \mu_+(P_k) \\ & \quad + (\delta_t - \delta) \Pr(M_1(\mathbf{x}^0) \in P_k). \end{aligned} \quad (8)$$

Next, we consider all partitions P_k and slices \mathcal{V}_k together. Since M_1 is $(\{P_k\}_{k \in \mathcal{K}}, \mathcal{E}, \delta)$ -ODP, for all $S \subseteq \mathcal{O}_1$, we have

$$\Pr(M_1(\mathbf{x}^0) \in S) \leq \delta + \sum_{k \in \mathcal{K}} e^{\mathcal{E}(P_k)} \Pr(M_1(\mathbf{x}^1) \in S \cap P_k),$$

and thus $\mu(S) \leq \delta$. It follows that for all $S \subseteq \mathcal{O}_1$,

$$\mu_+(S) \leq \delta. \quad (9)$$

Thus,

$$\begin{aligned} & \Pr(V^0 \in \mathcal{V}) = \Pr(V^0 \in \bigcup_k \mathcal{V}_k) = \sum_k \Pr(V^0 \in \mathcal{V}_k) \\ & \stackrel{(8)}{\leq} \sum_k [e^{\varepsilon_t} \Pr(V^1 \in \mathcal{V}_k) + \mu_+(P_k) \\ & \quad + (\delta_t - \delta) \Pr(M_1(\mathbf{x}^0) \in P_k)] \\ & = e^{\varepsilon_t} \sum_k \Pr(V^1 \in \mathcal{V}_k) + \sum_k \mu_+(P_k) \end{aligned}$$

$$\begin{aligned}
& + \sum_k (\delta_t - \delta) \Pr(M_1(\mathbf{x}^0) \in P_k) \\
& = e^{\varepsilon_t} \Pr(V^1 \in \mathcal{V}) + \mu_+(\mathcal{O}_1) + (\delta_t - \delta) \Pr(M_1(\mathbf{x}^0) \in \mathcal{O}_1) \\
& \stackrel{(9)}{\leq} e^{\varepsilon_t} \Pr(V^1 \in \mathcal{V}) + \delta + (\delta_t - \delta) \times 1 \\
& = e^{\varepsilon_t} \Pr(V^1 \in \mathcal{V}) + \delta_t. \quad \square
\end{aligned}$$

Theorem 7. For every adversary \mathcal{A} and for every set of views \mathcal{V} of \mathcal{A} returned by Alg. 1 we have that

$$\Pr(V^0 \in \mathcal{V}) \leq e^{\varepsilon_t} \Pr(V^1 \in \mathcal{V}) + \delta_t.$$

Proof. Since the adversary's coins are tossed before any computation on the data is done, we can fix the randomness $R_{\mathcal{A}}^b$ of the adversary and thus assume a deterministic adversary.

We proceed by induction on the number of iterations I of Alg. 1. The theorem holds for $I = 1$ by Lemma 4. By the induction hypothesis, we can assume that the theorem holds for I iterations and we must show that it holds for $I + 1$ iterations.

Since the adversary is deterministic, the databases $\mathbf{x}^{1,0}$ and $\mathbf{x}^{1,1}$, the triple $(\mathcal{P}_1 = \{P_{1,k}\}_{k \in \mathcal{K}_1}, \mathcal{E}_1, \delta_1)$, and the ODP mechanism M_1 are fixed.

Let v be a possible output of M_1 . Let $\mathcal{A}_r(v)$ be the adversary that behaves like adversary \mathcal{A} after adversary \mathcal{A} has seen M_1 produce output v . Let $k \in \mathcal{K}_1$ such that $v \in P_{1,k}$. We invoke the induction hypothesis using $\mathcal{A}_r(v)$ for \mathcal{A} , $\varepsilon_t - \mathcal{E}_1(P_{1,k})$ for ε_t , and $\delta_t - \delta_1$ for δ_t to obtain

$$\Pr(V_r^0(v) \in \mathcal{V}_r) \leq e^{\varepsilon_t - \mathcal{E}_1(P_{1,k})} \Pr(V_r^1(v) \in \mathcal{V}_r) + \delta_t - \delta_1,$$

for every set of views \mathcal{V}_r of $\mathcal{A}_r(v)$ returned by Alg. 1, where $V_r^b(v)$ is the random variable describing the view of $\mathcal{A}_r(v)$ returned by Alg. 1 for bit b with total budgets $\varepsilon_t - \mathcal{E}_1(P_{1,k})$ and $\delta_t - \delta_1$.

We can thus apply Lemma 20 using M_1 for M_1 and $V_r^b(v)$ for $M_2^b(v)$, yielding

$$\begin{aligned}
& \Pr((M_1(\mathbf{x}^{1,0}), V_r^0(M_1(\mathbf{x}^{1,0}))) \in \mathcal{V}) \\
& \leq e^{\varepsilon_t} \Pr((M_1(\mathbf{x}^{1,1}), V_r^1(M_1(\mathbf{x}^{1,1}))) \in \mathcal{V}) + \delta_t.
\end{aligned}$$

Since $V^b = (M_1(\mathbf{x}^{1,b}), V_r^b(M_1(\mathbf{x}^{1,b})))$, this completes the induction step and the proof. \square

C Proofs from Sec. 5.2

We first prove Lemma 14, which we use in the proof of Lemma 13.

Lemma 14. Let $f = (f_1, \dots, f_{k_n})$ be a randomized function, where f_k may depend on the output of f_1, \dots, f_{k-1} , and let $M = \text{M}_{\text{iter}}[\{k_i\}_{i=1}^n, \{M_k\}_{k=1}^{k_n}, \{h^{k_i} \circ (f_1, \dots, f_{k_i})\}_{i=1}^{n-1}]$ be an iterative mechanism that fulfills $(\mathcal{P}^M = \{P_i^M\}_{i=1}^n, \mathcal{E}, \delta)$ -ODP. Assume that the outputs of each mechanism M_k , $k = 2, \dots, k_n$, only depends on the database but not on the outputs of M_1, \dots, M_{k-1} . Then the iterative mechanism $M' = \text{M}_{\text{iter}}[\{k_i\}_{i=1}^n, \{f_k \circ M_k\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}]$ fulfills $(\mathcal{P}^{M'} = \{P_i^{M'}\}_{i=1}^n, \mathcal{E}', \delta)$ -ODP, where, for $i = 1, \dots, n$, $\mathcal{E}'(P_i^{M'}) = \mathcal{E}(P_i^M)$.

Proof. First assume that f is deterministic. Let \mathbf{x}, \mathbf{x}' be neighboring databases and let $S' \subseteq \text{Range}(M')$. Let

$$S = \{s \in \text{Range}(M) \mid (f_1, \dots, f_{\dim(s)})(s) \in S'\},$$

where $\dim(s)$ denotes the dimensionality of the vector s . Then

$$\begin{aligned}
& \Pr(M'(\mathbf{x}) \in S') = \Pr(M(\mathbf{x}) \in S) \\
& \leq \delta + \sum_{i=1}^n e^{\mathcal{E}(P_i^M)} \Pr(M(\mathbf{x}) \in S \cap P_i^M) \\
& = \delta + \sum_{i=1}^n e^{\mathcal{E}(P_i^M)} \Pr(M'(\mathbf{x}) \in S' \cap P_i^{M'}).
\end{aligned}$$

This proves the statement for deterministic functions f . Every randomized function f can be written as a random convex combination of deterministic functions. Since every convex combination of $(\mathcal{P}, \mathcal{E}, \delta)$ -ODP mechanisms fulfills $(\mathcal{P}, \mathcal{E}, \delta)$ -ODP, the statement follows for randomized functions f . \square

Lemma 13. For any $\{k_i\}_{i=1}^n$, $\{(\varepsilon_k, \delta_k)\}_{k=1}^{k_n}$ and any \mathcal{E} we have that

$$\begin{aligned}
& \text{OptDel}(\{k_i\}_{i=1}^n, \{(\varepsilon_k, \delta_k)\}_{k=1}^{k_n}, \mathcal{E}) \\
& = \sup_{\{h^{k_i}\}_{i=1}^{n-1}} \text{OptDel}(\{k_i\}_{i=1}^n, \{\tilde{M}_{(\varepsilon_k, \delta_k)}\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}, \mathcal{E}).
\end{aligned}$$

Proof of Lemma 13. Since, for $k = 1, \dots, k_n$, $\tilde{M}_{(\varepsilon_k, \delta_k)}$ fulfills $(\varepsilon_k, \delta_k)$ -DP, we have that

$$\begin{aligned}
& \text{OptDel}(\{k_i\}_{i=1}^n, \{(\varepsilon_k, \delta_k)\}_{k=1}^{k_n}, \mathcal{E}) \\
& = \sup_{\substack{\{M_k\}_{k=1}^{k_n} \\ \{h^{k_i}\}_{i=1}^{n-1}}} \{\text{OptDel}(\{k_i\}_{i=1}^n, \{M_k\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}, \mathcal{E}) \\
& \quad \mid M_k \text{ is } (\varepsilon_k, \delta_k)\text{-DP}, k = 1, \dots, k_n\} \\
& \geq \sup_{\{h^{k_i}\}_{i=1}^{n-1}} \text{OptDel}(\{k_i\}_{i=1}^n, \{\tilde{M}_{(\varepsilon_k, \delta_k)}\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}, \mathcal{E}).
\end{aligned}$$

For the converse, define an iterative algorithm $M = \text{M}_{\text{iter}}[\{k_i\}_{i=1}^n, \{M_k\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}]$, where, for $k =$

$1, \dots, k_n$, M_k fulfills $(\varepsilon_k, \delta_k)$ -DP. Since we allow adaptive composition, M_k may depend on the outputs of M_1, \dots, M_{k-1} , which we denote by s_1, \dots, s_{k-1} . We write $M_k^{s_1, \dots, s_{k-1}}$ when we want to make this dependency explicit. Fix a pair of neighboring databases $\mathbf{x}^0, \mathbf{x}^1$. Due to Lemma 12 there exists, for every k and for every sequence of previous outputs s_1, \dots, s_{k-1} , a function $T_k^{s_1, \dots, s_{k-1}}$ such that $T_k^{s_1, \dots, s_{k-1}}(\tilde{M}_{(\varepsilon_k, \delta_k)}(b))$ follows the same distribution as $M_k^{s_1, \dots, s_{k-1}}(\mathbf{x}^b)$ for $b = 0, 1$. In following Murtagh and Vadhan, we define a function $\hat{T}(z_1, \dots, z_{k_n})$, where $z_1, \dots, z_{k_n} \in \{0, 1, 2, 3\}$, as follows:

$$\hat{T}_k(z_k) = T_k^{T_1(z_1), \dots, T_{k-1}(z_{k-1})}(z_k).$$

Then, for $b = 0, 1$,

$$M(\mathbf{x}^b) = M_{\text{iter}}[\{k_i\}_{i=1}^n, \{M_k\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}](\mathbf{x}^b)$$

follows the same distribution as

$$M'(\mathbf{x}^b) := M_{\text{iter}}[\{k_i\}_{i=1}^n, \{\hat{T}_k \circ \tilde{M}_{(\varepsilon_k, \delta_k)}\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}](\mathbf{x}^b).$$

Let

$$M'' = M_{\text{iter}}[\{k_i\}_{i=1}^n, \{\tilde{M}_{(\varepsilon_k, \delta_k)}\}_{k=1}^{k_n}, \{h^{k_i} \circ (\hat{T}_1, \dots, \hat{T}_{k_i})\}_{i=1}^{n-1}].$$

From Lemma 13 it follows that any ODP guarantee that holds for M'' also hold for M' and thus for M . Hence,

$$\begin{aligned} & \text{OptDel}(\{k_i\}_{i=1}^n, \{M_k\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}, \mathcal{E}) \\ & \leq \text{OptDel}(\{k_i\}_{i=1}^n, \{\tilde{M}_{(\varepsilon_k, \delta_k)}\}_{k=1}^{k_n}, \\ & \quad \{h^{k_i} \circ (\hat{T}_1, \dots, \hat{T}_{k_i})\}_{i=1}^{n-1}, \mathcal{E}). \end{aligned}$$

Taking the supremum over the mechanisms and the stopping criteria on both sides yields

$$\begin{aligned} & \text{OptDel}(\{k_i\}_{i=1}^n, \{(\varepsilon_k, \delta_k)\}_{k=1}^{k_n}, \mathcal{E}) \\ & = \sup_{\substack{\{M_k\}_{k=1}^{k_n} \\ \{h^{k_i}\}_{i=1}^{n-1}}} \{\text{OptDel}(\{k_i\}_{i=1}^n, \{M_k\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}, \mathcal{E}) \\ & \quad | M_k \text{ is } (\varepsilon_k, \delta_k)\text{-DP}, k = 1, \dots, k_n\} \\ & \leq \sup_{\hat{T}, \{h^{k_i}\}_{i=1}^{n-1}} \text{OptDel}(\{k_i\}_{i=1}^n, \{\tilde{M}_{(\varepsilon_k, \delta_k)}\}_{k=1}^{k_n}, \\ & \quad \{h^{k_i} \circ (\hat{T}_1, \dots, \hat{T}_{k_i})\}_{i=1}^{n-1}, \mathcal{E}) \\ & = \sup_{\{h^{k_i}\}_{i=1}^{n-1}} \text{OptDel}(\{k_i\}_{i=1}^n, \{\tilde{M}_{(\varepsilon_k, \delta_k)}\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}, \mathcal{E}). \end{aligned}$$

□

Theorem 15. Let $\{k_i\}_{i=1}^n$ be numbers of iterations, let $\{(\varepsilon_k, \delta_k)\}_{k=1}^{k_n}$ be DP parameters and let \mathcal{E} be a function

that assigns ε -values to outputs of different lengths. For $i = 1, \dots, n$, define, for every $Q \in \{0, 1, 2, 3\}^{k_i}$ and for $b = 0, 1$,

$$\Pr^{k_i}(Q) = \Pr((\tilde{M}_{(\varepsilon_1, \delta_1)}(b), \dots, \tilde{M}_{(\varepsilon_{k_i}, \delta_{k_i})}(b)) \in Q).$$

For sets $Q_{k_i} \in \{0, 1, 2, 3\}^{k_i}$ and $Q_{k_j} \in \{0, 1, 2, 3\}^{k_j}$ with $i < j$, write, with a slight abuse of notation, $Q_{k_i} \cap Q_{k_j} = \emptyset$ if $q_1, \dots, q_{k_i} \neq q'_1, \dots, q'_{k_i}$ for all $q \in Q_{k_i}$, $q' \in Q_{k_j}$. Then

$$\begin{aligned} & \text{OptDel}(\{k_i\}_{i=1}^n, \{(\varepsilon_k, \delta_k)\}_{k=1}^{k_n}, \mathcal{E}) \\ & = \max_{\substack{Q_{k_i} \in \{0, 1, 2, 3\}^{k_i} \\ i=1, \dots, n \\ Q_{k_j} \cap Q_{k_l} = \emptyset \text{ for all } j < l}} \sum_{i=1}^n \left[\Pr_0^{k_i}(Q_{k_i}) - e^{\mathcal{E}(P_i)} \Pr_1^{k_i}(Q_{k_i}) \right]. \end{aligned}$$

Proof. From Lemma 13 we know that we only have to compute

$$\sup_{\{h^{k_i}\}_{i=1}^{n-1}} \text{OptDel}(\{k_i\}_{i=1}^n, \{\tilde{M}_{(\varepsilon_k, \delta_k)}\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}, \mathcal{E}). \quad (10)$$

Fix a sequence of (potentially randomized) stopping criteria $\{\tilde{h}^{k_i}\}_{i=1}^{n-1}$. For $q \in \{0, 1, 2, 3\}^{k_i}$, write $\Pr_b^{k_i}(q) = \Pr_b^{k_i}(\{q\})$ and

$$\begin{aligned} & R(\{\tilde{h}^{k_j}\}_{j=1}^i, q) \\ & = \Pr(\tilde{h}^{k_j}(q_1, \dots, q_j) = 0 \text{ for all } j < i, \tilde{h}^{k_i}(q) = 1). \end{aligned}$$

Then $\text{OptDel}(\{k_i\}_{i=1}^n, \{\tilde{M}_{(\varepsilon_k, \delta_k)}\}_{k=1}^{k_n}, \{\tilde{h}^{k_i}\}_{i=1}^{n-1}, \mathcal{E})$ is the minimal value $\tilde{\delta}$ such that, for all $Q_{k_i} \in \{0, 1, 2, 3\}^{k_i}$, $i = 1, \dots, n$,

$$\begin{aligned} & \sum_{i=1}^n \sum_{q \in Q_{k_i}} R(\{\tilde{h}^{k_j}\}_{j=1}^i, q) \Pr_0^{k_i}(q) \\ & \leq \tilde{\delta} + \sum_{i=1}^n \sum_{q \in Q_{k_i}} e^{\mathcal{E}(P_i)} R(\{\tilde{h}^{k_j}\}_{j=1}^i, q) \Pr_1^{k_i}(q), \end{aligned}$$

i.e.,

$$\sum_{i=1}^n \sum_{q \in Q_{k_i}} R(\{\tilde{h}^{k_j}\}_{j=1}^i, q) (\Pr_0^{k_i}(q) - e^{\mathcal{E}(P_i)} \Pr_1^{k_i}(q)) \leq \tilde{\delta}. \quad (11)$$

Let H be the set of all sequences $\{h^{k_i}\}_{i=1}^{n-1}$ of deterministic stopping criteria. Since the domain and the range of all stopping criteria are finite sets, H is also a finite

set. We can thus write

$$\begin{aligned}
& \sum_{i=1}^n \sum_{q \in Q_{k_i}} R(\{\tilde{h}^{k_j}\}_{j=1}^i, q) \left(\Pr_0^{k_i}(q) - e^{\mathcal{E}(P_i)} \Pr_1^{k_i}(q) \right) \\
&= \sum_{\{h^{k_i}\}_{i=1}^{n-1} \in H} \sum_{i=1}^n \sum_{q \in Q_{k_i}} [R(\{\tilde{h}^{k_j}\}_{j=1}^i, q) \\
&\quad \Pr(\{\tilde{h}^{k_i}\}_{i=1}^{n-1} = \{h^{k_i}\}_{i=1}^{n-1}) \left(\Pr_0^{k_i}(q) - e^{\mathcal{E}(P_i)} \Pr_1^{k_i}(q) \right)] \\
&\leq \sum_{\{h^{k_i}\}_{i=1}^{n-1} \in H} \sum_{i=1}^n \sum_{q \in Q_{k_i}} \left[\max_{\{\tilde{h}^{k_i}\}_{i=1}^{n-1} \in H} R(\{\tilde{h}^{k_j}\}_{j=1}^i, q) \right. \\
&\quad \left. \Pr(\{\tilde{h}^{k_i}\}_{i=1}^{n-1} = \{h^{k_i}\}_{i=1}^{n-1}) \left(\Pr_0^{k_i}(q) - e^{\mathcal{E}(P_i)} \Pr_1^{k_i}(q) \right) \right] \\
&= \sum_{i=1}^n \sum_{q \in Q_{k_i}} \left[\max_{\{\tilde{h}^{k_i}\}_{i=1}^{n-1} \in H} R(\{\tilde{h}^{k_j}\}_{j=1}^i, q) \right. \\
&\quad \left. \left(\Pr_0^{k_i}(q) - e^{\mathcal{E}(P_i)} \Pr_1^{k_i}(q) \right) \sum_{\{h^{k_i}\}_{i=1}^{n-1} \in H} \Pr(\{\tilde{h}^{k_i}\}_{i=1}^{n-1} = \{h^{k_i}\}_{i=1}^{n-1}) \right] \\
&= \sum_{i=1}^n \sum_{q \in Q_{k_i}} \left[\max_{\{\tilde{h}^{k_i}\}_{i=1}^{n-1} \in H} R(\{\tilde{h}^{k_j}\}_{j=1}^i, q) \right. \\
&\quad \left. \left(\Pr_0^{k_i}(q) - e^{\mathcal{E}(P_i)} \Pr_1^{k_i}(q) \right) \right].
\end{aligned}$$

Thus, the worst-case in terms of δ is achieved for a sequence of deterministic stopping criteria. In Eq. 10 we therefore only have to take the supremum over deterministic stopping criteria, and can replace it with a maximum, since the set H of these criteria is finite. For deterministic stopping criteria $\{h^{k_i}\}_{i=1}^{n-1}$, the values

$$\begin{aligned}
& R(\{\tilde{h}^{k_j}\}_{j=1}^i, q) \\
&= \Pr(h^{k_j}(q_1, \dots, q_j) = 0 \text{ for all } j < i, h^{k_i}(q) = 1)
\end{aligned}$$

are each either 0 or 1. We have, using $\mathbf{1}$ as the indicator function:

$$\begin{aligned}
& \max_{\{h^{k_i}\}_{i=1}^{n-1}} \text{OptDel}(\{k_i\}_{i=1}^n, \{\tilde{M}(\varepsilon_k, \delta_k)\}_{k=1}^{k_n}, \{h^{k_i}\}_{i=1}^{n-1}, \mathcal{E}) \\
&= \max_{\{h^{k_i}\}_{i=1}^{n-1} \in H} \max_{\substack{Q_{k_i} \in \{0,1,2,3\}^{k_i} \\ i=1, \dots, n}} \\
&\quad \sum_{i=1}^n \sum_{q \in Q_{k_i}} [\mathbf{1}(h^{k_j}(q_1, \dots, q_j) = 0 \\
&\quad \text{for all } j < i, h^{k_i}(q) = 1) \\
&\quad \left(\Pr_0^{k_i}(q) - e^{\mathcal{E}(P_i)} \Pr_1^{k_i}(q) \right)]
\end{aligned} \tag{12}$$

$$\begin{aligned}
&= \max_{\substack{Q_{k_i} \in \{0,1,2,3\}^{k_i} \\ i=1, \dots, n \\ Q_{k_j} \cap Q_{k_l} = \emptyset \text{ for all } j < l}} \sum_{i=1}^n \sum_{q \in Q_{k_i}} \left[\Pr_0^{k_i}(q) - e^{\mathcal{E}(P_i)} \Pr_1^{k_i}(q) \right] \\
&= \max_{\substack{Q_{k_i} \in \{0,1,2,3\}^{k_i} \\ i=1, \dots, n \\ Q_{k_j} \cap Q_{k_l} = \emptyset \text{ for all } j < l}} \sum_{i=1}^n \left[\Pr_0^{k_i}(Q_{k_i}) - e^{\mathcal{E}(P_i)} \Pr_1^{k_i}(Q_{k_i}) \right].
\end{aligned} \tag{13}$$

The equality between Eq. 12 and Eq. 13 holds because of the following:

(12) \leq (13) since for every sequence of tests $\{h^{k_i}\}_{i=1}^{n-1}$ and for every sequence of sets $\{Q_{k_i}\}_{i=1}^n$ we get the same quantity in Eq. 13 as in Eq. 12 by choosing that sequence of subsets of the sets $\{Q_{k_i}\}_{i=1}^n$ that excludes those elements q for which

$$\mathbf{1}(h^{k_j}(q_1, \dots, q_j) = 0 \text{ for all } j < i, h^{k_i}(q) = 1) = 0.$$

This sequence of subsets fulfills $Q_{k_j} \cap Q_{k_l} = \emptyset$ for all $j < l$ because of the following observation: let $q \in \{0, 1, 2, 3\}^{k_i}$ for some $i > 1$. If $h^{k_l}(q_1, \dots, q_l) = 1$ for some $l < i$, then

$$\mathbf{1}(h^{k_j}(q_1, \dots, q_j) = 0 \text{ for all } j < i, h^{k_i}(q) = 1) = 0.$$

Hence, the union of the sequence of subsets will not include elements q_1, \dots, q_l and q_1, \dots, q_i , i.e., elements where one is the prefix of the other, at the same time.

(13) \leq (12) since for every sequence of sets $\{Q_{k_i}\}_{i=1}^n$ that fulfills $Q_{k_j} \cap Q_{k_l} = \emptyset$ for all $j < l$, we can define a sequence of tests $\{h^{k_i}\}_{i=1}^{n-1}$ such that, for every $i = 1, \dots, n$ and every $q \in Q_{k_i}$, we have $h^{k_j}(q_1, \dots, q_j) = 0$ for all $j < i$ and $h^{k_i}(q) = 1$. \square

D Proof of Thm. 17

Theorem 17. Let $\varepsilon_1, \varepsilon_2 > 0$ and let

$$a = \max \left(\frac{2}{n^{\text{test}}}, 2 \left(\exp \left(\frac{2}{n^{\text{train}} \Lambda} \right) - 1 \right) \right).$$

Then Alg. 5 is $(\{\mathbb{R}^d, \{\perp\}\}, \mathcal{E}, 0)$ -ODP with

$$\mathcal{E}(\mathbb{R}^d) = \max \left(\varepsilon_1, \frac{2/n^{\text{test}}}{a} \varepsilon_2 \right)$$

and

$$\mathcal{E}(\{\perp\}) = \min(\mathcal{E}(\mathbb{R}^d), \varepsilon_2)$$

w.r.t. a change in either $\mathbf{x}^{\text{train}}$ or \mathbf{x}^{test} .

The main ingredient for the proof of Thm. 17 is the following lemma about the error function s :

Lemma 21. Let $\varepsilon_1, \varepsilon_2 > 0$. Let Q be a random vector with element-wise density proportional to

$$\exp\left(-\frac{n^{\text{train}}\Lambda\varepsilon_1}{2}\|q\|_2\right).$$

Let further

$$a = \max\left(\frac{2}{n^{\text{test}}}, 2\left(\exp\left(\frac{2}{n^{\text{train}}\Lambda}\right) - 1\right)\right),$$

and let R be distributed according to $\text{Lap}(a/\varepsilon_2)$. Then:

1. Let \mathbf{x}^{test} be a fixed database. The function

$$g_1(\mathbf{x}^{\text{train}}) = s(p_{\min}(\mathbf{x}^{\text{train}}) + Q, \mathbf{x}^{\text{test}}) + R$$

fulfills ε_2 -DP.

2. Let p be a fixed vector. The function

$$g_2(\mathbf{x}^{\text{test}}) = s(p, \mathbf{x}^{\text{test}}) + R$$

fulfills $\left(\frac{2/n^{\text{test}}}{a}\varepsilon_2\right)$ -DP.

Proof of Thm. 17. In the case where the noisy value of the error function is not larger than the threshold in Alg. 5, the adversary learns the noisy model parameters and thereby implicitly the result of the threshold comparison in line 4. If the noisy error exceeds the threshold, the adversary only learns the result of the threshold comparison, i.e., a subset of the information of the first case. Thus, a DP guarantee for Alg. 5 can be given by composing the guarantee of Alg. 4 with a guarantee for the comparison. Since the comparison is a post-processing of the noisy error, and the vector input to the error function is already differentially private, we can instead use the guarantee for g_1 . Alg. 4 only accesses $\mathbf{x}^{\text{train}}$ and g_1 only accesses \mathbf{x}^{test} , and thus we can use parallel composition. This yields a DP-guarantee of

$$\max\left(\varepsilon_1, \frac{2/n^{\text{test}}}{a}\varepsilon_2\right),$$

and the same subset DP-guarantees for the sets \mathbb{R}^d and $\{\perp\}$ according to Lemma 3.

To get a refined privacy guarantee for the case of a \perp -output, we use the fact that in this case the adversary does not learn the noisy parameter vector but only the result of the comparison. As above, we can instead assume that the adversary receives the noisy error directly. We hence compute a DP guarantee for the computation of the error function s in line 4 w.r.t. a change of one record in either $\mathbf{x}^{\text{train}}$ or \mathbf{x}^{test} . This is given as the maximum of the DP guarantees of g_1 and g_2 , which is

$$\max\left(\varepsilon_2, \frac{2/n^{\text{test}}}{a}\varepsilon_2\right) = \varepsilon_2.$$

Thus, Alg. 5 is $(\{\perp\}, \tilde{\varepsilon}, 0)$ -subset DP for

$$\tilde{\varepsilon} = \min\left(\max\left(\varepsilon_1, \frac{2/n^{\text{test}}}{a}\varepsilon_2\right), \varepsilon_2\right).$$

□

Proof of Lemma 21.

Privacy w.r.t. $\mathbf{x}^{\text{train}}$. We first analyze the sensitivity of the (non-noisy) error function s w.r.t. $\mathbf{x}^{\text{train}}$ when adding a fixed vector q to the parameter vector. For neighboring databases $\mathbf{x}_0^{\text{train}}$, $\mathbf{x}_1^{\text{train}}$, and $p = p_{\min}(\mathbf{x}_0^{\text{train}})$, $p' = p_{\min}(\mathbf{x}_1^{\text{train}})$ we have

$$\begin{aligned} & |s(p+q, \mathbf{x}^{\text{test}}) - s(p'+q, \mathbf{x}^{\text{test}})| \\ &= \frac{1}{n^{\text{test}}} \sum_{(x,y) \in \mathbf{x}^{\text{test}}} |h_{p+q}(x) - y| - |h_{p'+q}(x) - y|. \end{aligned}$$

For any real numbers u, u', v it holds that

$$||u - v| - |u' - v|| \leq |u - u'|, \quad (14)$$

which can easily be shown by a case-by-case analysis. Further, we know from Thm. 16 that $\|p - p'\|_2 \leq \frac{2}{n^{\text{train}}\Lambda}$ and hence, by the Cauchy-Schwarz inequality,

$$\begin{aligned} |p^T x - p'^T x| &= |(p - p')^T x| \\ &\leq \|p - p'\|_2 \|x\|_2 \\ &\leq \frac{2}{n^{\text{train}}\Lambda} \times 1. \end{aligned} \quad (15)$$

Let $w_1 = (p + q)^T x$ and $w_2 = \frac{2}{n^{\text{train}}\Lambda}$. Then

$$\begin{aligned} & ||h_{p+q}(x) - y| - |h_{p'+q}(x) - y|| \\ &\stackrel{(14)}{\leq} |h_{p+q}(x) - h_{p'+q}(x)| \\ &= 2 \left| \frac{1}{1 + \exp(-(p+q)^T x)} - \frac{1}{1 + \exp(-(p'+q)^T x)} \right| \\ &\stackrel{(15)}{\leq} 2 \left| \frac{1}{1 + \exp(-(p+q)^T x)} \right. \\ &\quad \left. - \frac{1}{1 + \exp(-(p+q)^T x + \frac{2}{n^{\text{train}}\Lambda})} \right| \\ &= 2 \left(\frac{1}{1 + e^{-w_1}} - \frac{1}{1 + e^{-w_1 + w_2}} \right) \\ &= 2 \left(\frac{e^{w_1}}{e^{w_1} + 1} - \frac{e^{w_1}}{e^{w_1} + e^{w_2}} \right) \\ &= 2 \frac{e^{w_1}(e^{w_2} - 1)}{(e^{w_1} + 1)(e^{w_1} + e^{w_2})} \\ &\leq 2 \frac{e^{w_1}(e^{w_2} - 1)}{(e^{w_1} + 1)e^{w_1}} \\ &= 2 \frac{e^{w_2} - 1}{e^{w_1} + 1} \\ &\leq 2(e^{w_2} - 1) \end{aligned}$$

$$= 2 \left(\exp \left(\frac{2}{n^{\text{train}} \Lambda} \right) - 1 \right).$$

Hence,

$$\begin{aligned} & |s(p + q, \mathbf{x}^{\text{test}}) - s(p' + q, \mathbf{x}^{\text{test}})| \\ & \leq 2 \left(\exp \left(\frac{2}{n^{\text{train}} \Lambda} \right) - 1 \right). \end{aligned}$$

Let C be a set of real numbers and denote by f_R the density of R . According to the Laplace mechanism, we have

$$\begin{aligned} & \int_{c \in C} f_R(c - s(p + q, \mathbf{x}^{\text{test}})) dc \\ & \leq \exp \left(\varepsilon_2 \frac{2 \left(\exp \left(\frac{2}{n^{\text{train}} \Lambda} \right) - 1 \right)}{a} \right) \\ & \quad \int_{c \in C} f_R(c - s(p' + q, \mathbf{x}^{\text{test}})) dc \\ & \leq e^{\varepsilon_2} \int_{c \in C} f_R(c - s(p' + q, \mathbf{x}^{\text{test}})) dc. \end{aligned}$$

Therefore, writing f_Q for the density of Q :

$$\begin{aligned} & \Pr(s(p + Q, \mathbf{x}^{\text{test}}) + R \in C) \\ & = \int_{-\infty}^{\infty} f_Q(q) \int_{c \in C} f_R(c - s(p + q, \mathbf{x}^{\text{test}})) dc dq \\ & \leq \int_{-\infty}^{\infty} f_Q(q) \int_{c \in C} e^{\varepsilon_2} f_R(c - s(p' + q, \mathbf{x}^{\text{test}})) dc dq \\ & = e^{\varepsilon_2} \Pr(s(p' + Q, \mathbf{x}^{\text{test}}) + R \in C). \end{aligned}$$

Privacy w.r.t. \mathbf{x}^{test} . Since $y \in [-1, 1]$ and $h_p(x) \in [-1, 1]$, the sensitivity of $s(p, \mathbf{x}^{\text{test}})$ (with fixed vector p) w.r.t. \mathbf{x}^{test} is $\frac{2}{n^{\text{test}}}$. Thus, adding noise from $\text{Lap}(\frac{2}{n^{\text{test}}}/\varepsilon_2)$ would make the output of the error function ε_2 -DP. Since we add noise from $\text{Lap}(a/\varepsilon_2)$ instead, the output is $\left(\frac{2/n^{\text{test}}}{a}\varepsilon_2\right)$ -DP. \square