On User-Level Differential Privacy and Timing Attacks

Elena Ghazi* Zachary Ratliff†

Abstract

We initiate the study of *user-level* differential privacy (DP) in the presence of timing side channels. While DP controls what an algorithm's output reveals about any one user, these guarantees can be undermined by information leaked through the algorithm's runtime. Prior work achieved joint output/timing privacy (JOT-DP) for programs that are *timing-stable*, meaning that their runtime changes by only a bounded amount on adjacent inputs. In the user-level setting, however, privacy is defined with respect to adding or removing *all* records contributed by one user, and many natural algorithms are no longer timing-stable. In particular, if one user may contribute arbitrarily many records, then that user can induce arbitrarily large changes in runtime.

We characterize the feasibility of JOT-DP in this setting. When each user contributes at most a bounded number of records, we show that any user-level DP RAM program can be generically compiled into a JOT-DP program. In contrast, in the unbounded-contributions setting, we prove that no algorithm can achieve even approximate JOT-DP while retaining nontrivial utility on all datasets. This impossibility also yields a separation from the output-only setting, showing that unlike output-private algorithms, timing-private programs cannot always be extended from a restricted dataset domain to the full domain while preserving useful behavior on that restricted subclass. We then recover positive results under natural additional assumptions. For γ-balanced datasets, where no user contributes more than a constant fraction of the total records, we construct JOT-DP programs whose output distributions remain close to those of analogous algorithms that assume a public upper bound on the input size. Finally, assuming an unobservable offline preprocessing phase that groups the input by user identifier, we give efficient algorithms for timing-private user-level DP in the *upper-bounded contributions* setting, with runtime substantially improved over generic worst-case padding schemes.

Taken together, our results identify which assumptions make timing-private user-level DP feasible and provide guidance for designing user-level DP systems that are resilient to timing attacks.

*Harvard University. Email: elenaghazi@g.harvard.edu. Supported in part by NSF grant TIP-2453009 and a grant from the Sloan Foundation.

†Harvard University. Email: zacharyratliff@g.harvard.edu. Supported in part by the Siebel Scholarship and a grant from the Sloan Foundation.

Contents

1 Introduction

Differential privacy (DP) has become a standard foundation for privacy-preserving data analysis. Informally, it guarantees that the output of a randomized computation does not change much when any one individual's data is added or removed [DMNS06]. In many modern deployments, however, privacy is needed at the level of a *user*, not a single record. A user may contribute many correlated records, such as a purchase history, a sequence of searches and clicks, a mobility trace, a stream of device telemetry, or a long-running interaction log. This motivates *user-level* differential privacy, which aims to hide whether an entire user's contribution is present in the dataset. In practice, user-level DP is often implemented by enforcing a per-user contribution bound and then applying standard DP primitives for counts, histograms, and related aggregates, with the usual bias–variance tradeoff introduced by truncation and noise [AKMV19, EMM+20].

The definition of DP is usually stated for an adversary who observes only the algorithm's output. In interactive query systems, however, an analyst can often also measure response latency, which serves as a direct proxy for runtime. When runtime depends on the input, this timing information can undermine the intended privacy guarantee. Prior work has demonstrated timing attacks that recover sensitive information through data-dependent control flow and even through the timing behavior of the noise-generation procedure itself [HPN11, JMRO22, RBM25]. Motivated by these observations, recent work has begun to study settings in which the adversary observes both the output and the runtime, and has proposed mechanisms satisfying *joint* output/timing privacy (JOT-DP) [BDDNT23, RV24, RV25].

Definition 1.1 ((ε, δ)-Joint Output/Timing Privacy [BDDNT23, RV24])**.** Let $\mathcal{X}$ be a dataset space with an adjacency relation, $\mathcal{E}$ be a set of execution environments for a computational model, $\mathcal{Y}$ be an output space, $\mathcal{T} \subseteq \mathbb{R}^{\geq 0}$ a set of possible runtimes in the model, and $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ a randomized program in the model. Then we say that P is (ε, δ)-*jointly output/timing-private* (JOT-DP) if for all adjacent $x, x' \in \mathcal{X}$, all pairs of input-compatible execution environments $\mathtt{env}, \mathtt{env}' \in \mathcal{E}$, and all $S \subseteq \mathcal{Y} \times \mathcal{T}$

$$\Pr[(Y, T) \in S] \leq e^{\varepsilon} \cdot \Pr[(Y', T') \in S] + \delta$$

where $Y = \mathtt{out}(P(x, \mathtt{env}))$ and $T = T_P(x, \mathtt{env})$ denote the output and runtime of P on input x, respectively, and $Y' = \mathtt{out}(P(x', \mathtt{env}'))$ and $T' = T_P(x', \mathtt{env}')$ are defined analogously. When $\delta = 0$, we say that P achieves *pure* JOT-DP, and when $\delta > 0$, we say that P achieves *approximate* JOT-DP.

Definition 1.1 requires privacy to hold jointly over a program's output and runtime, relative to a specified adjacency relation on datasets. As in standard differential privacy, this adjacency relation determines the unit of protection. Here, we focus on *user-level* adjacency, under which two datasets are adjacent if they differ by the addition or removal

of all records contributed by a single user. We denote the corresponding user insert-delete distance by d_{SymUser} and define it formally in Section 2. Throughout the paper, we work in the setting where the number of users is not assumed to be public or bounded.

Prior work on JOT-DP has largely focused on the *record-level* setting, where neighboring datasets differ by a single record. Extending those guarantees to the user-level setting introduces new challenges. In particular, existing approaches achieve timing privacy by adding randomized delay to programs that are *timing-stable*, meaning that their runtime changes by only a bounded amount on adjacent inputs. However, many programs do not possess this property under the user-level adjacency relation. When one user may contribute many records, adding or removing that user can cause large, and potentially unbounded, changes in runtime, even for algorithms with bounded output sensitivity. The difficulty is especially clear when per-user contributions are unbounded, since then adding or removing one user can change the number of records processed by an arbitrary amount. In that regime, even simply processing the input record by record can produce unbounded changes in runtime under user-level adjacency. Existing JOT-DP techniques therefore do not directly apply, and a central question is whether, and under what assumptions, timing privacy can be achieved in the user-level setting.

1.1 Motivating Examples

We begin with two basic examples showing why achieving JOT-DP in the user-level setting is not straightforward using existing techniques.

Aggregating User Records. Consider a dataset x of transactions collected over time, where each record contains a user identifier u and a value $v \in [0, b]$. Because the records are ordered by time, a single user's transactions need not appear contiguously in memory. A basic analytic task is to release the total transaction value

$$q(x) = \sum_{(u,v)\in x} v$$

where each record (u, v) of user u contributes a value in $v \in [0, b]$. Under user-level adjacency, adding or removing one user adds or removes all of that user's transactions. If per-user contributions are unbounded, then q has unbounded global sensitivity with respect to d_{SymUser}. That is, adding or removing a single user can change the value of $q(x)$ by an arbitrarily large amount, since one user may contribute arbitrarily many records.

In the setting where only output privacy is required, the standard remedy is to enforce a per-user contribution cap. For example, one can truncate x so that each user u contributes at most B transactions, say by keeping only the first B in time order, and then compute q on the truncated dataset. After this preprocessing step, q has global sensitivity at most $B \cdot b$ with respect to d_{SymUser}, so it can be released using standard DP mechanisms. This

truncation may introduce bias when some users contribute more than B records, but that bias is a familiar tradeoff in user-level DP.

However, this approach does not automatically yield JOT-DP. Enforcing a per-user cap requires the algorithm to identify and count each user's records across the dataset, which in general means scanning the input and maintaining per-user state, such as counters. If one user contributes far more than the cap, the algorithm must still process those records in order to identify and discard the excess. As a result, the runtime can change by an unbounded amount between d_{SymUser}-neighboring datasets. An observer who measures execution time may therefore detect the presence or absence of a heavy contributor even when the output itself is differentially private.

Counting Users. An even simpler example is the task of counting the number of distinct users in the dataset. From the perspective of output privacy, this query is straightforward. Under user-level adjacency, the number of users has sensitivity 1, since adding or removing one user changes the count by exactly one. In particular, this query does not require any a priori bound on the number of records contributed by a single user.

From the perspective of timing privacy, however, the picture changes. To count the number of distinct users, the algorithm must inspect the records in the dataset and maintain a history of which users have already been seen. The runtime therefore depends on the total number of records that must be examined, not just on the number of users. If one user contributes many records, then the presence or absence of that user can substantially change the runtime, creating timing leakage even for a query with small output sensitivity.

1.2 Contributions

In this work, we initiate the study of JOT-DP in the user-level setting, where privacy is defined with respect to the addition or removal of an entire user's contribution. We do not assume that records are initially grouped by user, sorted by user, or equipped with auxiliary links or indices that reveal a user's contribution structure. This distinction is important because such structure can make some user-level tasks easier from the perspective of timing privacy, as discussed in Section 5.

Throughout this paper, the main distinction we use is how much structure we assume about each user's contribution. The first case is the bounded-contributions setting, where each user contributes at most a public constant number of records, but the number of users remains private and unbounded. In this setting, many linear-time record-processing algorithms become timing-stable under user-level adjacency, since adding or removing one user changes the number of processed records by at most B. However, timing stability still does not cover all useful DP computations. In particular, timing-stable programs have runtime $O(n)$ as a function of the number of records, whereas standard implementations of private medians, quantiles, and trimmed means first sort the input and therefore have

Per-user contributions	**Preproc. assumptions**	**Main result**	**Reference**
Bounded	None	We give an efficient compiler from user-level DP to JOT-DP, with arbitrarily small additional privacy loss and arbitrarily small total variation distance between the output distributions of the compiled program and the original program.	Theorem 3.1
Unbounded	None	We show that, for any query whose values can be arbitrarily far apart on well-behaved datasets, every approximately JOT-DP algorithm must incur large error on some dataset with high probability.	Theorem 4.1
Unbounded, γ-balanced datasets	None	We recover positive results on this restricted domain. In particular, if no user contributes more than a constant fraction of the dataset, then upper-bounded JOT-DP algorithms can be extended to this setting while preserving their output distribution up to arbitrarily small total variation distance and incurring only arbitrarily small additional privacy loss.	Theorem 4.8
Upper-Bounded	Grouped by user-ID	Given an input representation grouped by user, we obtain efficient JOT-DP algorithms when each user has a public but possibly very loose contribution bound B. The algorithms may choose a smaller truncation bound $b \ll B$ and run in time depending on b, rather than on the loose worst-case bound B.	Theorem 5.2

Table 1: Summary of the settings considered in this paper and the corresponding feasibility results for user-level differential privacy against timing attacks. When per-user contributions are bounded, generic compilation to JOT-DP is possible. When per-user contributions are unbounded, no general accurate approximately JOT-DP algorithm exists. Positive results can be recovered under additional structure, such as γ-balanced datasets or sorted-by-user representations that enable efficient preprocessing.

$\Omega(n \log n)$ runtime. Our first result shows that, under a per-user contribution bound, existing user-level DP algorithms can nevertheless be compiled generically into JOT-DP ones.

Theorem 1.2 (Informal Restatement of Theorem 3.1)**.** For any constants $\varepsilon' > \varepsilon$, $\delta' > \delta$, and $\beta > 0$, any (ε, δ)-DP RAM program in the user-level setting in which each user can contribute at most $B \in \mathbb{N}$ records can be transformed into a (ε', δ')-JOT-DP RAM program whose output distribution, on every input dataset x in which each user contributes at most

B records, is β-close in total variation distance to that of the original program on x.

In contrast, when per-user contributions are unbounded, timing privacy imposes a much stronger barrier. We prove that no algorithm can achieve even approximate JOT-DP while retaining nontrivial utility in full generality. This obstruction already appears for basic tasks such as counting users, but our impossibility result is more general. It applies to arbitrary target queries and continues to hold even in a *promise* setting where privacy must hold on all datasets, but accuracy is required only on well-behaved datasets in which each user contributes at most a bounded number of records,

Theorem 1.3 (Informal Restatement of Theorem 4.1)**.** Fix any bound $k \in \mathbb{N}$ and let Good_k denote the class of datasets in which every user contributes at most k records. Even if accuracy is required only on datasets in Good_k, any (ε, δ)-JOT-DP RAM program whose privacy guarantee must hold on all datasets must incur large error on some dataset in Good_k whenever the target query has unbounded range on Good_k.

Thus, without additional assumptions on the input domain, one cannot in general hope to obtain useful user-level JOT-DP algorithms in the unbounded-contributions setting. More broadly, this also yields a separation from the output-only setting. Unlike in the pure output-only DP setting, where private algorithms can always be extended from a restricted domain to the full domain [BCSZ18], timing-private executable RAM programs cannot always be extended from a well-behaved subclass to the full domain while preserving useful behavior on that subclass (Corollary 4.2).

We then show that this impossibility can be circumvented under a different type of assumption, namely a natural *restricted-domain* assumption that rules out datasets dominated by a single user. In particular, we consider the domain X_γ of γ-balanced datasets, where each user contributes at most a constant fraction of the total number of records. For this domain, we construct approximate JOT-DP programs by reducing to the *upper-bounded setting*, where a public upper bound on the input size is known in advance. Concretely, we show how to transform any timing-private algorithm for that upper-bounded setting into one for the unbounded setting that remains accurate on X_γ. The resulting program satisfies (ε, δ)-JOT-DP with respect to user-level insert/delete adjacency restricted to X_γ, meaning that privacy is required only over adjacent pairs $x, x' \in X_\gamma$. Moreover, its output distribution remains within a small constant β in total variation distance of that of the original upper-bounded algorithm.

Theorem 1.4 (Informal Restatement of Theorem 4.8)**.** Fix a constant $\gamma \in (0, 1)$ and let $\mathcal{X}_\gamma$ denote the space of datasets in which each user contributes at most a γ-fraction of the total number of records. For any (ε, δ)-JOT-DP RAM program with respect to d_{SymUser} in the *upper-bounded setting*, and for any $\varepsilon' > \varepsilon$, $\delta' > \delta$, and $\beta > 0$, there exists a compiler that transforms the program into a (ε', δ')-JOT-DP RAM program for the unbounded setting whose output distribution is β-close in total variation distance to that of the original program on all $x \in \mathcal{X}_\gamma$.

Finally, we consider the case where the online DP computation is not run on an arbitrary sequence of records. Instead, we assume that the input has already been preprocessed so that records are grouped by user identifier. This is a stronger assumption than the one used in our preceding results, since the preprocessing step itself may have data-dependent runtime and must either be unobservable to the adversary or performed outside the timing-sensitive computation. Under this assumption, the online program can access each user's contribution as a contiguous block, and enforcing a per-user truncation rule no longer requires searching through the entire dataset to find all records belonging to the same user.

Theorem 1.5 (Informal Restatement of Theorem 5.2)**.** Suppose the input is a preprocessed dataset in which records are grouped by user identifier, and suppose there is a public per-user contribution bound B. Then, for any truncation cap $b \leq B$, user-level DP computations that first truncate each user's contribution to at most b records and then run a DP computation on the truncated dataset can be implemented as JOT-DP RAM programs whose runtime depends on the number of users and the chosen cap b, rather than on the worst-case bound B.

Taken together, these results clarify when JOT-DP is achievable in the user-level setting. Without additional assumptions, accurate timing-private computation is impossible when per-user contributions are unbounded. Positive results become possible once one assumes a bound on each user's contribution, restricts attention to datasets in which no user dominates the input, or changes the input representation so that each user's contribution structure is directly accessible.

Beyond their theoretical interest, our results also provide guidance for practitioners designing user-level DP systems. They identify which input assumptions make timing privacy feasible, when timing leakage is unavoidable, and when preprocessing or data layout can materially simplify the problem. More broadly, they provide the first general constructions of efficient user-level DP programs that are also resistant to timing attacks and that closely match the accuracy of their non-timing-private counterparts.

The remainder of the paper is organized as follows. In Section 2, we review the computational model and the definition of JOT-DP. Section 3 studies the bounded-contributions setting and presents a general compiler for achieving JOT-DP. Section 4 establishes impossibility results for the unbounded-contributions setting and then gives positive results under a restricted-domain assumption. Section 5 studies preprocessed datasets with additional structure and shows how this can make timing privacy substantially easier to achieve. We conclude in Section 6 with a discussion of practical implications for designing user-level DP computations that are resistant to timing attacks.

2 Preliminaries

2.1 Computational Model

In this work, we use the **RAM model** of computation. Memory consists of an infinite sequence of cells, each capable of storing an arbitrarily large natural number. Program variables are stored in registers, and the model supports basic arithmetic operations such as addition, subtraction, multiplication, and integer division, as well as reading and writing memory. Programs may branch using conditional jumps such as **if** CONDITION **goto** LINE, and may invoke a $\texttt{RAND}(n)$ instruction that samples uniformly from $\{0, \ldots, n\}$.

The output and runtime of a RAM program may depend not only on the input, but also on its *execution environment*, which captures auxiliary machine state that may influence execution.

Definition 2.1 (RAM Execution Environment)**.** The execution environment `env` of a RAM program is the infinite sequence $(v_0, v_1, \ldots)$ such that $M[i] = v_i$ for all i, together with the values stored in built-in variables such as `input_ptr`, `input_len`, `output_ptr`, and `output_len`.

The runtime of a RAM program is defined as the number of executed basic instructions before halting. For a randomized RAM program P, input x, and compatible execution environment `env`, we write $\texttt{out}(P(x, \texttt{env}))$ for the output random variable induced by executing P on $(x, \texttt{env})$, and $T_P(x, \texttt{env})$ for the corresponding runtime random variable. We assume built-in variables `input_ptr` and `input_len`, and similarly `output_ptr` and `output_len`, that store the location and length of the program's input and output respectively.

We say that a RAM program is *output-pure* if the distribution of $\texttt{out}(P(x, \texttt{env}))$ is independent of the execution environment for all input-compatible[1] environments. Similarly, we say that a RAM program is *timing-pure* if the distribution of $T_P(x, \texttt{env})$ is independent of the execution environment for all input-compatible environments. All RAM programs in this paper satisfy both properties.

2.2 Datasets and Adjacency

We model datasets as collections of user-labeled records. Let $\mathcal{D}$ denote the record domain, and let $\mathcal{U}$ denote an abstract universe of user identifiers. The input space is

$$\mathcal{X} \;=\; \bigcup_{n \geq 0} (\mathcal{U} \times \mathcal{D})^n$$

[1]Not every execution environment is compatible with every input. For example, an environment in which memory is zeroed out is incompatible with a nonempty input whose records are expected to reside at `input_ptr`. For incompatible pairs $(x, \texttt{env})$, the program's execution is undefined. All definitions here are therefore implicitly quantified over *input-compatible* environments only.

where each element of $x \in \mathcal{X}$ is a pair (u, d) consisting of a user identifier $u \in \mathcal{U}$ and an associated record $d \in \mathcal{D}$. For a dataset x, we write $\texttt{users}(x) \subseteq \mathcal{U}$ for the set of user identifiers that appear in x.

Adjacency is defined with respect to a dataset distance metric $d_{\mathcal{X}}$. Two datasets $x, x' \in \mathcal{X}$ are adjacent if $d_{\mathcal{X}}(x, x') \leq 1$. Unless stated otherwise, we consider user-level adjacency, where $d_{\mathcal{X}}$ counts the minimum number of user insertions or deletions required to transform x into x'.

Definition 2.2 (User Insert-Delete Distance)**.** For $x \in \mathcal{D}^*$, an insertion to x is an addition of an element z to a location in x resulting in a new input $x' = [x_1, \ldots, x_i, z, x_{i+1}, \ldots, x_n]$. Likewise, a deletion from x is the removal of an element from a location i, giving a new input $x' = [x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]$. A *user insertion* to x is the k insertions of user u's records $x_u = \{x_{u_1}, x_{u_2}, \ldots, x_{u_k}\}$, giving a new dataset x' that contains all of user u's records. Similarly, a *user deletion* from x is the k deletions of user u's records $x_u = \{x_{u_1}, x_{u_2}, \ldots, x_{u_k}\}$, giving a new dataset x' without any of u's records present.

We define the *user insert-delete distance*, denoted d_{SymUser}, of inputs $x, x' \in \mathcal{D}^*$ to be the minimum number of *user* insertion and *user* deletion operations needed to transform x into x'.

This distance treats each user's entire contribution as an atomic unit. Replacing one user's data with another's therefore requires two operations, first a deletion and then an insertion.

We will consider two different types of assumptions on inputs throughout the paper.

Contribution regimes. In the *bounded-contributions setting*, each user contributes at most B records for a public constant B; we write $\mathcal{X}_B \subseteq \mathcal{X}$ for the set of such datasets. In the *unbounded-contributions setting*, no a priori bound is assumed on the number of records contributed by a single user.

Input-size regimes. In the *upper-bounded setting*, a public upper bound U on $|x|$ is known, while the exact value of $|x|$ remains private. This of course also implies a trivial per-user contribution bound of U and a trivial user bound of U, though these are not the structural assumptions of interest. In the *unbounded setting*, no public upper bound on $|x|$ is assumed.

2.3 Timing Privacy

In this work, we will use the definition of *output-conditional (OC) timing privacy*, which formalizes the requirement that a program's runtime should not leak much additional information about its input *beyond what is already revealed by the output.*

Definition 2.3 (Output-Conditional Timing Privacy [RV24])**.** Let $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ be a program and let $d_{\mathcal{X}}$ be an adjacency relation on datasets. We say that P is (ε, δ)-*OC-timing-private* with respect to $d_{\mathcal{X}}$ if for all adjacent datasets $x, x' \in \mathcal{X}$, all compatible execution environments $\mathtt{env}, \mathtt{env}' \in \mathcal{E}$, all outputs $y \in \mathcal{Y}$ such that

$$\Pr[Y = y] > 0 \quad \text{and} \quad \Pr[Y' = y] > 0,$$

and all measurable $S \subseteq \mathcal{T}$,

$$\Pr[T \in S \mid Y = y] \;\le\; e^{\varepsilon} \cdot \Pr[T' \in S \mid Y' = y] \;+\; \delta,$$

where $Y = \mathtt{out}(P(x, \mathtt{env}))$, $T = T_P(x, \mathtt{env})$, $Y' = \mathtt{out}(P(x', \mathtt{env}'))$, and $T' = T_P(x', \mathtt{env}')$.

Informally, OC-timing privacy requires that for any two adjacent inputs, and for any output value that can arise on both inputs, the corresponding conditional runtime distributions are differentially private. We also consider the natural definition of *joint output/timing privacy* (JOT-DP).

Definition 2.4 (Joint Output/Timing Privacy [BDDNT23, RV24])**.** Let $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ be a program and let $d_{\mathcal{X}}$ be an adjacency relation on datasets. We say that P is (ε, δ)-*jointly output/timing-private* with respect to $d_{\mathcal{X}}$ if for all adjacent datasets $x, x' \in \mathcal{X}$, all compatible execution environments $\mathtt{env}, \mathtt{env}' \in \mathcal{E}$, and all measurable $S \subseteq \mathcal{Y} \times \mathcal{T}$,

$$\Pr[(Y, T) \in S] \;\le\; e^{\varepsilon} \cdot \Pr[(Y', T') \in S] \;+\; \delta,$$

where $Y = \mathtt{out}(P(x, \mathtt{env}))$, $T = T_P(x, \mathtt{env})$, $Y' = \mathtt{out}(P(x', \mathtt{env}'))$, and $T' = T_P(x', \mathtt{env}')$.

The framework of [RV24] shows that DP output combined with OC-timing privacy is sufficient for JOT-DP. In particular, if a program is DP in its output and also OC-timing-private, then it is JOT-DP (Lemma A.1, Appendix). Accordingly, many of our constructions proceed by establishing output privacy and OC-timing privacy separately, and then invoking Lemma A.1 to conclude full JOT-DP. A simple but important special case is that of constant-time DP programs, which are automatically JOT-DP (Lemma A.2, Appendix). Beyond this special case, one must reason more carefully about how a program's runtime depends on its input. In particular, it is useful to isolate how much the runtime may vary between adjacent inputs once the output is fixed.

Definition 2.5 (Output-Conditional Timing Stability)**.** Let $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ be a RAM program and let $d_{\mathcal{X}}$ be an adjacency relation on datasets. We say that P is t-*OC timing-stable* with respect to $d_{\mathcal{X}}$ if for all adjacent datasets $x, x' \in \mathcal{X}$, all compatible execution environments $\mathtt{env}, \mathtt{env}' \in \mathcal{E}$, and all outputs $y \in \mathcal{Y}$ such that $\Pr[Y = y] > 0$ and $\Pr[Y' = y] > 0$, there exists a coupling (r, r') of the conditional runtime distributions $T\,|_{(Y=y)}$ and $T'\,|_{(Y'=y)}$ such that

$$\Pr\big[\,|r - r'| \le t\,\big] = 1$$

where $Y = \mathtt{out}(P(x, \mathtt{env}))$, $T = T_P(x, \mathtt{env})$, $Y' = \mathtt{out}(P(x', \mathtt{env}'))$, and $T' = T_P(x', \mathtt{env}')$.

Intuitively, OC-timing stability requires that once the output is fixed, small changes to the input can only induce bounded changes in execution time. The following theorem shows that OC-timing stability is sufficient for achieving timing privacy. In particular, bounded OC-timing stability can be converted into formal OC-timing privacy by adding an appropriate randomized delay.

Theorem 2.6 (Timing Privacy for OC-Timing-Stable Programs [RV24])**.** Let $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ be a RAM program that is t-OC-timing stable with respect to adjacency relation $d_{\mathcal{X}}$. Let D be a distribution such that, for all $\Delta \leq t$, the distributions D and $D + \Delta$ are (ε, δ)-differentially private. Then the program $\widetilde{P}$ obtained by running P and delaying its output by an independent sample from D is (ε, δ)-OC-timing private with respect to $d_{\mathcal{X}}$.

Chaining and Composition. We will also make frequent use of *chaining* of RAM programs. Given programs P_1 and P_2, the chained program $P_2 \circ P_1$ first executes P_1 and then executes P_2 using the original input together with the output of P_1. In the RAM model, chaining provides a natural way to express *composition*, where a program invokes multiple differentially private subroutines and releases their outputs in sequence. This pattern plays a central role in our constructions, which often decompose a mechanism into simpler subroutines. Moreover, composing two JOT-DP RAM programs again yields a JOT-DP RAM program, with privacy parameters that add under composition (Theorem A.3, Appendix).

Discrete Laplace Mechanism. Many of our constructions rely on adding discrete noise to integer-valued statistics. For this purpose, we recall the Discrete Laplace distribution, which serves as a standard building block in differential privacy.

Definition 2.7 (Discrete Laplace Distribution)**.** The Discrete Laplace distribution with shift $\mu \in \mathbb{Z}$ and scale $s > 0$ has probability mass function

$$p(x \mid \mu, s) = \frac{e^{1/s} - 1}{e^{1/s} + 1} \cdot e^{-|x-\mu|/s}.$$

We will often use a censored, or truncated, version of this distribution, denoted

$$\texttt{CensoredDiscreteLaplace}(\mu, s, \ell, u),$$

obtained by clamping the support to $[\ell, u]$. This mechanism is DP (Lemma A.4, Appendix), and constant-time implementations are also known (Lemma A.5, Appendix).

Finally, we adapt the pure JOT-DP length-estimation mechanism of [RV25] to the user-level setting. At a high level, the mechanism repeatedly compares the input length against progressively larger thresholds and returns a threshold that upper bounds the true input length with high probability (Lemma A.7, Appendix). The original construction is stated for the insert-delete metric in the unbounded setting, where adding or removing one record changes the input length by at most 1, so the relevant sensitivity is 1. By replacing this sensitivity parameter with B, the same construction gives the following user-level analogue.

Lemma 2.8. Fix $B \in \mathbb{N}$ and let $\mathcal{X}_B \subseteq \mathcal{X}$ denote the set of datasets in which each user contributes at most B records. For every $\varepsilon > 0$ and every $\beta \in (0,1)$, there exists a pure ε-JOT-DP RAM program $P_{\mathrm{len},B} : \mathcal{X}_B \times \mathcal{E} \to \mathbb{N} \times \mathcal{E}$ with respect to d_{SymUser} such that the following holds. For every $x \in \mathcal{X}_B$ with $n = |x|$, and every input-compatible execution environment $\mathtt{env} \in \mathcal{E}$,

$$\Pr[\mathtt{out}(P_{\mathrm{len},B}(x, \mathtt{env})) \geq n] \geq 1 - \beta.$$

Moreover, $\mathtt{out}(P_{\mathrm{len},B}(x, \mathtt{env})) = O(n)$ and runs in time $O(n)$ with high probability. In addition, there exists a deterministic function g such that $T_{P_{\mathrm{len},B}}(x, \mathtt{env}) = g(\mathtt{out}(P_{\mathrm{len},B}(x, \mathtt{env})))$ for every $x \in \mathcal{X}_B$ and every input-compatible execution environment $\mathtt{env} \in \mathcal{E}$.

Throughout the paper, we use this mechanism, together with its RAM implementation, as a black box.

3 Bounded User-Contributions Setting

We begin with the *bounded-contributions* regime, in which each user may contribute at most $B \in \mathbb{N}$ records to the dataset. This assumption limits how much any one user can affect both the output and the runtime of an algorithm. In many common algorithms, runtime is a monotone function of the total number of records processed. Thus, if each user contributes at most B records and the total number of users is public information, as in the *bounded*-DP setting, then the total number of records is bounded in advance by B times that user bound. In that case, we can pad every execution to the worst-case runtime for inputs of the maximum possible size.

The more interesting case is when the number of users is unbounded. Even in this setting, many common data-processing algorithms are timing-stable with respect to d_{SymUser} when per-user contributions are bounded. This includes algorithms whose runtime is linear in the total number of records and that perform only bounded work per record. For such algorithms, changing one user's data changes the runtime by only a bounded amount. As a result, Theorem 2.6 applies directly once the input metric is instantiated as d_{SymUser}. Still, timing-stable programs necessarily have $O(n)$ runtime [RV24], which rules out many natural data-analysis tasks.[2] For example, even counting the number of users in a dataset may require hashing or sorting user identifiers, and such computations need not be timing-stable with respect to d_{SymUser}.

Nevertheless, when per-user contributions are bounded *a priori*, we can still achieve JOT-DP for arbitrary user-level DP computations. At a high level, we first privately compute a size bound that, with high probability, is large enough to cover both the input

[2] Intuitively, timing stability bounds the change in runtime caused by adding or removing a single record. Starting from the empty dataset and adding records one at a time, the total runtime can therefore increase by at most a constant amount per record, yielding an overall $O(n)$ bound.

and any neighboring dataset. We then run the original DP algorithm if the input fits within this bound, and otherwise run it on the empty dataset, padding the execution to the worst-case runtime for that bound. On the high-probability event that the bound is large enough, the empty-dataset fallback is not used on either adjacent input, so the output distribution is exactly that of the original program and the runtime depends only on the privatized size bound. The resulting construction is therefore approximately JOT-DP.

Theorem 3.1. Fix a per-user contribution bound $B \in \mathbb{N}$ and let $\mathcal{X}_B \subseteq \mathcal{X}$ denote the set of datasets in which each user contributes at most B records. Let $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ be an (ε, δ)-DP RAM program with respect to d_{SymUser} on $\mathcal{X}_B$, and suppose that P has a known worst-case runtime bound $T(n)$ on inputs of length at most n. Fix privacy parameters $\varepsilon' > \varepsilon$ and $\delta' > \delta$, and an error parameter $\beta \in (0, 1)$. Then there exists a RAM program $\widehat{P} : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ that is (ε', δ')-JOT-DP with respect to d_{SymUser} on $\mathcal{X}_B$, and for every $x \in \mathcal{X}_B$ and every input-compatible $\mathtt{env}$,

$$d_{\mathrm{TV}}\Big(\mathtt{out}(\widehat{P}(x, \mathtt{env})), \mathtt{out}(P(x, \mathtt{env}))\Big) \leq \beta.$$

Furthermore, there exists a constant $c \geq 1$ such that for every nonempty $x \in \mathcal{X}_B$ of length n, the program $\widehat{P}$ runs in time $O\big(n + T(cn)\big)$ with high probability.

Proof. On $\mathcal{X}_B$, adding or removing one user changes the input length by at most B. Thus the length function has user-level sensitivity at most B.

Let $\varepsilon_{\mathrm{len}} := \varepsilon' - \varepsilon$ and $\beta_{\mathrm{len}} := \min\{\beta, \delta' - \delta\}$. Run the pure $\varepsilon_{\mathrm{len}}$-JOT-DP length upper-bound routine from Lemma 2.8 with failure probability β_{len}. Let its output be $\widehat{n}$, and set $U := \widehat{n} + B$. For every input $x \in \mathcal{X}_B$ of length n,

$$\Pr[U \geq n + B] \geq 1 - \beta_{\mathrm{len}}.$$

Also, by post-processing, U is pure $\varepsilon_{\mathrm{len}}$-DP.

Define $\widehat{P}$ as follows. First compute U. Then:

- if $\mathtt{input\_len}(x) \leq U$, run P on x;
- otherwise, run P on the empty dataset.

In either case, pad the entire remaining execution to time $T(U)$. Since the runtime of the length routine is a deterministic function of $\widehat{n}$, and the remaining execution is padded to a runtime depending only on U, there is a deterministic function g such that

$$T_{\widehat{P}}(x, \mathtt{env}) = g(U).$$

We now prove JOT-DP. Fix adjacent $x, x' \in \mathcal{X}_B$, compatible environments $\mathtt{env}, \mathtt{env}'$, and a measurable set $S \subseteq \mathcal{Y} \times \mathbb{N}$. Since x, x' are adjacent in $\mathcal{X}_B$, $\big||x| - |x'|\big| \leq B$. For each integer u, define

$$S_u := \{y \in \mathcal{Y} : (y, g(u)) \in S\}.$$

Let U_x and $U_{x'}$ denote the random values of U on inputs x and x', respectively. Also write

$$Y := \mathtt{out}(P(x, \mathtt{env})), \qquad Y' := \mathtt{out}(P(x', \mathtt{env}')),$$

$$\widehat{Z}_x := (\mathtt{out}(\widehat{P}(x, \mathtt{env})), T_{\widehat{P}}(x, \mathtt{env})),$$

$$\widehat{Z}_{x'} := (\mathtt{out}(\widehat{P}(x', \mathtt{env}')), T_{\widehat{P}}(x', \mathtt{env}')).$$

We have

$$\begin{aligned}
\Pr[\widehat{Z}_x \in S] &\le \Pr[U_x < |x| + B] + \sum_{u \ge |x|+B} \Pr[U_x = u] \Pr[Y \in S_u] \\
&\le \beta_{\mathrm{len}} + \sum_{u \ge |x|+B} \Pr[U_x = u] \left(e^{\varepsilon} \Pr[Y' \in S_u] + \delta\right) \\
&\le \beta_{\mathrm{len}} + \delta + e^{\varepsilon} \sum_{u \ge |x|+B} \Pr[U_x = u] \Pr[Y' \in S_u].
\end{aligned}$$

Since U is pure $\varepsilon_{\mathrm{len}}$-DP,

$$\Pr[U_x = u] \le e^{\varepsilon_{\mathrm{len}}} \Pr[U_{x'} = u]$$

for every u. Therefore

$$\Pr[\widehat{Z}_x \in S] \le \beta_{\mathrm{len}} + \delta + e^{\varepsilon'} \sum_{u \ge |x|+B} \Pr[U_{x'} = u] \Pr[Y' \in S_u].$$

For every $u \ge |x| + B$, we also have $u \ge |x'|$, so on input x' the program $\widehat{P}$ runs P on the true input x' and has runtime $g(u)$. Hence

$$\sum_{u \ge |x|+B} \Pr[U_{x'} = u] \Pr[Y' \in S_u] \le \Pr[\widehat{Z}_{x'} \in S].$$

Thus,

$$\Pr[\widehat{Z}_x \in S] \le e^{\varepsilon'} \Pr[\widehat{Z}_{x'} \in S] + \delta + \beta_{\mathrm{len}}.$$

This proves that $\widehat{P}$ is $(\varepsilon + \varepsilon_{\mathrm{len}}, \delta + \beta_{\mathrm{len}})$-JOT-DP, and hence (ε', δ')-JOT-DP.

Next we prove the output guarantee. Fix $x \in \mathcal{X}_B$ with $n = |x|$ and an input-compatible $\mathtt{env}$. On the event $U \ge n + B$, we have in particular $U \ge n$, so $\widehat{P}$ runs P on the true input x. Therefore $\mathtt{out}(\widehat{P}(x, \mathtt{env}))$ differs from $\mathtt{out}(P(x, \mathtt{env}))$ only on an event of probability at most β_{len}. Hence

$$d_{\mathrm{TV}}\Big(\mathtt{out}(\widehat{P}(x, \mathtt{env})), \mathtt{out}(P(x, \mathtt{env}))\Big) \le \beta_{\mathrm{len}} \le \beta.$$

Finally, let $x \in \mathcal{X}_B$ be nonempty with $n = |x|$. By Lemma 2.8, with high probability the length routine outputs $\widehat{n} \le c_0 n$, for some constant c_0 depending only on $B, \varepsilon_{\mathrm{len}}, \beta_{\mathrm{len}}$, and

runs in time $O(n)$. Since $U = \widehat{n} + B$, B is fixed, and $n \geq 1$, there is a constant $c \geq 1$ such that $U \leq cn$ on this event. The final execution is padded to time $T(U) \leq T(cn)$. Therefore $\widehat{P}$ runs in time

$$O(n) + T(U) = O(n + T(cn))$$

with high probability. □

In practice, per-user contribution bounds are often enforced algorithmically rather than assumed of the input *a priori*. For example, an algorithm may retain only the first B records contributed by each user. This is enough to bound output sensitivity. However, enforcing such a cap typically requires identifying and counting each user's records, which may require scanning the entire dataset. If a single user contributes many more than B records, then the cost of detecting and discarding the excess can itself depend heavily on that user's data, introducing data-dependent runtime.

This motivates the study of timing privacy in regimes where per-user contribution bounds are not assumed *a priori*, which we consider next.

4 The Unbounded Contributions Setting

The positive results of Section 3 rely on the assumption that per-user contributions are bounded a priori. When this assumption is removed, the situation changes fundamentally. In the unbounded-contributions setting, timing privacy severely limits what can be computed, even for the simplest user-level statistics. In particular, we show that no (ε, δ)-JOT-DP algorithm can achieve nontrivial accuracy on all datasets, even when accuracy is only required on well-behaved inputs.

At a high level, the impossibility arises because a single heavy-contributing user can dominate the dataset. When per-user contributions are unbounded, it is possible to construct datasets in which, with high probability, a timing-private algorithm observes only records belonging to one user before halting. Joint output/timing privacy then requires that the joint distribution of outputs and runtimes on such a dataset be close to that on a neighboring dataset in which that user is removed. As a result, the algorithm's output distribution must be nearly the same on datasets with many users and on datasets with only one, precluding accurate estimation of queries whose value depends on the number or diversity of users.

Theorem 4.1 (General Impossibility Result for Unbounded-Contributions Setting)**.** Fix any bound $k \in \mathbb{N}$. Let $\mathsf{Good}_k \subseteq \mathcal{X}$ denote the set of *good* datasets in which every user contributes at most k records:

$$\mathsf{Good}_k \;:=\; \big\{x \in \mathcal{X} : \max_u \; \#\mathtt{records}_x(u) \;\leq\; k\big\}.$$

Let $(\mathcal{Z}, d)$ be a countable metric space and let $q : \mathcal{X} \to \mathcal{Z}$ be any target query. Let $\varepsilon > 0$, $\delta \in [0, 1)$, and let $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Z} \times \mathcal{E}$ be an (ε, δ)-JOT-DP RAM program with

respect to d_{SymUser} on the full unbounded-contributions domain $\mathcal{X}$. For every β with $0 < \beta < 1 - \delta \cdot (e^{\varepsilon} + 1)$, there exists a bounded set $B \subseteq \mathcal{Z}$ with $\mathrm{diam}(B) < \infty$ such that the following holds.

For any accuracy radius $\rho \geq 0$ and any two good datasets $x, x' \in \mathsf{Good}_k$, if $d(q(x), q(x')) > \mathrm{diam}(B) + 2\rho$, then

$$\max\{\Pr[d(Y, q(x)) > \rho],\ \Pr[d(Y', q(x')) > \rho]\} \ > \ \beta,$$

where $Y = \mathtt{out}(P(x, \mathtt{env}))$ and $Y' = \mathtt{out}(P(x', \mathtt{env}'))$. In particular, if q has unbounded diameter on Good_k, then for every $\rho \geq 0$ there exists $x \in \mathsf{Good}_k$ such that

$$\Pr[d(\mathtt{out}(P(x, \mathtt{env})), q(x)) > \rho] \ > \ \beta.$$

Proof. Fix $\varepsilon \geq 0$, $\delta \in [0, 1)$, and β with $0 < \beta < 1 - \delta \cdot (e^{\varepsilon} + 1)$. Let λ denote the empty dataset. Fix parameters $p, \alpha, \eta \in (0, 1)$ to be chosen later. Let

$$t \ := \ \min\{s \in \mathbb{N} : \Pr[T_P(\lambda, \mathtt{env}_\lambda) \leq s] \geq 1 - p\}$$

so that $\Pr[T_P(\lambda, \mathtt{env}_\lambda) > t] \leq p$. Since $\mathcal{Z}$ is countable, choose a finite set $B \subseteq \mathcal{Z}$ such that

$$\Pr[\mathtt{out}(P(\lambda, \mathtt{env}_\lambda)) \in B] \ \geq \ 1 - \alpha.$$

Since B is finite, $\mathrm{diam}(B) < \infty$. Write $D := \mathrm{diam}(B)$.

Now fix an arbitrary $x \in \mathsf{Good}_k$ and let $r := |x|$. Choose an integer $m \geq \max\{r, t\}$ (to be set later), and let $x_\star$ be a dataset consisting of m records from a single user $u_\star$. Since λ and $x_\star$ are adjacent and P is (ε, δ)-JOT-DP, applying privacy to the event

$$J \ := \ \{\, T_P(\cdot, \cdot) > t \ \text{ or } \ \mathtt{out}(P(\cdot, \cdot)) \notin B \,\}$$

gives

$$\Pr[J \text{ on } x_\star] \ \leq \ e^{\varepsilon} \cdot \Pr[J \text{ on } \lambda] + \delta.$$

By a union bound on λ, $\Pr[J \text{ on } \lambda] \leq p + \alpha$, and hence

$$\Pr[T_P(x_\star, \mathtt{env}_\star) \leq t \ \cap \ \mathtt{out}(P(x_\star, \mathtt{env}_\star)) \in B] \ \geq \ 1 - \big(e^{\varepsilon}(p + \alpha) + \delta\big). \tag{1}$$

Choose the user $u_\star$ so that $u_\star \notin \mathtt{users}(x)$. Construct $\hat{x}$ from $x_\star$ by selecting a subset $I \subseteq [m]$ of size r and replacing the records of $u_\star$ at indices in I with the r records of x. Then x and $\hat{x}$ are adjacent, since $\hat{x}$ can be obtained from x by inserting the single user $u_\star$.

Couple executions of $P(x_\star, \mathtt{env}_\star)$ and $P(\hat{x}, \hat{\mathtt{env}})$ using the same internal randomness. Let $S \subseteq [m]$ denote the (random) set of input locations read by P before halting on input $x_\star$.[3] On the event $\{T_P(x_\star, \mathtt{env}_\star) \leq t\}$ we have $|S| \leq t$. Let H be the event that none of these probed locations lies in I, i.e., $H := \{S \cap I = \varnothing\}$. Conditioning on both S and the

[3] Without loss of generality, assume the m input records occupy the first m memory locations.

event $\{T_P(x_\star, \mathtt{env}_\star) \le t\}$, and using that I is a uniform r-subset of $[m]$, a union bound gives

$$\Pr[\overline{H} \mid S,\ T_P(x_\star, \mathtt{env}_\star) \le t] \ =\ \Pr[\overline{H} \mid S] \ \le\ |S| \cdot \frac{r}{m} \ \le\ t \cdot \frac{r}{m}.$$

Averaging over S therefore yields

$$\Pr[\overline{H} \mid T_P(x_\star, \mathtt{env}_\star) \le t] \ \le\ t \cdot \frac{r}{m}. \tag{2}$$

Choose m large enough that $t \cdot r/m \le \eta$, and fix a choice of I (and thus $\hat{x}$) for which (2) holds.[4]

Let

$$T_\star \ :=\ T_P(x_\star, \mathtt{env}_\star) \qquad \text{and} \qquad Y_\star \ :=\ \mathtt{out}(P(x_\star, \mathtt{env}_\star)).$$

On the event $\{T_\star \le t\} \cap H$, the execution of P on input $\hat{x}$ reads exactly the same sequence of memory contents as on input $x_\star$ before halting, and therefore produces the same output. In particular, on this event we have $\mathtt{out}(P(\hat{x}, \hat{\mathtt{env}})) = Y_\star$.

Combining (1) and (2), we obtain

$$\begin{aligned}
\Pr\big[\mathtt{out}(P(\hat{x}, \hat{\mathtt{env}})) \in B\big] &\ge \Pr\big[T_\star \le t,\ Y_\star \in B,\ H\big] \\
&\ge \Pr\big[T_\star \le t,\ Y_\star \in B\big] - \Pr\big[T_\star \le t,\ \overline{H}\big] \\
&\ge 1 - \big(e^\varepsilon(p+\alpha) + \delta\big) \ -\ \eta
\end{aligned} \tag{3}$$

where the last inequality uses (1) and the bound $\Pr[\overline{H} \mid T_\star \le t] \le \eta$.

Since x and $\hat{x}$ are adjacent and P is (ε, δ)-JOT-DP, applying privacy to the event $\{\mathtt{out} \notin B\}$ gives

$$\Pr\big[\mathtt{out}(P(x, \mathtt{env})) \notin B\big] \ \le\ e^\varepsilon \cdot \Pr\big[\mathtt{out}(P(\hat{x}, \hat{\mathtt{env}})) \notin B\big] \ +\ \delta$$

and hence, using (3),

$$\begin{aligned}
\Pr\big[\mathtt{out}(P(x, \mathtt{env})) \in B\big] &\ge 1 - e^\varepsilon \cdot \big(1 - \Pr[\mathtt{out}(P(\hat{x}, \hat{\mathtt{env}})) \in B]\big) - \delta \\
&\ge 1 - e^\varepsilon \cdot \big(e^\varepsilon(p+\alpha) + \delta + \eta\big) - \delta \\
&= 1 - e^{2\varepsilon}(p+\alpha) - \delta \cdot (e^\varepsilon + 1) - e^\varepsilon \eta.
\end{aligned}$$

Because $\beta < 1 - \delta \cdot (e^\varepsilon + 1)$, we can choose p, α, η small enough that the right-hand side exceeds β. Since $x \in \mathsf{Good}_k$ was arbitrary, this yields

$$\Pr\big[\mathtt{out}(P(x, \mathtt{env})) \in B\big] \ >\ \beta \qquad \text{for all } x \in \mathsf{Good}_k. \tag{4}$$

[4] For a uniform random choice of I, the left-hand side of (2) is at most $t \cdot r/m$, so such an I exists by averaging.

Now fix $\rho \geq 0$ and take $x, x' \in \mathsf{Good}_k$ with $d(q(x), q(x')) > D + 2\rho$. Writing $d(z, B) := \inf_{y \in B} d(z, y)$, the triangle inequality implies

$$d\big(q(x), q(x')\big) \;\leq\; d\big(q(x), B\big) + D + d\big(q(x'), B\big).$$

Therefore $\max\{d(q(x), B),\, d(q(x'), B)\} > \rho$. Assume without loss of generality that $d(q(x), B) > \rho$. Then the event $\{\mathtt{out}(P(x, \mathtt{env})) \in B\}$ implies $d(\mathtt{out}(P(x, \mathtt{env})), q(x)) > \rho$, and by (4),

$$\Pr\big[d(\mathtt{out}(P(x, \mathtt{env})), q(x)) > \rho\big] \;\geq\; \Pr\big[\mathtt{out}(P(x, \mathtt{env})) \in B\big] \;>\; \beta.$$

The other case is symmetric, proving the first claim. The final claim follows by taking $x, x' \in \mathsf{Good}_k$ with $d(q(x), q(x'))$ arbitrarily large. □

Theorem 4.1 applies to (ε, δ)-JOT-DP, allows arbitrary target queries, and is stated in a *promise* setting in which accuracy is required only on "good" datasets where each user contributes at most k records, while privacy must hold on the full unbounded-contributions domain. In particular, if a query has unbounded range on Good_k, then there must exist some good dataset on which the algorithm incurs large error with nontrivial probability.

Importantly, this impossibility persists even under extremely strong distributional assumptions. For example, suppose the dataset is drawn from a distribution that places probability $1 - 10^{-100}$ on Good_k. Even then, there exists a constant N such that for all $n > N$, there is a dataset with n users in which every user contributes at most k records, yet any (ε, δ)-JOT-DP algorithm that attempts to count the number of users must incur error $\Omega(n)$ on that dataset with nontrivial probability (Corollary A.10).

Timing Private Algorithms Cannot Always Be Extended. A useful point of comparison comes from the output-only setting. Borgs, Chayes, Smith, and Zadik [BCSZ18] show that if an ε-DP randomized algorithm is defined on a restricted domain $\mathcal{H} \subseteq \mathcal{X}$, then there exists a 2ε-DP randomized algorithm on all of $\mathcal{X}$ whose output distribution agrees exactly with the original algorithm on every input in $\mathcal{H}$. Thus, in ordinary output-only differential privacy, any ε-DP mechanism defined on a well-behaved subclass admits an extension to the full domain, with only a constant-factor loss in privacy and exact preservation of its output distribution on that subclass. One might hope for an analogous extension principle in the timing privacy setting, however, Theorem 4.1 rules out such a result.

Corollary 4.2 (Timing-private algorithms cannot always be extended)**.** There exist queries and user-level JOT-DP programs defined on a restricted domain $\mathcal{H} \subseteq \mathcal{X}$ that are accurate on $\mathcal{H}$, but that do not admit any extension to the full domain $\mathcal{X}$ that satisfies global JOT-DP and preserves useful accuracy on $\mathcal{H}$.

4.1 Privacy for γ-Balanced Datasets

Impossibility results in the unbounded-contributions setting are driven by the fact that a neighboring dataset may differ by a user whose contribution is so large that it can dominate the computation and radically change the runtime distribution.

We therefore study a relaxed setting in which no single user can dominate the dataset. Concretely, we restrict attention to balanced datasets, where every user contributes at most a γ-portion of the records, for $\gamma \in (0, 1)$. We keep the standard user-level insert/delete distance d_{SymUser}, but require the JOT-DP guarantee only for adjacent pairs that remain within the balanced class.

Definition 4.3 (γ-Balanced datasets)**.** We say that a dataset $x \in \mathcal{X}$ is γ-balanced if for all users $u \in \texttt{users}(x)$,

$$c_u(x) \leq \gamma |x|.$$

We let $\mathcal{X}_\gamma \subseteq \mathcal{X}$ denote the set of γ-balanced datasets.

Restricted sensitivity on balanced domains. Our positive result follows the restricted-sensitivity philosophy of Blocki, Blum, Datta, and Sheffet [BBDS13]: a statistic may have unbounded sensitivity on the full domain, but bounded sensitivity on a structurally restricted class. Here the restricted class is $\mathcal{X}_\gamma$. Although input length has unbounded user-level sensitivity on the full unbounded-contributions domain, adjacent datasets that both lie in $\mathcal{X}_\gamma$ have comparable sizes. Consequently, the coarsened length statistic

$$I_b(|x|) = \lfloor \log_b |x| \rfloor$$

has bounded one-step restricted sensitivity on $\mathcal{X}_\gamma$.

Unlike projection-based mechanisms for restricted sensitivity, our result in this section is a restricted-domain JOT-DP guarantee: both datasets in the adjacent pair are required to lie in $\mathcal{X}_\gamma$.

A subtlety with restricting attention to balanced datasets is that a user-level delete can either preserve balance or break it, depending on the rest of the dataset. For instance, the dataset with user IDs $[a, a, b, b, c, c]$ is $(1/2)$-balanced, and deleting user a yields $[b, b, c, c]$, which is still balanced. In contrast, deleting user a from $[a, a, b, b, b, c, c, c, c, c]$ yields $[b, b, b, c, c, c, c, c]$, which is not $(1/2)$-balanced because user c contributes more than half of the remaining records. Such deletes are outside the restricted threat model.

This restriction does not make the privacy notion vacuous. Many natural substitution-style changes remain protected up to a constant-factor loss in the privacy parameters. For example, replacing one user's contribution by another user's contribution of the same size can be routed through a short sequence of balanced insert/delete moves: first insert the new user as an auxiliary user, then delete the old user. Lemma A.11 formalizes this by showing that such substitutions have distance at most 2 in the balanced adjacency graph.

Lemma 4.4. Fix $\gamma \in (0,1)$. For any two datasets x and x' that are γ-balanced and satisfy $d_{\mathrm{SymUser}}(x, x') = 1$, it follows that

$$\left(\frac{1}{1-\gamma}\right) \cdot n' \;\geq\; n \;\geq\; (1-\gamma) \cdot n'$$

and hence

$$\max\left\{\frac{n}{n'}, \frac{n'}{n}\right\} \leq \frac{1}{1-\gamma}$$

where $n = |x|$ and $n' = |x'|$.

Proof. First note that $n, n' > 0$. Indeed, if one of the two datasets were empty, then the other would consist of the records of a single user, and hence that user would contribute all records, contradicting $\gamma < 1$ and γ-balance. Assume without loss of generality that x is obtained from x' by adding one user's contribution (the removal case is symmetric). Then $n = n' + m$ where $m = c_u(x)$ for the added user u. Since x is γ-balanced, $m \leq \gamma n = \gamma(n' + m)$, so $(1-\gamma) \cdot m \leq \gamma n'$ and therefore

$$n \;=\; n' + m \;\leq\; n' + \left(\frac{\gamma}{1-\gamma}\right) \cdot n' \;=\; \left(\frac{1}{1-\gamma}\right) \cdot n'.$$

For the other direction, apply the same inequality with the roles of x and x' swapped, which yields $n \geq (1-\gamma) \cdot n'$. □

Restricted sensitivity of logarithmic size buckets.

Lemma 4.5. Fix $\gamma \in (0,1)$ and restrict attention to γ-balanced datasets. Let $b \geq 2$ and define

$$I_b(n) := \begin{cases} 0 & \text{if } n = 0, \\ \lfloor \log_b n \rfloor & \text{if } n \geq 1. \end{cases}$$

If x, x' are user-level neighbors with $n = |x|$ and $n' = |x'|$, then

$$\left|I_b(n) - I_b(n')\right| \leq \left\lceil \log_b\left(\frac{1}{1-\gamma}\right) \right\rceil.$$

In particular, if $b \geq \frac{1}{1-\gamma}$ then $|I_b(n) - I_b(n')| \leq 1$ for all neighboring γ-balanced datasets.

Proof. First define $\kappa = 1/(1-\gamma)$ and let $s := \lceil \log_b(\kappa) \rceil$. By Lemma 4.4, we have $n \leq \kappa \cdot n'$ and $n' \leq \kappa \cdot n$. Assume without loss of generality that $n \geq n'$ so that $n \leq \kappa \cdot n'$. Let $j' = I_b(n')$, so $b^{j'} \leq n' < b^{j'+1}$. Multiplying by κ gives $n \leq \kappa \cdot n' < \kappa \cdot b^{j'+1}$. By the definition of s, we have $\kappa \leq b^s$, and hence $n < b^s \cdot b^{j'+1} = b^{j'+s+1}$. Since $n \geq n' \geq b^{j'}$, it follows that $n \in [b^{j'}, b^{j'+s+1})$, which implies $I_b(n) \leq j' + s$. Therefore $I_b(n) - I_b(n') \leq s$. The case $n < n'$ is symmetric, giving $|I_b(n) - I_b(n')| \leq s$. □

Program 1 Compute the bucket index $I_b(|x|)$

Description: A dataset x and a public integer base $b \geq 2$. The program outputs $I_b(|x|)$.

```
n = input_len
m = n
j = 0
while m ≥ b do
    m = ⌊m/b⌋
    j = j + 1
return j
```

Lemma 4.6. Fix $\gamma \in (0,1)$ and a public integer base $b \geq 2$, and let

$$s \coloneqq \left\lceil \log_b\left(\frac{1}{1-\gamma}\right) \right\rceil.$$

Define

$$I_b(n) \coloneqq \begin{cases} 0 & \text{if } n = 0, \\ \lfloor \log_b n \rfloor & \text{if } n \geq 1. \end{cases}$$

Then, in the RAM model, there is an $O(s)$-timing-stable program with respect to d_{SymUser} on $\mathcal{X}_\gamma$ that accepts a dataset $x \in \mathcal{X}_\gamma$ and outputs $I_b(|x|)$.

Proof. We give a proof by construction in Program 1. The program reads the input length $n = |x|$, initializes $m = n$ and $j = 0$, and then repeatedly divides m by b while incrementing j, until $m < b$. If $n \geq 1$, this loop performs exactly $\lfloor \log_b n \rfloor$ iterations, and hence outputs $I_b(n)$. If $n = 0$, the loop executes zero iterations and outputs $0 = I_b(0)$.

Each loop iteration uses $O(1)$ RAM operations, so on inputs of lengths n and n', the difference in running time is

$$O\big(\big|I_b(n) - I_b(n')\big|\big).$$

Now let $x, x' \in \mathcal{X}_\gamma$ be d_{SymUser}-adjacent, and write $n = |x|$ and $n' = |x'|$. By Lemma 4.5,

$$\big|I_b(n) - I_b(n')\big| \leq s.$$

Therefore the running time changes by at most $O(s)$ on adjacent inputs in $\mathcal{X}_\gamma$, as claimed. □

Lemma 4.7. Fix $\gamma \in (0,1)$ and a public integer base $b \geq 2$ such that

$$b \geq \frac{1}{1-\gamma}.$$

Let $\mathcal{X}_\gamma \subseteq \mathcal{X}$ denote the set of γ-balanced datasets.

For every $\varepsilon > 0$, $\delta > 0$, and $\beta \in (0, 1]$, there exists an (ε, δ)-JOT-DP RAM program $P_{\mathrm{len},\gamma} : \mathcal{X}_\gamma \times \mathcal{E} \to \mathbb{N} \times \mathcal{E}$ with respect to d_{SymUser} on $\mathcal{X}_\gamma$ such that, for every $x \in \mathcal{X}_\gamma$ and every input-compatible $\mathtt{env}$,

$$\Pr[\mathtt{out}(P_{\mathrm{len},\gamma}(x, \mathtt{env})) \geq I_b(|x|)] \geq 1 - \beta.$$

Proof. Let

$$s \coloneqq \left\lceil \log_b\left(\frac{1}{1-\gamma}\right) \right\rceil$$

and let P_b be the deterministic RAM program from Program 1, which outputs $j = I_b(|x|)$.

By Lemma 4.6, P_b is $O(s)$-timing stable with respect to d_{SymUser} on $\mathcal{X}_\gamma$. Moreover, the map $x \mapsto I_b(|x|)$ has one-step restricted sensitivity at most 1 on $\mathcal{X}_\gamma$.

Fix any $\varepsilon' \in (0, \varepsilon)$. Set

$$a \coloneqq \left\lceil \frac{\ln(1/\beta)}{\varepsilon'} \right\rceil.$$

We use ε' for the output noise and the remaining budget $\varepsilon - \varepsilon'$, together with δ, for the timing-private delay in Lemma A.6. By Lemma A.6, there exists an (ε, δ)-JOT-DP RAM program that samples

$$\tilde{j} \sim \mathtt{DiscreteLaplace}\left(I_b(|x|) + a, \frac{1}{\varepsilon'}\right).$$

The program releases

$$\hat{j} := \max\{0, \tilde{j}\}.$$

Let $j = I_b(|x|)$. Since $j \geq 0$, whenever $\tilde{j} \geq j$ we also have $\hat{j} \geq j$. Therefore

$$\Pr[\hat{j} < j] \leq \Pr[\tilde{j} \leq j - 1].$$

Writing

$$\tilde{j} = j + a + Z$$

where $Z \sim \mathtt{DiscreteLaplace}(0, 1/\varepsilon')$, we have

$$\Pr[\tilde{j} \leq j - 1] = \Pr[Z \leq -a - 1].$$

For the discrete Laplace distribution,

$$\Pr[Z \leq -a - 1] = \frac{e^{-(a+1)\varepsilon'}}{1 + e^{-\varepsilon'}} \leq e^{-(a+1)\varepsilon'} \leq \beta.$$

Thus

$$\Pr[\hat{j} \geq I_b(|x|)] \geq 1 - \beta.$$

Finally, since $\hat{j}$ is obtained from $\tilde{j}$ by post-processing, the released output inherits the (ε, δ)-JOT-DP guarantee. □

Theorem 4.8. Fix $\gamma \in (0,1)$ and an integer $b \geq \max\{2, \lceil 1/(1-\gamma) \rceil\}$. Let $\mathcal{X}_\gamma \subseteq \mathcal{X}$ denote the set of γ-balanced datasets. Let P be an (ε, δ)-JOT-DP RAM program with respect to d_{SymUser} in the upper-bounded setting. Fix privacy parameters $\varepsilon' > \varepsilon$, $\delta' > \delta$, and an error parameter $\beta \in (0,1)$. Then there exists a RAM program $\hat{P} : \mathcal{X}_\gamma \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ for the unbounded setting such that $\hat{P}$ is (ε', δ')-JOT-DP with respect to d_{SymUser} on $\mathcal{X}_\gamma$, and for every $x \in \mathcal{X}_\gamma$ and every input-compatible $\texttt{env}$,

$$d_{\text{TV}}\Big(\texttt{out}(\hat{P}(x, \texttt{env})), \texttt{out}(P(x, \texttt{env}))\Big) \leq \beta.$$

Proof. Let $\varepsilon_{\text{len}} := \varepsilon' - \varepsilon$. Fix any $\eta \in (0, \min\{\beta, \delta' - \delta\})$, and set $\delta_{\text{len}} := e^{-\varepsilon}(\delta' - \delta - \eta)$. Also let $j(x) := I_b(|x|)$. By Lemma 4.7, there exists an $(\varepsilon_{\text{len}}, \delta_{\text{len}})$-JOT-DP private length-estimation subroutine P_{len} that releases an integer $\hat{j}$ such that, for every $x \in \mathcal{X}_\gamma$, we have $\Pr[\hat{j} \geq j(x)] \geq 1 - \eta$. For each public upper bound $U \in \mathbb{N}$, let P_U denote the upper-bounded implementation of P on inputs of length at most U. Define $\hat{P}$ as follows. On input $x \in \mathcal{X}_\gamma$, first run P_{len} to obtain $\hat{j}$, and set $U := b^{\hat{j}+2}$. If $|x| \leq U$, run P_U on input x. Otherwise, run P_U on the empty dataset.

We first prove the utility guarantee. Fix $x \in \mathcal{X}_\gamma$, and let $j := j(x)$.

On the event $G_x := \{\hat{j} \geq j\}$, we have $U = b^{\hat{j}+2} \geq b^{j+2} > |x|$. Thus, on G_x, the program runs the valid upper-bounded execution P_U on input x.

Therefore

$$d_{\text{TV}}\Big(\texttt{out}(\hat{P}(x, \texttt{env})), \texttt{out}(P(x, \texttt{env}))\Big) \leq \Pr[G_x^c] \leq \eta \leq \beta.$$

We now prove privacy. Fix adjacent datasets $x, x' \in \mathcal{X}_\gamma$, compatible environments $\texttt{env}, \texttt{env}'$, and a measurable set $S \subseteq \mathcal{Y} \times \mathbb{N}$. Let $j := I_b(|x|)$. Since I_b has one-step restricted sensitivity at most 1 on $\mathcal{X}_\gamma$, we have $I_b(|x'|) \leq j + 1$.

Let $(\hat{Y}, \hat{T})$ and $(\hat{Y}', \hat{T}')$ denote the joint observations of $\hat{P}$ on inputs $(x, \texttt{env})$ and $(x', \texttt{env}')$, respectively. Let H_x and $H_{x'}$ denote the joint observations of P_{len} on inputs x and x', respectively.

For each possible joint observation h of P_{len}, let $\hat{j}_h$ be the released noisy bucket index contained in h, and set $U_h := b^{\hat{j}_h+2}$.

Let S_h be the set of possible joint observations of the remaining execution after P_{len} that, together with h, yield an overall joint observation in S. Let (Y_h, T_h) and (Y_h', T_h') denote the joint observations of this remaining execution on inputs $(x, \texttt{env})$ and $(x', \texttt{env}')$, respectively, when the joint observation of P_{len} is fixed to h.

Now fix any such h with $\hat{j}_h \geq j$. Then $U_h \geq b^{j+2}$. Also $|x| < b^{j+1}$ and $|x'| < b^{I_b(|x'|)+1} \leq b^{j+2}$, so both x and x' have length at most U_h. Therefore, when the joint observation of P_{len} is fixed to such an h, the remaining execution runs P_{U_h} on inputs x and x'. For such an h, both inputs take the same branch, and the branch check and setup contribute the same deterministic runtime offset before the call to P_{U_h}. Since P_{U_h} is (ε, δ)-JOT-DP, we have

$$\Pr[(Y_h, T_h) \in S_h] \leq e^{\varepsilon} \Pr[(Y_h', T_h') \in S_h] + \delta.$$

Now define

$$\psi(h) := \begin{cases} \Pr[(Y'_h, T'_h) \in S_h] & \text{if } \hat{j}_h \geq j \\ 0 & \text{otherwise.} \end{cases}$$

Then $\psi(h) \in [0,1]$ for every h. Using the previous bound and the fact that $\Pr[\hat{j} < j] \leq \eta$, we get

$$\begin{aligned} \Pr[(\hat{Y}, \hat{T}) \in S] &\leq \Pr[\hat{j} < j] + \sum_h \Pr[H_x = h] \mathbf{1}\{\hat{j}_h \geq j\} \Pr[(Y_h, T_h) \in S_h] \\ &\leq \eta + \delta + e^{\varepsilon} \sum_h \Pr[H_x = h] \psi(h). \end{aligned}$$

Since P_{len} is $(\varepsilon_{\text{len}}, \delta_{\text{len}})$-JOT-DP, applying it to the randomized test that accepts h with probability $\psi(h)$ gives

$$\sum_h \Pr[H_x = h] \psi(h) \leq e^{\varepsilon_{\text{len}}} \sum_h \Pr[H_{x'} = h] \psi(h) + \delta_{\text{len}}.$$

Substituting this bound yields

$$\Pr[(\hat{Y}, \hat{T}) \in S] \leq \eta + \delta + e^{\varepsilon} \delta_{\text{len}} + e^{\varepsilon'} \sum_h \Pr[H_{x'} = h] \psi(h).$$

Finally, by the definition of ψ,

$$\begin{aligned} \sum_h \Pr[H_{x'} = h] \psi(h) &\leq \sum_h \Pr[H_{x'} = h] \Pr[(Y'_h, T'_h) \in S_h] \\ &= \Pr[(\hat{Y}', \hat{T}') \in S]. \end{aligned}$$

Hence

$$\begin{aligned} \Pr[(\hat{Y}, \hat{T}) \in S] &\leq e^{\varepsilon'} \Pr[(\hat{Y}', \hat{T}') \in S] + \eta + \delta + e^{\varepsilon} \delta_{\text{len}} \\ &= e^{\varepsilon'} \Pr[(\hat{Y}', \hat{T}') \in S] + \eta + \delta + (\delta' - \delta - \eta) \\ &= e^{\varepsilon'} \Pr[(\hat{Y}', \hat{T}') \in S] + \delta'. \end{aligned}$$

Therefore $\hat{P}$ is (ε', δ')-JOT-DP with respect to d_{SymUser} on $\mathcal{X}_\gamma$. □

5 Preprocessed Datasets

The impossibility result of Theorem 4.1 is partly driven by the input representation. Throughout this paper, we model inputs as arbitrary sequences of user-labeled records. It is therefore natural to ask what changes if the dataset is instead stored in a representation

that is more aligned with user-level computation. In this section, we consider a different kind of relaxation, in which the dataset is first transformed by a one-time preprocessing step that is not observable to the adversary.

A canonical preprocessing step is to group the records by user identifier, so that all records contributed by the same user appear contiguously in memory. Once the input has this structure, tasks such as counting users or truncating each user's contribution become much more tractable from the perspective of timing privacy. The reason is that the computation no longer needs to recover a user's contribution from an arbitrary interleaving of records. Instead, each user's contribution appears as a single contiguous block, so once the program reaches the beginning of that block, it can process only the portion it needs and then move directly to the next user.

A particularly convenient representation is one in which the first record of each user's block stores auxiliary metadata giving the size of that block. Under this stronger assumption, after reading the first B records of a user's block, the program can immediately jump to the next block. Thus many natural user-level preprocessing tasks become timing-stable on the already-preprocessed representation. For example, user counting becomes $O(1)$-timing-stable (Lemma A.13, Appendix), and truncating each user to their first B records becomes $O(B)$-timing-stable (Lemma A.14, Appendix).

Even without explicit block-length metadata, a grouped representation[5] is already enough to obtain efficient timing-stable algorithms when there is a large public upper bound N on the number of records contributed by any one user. This remains meaningful even when N is enormous. The key point is that if the current user's block starts at position i, then the next user's block must begin at some position in $\{i+1, \ldots, i+N\}$. Since the data are sorted by user ID, the end of the current user's block can be found by binary search. Thus advancing from one user to the next requires only $O(\log N)$ additional work.

Lemma 5.1. Fix public bounds $B, N \in \mathbb{N}$ with $B \leq N$, and let $\mathcal{X}_N^{\text{sort}} \subseteq \mathcal{X}$ denote the set of datasets that are sorted by user identifier and in which each user contributes at most N records. Then there exists a deterministic RAM program $P : \mathcal{X} \times \mathcal{E} \to \mathcal{X} \times \mathcal{E}$ such that, for every $x \in \mathcal{X}_N^{\text{sort}}$, the output of P is the dataset obtained from x by keeping only the first B records of each user. Moreover, P is t-timing-stable with respect to d_{SymUser}, where $t = O(B + \log N)$.

Proof. Let P be the deterministic RAM program that processes the input block by block. Suppose the current block starts at index i, and let u be the user identifier in record i. Since the dataset is grouped by user identifier and each user contributes at most N records, the first record not belonging to user u must lie in the interval

$$\{i+1, \ldots, \min\{i+N, |x|\}\}.$$

[5]For example, sorting the dataset by user IDs.

Thus, the program can use binary search to find the smallest index j in this interval whose user identifier is strictly larger than u, or set $j = |x| + 1$ if no such position exists. The records $x_i, \ldots, x_{j-1}$ are exactly the block contributed by user u.

The program copies the first $\min\{B, j - i\}$ records of this block to the output, discards the remainder, and then continues from position j. Repeating this procedure until the end of the input yields exactly the dataset obtained by keeping only the first B records of each user.

We now analyze timing stability. Each iteration performs one binary search over an interval of length at most N, which takes $O(\log N)$ time, together with copying at most B records and $O(1)$ additional work. Hence the total work per user block is $O(B + \log N)$.

If two datasets $x, x' \in \mathcal{X}_N^{\text{sort}}$ are adjacent under d_{SymUser}, then one is obtained from the other by adding or removing one contiguous user block. Therefore the executions of P on x and x' differ by one additional, or one fewer, block-processing iteration, and all other iterations are identical. It follows that

$$|T_P(x, \texttt{env}) - T_P(x', \texttt{env}')| = O(B + \log N)$$

for all input-compatible environments $\texttt{env}, \texttt{env}'$. Thus P is t-timing-stable for $t = O(B + \log N)$, which is a constant. □

Theorem 5.2. Fix public bounds $B, N \in \mathbb{N}$ with $B \leq N$, and let $\mathcal{X}_N^{\text{sort}} \subseteq \mathcal{X}$ denote the set of datasets that are sorted by user identifier and in which each user contributes at most N records. Let $\mathcal{X}_B \subseteq \mathcal{X}$ denote the set of datasets in which each user contributes at most B records. Let $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ be an (ε, δ)-DP RAM program with respect to d_{SymUser} on $\mathcal{X}_B$, and suppose that P has worst-case runtime $T(n)$ on inputs of length at most n. Fix privacy parameters $\varepsilon' > \varepsilon$ and $\delta' > \delta$, and a failure parameter $\beta \in (0, 1)$. Then there exists a RAM program $\widehat{P} : \mathcal{X}_N^{\text{sort}} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ that is (ε', δ')-JOT-DP with respect to d_{SymUser} on $\mathcal{X}_N^{\text{sort}}$, and for every $x \in \mathcal{X}_N^{\text{sort}}$,

$$d_{\text{TV}}\Big(\texttt{out}(\widehat{P}(x, \texttt{env})), \texttt{out}(P(\texttt{TruncUser}_B(x), \texttt{env}))\Big) \leq \beta$$

where $\texttt{TruncUser}_B(x)$ denotes the dataset obtained from x by keeping only the first B records of each user.

Furthermore, there exists a constant $c \geq 1$ such that for every $x \in \mathcal{X}_N^{\text{sort}}$ with $m = |\texttt{users}(x)|$, the program $\widehat{P}$ runs in time $O(m + T(cm))$ with high probability.

Proof. See appendix. □

6 Discussion

Our results should be read as part of a broader lesson from side-channel security. Cryptographic systems have repeatedly shown that protecting the mathematical interface of

a primitive is not enough when implementations expose timing, cache behavior, power, or other execution-dependent signals [Koc96, BB05, BT11, AFP13, AP16, Ber05, OST06, WPH+22, KHF+20]. Differential privacy faces an analogous implementation problem. The formal guarantee is usually stated for the released output, but an interactive DP system also exposes latency. Existing attacks on DP implementations show that this latency can be correlated with filtering, floating-point behavior, noise generation, and other data-dependent computation [HPN11, Mir12, JMRO22, RBM25]. Our results show that the user-level setting introduces a distinct challenge, since adding or removing one user may add or remove an unbounded number of records from the computation.

This distinction matters for defenses. In cryptography, constant-time programming is often the right abstraction because secrets are usually fixed-size values, and the goal is to make execution independent of those values [CSB+17, BCS13]. In user-level DP, the protected unit is an entire user's contribution, and the size of that contribution may be unknown and unbounded. When these quantities are unbounded, there is no finite worst-case runtime to pad to. Even when a loose public bound exists, padding to that bound may be too expensive to be useful.

A second implication is that contribution bounding is not a free preprocessing step once runtime is part of the privacy guarantee. In output-only user-level DP, contribution bounding is often used to reduce sensitivity, with a bias-variance tradeoff determined by the cap [AKMV19, EMM+20]. However, enforcing the cap may itself require scanning an arbitrary sequence of records to find and discard excess contributions. The runtime of this scan can reveal the presence of a heavy contributor even when the final output is differentially private. Thus, systems should not rely on the claim that heavy contributors are rare unless that assumption is part of the privacy model. A deployment that wants user-level timing privacy should either enforce public contribution bounds before the observable computation begins (Section 3) or restrict the domain in a way that rules out dominant users (Section 4).

Our results also identify data representation as a privacy-relevant systems choice. When records are stored as an arbitrary sequence, the program may need to recover the set of records contributed by each user before it can enforce contribution bounds or run a user-level DP algorithm. This recovery step can itself leak through runtime. If records are grouped by user, sorted by user identifier, or annotated with block lengths before the adversary can observe the computation, then enforcing per-user contribution bounds and running the resulting bounded-contribution DP computation become much easier to protect.

Finally, we prove our theorems in the RAM model because it gives a clean way to study these algorithmic issues without committing to a particular processor, runtime system, or distributed execution engine. The same approach can be applied in more realistic computational models, including models that account for caches, branch prediction, memory allocation, scheduling, network effects, or distributed execution. The practical lesson is that user-level timing privacy depends not only on the DP mechanism, but also on how

user contributions are represented, bounded, and processed by the implementation.

References

[AFP13] Nadhem J Al Fardan and Kenneth G Paterson. Lucky thirteen: Breaking the tls and dtls record protocols. In *2013 IEEE symposium on security and privacy*, pages 526–540. IEEE, 2013.

[AKMV19] Kareem Amin, Alex Kulesza, Andres Munoz, and Sergei Vassilvtiskii. Bounding user contributions: A bias-variance trade-off in differential privacy. In *International Conference on Machine Learning*, pages 263–271. PMLR, 2019.

[AP16] Martin R Albrecht and Kenneth G Paterson. Lucky microseconds: A timing attack on amazon's s2n implementation of tls. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I 35*, pages 622–643. Springer, 2016.

[BB05] David Brumley and Dan Boneh. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005.

[BBDS13] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 87–96, 2013.

[BCS13] Daniel J Bernstein, Tung Chou, and Peter Schwabe. Mcbits: fast constant-time code-based cryptography. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 250–272. Springer, 2013.

[BCSZ18] Christian Borgs, Jennifer Chayes, Adam Smith, and Ilias Zadik. Private algorithms can always be extended. *arXiv preprint arXiv:1810.12518*, 2018.

[BDDNT23] Yoav Ben Dov, Liron David, Moni Naor, and Elad Tzalik. Resistance to timing attacks for sampling and privacy preserving schemes. In *4th Symposium on Foundations of Responsible Computing (FORC 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.

[Ber05] Daniel J Bernstein. Cache-timing attacks on aes. 2005.

[BT11] Billy Bob Brumley and Nicola Tuveri. Remote timing attacks are still practical. In *European Symposium on Research in Computer Security*, pages 355–371. Springer, 2011.

[CSB+17] Sunjay Cauligi, Gary Soeller, Fraser Brown, Brian Johannesmeyer, Yunlu Huang, Ranjit Jhala, and Deian Stefan. Fact: A flexible, constant-time programming language. In *2017 IEEE Cybersecurity Development (SecDev)*, pages 69–76. IEEE, 2017.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pages 265–284. Springer, 2006.

[EMM+20] Alessandro Epasto, Mohammad Mahdian, Jieming Mao, Vahab Mirrokni, and Lijie Ren. Smoothly bounding user contributions in differential privacy. *Advances in Neural Information Processing Systems*, 33:13999–14010, 2020.

[GRS12] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. *SIAM Journal on Computing*, 41(6):1673–1693, 2012.

[HPN11] Andreas Haeberlen, Benjamin C Pierce, and Arjun Narayan. Differential privacy under fire. In *20th USENIX Security Symposium (USENIX Security 11)*, 2011.

[JMRO22] Jiankai Jin, Eleanor McMurtry, Benjamin IP Rubinstein, and Olga Ohrimenko. Are we there yet? timing and floating-point attacks on differential privacy systems. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 473–488. IEEE, 2022.

[KHF+20] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, et al. Spectre attacks: Exploiting speculative execution. *Communications of the ACM*, 63(7):93–101, 2020.

[Koc96] Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16*, pages 104–113. Springer, 1996.

[Mir12] Ilya Mironov. On significance of the least significant bits for differential privacy. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 650–661, 2012.

[OST06] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: the case of aes. In *Topics in Cryptology–CT-RSA 2006: The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2005. Proceedings*, pages 1–20. Springer, 2006.

[RBM25] Zachary Ratliff, Nicolas Berrios, and James Mickens. Timing attacks on differential privacy are practical. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pages 3694–3707, 2025.

[RV24] Zachary Ratliff and Salil Vadhan. A framework for differential privacy against timing attacks. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 3615–3629, 2024.

[RV25] Zachary Ratliff and Salil Vadhan. Securing unbounded differential privacy against timing attacks. In *Theory of Cryptography Conference*, pages 387–414. Springer, 2025.

[WPH+22] Yingchen Wang, Riccardo Paccagnella, Elizabeth Tang He, Hovav Shacham, Christopher W Fletcher, and David Kohlbrenner. Hertzbleed: Turning power {Side-Channel} attacks into remote timing attacks on x86. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 679–697, 2022.

A Appendix

Lemma A.1 (DP and OC-Timing Privacy Imply JOT-DP [RV24])**.** Let$P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ be a RAM program. If P is $(\varepsilon_1, \delta_1)$-differentially private in its output and $(\varepsilon_2, \delta_2)$-OC-timing private, then P is $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-JOT-DP.

Lemma A.2 (Constant-Time DP Programs are JOT-DP [RV24])**.** If a program P is (ε, δ)-DP in its output and has runtime $T_P(x, \texttt{env}) = c$ for all x and all compatible $\texttt{env}$, then P is (ε, δ)-JOT-DP.

Theorem A.3 (Composition of JOT-DP RAM Programs [RV24])**.** Let $P_1 : \mathcal{X} \times \mathcal{E} \to \mathcal{Y}_1 \times \mathcal{E}$ be an $(\varepsilon_1, \delta_1)$-JOT-DP RAM program and let $P_2 : \mathcal{X} \times \mathcal{Y}_1 \times \mathcal{E} \to \mathcal{Y}_2 \times \mathcal{E}$ be an $(\varepsilon_2, \delta_2)$-JOT-DP RAM program. Then the chained RAM program $P_2 \circ P_1 : \mathcal{X} \times \mathcal{E} \to (\mathcal{Y}_1 \times \mathcal{Y}_2) \times \mathcal{E}$ that executes P_1 on input x, then executes P_2 on the pair (x, y_1) where y_1 is the output of P_1, and releases both outputs, is $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-JOT-DP.

Lemma A.4 (Censored Discrete Laplace is DP [GRS12])**.** Let $f : \mathcal{X} \to \mathbb{Z}$ have global sensitivity Δ, and fix integers $\ell \leq u$. Define

$$\mu(x) \;:=\; \min\{u, \max\{\ell, f(x)\}\}.$$

Then the mechanism outputting a sample from

$$\texttt{CensoredDiscreteLaplace}(\mu(x), \Delta/\varepsilon, \ell, u)$$

is ε-DP.

Lemma A.5 (Censored Discrete Laplace Sampling in Constant Time)**.** Fix integers $\ell \le u$. There exists a RAM program that, given $\mu \in \mathbb{Z}$ and $s > 0$, computes $\tilde{\mu} = \min\{u, \max\{\ell, \mu\}\}$ and then outputs a sample from

$$\texttt{CensoredDiscreteLaplace}(\tilde{\mu}, s, \ell, u),$$

and halts after $O(u - \ell)$ basic instructions.

Lemma A.6 (JOT-DP Discrete Laplace [RV24])**.** Let $f : \mathcal{X} \to \mathbb{Z}$ be a deterministic function with global sensitivity Δ. Assume f is computable by a RAM program that is t-timing stable. Fix $\varepsilon > \varepsilon' > 0$ and $\delta > 0$. Then there exists a RAM program $P : \mathcal{X} \times \mathcal{E} \to \mathbb{Z} \times \mathcal{E}$ such that, for every $x \in \mathcal{X}$,

$$\texttt{out}(P(x, \texttt{env})) \sim \texttt{DiscreteLaplace}(f(x), \Delta/\varepsilon')$$

and P is (ε, δ)-JOT-DP.

Lemma A.7 (Pure JOT-DP upper bound on input length [RV25])**.** For every $\varepsilon > 0$ and every $\beta \in (0, 1)$, there exists a pure ε-JOT-DP RAM program $P_{\text{len}} : \mathcal{X} \times \mathcal{E} \to \mathbb{N} \times \mathcal{E}$ with respect to the record-level insert-delete distance d_{ID} such that the following holds. For every input $x \in \mathcal{X}$ with $n = |x|$, and every input-compatible execution environment $\texttt{env} \in \mathcal{E}$,

$$\Pr[\texttt{out}(P_{\mathsf{len}}(x, \texttt{env})) \ge n] \ge 1 - \beta$$

Moreover, $P_{\text{len}}(x, \texttt{env})$ runs in time $O(n)$ with high probability.

Lemma A.8. Fix $m \in \mathbb{N}$ and let $\mathcal{X}_B \subseteq \mathcal{X}$ denote the set of datasets in which each user contributes at most B records. If $x, x' \in \mathcal{X}_B$ are d_{SymUser}-adjacent, then

$$d_{\text{SymUser}}\big(\texttt{Trunc}(x, m), \texttt{Trunc}(x', m)\big) \le 2B + 1.$$

Proof. Suppose that x' is obtained from x by inserting all records of a single user u. Let $t \le B$ be the number of records contributed by u. Write

$$y := \texttt{Trunc}(x, m) \qquad \text{and} \qquad y' := \texttt{Trunc}(x', m).$$

If none of the records of u appear among the first m positions of x', then $y' = y$ and there is nothing to prove.

Otherwise, let $t' \in \{1, \ldots, t\}$ be the number of records of u that appear among the first m positions of x'. Then y' is obtained from y by inserting these t' records of u into the prefix and dropping the last t' records of y. Let A be the set of users that appear among

those last t' records of y. Since those t' records contribute at most one new user each, we have

$$|A| \le t' \le B.$$

We now transform y into y' using user insertions and deletions. First, for each user $v \in A$, delete all records of v from y. This costs $|A|$ user deletions. Next, insert the user u with exactly the records of u that appear in y'. This costs one user insertion. Finally, for each user $v \in A$, insert the records of v as they appear in y'. This costs another $|A|$ user insertions.

The resulting dataset is exactly y'. Therefore,

$$d_{\mathrm{SymUser}}(y, y') \le 2|A| + 1 \le 2B + 1.$$

□

The important point for our purposes is that, unlike in the record-level setting, prefix truncation is not 1-stable under user-level adjacency.

Lemma A.9. Prefix truncation $\texttt{Trunc}(\cdot, m)$ is not 1-stable under user-level adjacency.

Proof. Let

$$x = [(a, d_1), (b, d_2)] \quad \text{and} \quad x' = [(u, e_1), (u, e_2), (a, d_1), (b, d_2)].$$

Then $d_{\mathrm{SymUser}}(x, x') = 1$. But

$$\texttt{Trunc}(x, 2) = [(a, d_1), (b, d_2)] \quad \text{and} \quad \texttt{Trunc}(x', 2) = [(u, e_1), (u, e_2)],$$

and these truncated datasets have user-level distance 3, since one must delete users a and b and insert user u. Hence

$$d_{\mathrm{SymUser}}(\texttt{Trunc}(x, 2), \texttt{Trunc}(x', 2)) > d_{\mathrm{SymUser}}(x, x').$$

The same example works for every $m \ge 2$ by appending identical trailing records to both inputs. □

Corollary A.10 (Impossibility for Approximate JOT-DP User Counts)**.** Fix any $k \in \mathbb{N}$ and let $\mathsf{Good}_k \subseteq \mathcal{X}$ be the set of datasets in which every user contributes at most k records. Let $P : \mathcal{X} \times \mathcal{E} \to \mathbb{N} \times \mathcal{E}$ be an (ε, δ)-JOT-DP RAM program (with respect to d_{SymUser} on all of $\mathcal{X}$) that outputs an estimate of $|\texttt{users}(x)|$. For every $\varepsilon \ge 0$, $\delta \in [0, 1)$, and every failure probability β with $0 < \beta < 1 - \delta \cdot (e^{\varepsilon} + 1)$, there exist constants $c, N \in \mathbb{N}$ such that for all $n > N$ there exists a dataset $x \in \mathsf{Good}_k$ with $|\texttt{users}(x)| = n$ for which

$$\Pr\big[\, |\texttt{out}(P(x, \texttt{env})) - n| > n - c \,\big] \;>\; \beta.$$

Proof. Apply Theorem 4.1 with $(\mathcal{Z}, d) = (\mathbb{N}, |\cdot|)$ and $q(x) = |\texttt{users}(x)|$. Let $B \subseteq \mathbb{N}$ be the bounded set produced there. Since $\mathrm{diam}(B) < \infty$ and $B \subseteq \mathbb{N}$, the set B is finite. Thus, define

$$c \;:=\; 1 + \max(B) \qquad \text{and} \qquad N \;:=\; c$$

For any $n > N$, let x be any dataset consisting of n distinct users, each contributing exactly one record (so $x \in \texttt{Good}_1 \subseteq \texttt{Good}_k$ and $|\texttt{users}(x)| = n$). By the construction of B in the proof of Theorem 4.1, we have $\Pr[\texttt{out}(P(x, \texttt{env})) \in B] > \beta$. On the event $\{\texttt{out}(P(x, \texttt{env})) \in B\}$ we have $\texttt{out}(P(x, \texttt{env})) \le \max(B) = c - 1$, and hence $|\texttt{out}(P(x, \texttt{env})) - n| > n - c$. Therefore,

$$\Pr\big[\, |\texttt{out}(P(x, \texttt{env})) - n| > n - c \,\big] \;\ge\; \Pr[\texttt{out}(P(x, \texttt{env})) \in B] \;>\; \beta.$$

□

Lemma A.11. Let x be γ-balanced for $\gamma \in (0, 1)$ and let $n = |x|$. Fix a user $b \in \texttt{users}(x)$ with $t = c_b(x)$, and let $M = \max(\{\, c_u(x) : u \in \texttt{users}(x) \setminus \{b\} \,\} \cup \{0\})$. Let t' be an integer satisfying

$$\max\left\{0, \frac{M}{\gamma} - (n - t)\right\} \le t' \le \frac{\gamma}{1 - \gamma}(n - t) \tag{5}$$

and let $s \notin \texttt{users}(x)$ be a user with t' contributions. Then the dataset x' that is obtained by deleting the user b from x and inserting the user s is both γ-balanced and satisfies $d_{\mathrm{SymUser}}(x, x') \le 2$.

Proof. 1. We have $|x'| = n - t + t'$.

- For the new user s, $c_s(x') = t'$, so γ-balance requires $t' \le \gamma|x'| = \gamma(n - t + t')$, equivalently $t' \le \frac{\gamma}{1-\gamma}(n - t)$.
- For any original user $u \in \texttt{users}(x) \setminus \{b\}$ we have $c_u(x') = c_u(x) \le M$, so it suffices that $M \le \gamma|x'| = \gamma(n - t + t')$, equivalently $t' \ge \frac{M}{\gamma} - (n - t)$.

Together with $t' \ge 0$, this is exactly (5).

2. Assume (5) holds. Define y by inserting the new user s with t' records into x. Then $|y| = n + t'$. For any original user $u \in \texttt{users}(x)$,

$$c_u(y) = c_u(x) \le \gamma n \le \gamma(n + t') = \gamma|y|,$$

so all original users satisfy γ-balance in y. For the new user s, we need $t' \le \gamma|y| = \gamma(n + t')$, equivalently $t' \le \frac{\gamma}{1-\gamma}n$. But from (5), we already have $t' \le \frac{\gamma}{1-\gamma}(n - t) \le \frac{\gamma}{1-\gamma}n$. Hence, y is γ-balanced. Deleting user b from y gives x', and x' is γ-balanced by part (1). Therefore, $x \to y \to x'$ is a γ-balanced path of length at most 2.

□

Definition A.12 (User-block representation)**.** A dataset $x \in \mathcal{X}$ is in *user-block form* if all records contributed by the same user appear contiguously in memory. We say that x is in *length-annotated user-block form* if, in addition, the first record of each user's block stores the length of that block.

Lemma A.13 (User counting is $O(1)$-timing-stable on length-annotated user blocks)**.** Fix the user-level insert-delete distance d_{SymUser}. Let P_{count} be the RAM program that, on an input in length-annotated user-block form, counts the number of user blocks by reading the first record of each block and jumping directly to the next block using the stored block length. Then P_{count} is $O(1)$-timing-stable with respect to d_{SymUser}.

Proof. Let x, x' be adjacent datasets, so x' is obtained from x by inserting or deleting all records of a single user. Because the input is in length-annotated user-block form, the program processes the dataset block-by-block, performing a fixed amount of work on each block: it reads the first record, increments a counter, reads the block length, and jumps to the first record of the next block.

Passing from x to x' changes the block decomposition by at most one user block. Thus the execution of P_{count} on x and x' differs by only one iteration of this fixed-cost loop. Therefore the runtime changes by at most a constant, so P_{count} is $O(1)$-timing-stable. □

Lemma A.14 (Per-user truncation is $O(B)$-timing-stable on length-annotated user blocks)**.** Fix $B \in \mathbb{N}$. Let $P_{\mathsf{trunc},B}$ be the RAM program that, on an input in length-annotated user-block form, outputs the dataset obtained by keeping only the first B records from each user block. Then $P_{\mathsf{trunc},B}$ is $O(B)$-timing-stable with respect to d_{SymUser}.

Proof. Let x, x' be adjacent datasets. Then x' differs from x by the insertion or deletion of one user's entire block. On every user block other than this one, the program performs exactly the same computation on x and x'.

Now consider the unique block on which the two inputs differ. Because the input is in length-annotated user-block form, the program reads the block header, copies at most the first B records of the block to the output, and then jumps directly to the next block using the stored block length. Thus the amount of work contributed by this block is $O(B)$, independent of the total number of records in the block.

Therefore the total runtime difference between executions on x and x' is at most $O(B)$, and so $P_{\mathsf{trunc},B}$ is $O(B)$-timing-stable. □

Theorem 5.2. Fix public bounds $B, N \in \mathbb{N}$ with $B \le N$, and let $\mathcal{X}_N^{\mathrm{sort}} \subseteq \mathcal{X}$ denote the set of datasets that are sorted by user identifier and in which each user contributes at most N records. Let $\mathcal{X}_B \subseteq \mathcal{X}$ denote the set of datasets in which each user contributes at most B records. Let $P : \mathcal{X} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ be an (ε, δ)-DP RAM program with respect to d_{SymUser} on $\mathcal{X}_B$, and suppose that P has worst-case runtime $T(n)$ on inputs of length at most n. Fix privacy parameters $\varepsilon' > \varepsilon$ and $\delta' > \delta$, and a failure parameter $\beta \in (0, 1)$. Then there exists

a RAM program $\widehat{P} : \mathcal{X}_N^{\mathrm{sort}} \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ that is (ε', δ')-JOT-DP with respect to d_{SymUser} on $\mathcal{X}_N^{\mathrm{sort}}$, and for every $x \in \mathcal{X}_N^{\mathrm{sort}}$,

$$d_{\mathrm{TV}}\Big(\mathtt{out}(\widehat{P}(x, \mathtt{env})), \mathtt{out}(P(\mathtt{TruncUser}_B(x), \mathtt{env}))\Big) \le \beta$$

where $\mathtt{TruncUser}_B(x)$ denotes the dataset obtained from x by keeping only the first B records of each user.

Furthermore, there exists a constant $c \ge 1$ such that for every $x \in \mathcal{X}_N^{\mathrm{sort}}$ with $m = |\mathtt{users}(x)|$, the program $\widehat{P}$ runs in time $O(m + T(cm))$ with high probability.

Proof. Fix privacy parameters $\varepsilon_{\mathrm{pre}}, \varepsilon_{\mathrm{comp}} > 0$ and $\delta_{\mathrm{pre}}, \delta_{\mathrm{comp}} > 0$ such that

$$\varepsilon_{\mathrm{pre}} + \varepsilon_{\mathrm{comp}} = \varepsilon', \qquad \delta_{\mathrm{pre}} + \delta_{\mathrm{comp}} = \delta',$$

with $\varepsilon_{\mathrm{comp}} > \varepsilon$ and $\delta_{\mathrm{comp}} > \delta$.

Let P^0_{trunc} be the deterministic RAM program that, on input $x \in \mathcal{X}_N^{\mathrm{sort}}$, computes $f(x) := \mathtt{TruncUser}_B(x)$, writes $f(x)$ back into the input buffer, updates the input length, and outputs the constant 0. By Lemma 5.1, the runtime of P^0_{trunc} changes by at most $t = O(B + \log N)$ on adjacent inputs. Since the output is constant, P^0_{trunc} is t-OC-timing-stable with respect to d_{SymUser} on $\mathcal{X}_N^{\mathrm{sort}}$.

Let $\widetilde{P}_{\mathrm{trunc}}$ be the program obtained by running P^0_{trunc} and then adding an independent $(t \to (\varepsilon_{\mathrm{pre}}, \delta_{\mathrm{pre}}))$ timing-private delay. By Theorem 2.6, $\widetilde{P}_{\mathrm{trunc}}$ is $(\varepsilon_{\mathrm{pre}}, \delta_{\mathrm{pre}})$-OC-timing private. Moreover, since it always outputs the constant 0, it is $(0, 0)$-DP in its output. Thus, by Lemma A.1, $\widetilde{P}_{\mathrm{trunc}}$ is $(\varepsilon_{\mathrm{pre}}, \delta_{\mathrm{pre}})$-JOT-DP.

The map f sends $\mathcal{X}_N^{\mathrm{sort}}$ into $\mathcal{X}_B$ and preserves user-level adjacency. Indeed, if x' is obtained from x by inserting or deleting one user's block, then every other user's ordered list of records is unchanged, and hence the first B records retained for every other user are unchanged. Thus $f(x')$ is obtained from $f(x)$ by inserting or deleting only the retained prefix of that same user's block, so $d_{\mathrm{SymUser}}(f(x), f(x')) \le 1$.

Next, apply Theorem 3.1 to P with privacy parameters $\varepsilon_{\mathrm{comp}}, \delta_{\mathrm{comp}}$ and failure probability β. This gives a RAM program $P_{\mathrm{comp}} : \mathcal{X}_B \times \mathcal{E} \to \mathcal{Y} \times \mathcal{E}$ that is $(\varepsilon_{\mathrm{comp}}, \delta_{\mathrm{comp}})$-JOT-DP with respect to d_{SymUser} on $\mathcal{X}_B$, and such that for every $z \in \mathcal{X}_B$,

$$d_{\mathrm{TV}}(\mathtt{out}(P_{\mathrm{comp}}(z, \mathtt{env})), \mathtt{out}(P(z, \mathtt{env}))) \le \beta.$$

Define $\widehat{P}$ as follows. On input x, first run $\widetilde{P}_{\mathrm{trunc}}$, which rewrites the input buffer to contain $f(x)$, and then run P_{comp} on the dataset currently stored in the input buffer.

We claim that $\widehat{P}$ is (ε', δ')-JOT-DP. The invocation of $\widetilde{P}_{\mathrm{trunc}}$ is $(\varepsilon_{\mathrm{pre}}, \delta_{\mathrm{pre}})$-JOT-DP. After this invocation, neighboring inputs $x, x' \in \mathcal{X}_N^{\mathrm{sort}}$ have been rewritten as neighboring datasets $f(x), f(x') \in \mathcal{X}_B$. Therefore the subsequent invocation of P_{comp} is $(\varepsilon_{\mathrm{comp}}, \delta_{\mathrm{comp}})$-JOT-DP. By the standard JOT-DP composition theorem (Theorem A.3), the transcript revealing the final output together with the two runtimes separately is

$$(\varepsilon_{\mathrm{pre}} + \varepsilon_{\mathrm{comp}}, \delta_{\mathrm{pre}} + \delta_{\mathrm{comp}}) = (\varepsilon', \delta')$$

-DP. The actual observation of $\widehat{P}$ reveals only the final output and the sum of the two runtimes, which is a deterministic post-processing of that transcript. Hence $\widehat{P}$ is (ε', δ')-JOT-DP.

For utility, after the first invocation the input buffer contains $f(x) = \texttt{TruncUser}_B(x)$. Thus, for the resulting environment $\texttt{env}_B$,

$$\texttt{out}(\widehat{P}(x, \texttt{env})) = \texttt{out}(P_{\text{comp}}(f(x), \texttt{env}_B)).$$

Since $f(x) \in \mathcal{X}_B$, the guarantee for P_{comp} gives

$$d_{\text{TV}}\Big(\texttt{out}(\widehat{P}(x, \texttt{env})), \texttt{out}(P(f(x), \texttt{env}_B))\Big) \leq \beta.$$

Using output-purity of P to replace $\texttt{env}_B$ by $\texttt{env}$, and substituting $f(x) = \texttt{TruncUser}_B(x)$, we obtain

$$d_{\text{TV}}\Big(\texttt{out}(\widehat{P}(x, \texttt{env})), \texttt{out}(P(\texttt{TruncUser}_B(x), \texttt{env}))\Big) \leq \beta.$$

Finally, let $m = |\texttt{users}(x)|$. By Lemma 5.1, the preprocessing computation takes time

$$O(m(B + \log N)) = O(m),$$

since B and N are fixed constants. The added delay contributes only $O(1)$ additional time with high probability, and the preprocessed dataset has length at most $Bm = O(m)$. Applying the runtime guarantee from Theorem 3.1 to P_{comp} therefore gives total runtime

$$O(m + T(cm))$$

with high probability for some constant $c \geq 1$. □