Continual Observation under User-level Differential Privacy

Wei Dong, Qiyao Luo, and Ke Yi
Hong Kong University of Science and Technology, Hong Kong SAR, China
{wdongac,qluoak,yike}@cse.ust.hk

***Abstract*—In the foundational work of Dwork et al. [15] on continual observation under differential privacy (DP), two privacy models have been proposed: event-level DP and user-level DP. The latter provides a much stronger notion of privacy, as it allows a user to contribute an arbitrary number of items. Under event-level DP, their mechanisms match the optimal utility bounds in the static setting up to polylogarithmic factors for all union-preserving functions. Unfortunately, in contrast to this strong result for event-level DP, their user-level DP mechanisms have weak utility guarantees and many restrictions on the data. In this paper, we take an instance-specific approach, designing continual observation mechanisms for a number of fundamental functions under user-level DP. Our mechanisms do not need any *a priori* restrictions on the data, while providing utility guarantees that degrade gracefully as the hardness of the data increases. For the count and sum function, our mechanisms are down-neighborhood optimal, matching the static setting up to polylogarithmic factors. For other functions, they do not match the static case, but we prove that this is inevitable, which is the first separation result for continual observation under differential privacy.**

***Index Terms*—Differential privacy, Continual observation, User-level differential privacy**

I. Introduction

Data is seldom static. When private data evolves over time, there is a need to continually release sanitized query results about the data while preserving the privacy of the users who contribute to the data. This is precisely the problem of *continual observation under differential privacy*, introduced in the pioneering work of Dwork et al. [15]. Here, time is divided into discrete steps, and data is modeled as a (possibly infinite) stream of items arriving over time, one per time step. More formally, let $D = (x_1, x_2, \dots)$ be the stream, where item x_i arrives at time i. If no data arrives at time i, x_i is set to a dummy item $\perp$. Let $D_t = (x_1, \dots, x_t)$ be the data set received up until time t. With a slight abuse of notation, we also use D_t to denote the multiset of items contained in D_t, not including the dummy items. For a given function $F(\cdot)$, a continual observation mechanism $\mathcal{M}$ must output an $\tilde{F}(D_t)$ for every $t \in \mathbb{N}$ with the following three properties:

- Online: For each $t \in \mathbb{N}$, it must output $\tilde{F}(D_t)$ before x_{t+1} arrives.
- Differential privacy: The entire output is ε-indistinguishable over any two neighboring streams $D \sim D'$, i.e., for any $y_1, y_2, \dots$,
$$\begin{aligned}&\Pr[\tilde{F}(D_1) = y_1, \tilde{F}(D_2) = y_2, \dots]\\ \le& e^{\varepsilon} \cdot \Pr[\tilde{F}(D_1') = y_1, \tilde{F}(D_2') = y_2, \dots],\end{aligned}$$
where ε is the privacy parameter.
- Utility: For any $t \in \mathbb{N}$, the error $\|\tilde{F}(D_t) - F(D_t)\|$ is small with at least constant probability[1], for some appropriate norm $\|\cdot\|$.

To make this definition complete, we need to specify the neighboring relationship $D \sim D'$. Dwork et al. [15] proposed two natural definitions: In *event-level differential privacy*, or simply *event-DP*, two streams D and D' are neighbors if one can be obtained from the other by removing one item (i.e., setting it to $\perp$). In *user-level differential privacy*, or *user-DP*, each item x_i is associated with (the id of) a user u_i, and D and D' are neighbors if one can be obtained from the other by removing all or any subset of items associated with one user. Clearly, event-DP is a special case of user-DP by setting $u_i = i$ for all i, while the latter provides stronger privacy protection in situations where a user may contribute multiple items in the stream. The canonical example is a click stream that consists of websites visited by users. In this case, user-DP guarantees that, from the released results, the adversary would not be able to decide if any particular user is present in the stream with enough confidence (controlled by ε), even if the adversary is computationally unbounded and can observe the data contributed by all other users. Below, we use $\sim_E$ and $\sim_U$ to differentiate these two neighboring relationships, and use $\sim$ when this distinction is irrelevant. Furthermore, we use $d_E(D, D')$ and $d_U(D, D')$ to denote the shortest distance between D and D' where each step is $\sim_E$ and $\sim_U$ respectively.

A. Previous work on event-DP

The major result in [15] on event-DP is a black-box reduction to the static problem with only a $\operatorname{poly}\log(T)$-factor increase in the error for any *union-preserving* function F, where T is an upper bound on the stream length. Chan et al. [8] extend this result to infinite streams, with the $\operatorname{poly}\log(T)$ factor replaced by $\operatorname{poly}\log(t)$, where t is the current length of the stream. A union-preserving function F is one such that $F(D_1 \uplus D_2) = F(D_1) + F(D_2)$ for any D_1, D_2, where $\uplus$ denotes multiset union. Most natural functions (e.g., count, sum, histogram) are union-preserving or can be transformed to such a function (e.g., distinct count, k-selection). The exact $\operatorname{poly}\log(t)$ factor depends on the function F and the black-box static mechanism being used. In particular, for the count function $F_{\text{cnt}}(D) = \sum_i \mathbf{I}(x_i \ne \perp)$ with the Laplace mechanism as the black box which has error $O(\frac{1}{\varepsilon})$, the

[1] Most of the literature uses this definition of utility, which is also called a *one-shot* guarantee. This can be extended to an *all-time* guarantee, i.e., a bound on $\sup_{t\in\mathbb{N}} \|\tilde{F}(D_t) - F(D_t)\|$, with just a logarithmic-factor increase in the error bound, even over an infinite stream; see the remark after Lemma II.11.

continual observation mechanism of [15, 8] achieves an error of $O(\frac{1}{\varepsilon}\log^{1.5} t)$ for $F_{\text{cnt}}(D_t)$.

After the ground-laying work [15, 8] that established the theoretical equivalence of event-DP and the static problem (up to polylogarithmic factors), a slew of works have made further practical improvements. [16] has optimized the sparse-stream case where most items are $\perp$. [9] has proposed techniques to smooth out the query results, which may reduce the error during periods when the stream is stable. [11, 23, 22] use the matrix mechanism to improve the constant factors in the error of [15, 8], while relaxing the privacy definition to (ε, δ)-DP. Since we use [8] as a black box, such techniques can also be used in our mechanisms. Others have made improvements for particular functions, such as sum [30, 38, 4], histogram [6, 7, 32], linear functions [10], and graph statistics [21, 33]. Event-DP under the local model of differential privacy has been studied in [25, 38]. Finally, w-event-DP [27, 31, 40, 39, 37] has been proposed as an extension, where we protect the privacy for any w consecutive events.

B. Previous work on user-DP

In contrast to the strong result on event-DP, the result on user-DP from [15] is relatively weak and restrictive. It requires the function F to have a bounded global sensitivity GS and be (α_1, α_2)-*unvarying*, i.e., over any stream D and any α_1 number of time instances, there must exist two consecutive time instances between which the function value changes by no more than α_2. For such an F, the user-DP mechanism of [15] has error[2] $\tilde{O}(\text{GS} \cdot \alpha_1 + \alpha_2)$. For most natural functions, the restriction of a bounded GS also imposes an upper bound on the maximum user contribution: Taking the count function as an example, this means that each user is only allowed to contribute at most GS events. In fact, under such a restriction, simply running their event-DP mechanism with a privacy parameter of ε/GS using the group privacy property of DP (note that $d_E(D, D') \le \text{GS} \cdot d_U(D, D')$) can already achieve an error of $\tilde{O}(\text{GS})$. On the other hand, α_1, α_2 are unbounded for an infinite stream; even for a finite stream of length T, we need to set $\alpha_1 \cdot \alpha_2 \ge T$. Thus, for the count function, the user-DP mechanism of [15] is actually no better than their event-DP mechanism applied with group privacy. Unfortunately, [15] did not give a concrete F on which their user-DP mechanism is better.

In seeing the difficulty of user-DP, subsequent works introduced more assumptions on the data. For example, [19] assumes that data comes in batches and each user can only contribute one item in any batch; they also assume that the data in consecutive batches are correlated. [3] studies user-DP in the local model, while making similar assumptions. While these mechanisms satisfy privacy, they do not have theoretical guarantees on the utility.

C. Recent work on static user-DP

Earlier efforts on user-DP failed to yield satisfactory results, mostly because they aimed at optimizing the worst-case error, i.e., a bound on $\max_D \|\tilde{F}(D_t) - F(D_t)\|$, which is inevitably large unless strict restrictions are imposed on the user contributions. Recent work on user-DP in the static setting [2, 12, 20, 24] has taken an instance-specific approach, namely, the error depends on certain "hardness" parameter of D. Again taking the count function as an example, the mechanism in [24, 13] achieves an error of $\tilde{O}(\kappa(D))$ on any (static) data set D, where $\kappa(D)$ is the maximum number of items contributed by any user in D. Such an instance-specific approach has several advantages:

1) It offers better utilities on most real data sets in which the hardness parameter is small, which is more meaningful than worst-case bounds for problems where the worst case is atypical.
2) It works without making any assumptions on the data, such as restricting user contributions. In some sense, these restrictions become soft: if they are broken, the utility degrades gracefully (privacy is always guaranteed).
3) Very often, *down-neighborhood optimality* (see Section II-C for the formal definition) can be achieved, which is a natural relaxation of instance optimality and much stronger than worst-case optimality. For the count function, an error of $\tilde{O}(\kappa(D))$ has been shown to be down-neighborhood optimal. Intuitively, it means that any DP mechanism has to incur an error of $\Omega(\kappa(D))$ on either D or the data set after removing the most contributing user from D.

D. Our results

Inspired by the recent user-DP work in the static setting, in this paper we also take an instance-specific approach towards the dynamic problem. For a number of common functions F, we have designed dynamic user-DP mechanisms that inherit the first two advantages above from the static setting. Down-neighborhood optimality is also maintained for some functions like count and sum, but not for other functions like distinct count, histogram, and k-selection. Although there are down-neighborhood optimal mechanisms for these functions (in fact, for all monotonic functions [20]) in the static setting, we show that they do not exist in the dynamic setting. This is the first separation result between the static and dynamic case under differential privacy.

As representation, here we state our results on the count and distinct count function; results on other functions can be found in Section IV. Our continual observation mechanism for the count function F_{cnt} achieves an error of $O\Big(\frac{\kappa(D_t)}{\varepsilon} \cdot \log^{1.5} t \cdot \log^{1+\theta}(\kappa(D_t))\Big)$ for $F_{\text{cnt}}(D_t)$ where $\theta > 0$ can be any small constant. This matches the static error bound of [24, 13] up to polylogarithmic factors, hence down-neighborhood optimal. Furthermore, it incorporates event-DP as a special case: When $\kappa(D_t) = 1$, the bound precisely degenerates[3] into that of [15, 8]. We would like to stress that this incorporation is automatic, as our mechanism does not need to know the value of $\kappa(D_t)$;

[2] The $\tilde{O}$ notation suppresses dependencies on ε and polylogarithmic factors.

[3] We define $\log(x) = \max\{1, \log_2(x)\}$.

in fact, a main technical component in our mechanism is to estimate $\kappa(D_t)$ for all t in a differentially private manner.

For distinct count, our mechanism also has an error of $\tilde{O}(\kappa(D_t))$. While this is still a nontrivial instance-specific bound, it is not down-neighborhood optimal. For this function, a down-neighborhood optimal mechanism should achieve an error of $\tilde{O}(\mathrm{DS}^{(\rho)}(D_t))$ for some $\rho = \tilde{O}(1)$, where $\mathrm{DS}^{(\rho)}(D_t)$ is the *downward sensitivity* of D_t at distance ρ (see Definition II.8). For the distinct count function, this is the size of the largest set A of distinct items such that (1) there is one user who contributes at least one copy of each item of A; and (2) all copies of the items in A are contributed by at most $\rho + 1$ users. Note that $\mathrm{DS}^{(\rho)}(D_t) \leq \kappa(D_t)$ by definition. While a static mechanism can achieve error $\tilde{O}(\mathrm{DS}^{(\rho)}(D_t))$ for $\rho = \tilde{O}(1)$ [20], we show that this is impossible under the dynamic setting.

In addition to the nice theoretical guarantees, our mechanisms are also simple and practical. To demonstrate their empirical performance, we have conducted an extensive set of experiments using both synthetic and real data. The experimental results show that our algorithm largely outperforms the baselines even with strong prior knowledge on D, i.e., knowing $\kappa(D_T)$.

II. Preliminaries

A. Notation

Let $\mathcal{U}$ be the (possibly infinite) set of all users. Let $[n] := \{1, 2, \ldots, n\}$. The stream is an infinite sequence of item-user pairs $D := \Big((x_1, u_1), (x_2, u_2), \ldots\Big)$, where each x_i is taken from $\{\perp\} \cup [R]$ for some domain size R. Each item x_i is contributed by user $u_i \in \mathcal{U}$. Let $D_t := \Big((x_1, u_1), \ldots, (x_t, u_t)\Big)$. For any $t \in \mathbb{Z}^+$, let $\kappa(D_t)$ be the maximum number of items contributed by any user in D_t, i.e.,

$$\kappa(D_t) := \max_{u \in \mathcal{U}} \Big|\big\{(x_i, u_i) \in D_t : u_i = u, x_i \neq \perp \big\}\Big|.$$

Note that we do not assume any *a priori* upper bound on $\kappa(D_t)$.

For any $x \in \mathbb{N}$, let $\mathrm{Bin}_j(x) \in \{0, 1\}$ be the $(j+1)$-th least significant bit in the binary representation of x.

B. Properties of differential privacy

Lemma II.1 (Post Processing [18]). *If $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ satisfies ε-DP and $\mathcal{M}' : \mathcal{Y} \to \mathcal{Z}$ is any randomized mechanism, then $\mathcal{M}'(\mathcal{M}(D))$ satisfies ε-DP.*

Lemma II.2 (Basic Composition [18]). *If $\mathcal{M}_1 : \mathcal{X} \to \mathcal{Y}$ satisfies ε_1-DP and $\mathcal{M}_2 : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ satisfies ε_2-DP, then $\mathcal{M}_2(D, \mathcal{M}_1(D))$ satisfies $(\varepsilon_1 + \varepsilon_2)$-DP.*

Lemma II.3 (Parallel Composition [28]). *If $\mathcal{M}_1$, $\mathcal{M}_2$ satisfy ε_1-DP and ε_2-DP, and $\mathcal{X}_1, \mathcal{X}_2 \subseteq \mathcal{X}$ are two disjoint input domains, then $\big(\mathcal{M}_1(D \cap \mathcal{X}_1), \mathcal{M}_2(D \cap \mathcal{X}_1)\big)$ satisfies $\max(\varepsilon_1, \varepsilon_2)$-DP.*

Lemma II.4 (Group Privacy [18]). *If $\mathcal{M}$ is an ε_0-DP mechanism, then for any two datasets D, D' with $d(D, D') = k$, $\mathcal{M}$ satisfies $(k\varepsilon_0)$-DP.*

For any function $F : \mathcal{X} \to \mathbb{R}^d$, the *global sensitivity* is $\mathrm{GS} = \max_{D \sim D'} \big\|Q(D) - Q(D')\big\|_1$. A basic DP mechanism is the Laplace mechanism:

Lemma II.5 (Laplace Mechanism). *Given $F : \mathcal{X} \to \mathbb{R}^d$ with global sensitivity* GS*, the mechanism $\mathcal{M}(D) = F(D) + \gamma$ preserves ε-DP, where γ is a d-dimensional vector where each entry is independently drawn from the Laplace distribution* $\mathrm{Lap}(\mathrm{GS}_Q/\varepsilon)$.

The Laplace distribution enjoys a good concentration property.

Lemma II.6 (Concentration Bound of Laplace Distributions [8]). *Suppose $\gamma_1, \gamma_2, \ldots, \gamma_k$ are independent random variable, where each $\gamma_i \sim \mathrm{Lap}(b_i)$, then for any $\beta > 0$,*

$$\Pr\left[\left|\sum_i \gamma_i\right| \leq \sqrt{8 \sum_i b_i^2} \cdot \log(\frac{2}{\beta})\right] \leq \beta.$$

C. Down-neighborhood optimality

The Laplace mechanism is worst-case optimal. However, worst-case optimality is meaningless under user-DP, unless there is a small pre-defined limit on the user contributions. In the absence of such restrictions, *down-neighborhood optimality* [12, 20, 24] has been adopted as a more meaningful notion of optimality under user-DP.

Definition II.7 (Down-neighborhood Optimality [20]). *Let $\mathbb{M}$ be the class of all ε-DP mechanisms. Given any function F, the ρ-down neighborhood lower bound on dataset D is defined as*

$$\mathcal{L}(D, \rho) := \inf_{\mathcal{M}' \in \mathbb{M}} \max_{D' : D' \subseteq D, d_U(D, D') \leq \rho} \inf\left\{\xi : \Pr\Big[\big\|\mathcal{M}'(D') - F(D')\big\| \leq \xi\Big] \geq \frac{2}{3}\right\}.$$

A mechanism $\mathcal{M}$ is (ρ, c)-down neighborhood optimal if for any instance D,

$$\Pr\Big[\big\|\mathcal{M}(D) - F(D)\big\| \leq c \cdot \mathcal{L}(D, \rho)\Big] \geq \frac{2}{3}.$$

In plain language, a down-neighborhood optimal mechanism is one that, for any D, achieves an error as small as (up to a factor of c) the optimal mechanism tailor-made for D and its down neighborhood within distance ρ. Clearly, small ρ, c correspond to stronger optimality, and ideally we would want $\rho, c = \tilde{O}(1)$. In the static setting, this is achievable for all monotonic functions [20], i.e., any F such that $F(D') \leq F(D)$ for any $D' \subseteq D$.

Since comparing with $\mathcal{L}(D, \rho)$ directly is difficult, the *downward sensitivity* has been shown to be a good proxy for proving down-neighborhood optimality.

Definition II.8 (Downward Sensitivity). *Given any function F, for any dataset D, its downward sensitivity is*

$$\mathrm{DS}(D) = \max_{D' : D \sim D', D' \subseteq D} \|F(D) - F(D')\|,$$

Algorithm 1: SVT.

```
Input: $\eta$, $\varepsilon$, $Q_1(D), Q_2(D), \ldots$
$\tilde{\eta} \leftarrow \eta + \mathrm{Lap}(2/\varepsilon)$;
for $i \leftarrow 1, 2, \ldots$ do
    $\widetilde{Q}_i(D) \leftarrow Q_i(D) + \mathrm{Lap}(4/\varepsilon)$;
    if $\widetilde{Q}_i(D) > \tilde{\eta}$ then
        Break;
    end
end
return $i$;
```

and its downward sensitivity at distance ρ *is*

$$\mathrm{DS}^{(\rho)}(D) = \max_{D':D'\subseteq D, d(D,D')\leq\rho} \mathrm{DS}(D').$$

In particular, $\mathrm{DS}^{(0)}(D) = \mathrm{DS}(D)$.

Theorem II.9 ([20])**.** *Given any function* F*,* $\varepsilon \leq \ln 2$*, for any* D *and any* ρ*,* $\frac{1}{2}\cdot \mathrm{DS}^{(\rho-1)}(D) \leq \mathcal{L}(D,\rho) \leq \rho \cdot \mathrm{DS}^{(\rho-1)}(D)$.

Thus, a mechanism $\mathcal{M}$ is (ρ, c)-down neighborhood optimal if its error is at most $2c\cdot \mathrm{DS}^{(\rho-1)}(D)$ with probability at least $2/3$ on any D. On the other hand, if there exists a D on which its error is more than $c\rho \cdot \mathrm{DS}^{(\rho-1)}(D)$, then it cannot be (ρ, c)-down neighborhood optimal.

Overall, roughly speaking, down-neighborhood optimality requires a mechanism to achieve an error level corresponding to the maximum user contribution. It has been accepted as a standard optimality notion to quantify the utility for DP mechanisms under user-DP [12, 20, 24].

D. The sparse vector technique

The sparse vector technique (SVT) [17] has as input a (possibly infinite) sequence of queries, $Q_1, Q_2, \ldots$, where each query has global sensitivity 1, and a threshold η. It aims at finding the first i such that $Q_i(D) \geq \eta$ while satisfying ε-DP. The detailed algorithm is given in Algorithm 1. Due to the privacy noises, it cannot return such an i precisely, but something close, as formalized in the following lemma:

Lemma II.10 ([13])**.** *If there exists a* k *such that* $Q_k(D) \geq \eta + \frac{6}{\varepsilon}\log(2/\beta)$*, then with probability at least* $1-\beta$*,* SVT *returns an* $i \leq k$*.*

E. Continual counting under event-DP

We briefly review the continual observation mechanism [15, 8] under event-DP for the count function F_{cnt}, which will also be used in our user-DP mechanism. Consider the finite-stream case first. A binary decomposition over all the T time steps is built with $\log T$ levels of intervals, and the Laplace mechanism is invoked on each interval to return a noisy count. Then any $\tilde{F}(D_t)$ can be obtained by adding up at most $\log T$ such noisy counts, one from each level. To set the privacy budgets of these intervals, it suffices to allocate $\varepsilon/\log T$ to each interval by basic composition (across levels) and parallel composition (within a level). Thus, the noise from each interval is $O(\log T)$, which adds up to $O(\log^{1.5} T)$ by Lemma II.6. To extend to an infinite stream, the idea is to divide the stream into disjoint periods $[2^\ell, 2^{\ell+1}-1]$ for $\ell = 0, 1, 2, \ldots$, and invoke the finite-stream mechanism over each period. Now, each D_t spans $\log t$ periods. For all but the last period, it only queries the top-level interval; for the last period, it queries $\log t$ intervals. The detailed algorithm is shown in Algorithm 2 and Figure 1.

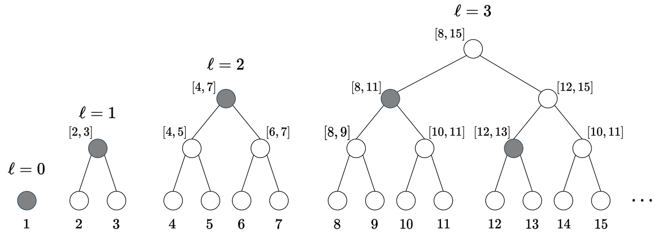


Fig. 1: An illustration of F_{cnt}EventDP. The counting result for time interval $[1, 13]$ can be obtained by adding the counters corresponding to black nodes.

Algorithm 2: F_{cnt}EventDP.

```
Input: $D = \Big((x_1, u_1), (x_2, u_2), \ldots\Big)$, $\varepsilon$
$\widetilde{F}_{\mathrm{cnt}} \leftarrow 0$;
for $\ell \leftarrow 0, 1, \ldots$ do
    $\alpha_j, \tilde{\alpha}_j \leftarrow 0,\ j = 0, 1, \ldots, \ell$;
    $\varepsilon' \leftarrow \varepsilon/(\ell+1)$;
    for $k \leftarrow 1, 2, \ldots, 2^\ell$ do
        $t \leftarrow 2^\ell - 1 + k$;
        Let $i := \min\{j : \mathrm{Bin}_j(k) \neq 0\}$;
        $\alpha_i = \sum_{j<i} \alpha_j + \mathbf{I}(x_t > 0)$;
        $\tilde{\alpha}_i \leftarrow \alpha_i + \mathrm{Lap}(\frac{1}{\varepsilon'})$;
        for $j \leftarrow 0, 1, \ldots, i-1$ do  $\alpha_j \leftarrow 0$ ;
        Output
            $\widetilde{F}_{\mathrm{cnt}}(D_t) \leftarrow \widetilde{F}_{\mathrm{cnt}} + \sum_{j\in[\ell]} \tilde{\alpha}_j \cdot \mathbf{I}(\mathrm{Bin}_j(k) = 1)$;
    end
    $\widetilde{F}_{\mathrm{cnt}} \leftarrow \widetilde{F}_{\mathrm{cnt}} + \tilde{\alpha}_\ell$;
end
```

Lemma II.11 ([8])**.** *Given any* $\varepsilon > 0$*, for any* D*, any* $t \in \mathbb{Z}^+$*, and any* $\beta > 0$*, with probability at least* $1-\beta$*,* F_{cnt}EventDP *returns an* $\widetilde{F}_{\mathrm{cnt}}(D_t)$ *such that*

$$\begin{aligned}&\left|\widetilde{F}_{\mathrm{cnt}}(D_t) - F_{\mathrm{cnt}}(D_t)\right| \leq \\ &\mathrm{Noise}_E(t, \varepsilon, \beta) := \frac{4}{\varepsilon}\cdot \lceil \log(t) \rceil^{1.5} \cdot \log(1/\beta).\end{aligned}$$

Remark. Lemma II.11 provides a one-shot guarantee that holds for any single $\widetilde{F}_{\mathrm{cnt}}(D_t)$. To extend it to an all-time guarantee for all t simultaneously, we can replace the failure probability β in the lemma with $\frac{\beta}{t^2}$. Then, a union bound over all t yields a total failure probability at most $\sum_{t=1}^{\infty} \frac{\beta}{t^2} = O(\beta)$, while the $\log(1/\beta)$ factor in the error bound becomes $O(\log(t/\beta))$.

Notation	Meaning
D	Input stream
D_t	Prefix stream at time t
$D\|_r$	Sub-stream restricted to item r
(x_i, u_i)	Item-user pairs
$[R]$	Item domain
$\mathcal{U}$	User domain
$\kappa(D_t)$	Maximum user contribution of D_t
$F(D_t)$	Query result at time t
$F(D_t)$	Query result at time t
$\mathrm{Bin}_j(x)$	$(j+1)$-th least significant binary bit of x
$d_E(D, D')$	Distance between D, D' under tuple-DP
$d_U(D, D')$	Distance between D, D' under user-DP
GS	Global sensitivity
$\mathrm{DS}(D)$	Downward sensitivity of D
$\mathrm{DS}^{(\rho)}(D)$	Downward sensitivity at distance ρ of D
ε	Privacy budget
β	High probablity parameter
$F_{\text{cnt}}\text{EventDP}$	Counting under tuple-DP
Noise_E	Error of counting under event-DP
F_{cnt}	Count query
F_{sum}	Sum query
F_{ht}	Histogram query
$F_{\text{k-sel}}$	k-selection query
F_0	Distinct count query
$F_{\max\text{-}f}$	Maximum frequency query

TABLE I: Notation used in the paper.

```
Algorithm 3: $\text{Truncate}F_{\text{cnt}}\text{UserDP}$.
Input: $D = \Big((x_1, u_1), (x_2, u_2), \dots\Big)$, $\varepsilon$, $\tau$
$\widehat{D} \leftarrow \text{Truncate}(D, \tau)$;
Run $F_{\text{cnt}}\text{EventDP}(\widehat{D}, \varepsilon/\tau)$;
```

III. Continual Counting under User-DP

In the static setting, the count function F_{cnt} can be returned with error $\tilde{O}(\kappa(D_t))$ [24, 13]. This is $(1, \tilde{O}(1))$-down neighborhood optimal since for F_{cnt}, we have $\mathrm{DS}(D_t) = \kappa(D_t)$. This is also our target in the dynamic setting. However, under user-DP, we cannot use parallel composition over disjoint time intervals, since a user may own items in multiple time steps. This renders the event-DP mechanism inapplicable.

A. Warm-up: two simple solutions

As warm-up, we first look at two simple solutions for the continual count queries under user-DP.

Basic/advanced composition. While parallel composition cannot be used under user-DP, we can still use basic composition. However, this would lose all utility. Just consider the finite stream case. We can allocate a privacy budget of ε/T to each D_t, and invoke the static user-DP mechanism [24, 13]. This results in an error of $\tilde{O}(T \cdot \kappa(D_t))$. Since $F(D_t) \le t \le T$ for all t, this is even worse than the trivial solution that always returns 0. If one is willing to relax the privacy guarantee to (ε, δ)-DP, then advanced composition [18] can be used to reduce the error to $\tilde{O}(\sqrt{T} \cdot \kappa(D_t))$. This is better than the trivial solution, but the error is still very large. In Section VII, we will empirically compare with advanced composition.

Truncating user contributions. Another idea is to truncate the user contributions: For some $\tau \in \mathbb{N}$, we only retain the first τ items from each user, while setting the remaining items to $\perp$. Let $\text{Truncate}(D, \tau)$ be the truncated stream. For any $D \sim_U D'$, it is clear that $d_E(\text{Truncate}(D, \tau), \text{Truncate}(D', \tau)) \le \tau$. Thus, by the group privacy property, we can run $F_{\text{cnt}}\text{EventDP}$ over $\text{Truncate}(D, \tau)$ with privacy budget ε/τ; see Algorithm 2. Let $\text{Count}(D_t, \tau)$ denote the number of users contributing more than τ items in D_t. Then the error of $\text{Truncate}F_{\text{cnt}}\text{UserDP}$ can be bounded as follows.

Lemma III.1. *Given $\varepsilon > 0$ and $\tau \in \mathbb{Z}$, for any D, and any $t \in \mathbb{Z}^+$, with probability at least $1 - \beta$,* $\text{Truncate}F_{\text{cnt}}\text{UserDP}$ *returns an $\widetilde{F}_{\text{cnt}}(D_t)$ such that*

$$\widetilde{F}_{\text{cnt}}(D_t) - F_{\text{cnt}}(D_t) \in \Big[- \text{Noise}_E\big(t, \frac{\varepsilon}{\tau}, \beta\big) \\ - \max(\kappa(D_t) - \tau, 0) \cdot \text{Count}(D_t, \tau), \text{Noise}_E\big(t, \frac{\varepsilon}{\tau}, \beta\big)\Big]$$

In the error bound above, the $\text{Noise}_E\left(t, \frac{\varepsilon}{\tau}, \beta\right)$ term is the noise from $F_{\text{cnt}}\text{EventDP}$, while the negative bias $-\max(\kappa(D_t) - \tau, 0) \cdot \text{Count}(D_t, \tau)$ is due to the truncation, since each of the $\text{Count}(D_t, \tau)$ users may have up to $\max(\kappa(D_t) - \tau, 0)$ items truncated. We see that if $\tau = \kappa(D_t)$, then the bias becomes 0 and $\text{Noise}_E\left(t, \frac{\varepsilon}{\tau}, \beta\right) = \tilde{O}(\kappa(D_t))$, thus achieving the desired error bound. The challenge, therefore, is to track $\kappa(D_t)$ continually and in a differentially private manner. Below, we will show how this can be done with only an $\tilde{O}(1)$ loss in the utility.

B. Continually bounding user contributions

The first observation is that $\kappa(D_t)$ monotonically increases over time, and we just need a constant-factor approximation. Thus, we can start with $\tau = 2$. Then, whenever some user has contributed more than τ items, we double τ. To detect the time instance when this happens, we can run an SVT over the queries $\text{Count}(D_t, \tau)$ for each successive time step t with threshold $\eta = 0$. Note that each such query has global sensitivity 1. This SVT will not immediately respond when $\kappa(D_t)$ exceeds τ, but will not be too late, either. By Lemma II.10, when $\text{Count}(D_t, \tau) \ge \frac{6}{\varepsilon} \log(2/\beta)$, it must have responded already. This means that we will miss at most $\tilde{O}(1)$ users, each of which may have up to $\kappa(D_t) - \tau$ items truncated. This will still lead to a bias, but it is at most $\tilde{O}(\kappa(D_t))$, which is within our target error bound.

Another danger we need to guard against is that the SVT may also stop too early. To see this, just imagine a long period in which all items are dummy, hence τ should not increase. However, due to the Laplace noises in the SVT, it will eventually stop with probability 1. To prevent this from happening, the idea is to use a discounted version of the $\text{Count}(D_t, \tau)$ queries:

$$\begin{aligned}&\overline{\text{Count}}(D_t, \tau, \beta) \\ =&\text{Count}(D_t, \tau) - \frac{6}{\varepsilon} \log(2/\beta) - \frac{8}{\varepsilon} \log(t+1). \qquad (1)\end{aligned}$$

Algorithm 4: EstimatingUserContribution.

```
Input: $D = \Big((x_1, u_1), (x_2, u_2), \dots\Big)$, $\varepsilon$, $\beta$, $\theta$
$i \leftarrow 1$;
$\varepsilon_1 \leftarrow \varepsilon\theta/2^{1+\theta}$;        // Initialize the parameters for the first SVT for $\tau = 2$
$\beta_1 \leftarrow \beta/2^2$;
$\tau_1 \leftarrow 2$;
$\tilde{\eta} \leftarrow \text{Lap}(2/\varepsilon_1)$;        // Start the first SVT for $\tau = 2$
for $t \leftarrow 1, 2, \dots$ do
    $\widetilde{\text{Count}}(D_t, \tau_i, \beta_i) \leftarrow \overline{\text{Count}}(D_t, \tau_i, \beta_i) + \text{Lap}(4/\varepsilon_i)$;
    while $\widetilde{\text{Count}}(D_t, \tau_i, \beta_i) > \tilde{\eta}$ do
        $i \leftarrow i + 1$;
        $\varepsilon_i \leftarrow \varepsilon\theta/(i+1)^{1+\theta}$;        // Update the parameters for the new SVT
        $\beta_i \leftarrow \beta/(i+1)^2$;
        $\tau_i \leftarrow 2^i$;
        $\tilde{\eta} \leftarrow \text{Lap}(2/\varepsilon_i)$;        // Start a new SVT
        $\widetilde{\text{Count}}(D_t, \tau_i, \beta_i) \leftarrow \overline{\text{Count}}(D_t, \tau_i, \beta_i) + \text{Lap}(4/\varepsilon_i)$;
    end
    $\tilde{\tau}_t \leftarrow \tau_i$;
    Output $\tilde{\tau}_t$;
end
```

Intuitively, the first discount term $-\frac{6}{\varepsilon}\log(2/\beta)$ cancels the Laplace noise that the SVT adds to the threshold $\eta = 0$, while the second discount term $-\frac{8}{\varepsilon}\log(t+1)$ negates the Laplace noises the SVT adds to each query. These terms are chosen in a way such that, when no users have contributions above τ, the SVT will stop with probability at most only β, even over an infinite stream. On the other hand, these discount terms may increase the number of users with $> \tau$ contributions that are missed by the SVT, but the this number is still bounded by $\tilde{O}(1)$.

Lastly, whenever an SVT instance stops, we double τ and start a new SVT instance. Under user-DP, these instances are no longer disjoint (in the user space), so we must use basic composition to split the privacy budget. To support an infinite stream, the idea is allocate privacy using a telescoping series, e.g., $\varepsilon_i = \varepsilon\theta/(i+1)^{1+\theta}$ for the i-th SVT instance. This ensures that the total privacy consumption is never more than $\sum_{i=1}^{\infty} \varepsilon_i = \varepsilon$.

The detailed algorithm for continually bounding user contributions is shown in Algorithm 4. Let $\tilde{\tau}_t$ denote the value of τ returned by the algorithm at time t. We formally prove its utility guarantee in the following lemma.

Lemma III.2. *Given any $\varepsilon > 0$, $\beta > 0$, $\theta > 0$, and any D, with probability at least $1 - \beta$, we have $\tilde{\tau}_t \leq 2 \cdot \kappa(D_t)$ and*

$$\text{Count}(D_t, \tau_i) = O\Big(\frac{1}{\varepsilon\theta} \cdot \log^{1+\theta}(\kappa(D_t)) \cdot \log(t/\beta)\Big),$$

for all $t \in \mathbb{Z}^+$ simultaneously.

Finally, we combine EstimatingUserContribution and TruncateF_{cnt}UserDP. We run them in parallel, each with $\frac{\varepsilon}{2}$ privacy. Whenever a new $\tilde{\tau}_t$ is returned by the former, we start a new instance of the latter, also using a telescoping series to allocate the privacy budget. Our final mechanism F_{cnt}UserDP is shown in Algorithm 5.

Algorithm 5: F_{cnt}UserDP.

```
Input: $D = \Big((x_1, u_1), (x_2, u_2), \dots\Big)$, $\varepsilon$, $\beta$, $\theta$
Start EstimatingUserContribution($D, \frac{\varepsilon}{2}, \frac{\beta}{2}, \theta$);
$i \leftarrow 1$;
$\tilde{\tau}_0 \leftarrow 1$;
for $t \leftarrow 1, 2, \dots$ do
    Get $\tilde{\tau}_t$ from EstimatingUserContribution;
    if $\tilde{\tau}_t \neq \tilde{\tau}_{t-1}$ then
        $\varepsilon_i \leftarrow \varepsilon\theta/(2(i+1)^{1+\theta})$;
        Start a new Truncate$F_{\text{cnt}}$UserDP($D, \varepsilon_i, \tilde{\tau}_t$);
        $i \leftarrow i + 1$;
    end
end
```

Theorem III.3. *Given $\varepsilon > 0$, $\beta > 0$, and $\theta > 0$, for any D, and any $t \in \mathbb{Z}^+$, with probability at least $1 - \beta$, F_{cnt}UserDP returns an $\widetilde{F}_{\text{cnt}}(D_t)$ such that*

$$\begin{aligned}\widetilde{F}_{\text{cnt}}(D_t) - F_{\text{cnt}}(D_t) \in \Big[&- \text{Noise}_U(\tilde{\tau}_t, t, \varepsilon, \beta, \theta) \\ &- \text{Bias}_U(D, t, \varepsilon, \beta, \theta), \text{Noise}_U(\tilde{\tau}_t, t, \varepsilon, \beta, \theta)\Big],\end{aligned}$$

where

$$\begin{aligned}&\mathrm{Noise}_U(\tilde{\tau}_t, t, \varepsilon, \beta, \theta)\\ =&\mathrm{Noise}_E\left(t, \frac{\varepsilon\theta}{2\tilde{\tau}_t \cdot \big(\log(\tilde{\tau}_t)+2\big)^{1+\theta}}, \frac{\beta}{2}\right)\\ =&O\Big(\frac{\kappa(D_t)}{\varepsilon\theta}\cdot\log^{1.5}(t)\cdot\log^{1+\theta}(\kappa(D_t))\cdot\log(1/\beta)\Big),\end{aligned}$$

$$\mathrm{Bias}_U(D,t,\varepsilon,\beta,\theta) = O\Big(\frac{\kappa(D_t)}{\varepsilon\theta}\cdot\log^{1+\theta}(\kappa(D_t))\cdot\log(t/\beta)\Big).$$

The proof directly follows from Lemma III.1 and III.2.

Remark. For conceptual simplicity, in Algorithm 5 we start each instance of TruncateF_{cnt}UserDP$(D, \varepsilon_i, \tilde{\tau}_t)$ from scratch. In the actual implementation, there is no need to rewind the stream (we are not allowed to change the released query results in the past anyway). Recall that in the event-DP counting algorithm, each interval has one counter, and the results are computed through these counters. Under user-DP, we need more effort to track the elements truncated in each interval. Whenever τ doubles, we first incrementally update the counters associated with the intervals in F_{cnt}EventDP with the user contributions previously truncated (up to the new τ), regenerate the Laplace noises with the new ε_i and τ, and then continue the execution of F_{cnt}EventDP. As argued in [8], only $O(\log t)$ intervals need to be maintained at time t. We also need to maintain a counter for each user recording the number of items she has contributed so far, and if it is larger than τ, the truncated items so that they can be added back when τ increases later. It can be shown that F_{cnt}UserDP can be implemented in $\tilde{O}(1)$ time amortized per item; we omit the detailed runtime analysis as it is not the focus.

IV. Other Functions under User-DP

Based on our continual counting mechanism, we can solve many other problems. Recall that each item x_i is taken from $\{\perp\} \cup [R]$ for some domain size R. For the count function, the domain is irrelevant, but the functions considered in this section may adopt a potentially large R. For simplicity, we assume that R is a power of 2.

For any $r \in [R]$, let $D|_r$ be the sub-stream restricted to item r with the contributing relationships kept, i.e., i-th item still belongs to u_i if it equals to r. More precisely, $D|_r := \Big((x_1|_r, u_1|_r), (x_2|_r, u_2|_r), \dots\Big)$, where

$$(x_t|_r, u_t|_r) = \begin{cases}(x_t, u_t), & \text{if } x_t = r;\\ (\perp,\perp), & \text{otherwise.}\end{cases}$$

Similarly, for any $[l, r]$, we define $D|_{[l,r]}$ as the sub-stream restricted to items in $[l, r]$.

A. Sum

We first consider the sum function $F_{\mathrm{sum}}(D_t) = \sum_{(x_i,u_i)\in D_t} x_i$. In the static setting, an error of $\tilde{O}(\varphi(D_t))$ can be achieved [24, 13], where

$$\varphi(D_t) := \max_{u\in\mathcal{U}} \sum_{(x_i,u_i)\in D_t, u_i=u} x_i.$$

This is $(1, \tilde{O}(1))$-down neighborhood optimal since $\mathrm{DS}(D_t) = \varphi(D_t)$.

To achieve such an error in the dynamic setting, we can reduce the sum problem to the count problem and then apply F_{cnt}userDP. We conceptually divide each time step into R sub-steps. For an item x_i arriving in this time step, we fill in the first x_i sub-steps with $\underline{1}$ while the remaining with $\perp$. Denote the new stream as $\overline{D}$. It is obvious that, $D \sim_U D'$ iff $\overline{D} \sim_U \overline{D}'$, and $\varphi(D_t) = \kappa(\overline{D}_{tR})$ for any t. Then privacy is preserved, and the error will be $O\Big(\frac{\varphi(D_t)}{\varepsilon\theta}\cdot\log^{1.5}(tR)\cdot\log^{1+\theta}(\varphi(D_t))\cdot\log(1/\beta)\Big)$, as desired. In the actual implementation, the R sub-steps need not be simulated one by one; they can be easily processed in a batch in $\tilde{O}(1)$ time.

B. Histogram

The histogram function, also known as the *frequency estimation* problem, simply returns the count of each item, i.e., $F_{\mathrm{ht}}(D_t) = (F_{\mathrm{cnt}}(D|_1), \dots, F_{\mathrm{cnt}}(D|_R))$. The function value is an R-dimensional vector, and the most commonly used utility metric is the maximum error $\|\tilde{F}_{\mathrm{ht}}(D_t) - F_{\mathrm{ht}}(D_t)\|_\infty$.

Under event-DP, the problem can be solved by simply running F_{cnt}EventDP over each $D|_r$ for $r \in [R]$ with parallel composition. This yields an estimate $\widetilde{F}_{\mathrm{cnt}}(D_t|_r)$ with a $(1-\beta)$-probability error bound of $\mathrm{Noise}_E(t, \varepsilon, \beta)$ for any one r. To obtain a $(1-\beta)$-probability error bound on $\|\tilde{F}_{\mathrm{ht}}(D_t) - F_{\mathrm{ht}}(D_t)\|_\infty$, one can replace β with β/R and apply a union bound.

Under user-DP, we cannot use parallel composition to decompose the histogram problem into R instances of F_{cnt}, as a user may contribute multiple different items. Nevertheless, we can run one instance of F_{cnt}UserDP except that, in Algorithm 2, we replace F_{cnt}EventDP with the event-DP histogram mechanism above. Privacy is still preserved, since the privacy of our mechanism only relies on that of the SVT and F_{cnt}EventDP. The utility guarantee can also be easily established:

Theorem IV.1. *Given $\varepsilon > 0$, $\beta > 0$, and $\theta > 0$, for any D, and any $t \in \mathbb{Z}^+$, with probability at least $1-\beta$, our user-DP mechanism for the continual histogram problem returns an $\tilde{F}_{\mathrm{ht}}(D_t)$ such that*

$$\|\tilde{F}_{\mathrm{ht}}(D_t) - F_{\mathrm{ht}}(D_t)\|_\infty = O\Big(\frac{\kappa(D_t)}{\varepsilon\theta}\cdot\log^{1+\theta}(\kappa(D_t))\cdot\log(tR/\beta)\Big).$$

Remark 1. For a large R, returning all R frequency estimates is both costly and unnecessary. A related problem, known as the *heavy hitters* problem, aims at only returning the items whose frequencies are above a threshold. A common technique is to reduce the domain $[R]$ to a smaller one via hashing/sketching, and then return the histogram in the reduced domain [14, 5, 34]. Our mechanism can also be used to solve this problem by simply running it on the reduced domain.

Remark 2. While our mechanism still achieves error $\tilde{O}(\kappa(D_t))$, this is not down-neighborhood optimal for the histogram problem. To see this, just consider a D_t in which each

Algorithm 6: $F_{\text{k-sel}}\text{UserDP}$.

```
Input: $D = \Big((x_1, u_1), (x_2, u_2), \dots\Big)$, $k$, $\varepsilon$, $\beta$, $\theta$
$\varepsilon' \leftarrow \varepsilon/(\log(R) + 1)$;
$\beta' \leftarrow \beta/(\log(R) + 1)$;
Start $\log(R) + 1$ $F_{\text{ht}}\text{UserDP}$ with each having parameters $\varepsilon', \beta', \theta$ to get all $\mathcal{F}_{\text{cnt}}(D|_I)$ for
    $I \in \Big\{\big[(j-1) * 2^{i-1} + 1, j * 2^{i-1}\big], i \in [\log(R) + 1], j \in [2^{i-1}]\Big\}$;
for $t \leftarrow 1, 2, \dots$ do
    Get all $\tilde{\tau}_t^i$'s with $i \in [\log(R) + 1]$ and all $\widetilde{F}_{\text{cnt}}(D_t|_I)$'s from $F_{\text{ht}}\text{UserDP}$'s;
    $\Delta = \sqrt{\sum_{i \in [\log(R)+1]} \left(\text{Noise}_U\left(\tilde{\tau}_t^i, t, \frac{\varepsilon}{\log(R)+1}, \frac{\beta}{R}, \theta\right)\right)^2}$;
    $r \leftarrow R$;
    while $r > 0$ do
        Compute $\widetilde{F}_{\text{cnt}}(D_t|_{[r,R]})$ with $\widetilde{F}_{\text{cnt}}(D_t|_I)$'s;
        if $\widetilde{F}_{\text{cnt}}(D_t^{[r,R]}) \geq k + \Delta$ then Break ;
        $r \leftarrow r - 1$;
    end
    Output $\tilde{o}_{t,k} \leftarrow r$;
end
```

user contributes one copy of every item. Then $\kappa(D_t) = R$ but $\text{DS}^{(\rho-1)}(D_t) = 1$ for any $\rho \geq 1$. Thus, as long as $R = \tilde{\omega}(1)$, it is not $(\tilde{O}(1), \tilde{O}(1))$-down neighborhood optimal by Theorem II.9. In Section V, we show that this is inevitable. In fact, even for the maximum frequency problem $F_{\text{max-}f}(D_t) = \|F_{\text{ht}}(D_t)\|_\infty$, which is a special case of the histogram problem, we show that down-neighborhood optimality is unachievable in the dynamic setting. On the other hand, since $F_{\text{max-}f}(D_t)$ is monotonic, $(\tilde{O}(1), \tilde{O}(1))$-down neighborhood optimality can be achieved in the static setting [20].

C. *k-selection*

The k-selection function $F_{\text{k-sel}}(D_t)$ returns the k-th largest item in D_t. Formally, let $o_{t,1}, o_{t,2}, \dots$ be the items in D_t in the descending order. Then $F_{\text{k-sel}}(D_t) := o_{t,k}$. Important special cases include the max function $F_{\max}(D_t) := o_{t,1}$, the minimum, and the median. For simplicity, for this function we assume that all items are distinct; if not, one may use a tie-breaker like the timestamp.

For this problem, the most common utility metric is the *rank error*, i.e., the difference between the rank of the estimate and the required rank k. However, for small k, in particular $F_{\max}$, the trivial solution that always returns R would have a rank error of 0. Thus, a *one-way* rank error is used more often for small k: Letting $\tilde{o}_{t,k}$ be the estimate, its one-way rank error is defined as[4]:

$$\xi(D_t, \tilde{o}_{t,k}) := \begin{cases} \infty, & \text{if } \tilde{o}_{t,k} > o_{t,k}; \\ \max\{k' - k : o_{t,k'} \geq \tilde{o}_{t,k}\}, & \text{otherwise.} \end{cases} \tag{2}$$

This definition thus requires us to find an estimate close (in the rank space) to $o_{t,k}$ but no greater. Symmetrically, for a large k close to $|D_t|$, in particular the minimum function, one would flip the one-way rank error definition around. For values of k close to neither end, like the median, a two-way rank error error could also be used. But obviously, a one-way rank error is only stronger.

The k-selection problem can be reduced to a counting problem. The high-level idea is to find the largest r such that $[r, R]$ contains at least k elements. To find such an r, we build $\log(R)$ histograms with bin sizes $1, 2, 4, \dots, R/2$. This allows us to find the count of $[r, R]$ for every r by decomposing it into $\log(R)$ bins. More precisely, we construct $\log R + 1$ instances of the histogram problem by building a binary decomposition of the domain $[R]$. In the i-th instance, $i \in [\log R + 1]$, all items in $[(j-1) \cdot 2^{i-1} + 1, j \cdot 2^{i-1}]$ are mapped to one item, namely, the i-th histogram instance has a domain size of $R/2^{i-1}$. We run our user-DP histogram mechanism for each instance with privacy budget $\varepsilon/(\log R + 1)$. These histograms allow us to do a binary search in $[R]$ to find the largest r such that $\widetilde{F}_{\text{cnt}}(D_t|_{[r,R]|}) \geq k$. Such an r can then be used as a good estimate of $F_{\text{k-sel}}(D_t)$. However, since the error in the count estimate $\widetilde{F}_{\text{cnt}}(D_t|_{[r,R]|})$ can be either positive or negative, this only yields a two-way rank error. To turn this into a one-way rank error, instead of aiming at a target rank of k, we aim at $k + \Delta$, where

$$\Delta = \sqrt{\sum_{i \in [\log(R)+1]} \left(\text{Noise}_U\left(\tilde{\tau}_t^i, t, \frac{\varepsilon}{\log(R)+1}, \frac{\beta}{R}, \theta\right)\right)^2}. \tag{3}$$

Here, $\tilde{\tau}_t^i$ is the estimate for $\kappa(D_t)$ outputted by the i-th instance of our user-DP histogram mechanism $F_{\text{ht}}\text{UserDP}$.

The detailed algorithm is shown in 6. Its privacy simply follows from basic composition and the post-processing properties of DP. The following theorem analyzes its utility. For

[4] Define $o_{t,i} = 0$ for $i > |D_t|$.

the space limitation, we move its proof to Appendix A-B.

Theorem IV.2. *Given $\varepsilon > 0$, $\beta > 0$, $\theta > 0$, and $k \geq 1$, for any D, and any $t \in \mathbb{Z}^+$, with probability at least $1-\beta$, $F_{\text{k-sel}}\text{UserDP}$ returns an $\tilde{o}_{t,k}$ with one-way rank error*

$$O\bigg(\frac{\kappa(D_t)}{\varepsilon\theta} \cdot \log^{1.5} t \cdot \log^{2.5} R \cdot \log^{1+\theta}(\kappa(D_t)) \cdot \log(1/\beta)\bigg).$$

Remark 1. Our mechanism above works for a publicly given k. For problems like median, where $k = |D_t|/2$ is also a sensitive value, we can first obtain a privatized $\tilde{k}$ by estimating $|D_t|$ first. This is precisely the count problem. Our mechanism for the count function can return such a $\tilde{k}$ with error $\tilde{O}(\kappa(D_t))$. Thus, it does not affect the rank error in Theorem IV.2 asymptotically.

Remark 2. Similar to the histogram problem, a one-way rank error of $\tilde{\kappa}(D_t)$ is not down-neighborhood optimal for the k-selection problem. Just consider the special case $k = 1$, i.e., the maximum function $F_{\max}$ and $D_t = \{(99, u_1), (98, u_2), \dots, (100-\rho, u_\rho), (1, u_1), (1, u_1), \dots\}$. On this instance, we have $\text{DS}^{(\rho-1)} = \rho$, so a ρ-down neighborhood optimal mechanism should achieve a one-way rank error of ρ, namely, return an estimate between 99 and $100-\rho$. However, $\kappa(D_t)$ in this case is the number of items contributed by u_1, which could be much larger. A one-way rank error of $\tilde{O}(1)$ is achievable in the static setting, since $F_{\max}$ is monotonic [20], but we will show in Section V that this is not possible under the dynamic setting.

D. Distinct count

The last problem we consider is the distinct count function $F_0(D_t) = \sum_{x\in[R]} \mathbf{I}\Big(x \in D_t\Big)$. This problem has not been explicitly studied under event-DP, but there is a simple, perhaps folklore, solution: The idea is to reduce it to the count problem, by converting the stream D into $\overline{D} = (\bar{x}_1, \bar{x}_2, \dots)$, where

$$\bar{x}_t = \begin{cases} 1, & \text{if } x_t \neq \perp \text{ and } x_t \notin D_{t-1}; \\ \perp, & \text{otherwise.} \end{cases}$$

It is obvious that $F_0(D_t) = F_{\text{cnt}}(\overline{D}_t)$ for all $t \in \mathbb{Z}^+$. However, acute readers may realize that the neighboring relationship is not precisely preserved. For example, assume item x arrives twice in D, at time t_1 and t_2, respectively. Then in $\overline{D}$, we have $\bar{x}_{t_1} = 1$ and $\bar{x}_{t_2} = \perp$. Deleting the x at time t_1 in D will change both $\bar{x}_{t_1}$ and $\bar{x}_{t_2}$. Nevertheless, $D \sim_E D'$ guarantees that $d_E(\overline{D}, \overline{D}') \leq 2$. Therefore, we just need to use group privacy and invoke the event-DP counting mechanism with privacy budget $\varepsilon/2$. This means that the distinct count problem can still be solved with error $\tilde{O}(1)$ under event-DP.

However, under user-DP, simply running $F_{\text{cnt}}\text{UserDP}$ over $\overline{D}$ does not work, because $D \sim_U D'$ does not yield any upper bound on $d_U(\overline{D}, \overline{D}')$: Just consider the stream $D = ((1, u_1), \dots, (k, u_1), (1, u_2), \dots, (k, u_{k+1}))$. We have $\overline{D} = ((1, u_1), \dots, (k, u_1), (\perp, u_2), \dots, (\perp, u_{k+1}))$. Now, deleting all items of u_1 from D would change $\overline{D}$ into $((\perp, u_1), \dots, (\perp, u_1), (1, u_2), \dots, (k, u_{k+1}))$, i.e., $d_U(\overline{D}, \overline{D}') = k+1$, while k can be made arbitrarily large. The same example also shows that the sensitivity of $\text{Count}(\overline{D}_t, \tau)$ is unbounded, so we cannot run EstimatingUserContribution on $\overline{D}$, either.

To get around these difficulties, we first run EstimatingUserContribution on the original stream D, obtaining a $\tilde{\tau}_t \leq 2 \cdot \kappa(D_t)$ for every t. Then as for the basic counting problem, we do a truncation on D with $\tilde{\tau}$, obtaining $\widehat{D}$. But when we call $F_{\text{cnt}}\text{EventDP}$ in $\text{Truncate}F_{\text{cnt}}\text{UserDP}$, we run it over the corresponding $\overline{D}$ with privacy budget divided by 2. More precisely, in $\text{Truncate}F_{\text{cnt}}\text{UserDP}$, we replace $F_{\text{cnt}}\text{EventDP}(\widehat{D}, \varepsilon/\tau)$ in the second line with $F_{\text{cnt}}\text{EventDP}(\overline{D}, \varepsilon/(2\tau))$, where $\overline{D}$ is constructed from $\widehat{D}$.

For privacy, it suffices to show that given any truncation threshold τ, the resulting $\overline{D}$ and $\overline{D}'$ differ by at most 2τ events. This directly follows the fact that $\widehat{D}$ and $\widehat{D}'$ differ by τ events and the factor-two observation above. The utility guarantee is just a factor-two larger than that for the basic counting problem:

Theorem IV.3. *Given $\varepsilon > 0$, $\beta > 0$, and $\theta > 0$, for any D, and any $t \in \mathbb{Z}^+$, with probability at least $1-\beta$, our user-DP distinct count mechanism returns an $\tilde{F}_0(D_t)$ such that*

$$\begin{aligned}\|\tilde{F}_0(D_t) - F_0(D_t)\| &\leq \text{Noise}_U(2\kappa(D_t), t, \varepsilon, \beta, \theta) \\ &\quad + \text{Bias}_U(D, t, \varepsilon, \beta, \theta) = \tilde{O}(\kappa(D_t)).\end{aligned}$$

V. The Difficulty of Continual Observation under User-DP

Our mechanisms in the previous sections have error $\tilde{O}(\kappa(D_t))$ or $\tilde{O}(\varphi(D_t))$ for various functions. For count and sum, this is $(1, \tilde{O}(1))$-down neighborhood optimal, since $\text{DS}(D_t) = \kappa(D_t)$ or $\varphi(D_t)$ for these two functions, respectively. However, it is not down neighborhood optimal for maximum frequency estimation, k-selection, and distinct count F_0. Although $(\tilde{O}(1), \tilde{O}(1))$-down neighborhood optimal mechanisms exist for these functions in the static setting [20], in this section, we show that in the dynamic setting, no ε-DP mechanisms can achieve $(O(\sqrt{T/\varepsilon}), c(T))$-down neighborhood optimality, where T is the length of the stream and $c(T)$ is an arbitrary function of T. In fact, for histogram and k-selection, we prove this hardness result for their special cases: the maximum frequency function $F_{\max\text{-}f}$ and the max function $F_{\max}$.

To unify the proofs, we define a class of (m, n)-*stable* functions, which include F_0, $F_{\max\text{-}f}$, and $F_{\max}$, and our lower bound will hold for any (m, n)-stable function.

Definition V.1. *Let F be a function that has a multiset as input and outputs an element in $[R]$. It is (m, n)-stable if there exists a multiset X whose items can be arranged in an $m \times n$ matrix such that:*

(i) For any $1 \leq i_1 < i_2 \leq m$, $F(X[1{:}i_1, 1]) \neq F(X[1{:}i_2, 1])$;

(ii) For any $i \in [m]$, $j \in [n]$, and any $X' \subseteq X[1{:}i, 1{:}n]$, $F(X[1{:}i, j] \uplus X') = F(X[1{:}i, 1])$.

Here, we use $X[i_1{:}i_2, j_1{:}j_2]$ to denote the multiset of items in rows $i_1 \sim i_2$ and columns $j_1 \sim j_2$ of this matrix.

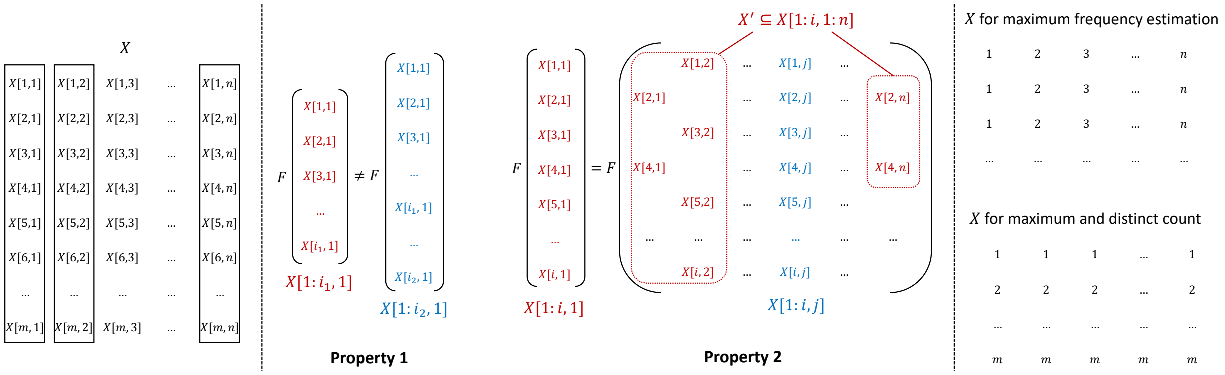


Fig. 2: Stable functions.

The two properties are illustrated in Figure 2. Intuitively, property (i) is just a non-triviality condition, requiring X to take distinct values on the first column, while (ii) is the stableness condition: For the first i rows for any i, all columns yield the same function value, even if “diluted” with other items. We can easily see that $F_{\text{max-}f}$ is (m, n)-stable for any m and any $n \leq R$, while Given $\varepsilon > 0$, $\beta > 0$, $\theta > 0$, and $k \geq 1$, for any D, and any $t \in \mathbb{Z}^+$, with probability at least $1 - \beta$, $F_{\text{k-sel}}$UserDP returns an $\tilde{o}_{t,k}$ with one-way rank error

$$O\left(\frac{\kappa(D_t)}{\varepsilon\theta} \cdot \log^{1.5} t \cdot \log^{2.5} R \cdot \log^{1+\theta}(\kappa(D_t)) \cdot \log(1/\beta)\right).$$

F_0 and $F_{\max}$ are (m, n)-stable for any $m \leq R$ and any n. The corresponding X is shown in Figure 2. On the other hand, count and sum are not (m, n)-stable for any $m, n \geq 2$.

Theorem V.2. *Fix any $\varepsilon > 0$. For any $m \in \mathbb{Z}^+$ and $n = \frac{m}{\varepsilon}$, let F be any (m, n)-stable function. For any ε-DP mechanism $\mathcal{M}$, there exists a stream D of length $T = 8mn = 8\varepsilon n^2$ and a time $t \in [T]$ such that $\mathrm{DS}^{(n-1)}(D_t) = 0$ but $\Pr[\mathcal{M}(D_t) \neq F(D_t)] \geq 1/3$.*

Before proving Theorem V.2, let’s see why it leads to the hardness result claimed earlier. By Theorem II.9, a (ρ, c)-down neighborhood optimal mechanism cannot have error $> c\rho \cdot \mathrm{DS}^{(\rho-1)}(D_t)$ on any D_t with probability more than $1/3$. Since the theorem shows that there exists a D_t on which $\mathrm{DS}^{(\rho-1)}(D_t) \leq \mathrm{DS}^{(n-1)}(D_t) = 0$ for any $\rho \leq n = O(\sqrt{T/\varepsilon})$, $\mathcal{M}(D_t) \neq F(D_t)$ implies that the error must be larger than $c\rho \cdot \mathrm{DS}^{(\rho-1)}(D_t)$ for any c. The complete proof of Theorem V.2 can be found in Appendix A-C.

VI. Dealing with Deletions

In this section, we briefly discuss the *fully-dynamic* setting in which items can be both inserted and deleted. For this case, we can simply separate the stream into two: an insertion stream that consists of all insertions and a deletion stream that consists of all deletions. We run a mechanism on each stream, each with privacy budget $\varepsilon/2$, and take the difference. The error is thus at most twice that of the insertion-only case, so the error is still $\tilde{O}(\kappa(D_t))$ for count, histogram, and $\tilde{O}(\varphi(D_t))$ for sum. For k-selection, which can be reduced to $\log R + 1$ instances of the histogram problem, we apply this technique on each of these instances. Then this also yields a one-way rank error of $\tilde{O}(\kappa(D_t))$ for the fully-dynamic case. Unfortunately, this technique does not work for the distinct count problem, since all copies of an item must be deleted for the distinct count to decrease by one. It remains an open problem if $\tilde{O}(\kappa(D_t))$ error can be achieved for this problem in the fully-dynamic case.

Note that in the fully-dynamic case, $\kappa(D_t)$ is the maximum number of events contributed by any user, which may be larger than $\mathrm{DS}(D_t)$. In the extreme case where all previously inserted item are deleted, we have $\mathrm{DS}(D_t) = 0$ while $\kappa(D_t)$ may be arbitrarily large. Thus, the simple mechanism above is not down-neighborhood optimal, as opposed to the insertion-only case. In Appendix B, we show that this is inevitable, even for the count function F_{cnt}. More precisely, we show the following lower bound:

Theorem VI.1. *For any ε-DP mechanism $\mathcal{M}$ and any T sufficiently large, there exists a fully-dynamic stream D of length T and a time $t \in [T]$ such that $\mathrm{DS}(D_t) = 1$ but $\Pr[|\mathcal{M}(D_t) - F_{\text{cnt}}(D_t)| \geq \zeta] \geq 1/3$, for $\zeta = \Omega(T^{1/3}/\varepsilon^{2/3})$.*

In addition, for count query under user-DP, for any $k > 0$ and any D_t, $\mathrm{DS}^{(k)}(D_t) \leq \mathrm{DS}(D_t)$. Thus this lower bound implies that it is not possible to achieve $(k, O(T^{1/3}/(k\varepsilon^{2/3})))$-down neighborhood optimality on F_{cnt} in the fully-dynamic case for any $k > 0$. This establishes a separation between the insertion-only case and the fully-dynamic case, since our mechanism is $(1, \tilde{O}(1))$-down neighborhood optimal in the insertion-only case.

VII. Experiments

In this section, we compare our algorithms with two baselines mentioned in Section III-A over four continual queries: count F_{cnt}, maximum frequency $F_{\text{max-}f}$, maximum function $F_{\max}$, and distinct count F_0.

Composition: We use advanced composition [18] to allocate ε', where $\varepsilon = \sqrt{2T\ln(1/\delta)}\varepsilon' + T\varepsilon'(e^{\varepsilon'} - 1)$, to each D_t and repeat running state-of-the-art algorithms in static setting

Dataset	T	$\lvert\mathcal{U}\rvert$	$\kappa(D_T)$	$F_{\text{max-}f}$		$F_{\max}$		F_0	
				Selected Item	R	Selected Item	R	Selected Item	R
AOL	1.76×10^7	4.7×10^5	1.51×10^5	Rank in SERP	500	Not selected			
MLens	2.5×10^7	1.63×10^5	3.22×10^4	Rating score	10	Rating score	10	Date + Rating score	1.57×10^{11}
Netf.	2.63×10^7	3.77×10^5	1.59×10^4	Not selected					
TLDR	3.85×10^6	1.46×10^6	3.76×10^5	Not selected				Time + Item ID	3.85×10^{11}

TABLE II: Real datasets used in the experiments.

Query type		F_{cnt}			$F_{\text{max-}f}$			$F_{\max}$			F_0		
Simulated Dataset		Unif.	Gaus.	Zipf.	Unif.	Gaus.	Zipf.	Unif.	Gaus.	Zipf.	Unif.	Gaus.	Zipf.
Our Mechanism	90%-max RE(%)	0.376	0.365	0.775	0.689	0.636	1.07	1.05	1.51	4.77	1.54	1.79	4.69
	Median RE(%)	0.197	0.203	0.523	0.281	0.316	0.676	0	0	1.10	1.10	1.17	2.89
	RT(s)	7.13×10^{-5}	4.77×10^{-5}	6.84×10^{-5}	8.27×10^{-3}	8.02×10^{-3}	8×10^{-3}	5.59×10^{-4}	6.04×10^{-4}	5.78×10^{-4}	6.9×10^{-5}	4.66×10^{-5}	6.73×10^{-5}
Advanced Composition	90%-max RE(%)	97.9	97.8	97.9	97.8	97.8	97.9	16.6	16.6	16.7	90.6	90.6	91.1
	Median RE(%)	96.2	96.2	96.5	96.0	96.1	96.5	2.17	2.17	2.18	89.9	89.8	90.6
	RT(s)	1.98×10^{-3}	2.04×10^{-3}	1.83×10^{-3}	0.336	0.335	0.367	9.02×10^{-5}	9.33×10^{-5}	9.02×10^{-5}	0.486	0.489	0.545
Truncation (random τ)	90%-max RE(%)	42.4	40.6	59	68.1	67.9	75.5	100	100	100	42.3	36.1	43
	Median RE(%)	14.2	13.3	16.8	15.0	13.4	22.6	20	20.1	21	21.6	21.2	32.1
	RT(s)	1.29×10^{-5}	1.28×10^{-5}	1.26×10^{-5}	8.11×10^{-3}	7.93×10^{-3}	8.94×10^{-3}	5.07×10^{-4}	4.96×10^{-4}	4.91×10^{-4}	1.30×10^{-5}	1.30×10^{-5}	1.33×10^{-5}
Truncation ($\tau = 1024$)	90%-max RE(%)	1.09	0.935	0.997	1.65	1.69	1.75	44.4	44.4	44.5	2.39	2.82	2.49
	Median RE(%)	0.235	0.24	0.254	0.379	0.357	0.385	6.98	7.00	6.99	1.3	1.31	1.38
	RT(s)	1.22×10^{-5}	1.24×10^{-5}	1.22×10^{-5}	8.13×10^{-3}	7.89×10^{-3}	8.08×10^{-3}	4.87×10^{-4}	4.83×10^{-4}	4.82×10^{-4}	1.29×10^{-5}	1.32×10^{-5}	1.26×10^{-5}

TABLE III: Comparison among our mechanism, advanced composition, and truncation mechanism on simulated datasets ($\varepsilon = 2$). RE and RT denote relative error and running time.

under user-DP. For F_{cnt} and $F_{\max}$, we use [20] and [24]. For $F_{\text{max-}f}$ and F_0, we use the ideas in Section IV-B and IV-D to extend F_{cnt} to support $F_{\text{max-}f}(D_t)$ and $F_0(D_t)$ instead of using the state-of-the-art polynomial algorithms proposed in [20] since they suffer from high computational cost, where each update requires more than 1 hour in our experiments, and they do not achieve any optimal utility.

Truncation: The truncation mechanism has as input a truncation threshold τ. Without prior knowledge, it is naturally impossible to select a proper τ before data comes. One idea is to randomly pick a τ from [GS], where GS is an upper bound for $\kappa(D_T)$ and is set 2^{20} in our experiments. However, using a random τ from [GS] achieves the same error level as using $\tau = \text{GS}$. To achieve better error, we use $2, 4, 8, \ldots, \text{GS}$ as the candidates. We also discuss the error with τ near $\kappa(D_T)$, which achieves the optimal error at final time T, but it remains unknown how to obtain $\kappa(D_T)$ in advance even under non-private setting.

A. Setup

Dataset We use both simulated data and real world data. For the simulated ones, we set $T = 5 \times 10^7$ and $|\mathcal{U}| = 10^6$. We use three distributions over $[1024]$ to simulate the number of items for each user: Zipf distribution $f(x) \propto (x + a)^{-b}$ with $a = 10$, $k = 1$; Gauss distribution with $\mu = 50$, $\sigma = 30$; and uniform distribution. Then, we use another independent step to assign values to the items. Each item value is drawn from some distribution dependent on the query function. For $F_{\max-f}$ and F_{cnt}, we use Zipf distribution with range $[1000]$ and parameters $a = 0$, $k = 2$. For F_0, we use uniform distribution with range $[10^7]$. For $F_{\max}$, we try a small R to save running time thus use uniform distribution with range $[20]$ for $F_{\max}$. Besides, to avoid the maximum value reaching 20 at a very early time. We reorder the dataset by the values.

In addition, we use four real datasets: AOL-user-ct-collection (AOL) [29], MovieLens (MLens) [35], Netflix-Prize (Netf.) [1], and Webis-TLDR-Corpus-2017 (TLDR) [36]. AOL is a collection of web queries with each record including the rank of the item the user clicked in search engine results page (SERP). MLens and Netf. are rating scores given by users to movies provided by movie recommendation website MovieLens and streaming-on-demand media provider Netflix. TLDR contains millions of posts from users on the social website Reddit. We test F_{cnt} on all four datasets and test the other queries on selected datasets. The details of datasets and items selected for each query are shown in Table II. For $F_{\max}$, we reorder the data in the same way as simulated data.

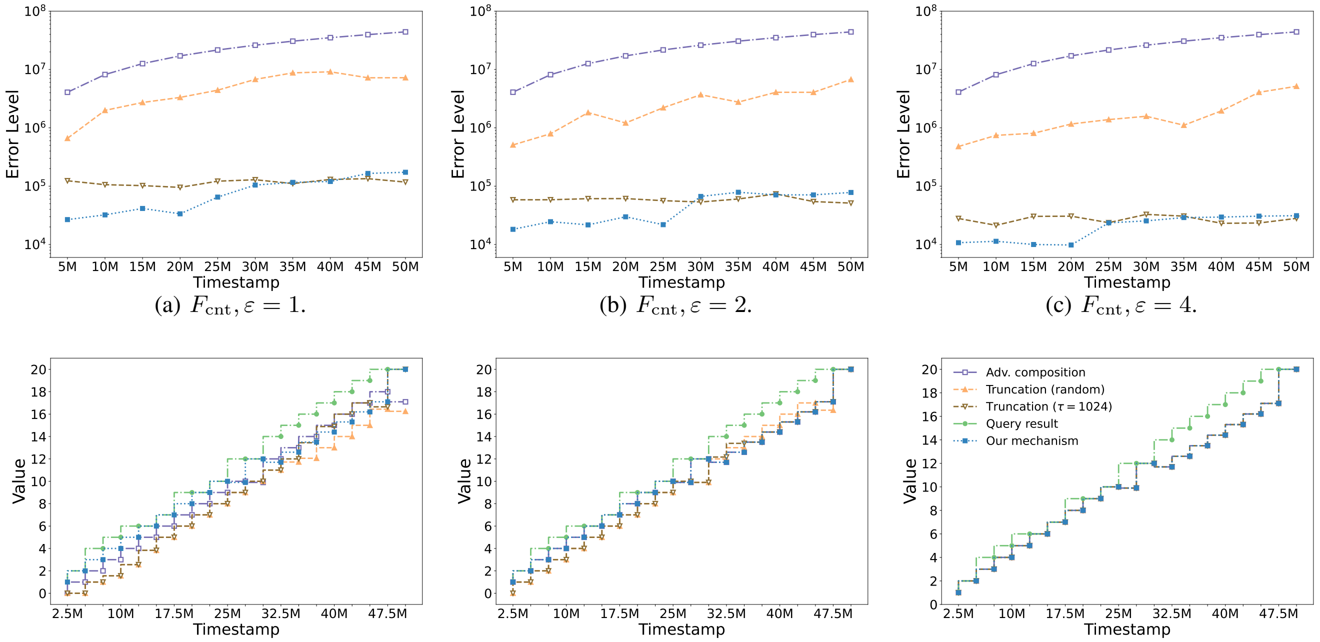
Experimental Parameters All experiments are conducted on a Linux server with a 24-core 48-thread 2.2GHz Intel Xeon CPU and 256GB memory. We report rank error for $F_{\max}$ and additive error for the others. Each experiment is repeated 30 times[5] and we collect the error every 10^5 times for the experiments over dataset TLDR and every 5×10^5 times for the others. For each selected timestamp, we remove 20% largest errors and 20% smallest errors and report the average error for the rest runs. For the privacy budget, we use $\varepsilon = 1, 2, 4$ and the default value is set to 2. In addition, we set the failure probability β to 0.1 and use $\theta = 1$. Furthermore, we also do some practical optimizations for our mechanisms (see Appendix C-A).

B. Experimental Results for Simulated Data

Utility and efficiency The errors and running times of all mechanisms over simulated data are shown in Table III. At each selected timestamp, we collect the relative error[6] and report their median and 90% maximum. For the truncation mechanism, besides using a random τ, we report the error with fixed $\tau = 1024$, which is used as the upper bound for user contribution in the data simulating process and nearly $\kappa(D_T)$ for all simulated datasets. The results indicate a clear superiority of our mechanism over the advanced composition and the truncation mechanism with a random τ in terms of

[5] For the truncation mechanism with a random τ, we repeat the experiments 20×30 times.

[6] For $F_{\max}$, the relative error equals to the rank error divided by the full rank.



(a) $F_{\text{cnt}}, \varepsilon = 1$.

(b) $F_{\text{cnt}}, \varepsilon = 2$.

(c) $F_{\text{cnt}}, \varepsilon = 4$.

(d) $F_{\max}, \varepsilon = 1$.

(e) $F_{\max}, \varepsilon = 2$.

(f) $F_{\max}, \varepsilon = 4$.

Fig. 3: Error levels/Values vs time of various mechanisms over $F_{\text{cnt}}/F_{\max}$ on Unif. dataset with $\varepsilon = 1, 2, 4$.

utility. What is more desirable is its robustness: Our median relative error is below 3% in all cases and is below 1% in all but four cases. In addition, we achieve slightly higher utility in F_{cnt} compared with $F_{\max\text{-}f}$, $F_{\max}$, and F_0. This confirms our theoretical analysis: We achieve down-neighborhood optimal error at each time in F_{cnt}, which is unachievable in the other three queries. Even though relaxing the privacy protection, the advanced composition still loses its utility in all queries but $F_{\max}$, where we also have higher utility and even achieve zero rank error in more than half of the time in two cases. For the truncation mechanism, using $\tau = 1024$ performs much better than using a random τ and achieves similar performance to ours on F_{cnt}, $F_{\max\text{-}f}$. The reason is, for simulated data, $\kappa(D_t)$ is relatively small and we start at $\tau = 64$ in the real implementation (see Appendix C-B), which only differs a constant factor from $\tau = 1024$. In addition, we need to separate the privacy budget to estimate $\kappa(D_t)$ and dynamically update τ. However, setting $\tau = 1024$ in practice is unachievable without enough prior knowledge as we mentioned before. Furthermore, in real data, our superiority over the truncation mechanism with a good τ is more clear and we will show this later.

In terms of running time, our mechanism runs a bit slower than the truncation mechanism but much faster than the advanced composition except F_{cnt} and $F_{\max}$. The reason is, both our mechanism and truncation mechanism only need $\tilde{O}(1)$ amortized updating time while we use more time to estimate the user contributions. Meanwhile, the advanced composition needs to re-execute the algorithm at each time. For F_{cnt} and $F_{\max}$, we observe some tricks in [24, 20] so they also do not need to run the whole algorithm after each update.

Error with time We also conducted experiments to see how the error changes with time for various mechanisms. We tested Unif. dataset over all four queries with different $\varepsilon = 1, 2, 4$. We plot the results for F_{cnt} and $F_{\max}$ in Figure 3 and defer the results for $F_{\max\text{-}f}$ and F_0 to Appendix C-B. The message from the plot is our mechanism has the error increasing with time and always achieves the highest utility except at the final stage, where the truncation mechanism with $\tau = 1024$ has a similar error to ours. It matches our theoretical guarantee that we achieve an error proportional to $\kappa(D_t)$ at each time, which is strictly smaller than $\kappa(D_T)$ and is close to $\kappa(D_T)$ at the end. In contrast, the error level of the truncation mechanism with $\tau = \kappa(D_T)$ always keeps at $\kappa(D_T)$.

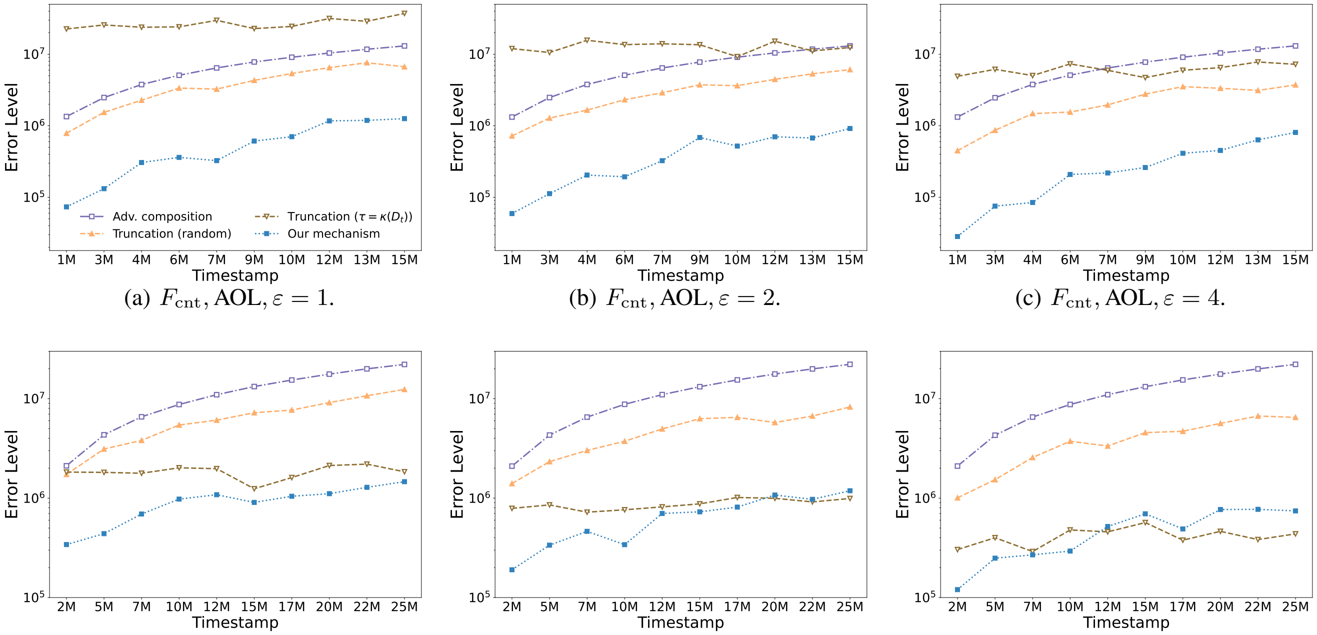
C. Experimental Results for Real Data

Utility and efficiency The experimental results over real data are shown in Table IV. For the truncation mechanism, we additionally report the result with $\tau = \kappa(D_T)$.[7] For real data, all mechanisms have worse utilities since the data are more skewed, which has already been revealed in Table II. But our mechanism still has high utilities: the median relative error is below 10% in six cases and is below 20% in all cases except $F_{\max}$ over MLens, where data are too skewed and $\kappa(D_t)$ is large in most time t thus using composition or a small truncation threshold τ can lead to a better result. Another interesting observation is, in real data, the superiority of our mechanism over the truncation mechanism with $\tau = \kappa(D_T)$ is more clear. Indeed, using $\tau = \kappa(D_T)$ does not always imply a better performance than using a random τ, indicating the

[7] $\tau = 2^s$, where $s = \lceil \log(\kappa(D_T)) \rceil$.

Query type		Count				Max Frequency		Max	Distinct Count	
Real Dataset		AOL	MLens	Netf.	TLDR	AOL	MLens	MLens	MLens	TLDR
Our Mechanism	90%-max RE(%)	6.36	11.7	7.91	10.5	11	21.5	62.7	20.7	10.8
	Median RE(%)	4.55	8.58	5.37	3.21	7.75	15.6	35.8	15.1	4.84
	RT(s)	6.82×10^{-5}	6.94×10^{-5}	8.55×10^{-5}	6.25×10^{-5}	4.2×10^{-3}	1.05×10^{-4}	3.13×10^{-4}	6.79×10^{-5}	4.36×10^{-5}
Advanced Composition	90%-max RE(%)	97.3	99.4	98.5	63.2	96.1	99.8	37.2	99.4	63.3
	Median RE(%)	95.9	99.3	97.8	61.5	94.3	99.3	0	99.2	61.7
	RT(s)	2.94×10^{-3}	8.69×10^{-4}	1.71×10^{-3}	2.69×10^{-3}	0.608	0.188	1.44×10^{-4}	0.335	0.484
Truncation (random τ)	90%-max RE(%)	73.8	79.7	66.6	48.2	88.2	94.7	100	94.6	48.2
	Median RE(%)	39.2	50.9	40.6	20.0	59.5	77.9	10.8	70.8	29.5
	RT(s)	1.23×10^{-5}	1.23×10^{-5}	1.21×10^{-5}	1.2×10^{-5}	4.99×10^{-3}	9.41×10^{-5}	3.03×10^{-4}	1.22×10^{-5}	1.27×10^{-5}
Truncation ($\tau = \kappa(D_T)$)	90%-max RE(%)	800	52.4	31.5	6300	7740	278	100	183	12900
	Median RE(%)	162	14.0	6.53	1140	1470	49.6	100	30.1	2280
	RT(s)	1.26×10^{-5}	1.28×10^{-5}	1.17×10^{-5}	1.23×10^{-5}	4.29×10^{-3}	9.68×10^{-5}	2.99×10^{-4}	1.22×10^{-5}	1.25×10^{-5}

TABLE IV: Comparison among our mechanism, advanced composition, and truncation mechanism on on real datasets ($\varepsilon = 2$). RE and RT denote relative error and running time.



(a) F_{cnt}, AOL, $\varepsilon = 1$.

(b) F_{cnt}, AOL, $\varepsilon = 2$.

(c) F_{cnt}, AOL, $\varepsilon = 4$.

(d) F_{cnt}, Netf., $\varepsilon = 1$.

(e) F_{cnt}, Netf., $\varepsilon = 2$.

(f) F_{cnt}, Netf., $\varepsilon = 4$.

Fig. 4: Error levels vs time of various mechanisms over F_{cnt} on AOL and Netf. datasets with $\varepsilon = 1, 2, 4$.

difficulty of selecting a good τ. Besides, the running times of all mechanisms are quite similar to the simulated data, where both our mechanism and truncation mechanism are with high efficiency.

Error with time We also tested F_{cnt} over AOL and Netf. to show how the error changes with time in real data. The results are plotted in Figure 4. Compared with simulated data, our superiority in terms of utility is larger. This is especially obvious in AOL dataset, where all mechanisms lose utility but we still have high utility. Using $\tau = \kappa(D_T)$ in the truncation mechanism does not always lead to a good performance. In Netf., they still can achieve similar errors as ours in the final steps while in AOL dataset, their error level is always above the query result.

Acknowledgment

This work has been supported by HKRGC under grants 16201819, 16205420, and 16205422. We would also like to thank the anonymous reviewers who have made valuable suggestions on improving the presentation of the paper.

References

[1] Netflix prize dataset. https://archive.org/download/nf_prize_dataset.tar. Accessed: 2016-02-04.

[2] Asi, H., and Duchi, J. C. Instance-optimality in differential privacy via approximate inverse sensitivity mechanisms. *Advances in Neural Information Processing Systems 33* (2020).

[3] Bao, E., Yang, Y., Xiao, X., and Ding, B. Cgm: an enhanced mechanism for streaming data collection with local differential privacy. *Proceedings of the VLDB Endowment 14*, 11 (2021), 2258–2270.

[4] Bolot, J., Fawaz, N., Muthukrishnan, S., Nikolov, A., and Taft, N. Private decayed predicate sums on streams. In *Proceedings of the 16th International Conference on Database Theory* (2013), pp. 284–295.

[5] Bun, M., Nissim, K., and Stemmer, U. Simultaneous private learning of multiple concepts. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science* (2016), pp. 369–380.

[6] Cardoso, A. R., and Rogers, R. Differentially private histograms under continual observation: Streaming selection into the unknown. In *International Conference on Artificial Intelligence and Statistics* (2022), PMLR, pp. 2397–2419.

[7] Chan, T.-H. H., Li, M., Shi, E., and Xu, W. Differentially private continual monitoring of heavy hitters from distributed streams. In *International Symposium on Privacy Enhancing Technologies Symposium* (2012), Springer, pp. 140–159.

[8] Chan, T.-H. H., Shi, E., and Song, D. Private and continual release of statistics. *ACM Transactions on Information and System Security* (2011).

[9] Chen, Y., Machanavajjhala, A., Hay, M., and Miklau, G. Pegasus: Data-adaptive differentially private stream processing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), pp. 1375–1388.

[10] Cummings, R., Krehbiel, S., Lai, K. A., and Tantipongpipat, U. Differential privacy for growing databases. *Advances in Neural Information Processing Systems 31* (2018).

[11] Denisov, S., McMahan, B., Rush, K., Smith, A., and Thakurta, A. Improved differential privacy for sgd via optimal private linear operators on adaptive streams. In *NeurIPS* (2022).

[12] Dong, W., Fang, J., Yi, K., Tao, Y., and Machanavajjhala, A. R2t: Instance-optimal truncation for differentially private query evaluation with foreign keys. In *Proc. ACM SIGMOD International Conference on Management of Data* (2022).

[13] Dong, W., and Yi, K. Universal private estimators. *arXiv preprint arXiv:2111.02598* (2021).

[14] Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference* (2006), Springer, pp. 265–284.

[15] Dwork, C., Naor, M., Pitassi, T., and Rothblum, G. N. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing* (2010), pp. 715–724.

[16] Dwork, C., Naor, M., Reingold, O., and Rothblum, G. N. Pure differential privacy for rectangle queries via private partitions. In *International Conference on the Theory and Application of Cryptology and Information Security* (2015), Springer, pp. 735–751.

[17] Dwork, C., Naor, M., Reingold, O., Rothblum, G. N., and Vadhan, S. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the forty-first annual ACM symposium on Theory of computing* (2009), pp. 381–390.

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

[19] Fan, L., and Xiong, L. An adaptive approach to real-time aggregate monitoring with differential privacy. *IEEE Transactions on knowledge and data engineering 26*, 9 (2013), 2094–2106.

[20] Fang, J., Dong, W., and Yi, K. Shifted inverse: A general mechanism for monotonic functions under user differential privacy. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (2022), pp. 1009–1022.

[21] Fichtenberger, H., Henzinger, M., and Ost, W. Differentially private algorithms for graphs under continual observation. In *29th Annual European Symposium on Algorithms (ESA 2021)* (2021), Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[22] Henzinger, M., and Upadhyay, J. Constant matters: Fine-grained complexity of differentially private continual observation using completely bounded norms. *arXiv preprint arXiv:2202.11205* (2022).

[23] Henzinger, M., Upadhyay, J., and Upadhyay, S. Almost tight error bounds on differentially private continual counting. *arXiv preprint arXiv:2211.05006* (2022).

[24] Huang, Z., Liang, Y., and Yi, K. Instance-optimal mean estimation under differential privacy. In *NeurIPS* (2021).

[25] Joseph, M., Roth, A., Ullman, J., and Waggoner, B. Local differential privacy for evolving data. *Advances in Neural Information Processing Systems 31* (2018).

[26] Kamath, G., Li, J., Singhal, V., and Ullman, J. Privately learning high-dimensional distributions. In *Conference on Learning Theory* (2019), PMLR, pp. 1853–1902.

[27] Kellaris, G., Papadopoulos, S., Xiao, X., and Papadias, D. Differentially private event sequences over infinite streams. *Proceedings of the VLDB Endowment 7*, 12 (2014), 1155–1166.

[28] McSherry, F. D. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In

Proceedings of the 2009 ACM SIGMOD International Conference on Management of data (2009), pp. 19–30.

[29] Pass, G., Chowdhury, A., and Torgeson, C. A picture of search. In *Proceedings of the 1st international conference on Scalable information systems* (2006).

[30] Perrier, V., Asghar, H. J., and Kaafar, D. Private continual release of real-valued data streams. In *26th Annual Network and Distributed System Security Symposium, NDSS 2016* (2019), Internet Society, pp. 1–13.

[31] Ren, X., Shi, L., Yu, W., Yang, S., Zhao, C., and Xu, Z. Ldp-ids: Local differential privacy for infinite data streams. In *Proc. ACM SIGMOD International Conference on Management of Data* (2022).

[32] Upadhyay, J. Sublinear space private algorithms under the sliding window model. In *International Conference on Machine Learning* (2019), PMLR, pp. 6363–6372.

[33] Upadhyay, J., Upadhyay, S., and Arora, R. Differentially private analysis on graph streams. In *International Conference on Artificial Intelligence and Statistics* (2021), PMLR, pp. 1171–1179.

[34] Vadhan, S. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*. Springer, 2017, pp. 347–450.

[35] Vig, J., Sen, S., and Riedl, J. The tag genome: Encoding community knowledge to support novel interaction. *ACM Trans. Interact. Intell. Syst. 2*, 3 (2012).

[36] Völske, M., Potthast, M., Syed, S., and Stein, B. TL;DR: Mining Reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization* (2017), pp. 59–63.

[37] Wang, Q., Zhang, Y., Lu, X., Wang, Z., Qin, Z., and Ren, K. Rescuedp: Real-time spatio-temporal crowd-sourced data publishing with differential privacy. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications* (2016), IEEE, pp. 1–9.

[38] Wang, T., Chen, J. Q., Zhang, Z., Su, D., Cheng, Y., Li, Z., Li, N., and Jha, S. Continuous release of data streams under both centralized and local differential privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (2021), pp. 1237–1253.

[39] Wang, Z., Liu, W., Pang, X., Ren, J., Liu, Z., and Chen, Y. Towards pattern-aware privacy-preserving real-time data collection. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications* (2020), IEEE, pp. 109–118.

[40] Wang, Z., Pang, X., Chen, Y., Shao, H., Wang, Q., Wu, L., Chen, H., and Qi, H. Privacy-preserving crowd-sourced statistical data publishing with an untrusted server. *IEEE Transactions on Mobile Computing 18*, 6 (2018), 1356–1367.

Appendix A
Proofs

A. Proof of Lemma III.2

Lemma III.2. *Given any $\varepsilon > 0$, $\beta > 0$, $\theta > 0$, and any D, with probability at least $1 - \beta$, we have $\tilde{\tau}_t \le 2 \cdot \kappa(D_t)$ and*

$$\mathrm{Count}(D_t, \tau_i) = O\Big(\frac{1}{\varepsilon\theta} \cdot \log^{1+\theta}(\kappa(D_t)) \cdot \log(t/\beta)\Big),$$

for all $t \in \mathbb{Z}^+$ simultaneously.

Proof. For any $i \in \mathbb{N}$, let t_i be the first time such that

$$\mathrm{Count}(D_{t_i}, \tau_i) \ge \frac{12}{\varepsilon_i}\log(2/\beta_i) + \frac{16}{\varepsilon_i}\log(t_i + 1), \tag{4}$$

where $\tau_i = 2^i$, $\varepsilon_i = \frac{\varepsilon\theta}{(i+1)^{1+\theta}}$, and $\beta_i = \frac{\beta}{(i+1)^2}$. By the tail bound of Laplace distribution, we have with probability at least $1 - \frac{\beta_i}{2}$,

$$|\tilde{\eta}| < \frac{2}{\varepsilon_i}\log(2/\beta_i). \tag{5}$$

Let $\widetilde{\mathrm{Count}}(D_t, \tau_i, \beta_i) = \overline{\mathrm{Count}}(D_t, \tau_i, \beta_i) + \mathrm{Lap}(4/\varepsilon)$, which is used internally in the SVT. For any $t \in \mathbb{Z}^+$, with probability at least $1 - \frac{\beta_i}{2(t+1)^2}$, we have

$$\begin{aligned} &\left|\widetilde{\mathrm{Count}}(D_t, \tau_i, \beta_i) - \overline{\mathrm{Count}}(D_t, \tau_i, \beta_i)\right| \\ <&\frac{4}{\varepsilon_i}\log(2/\beta_i) + \frac{8}{\varepsilon_i}\log(t+1). \end{aligned} \tag{6}$$

Applying a union bound and combining (4), (5), and (6), for any i, with probability at least $1 - \beta_i$, the i-th SVT stops at some t'_i such that

$$t'_i \le t_i, \tag{7}$$

and

$$\mathrm{Count}(D_{t_i}, \tau_i) > 0, \tag{8}$$

By another union bound over all the SVT's, with probability at least $1 - \beta$, (7) and (8) hold for all i. Furthermore, (8) implies, for all time $t \in \mathbb{Z}^+$,

$$\tilde{\tau}_t \le 2 \cdot \kappa(D_t),$$

and its corresponding ε_i have

$$\varepsilon_i \ge \frac{\varepsilon\theta}{\big(\log(\kappa(D_t)) + 1\big)^{1+\theta}} \tag{9}$$

Further combining (4), (8), and (9), we have

$$\mathrm{Count}(D_t, \tau_i) = O\Big(\frac{1}{\varepsilon\theta} \cdot \log^{1+\theta}(\kappa(D_t)) \cdot \log(t/\beta)\Big).$$

□

B. Proof of Theorem IV.2

Theorem IV.2. *Given $\varepsilon > 0$, $\beta > 0$, $\theta > 0$, and $k \geq 1$, for any D, and any $t \in \mathbb{Z}^+$, with probability at least $1-\beta$, $F_{\text{k-sel}}\text{UserDP}$ returns an $\tilde{o}_{t,k}$ with one-way rank error*

$$O\bigg(\frac{\kappa(D_t)}{\varepsilon\theta} \cdot \log^{1.5} t \cdot \log^{2.5} R \cdot \log^{1+\theta}(\kappa(D_t)) \cdot \log(1/\beta)\bigg).$$

Proof. Recall in each $F_{\text{ht}}\text{EventDP}$, we first run EstimatingUserContribution with half privacy budget to find the estimation of $\kappa(D_t)$. By Lemma III.2, with probability at least $1 - \frac{\beta}{2}$, for all $t \in \mathbb{Z}^+$ and all $i \in [\log(R) + 1]$, we have

$$\tilde{\tau}_t^i \leq 2 \cdot \kappa(D_t), \tag{10}$$

and

$$\begin{aligned}\text{Count}(D_t, \tilde{\tau}_t^i) =& O\bigg(\frac{1}{\varepsilon\theta} \cdot \log^{1+\theta}(\kappa(D_t)) \\ & \cdot \log(R) \cdot \log\Big(t \cdot \log(R)/\beta\Big)\bigg).\end{aligned} \tag{11}$$

Then, we do the truncation and run $F_{\text{ht}}\text{EventDP}$ to get $\widetilde{F}_{\text{cnt}}(D_t|_I)$ for all $t \in \mathbb{Z}^+$,

$$I \in \Big\{[(j-1) \cdot 2^{i-1} + 1, j \cdot 2^{i-1}], i \in [\log(R)+1], j \in [2^{i-1}]\Big\}.$$

(10) and (11) cause the noise and bias respectively.

Next, for any $t \in \mathbb{Z}^+$ and $r \in [R]$, let's analyze the error for each $\widetilde{F}_{\text{cnt}}(D_t|_{[r,R]})$. Since it is the combinations for at most $\log(R)$ $\widetilde{F}_{\text{cnt}}(D_t|_I)$'s thus inherits their biases and noises. For the bias term, we just linearly sum the biases of the combined $\widetilde{F}_{\text{cnt}}(D_t|_I)$'s, each of which is already bounded following (11). For the noise term, recall the noise of each $\widetilde{F}_{\text{cnt}}(D_t|_I)$ is from multiple independent Laplace noises, thus we can use Lemma II.6 to obtain a tighter bound than the linear sum. More precisely, conditioned on (10) and (11), with probability at least $1 - \frac{\beta}{2}$, for any $t \in \mathbb{Z}^+$ and all $r \in [R]$, we have

$$\begin{aligned}&F_{\text{cnt}}(D_t|_{[r,R]}) - \widetilde{F}_{\text{cnt}}(D_t|_{[r,R]}) \in \bigg[-\Delta, \Delta + O\Big(\frac{\kappa(D_t)}{\varepsilon\theta} \\ &\cdot \log^{1+\theta}(\kappa(D_t)) \cdot \log^2(R) \cdot \log\Big(t \cdot \log(R)/\beta\Big)\Big)\bigg],\end{aligned} \tag{12}$$

where $\Delta = (3)$.

Now, we begin to analyze the rank error for $\tilde{o}_{t,k}$. With (12), we have

$$k \leq F_{\text{cnt}}(D_t|_{[\tilde{o}_{t,k},R]}),$$

$$\begin{aligned}&F_{\text{cnt}}(D_t|_{[\tilde{o}_{t,k}-1,R]}) \leq k + 2\lambda + O\Big(\frac{\kappa(D_t)}{\varepsilon\theta} \\ &\cdot \log^{1+\theta}(\kappa(D_t)) \cdot \log^2(R) \cdot \log\Big(t \cdot \log(R)/\beta\Big)\Big)\end{aligned}$$

which further implies the claim. □

C. Proof of Theorem V.2

Theorem V.2. *Fix any $\varepsilon > 0$. For any $m \in \mathbb{Z}^+$ and $n = \frac{m}{\varepsilon}$, let F be any (m,n)-stable function. For any ε-DP mechanism $\mathcal{M}$, there exists a stream D of length $T = 8mn = 8\varepsilon n^2$ and a time $t \in [T]$ such that $\text{DS}^{(n-1)}(D_t) = 0$ but $\Pr[\mathcal{M}(D_t) \neq F(D_t)] \geq 1/3$.*

Proof. We will construct a family $\mathcal{D}$ of streams, each of length T, and argue that some $D \in \mathcal{D}$ satisfies the claim in the theorem.

Let X be the matrix that witnesses the (m,n)-stableness of F. There are n users $u_1, \ldots, u_n$, and the j-th column of items of X are contributed by u_j. We divide the time domain $[1,T]$ into $\frac{T}{n} = 8m$ buckets, each of length n. Each stream $D \in \mathcal{D}$ is constructed as follows. We choose m buckets, indexed by $(b_1, b_2, \ldots, b_m)$. For $i \in [n]$, we use the i-th row of X to fill bucket b_i. The remaining buckets are filled with dummy items. Since there are

$$\binom{\frac{T}{n}}{m} = \binom{8m}{m} \geq 8^m + 1 \tag{13}$$

choices for the buckets, we have $|\mathcal{D}| \geq 8^m + 1$.

It is easy to see that $D_{b_i n}$, i.e., the data set after the i-th bucket has arrived, contains exactly the first i rows of X. Then by property (i) of a stable function, $F(D_{b_{i_1} n}) \neq F(D_{b_{i_2} n})$ for any $1 \leq i_1 < i_2 \leq m$. Recall that the j-th column of X is contributed by user u_j. Then by property (ii) of a stable function, we have $\text{DS}^{(n-1)}(D_{\ell n}) = 0$ for any $\ell \in [\frac{T}{n}]$. Now that all $D \in \mathcal{D}$ and all $t = \ell n$ satisfy the condition $\text{DS}^{(n-1)}(D_t) = 0$, it only remains to show that on at least one of them, we have $\Pr[\mathcal{M}(D_t) \neq F(D_t)] \geq 1/3$.

We prove so by contradiction. Assume for any $D \in \mathcal{D}$ and $t \in [T]$, we have $\Pr[\mathcal{M}(D_t) \neq F(D_t)] < 1/3$. By running $O(\log T)$ independent instances of $\mathcal{M}$ and taking the majority answer for each t, we have that for each $D \in \mathcal{T}$,

$$\Pr[\mathcal{M}(D) \neq \mathcal{F}(D)] < 1/3,$$

where $\mathcal{F}(D) = (F(D_n), F(D_{2n}), \ldots, F(D_T)$ and $\mathcal{M}(D) = (\mathcal{M}(D_n), \mathcal{M}(D_{2n}), \ldots, \mathcal{M}(D_T))$ are respectively the vectors of function values and the outputs of $\mathcal{M}$ after each bucket. Because each $D \in \mathcal{D}$ chooses a different set of buckets, $\mathcal{F}(D)$ are different for all $D \in \mathcal{D}$.

Fix some $D^\circ \in \mathcal{D}$. We have

$$\begin{aligned}\frac{1}{3} >& \Pr\Big[\mathcal{M}(D^\circ) \neq \mathcal{F}(D^\circ)\Big] \\ \geq& \sum_{D \in \mathcal{D}, D \neq D^\circ} \Pr\Big[\mathcal{M}(D^\circ) = \mathcal{F}(D)\Big] \\ \geq& \sum_{D \in \mathcal{D}, D \neq D^\circ} \Pr\Big[\mathcal{M}(D) = \mathcal{M}(D)\Big] \cdot e^{-2\varepsilon n} \\ =& (|\mathcal{D}| - 1) \cdot \frac{2}{3} \cdot e^{-2\varepsilon n} \\ \geq& 8^m \cdot e^{-2\varepsilon \frac{m}{\varepsilon}} \cdot \frac{2}{3} \\ \geq& \frac{2}{3},\end{aligned} \tag{14}$$

which is a contradiction. In the above, the second inequality is because all $\mathcal{F}(D)$'s are distinct. The third inequality is due to group privacy and the fact that $d_U(D, D^\circ) \leq 2n$ as there are only n users. □

Appendix B
The challenge of data deletions (Proof of Theorem V.2)

In the previous sections, we show even only with insertions, continual observation can introduce new challenges to user-DP. For queries like maximum frequency estimation, maximum estimation, and distinct count, down-neighborhood optimal error as the static setting is unachievable. But for count/sum estimation, we still can achieve $(\tilde{O}(1), \tilde{O}(1))$-down neighborhood optimal error. Now, we show considering deletion can further enlarge this difficulty. Here, we only consider the count problem since the sum is more general and thus also holds the same lower bound.

Let's first build a connection between count queires in fully-dynamic setting under user-DP with high-dimensional bit-sum estimation, a very fundamental problem that has been researched largely. In that problem, the input $A = \{a_i\}_{i\in[n]} \subset \{0,1\}^d$ with each user $u^{(i)}$ having one d-dimensional data a_i, and we target to answer

$$\mathrm{BitSum}(A) = \sum_{i\in[n]} a_i.$$

There is one negative result.

Lemma B.1 (Derived from Lemma 6.2 [26]). *For any $\varepsilon > 0$, $d > 0$, and any $n \geq \frac{\sqrt{d}}{\varepsilon}$, if one mechanism $\mathcal{M} : \{0,1\}^{n\times d} \to [0,n]^d$, and for any input $A \in \{0,1\}^{n\times d}$,*

$$\mathsf{E}\left[\left\|\mathcal{M}(A) - \mathrm{BitSum}(A)\right\|_2^2\right] \leq \alpha^2,$$

then, $\alpha \geq \frac{d}{72\varepsilon}$.

Now, we construct D from $A \in \{0,1\}^{n\times d}$. Here, time interval $[1, 2nd]$ is divided into $2d$ buckets containing n time units. In the i-th odd bucket, we insert the data of i-th dimension in A while in i-th even bucket, we delete them. More precisely, for each $i \in [n]$, $j \in [d]$, let $t_1 = 2dj - 2d + i$, $t_2 = 2dj - d + i$ and we set $(x_{t_1}, y_{t_1}, u_{t_1}) = (a_{i,j}, 1, u^{(i)})$,[8] and $(x_{t_2}, y_{t_2}, u_{t_2}) = (a_{i,j}, -1, u^{(i)})$. With this setting, we see (1) Modifying the a_i in A will only modify the events of $u^{(i)}$ thus for $A \sim_E A'$, we have $D \sim_U D'$; (2) For each $j \in [d]$, let $t = 2jd - d$, then $F_{\text{cnt}}(D_t) = \sum_{i\in[n]} a_{i,d}$, and $\mathrm{DS}(D_t) \leq 1$. Therefore, we have

Lemma B.2. *For any $\varepsilon > 0$, $d > 0$, and any $n \geq \frac{\sqrt{d}}{\varepsilon}$, if one mechanism $\mathcal{M}$ can answer count queries in the fully-dynamic setting under ε-user DP and for any D, with $\frac{2}{3}$ probability, it answers $F_{\text{cnt}}(D_t)$ for all $t \in [dn]$ with error $\zeta \cdot \mathrm{DS}(D_t)$, then, we can construct an $\mathcal{M}'$ to answer d-dimensional bit sum estimation and for any and for any input $A \in \{0,1\}^{n\times d}$, with probability at least $\frac{2}{3}$,*

$$\mathsf{E}\left[\left\|\mathcal{M}(A) - \mathrm{BitSum}(A)\right\|_2^2\right] \leq d \cdot \zeta^2.$$

Since the result has a bounded range, i.e., $[0,n]^d$, by Chernoff bound, we can run $\mathcal{M}'$ $\tilde{O}(\log(nd))$ times to achieve bounded expected error. Therefore, with Lemma B.1, we know ζ should be at least $\Omega(\frac{\sqrt{d}}{\varepsilon\log(n)})$. By further setting $n = \frac{\sqrt{d}}{\varepsilon}$, we get Theorem V.2.

Appendix C
Additional Experiments

A. Practical Optimization for Our Mechanism

In EstimateUserContribution, we start at $\tau = 64$. Besides, when assign ε_i in line 8, we change $\varepsilon_i \leftarrow \varepsilon\theta/(i+1)^{1+\theta}$ to $\varepsilon_i \leftarrow \varepsilon\theta 3^\theta/(i+3)^{1+\theta}$. Both of them do not change DP requirement and theoretical utility guarantee but leads to better practical performance. Besides, recall in $F_{\text{k-sel}}\mathrm{UserDP}$, to guarantee one-way rank error, we increment k by Δ as (3). Δ is from our theoretical analysis and is not tight. In experiments, we show $\Delta/2$ is enough to guarantee the one-way rank error. Furthermore, as mentioned, to reduce the running time of experiments, we use small R for $F_{\max}$. To further reduce the time, we only use one histogram to answer the count queries used in $F_{\text{k-sel}}\mathrm{UserDP}$.

B. Additional Experiments over Simulated Data

Besides of F_{cnt} and $F_{\max}$, we also implement the experiments on $F_{\text{max-}f}$ and F_0 over Unif. dataset to show the change of error with time. The results are plotted in Figure 6.

[8] For convenience, we allow $x = 0$ for $u \neq \perp$ and we can just set $u = \perp$ in that case and this will not affect the result.

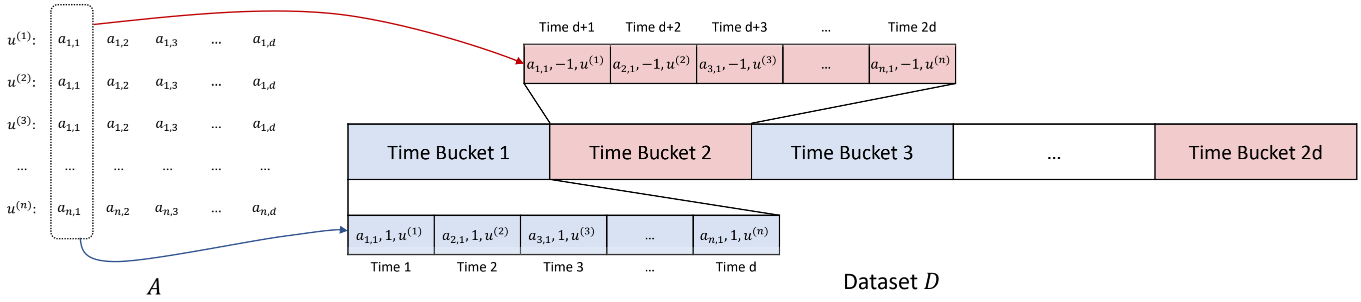
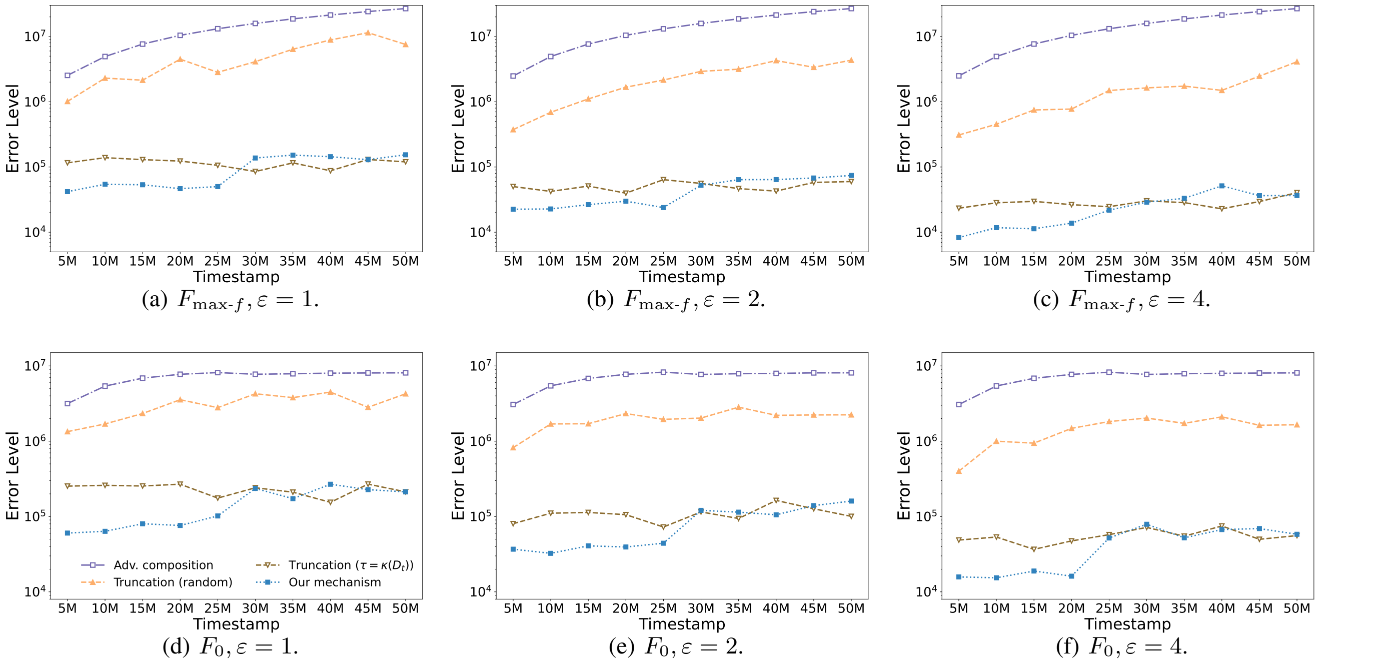


Fig. 5: Reduce high-dimensional sum estimation to count queries in the fully-dynamic setting.



(a) $F_{\text{max-}f}, \varepsilon = 1$.

(b) $F_{\text{max-}f}, \varepsilon = 2$.

(c) $F_{\text{max-}f}, \varepsilon = 4$.

(d) $F_0, \varepsilon = 1$.

(e) $F_0, \varepsilon = 2$.

(f) $F_0, \varepsilon = 4$.

Fig. 6: Error levels vs time of various mechanisms over $F_{\text{max-}f}$ and F_0 on Unif. dataset with $\varepsilon = 1, 2, 4$.