

# Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity

Úlfar Erlingsson\*    Vitaly Feldman\*    Ilya Mironov\*    Ananth Raghunathan\*  
Kunal Talwar\*    Abhradeep Thakurta†

November 2018

## Abstract

Sensitive statistics are often collected across sets of users, with repeated collection of reports done over time. For example, trends in users’ private preferences or software usage may be monitored via such reports. We study the collection of such statistics in the local differential privacy (LDP) model, and describe an algorithm whose privacy cost is polylogarithmic in the number of changes to a user’s value.

More fundamentally—by building on anonymity of the users’ reports—we also demonstrate how the privacy cost of our LDP algorithm can actually be much lower when viewed in the central model of differential privacy. We show, via a new and general privacy amplification technique, that any permutation-invariant algorithm satisfying  $\epsilon$ -local differential privacy will satisfy  $(O(\epsilon\sqrt{\log(1/\delta)/n}), \delta)$ -central differential privacy. By this, we explain how the high noise and  $\sqrt{n}$  overhead of LDP protocols is a consequence of them being significantly more private in the central model. As a practical corollary, our results imply that several LDP-based industrial deployments may have much lower privacy cost than their advertised  $\epsilon$  would indicate—at least if reports are anonymized.

## 1 Introduction

A frequent task in data analysis is the monitoring of the statistical properties of evolving data in a manner that requires repeated computation on the entire evolving dataset. Software applications commonly apply online monitoring, e.g., to establish trends in the software configurations or usage patterns. However, such monitoring may impact the privacy of software users, as it may directly or indirectly expose some of their sensitive attributes (e.g., their location, ethnicity, gender, etc.), either completely or partially. To address this, recent work has proposed a number of mechanisms that provide users with strong privacy-protection guarantees in terms of differential privacy [DMNS06, Dwo06, KLN<sup>+</sup>08] and, specifically, mechanisms that provide local differential privacy (LDP) have been deployed by Google, Apple, and Microsoft [EPK14, App17, DKY17].

The popularity and practical adoption of LDP monitoring mechanisms stems largely from their simple trust model: for any single LDP report that a user contributes about one of their sensitive attributes, the user will benefit from strong differential privacy guarantees even if the user’s report becomes public and all other parties collude against them.

However, this apparent simplicity belies the realities of most monitoring applications. Software monitoring, in particular, near always involves repeated collection of reports over time, either on a regular basis

---

\*Google Research – Brain, {ulfar, vitalyfm, mironov, kunal, pseudorandom}@google.com.

†UC Santa Cruz and Google Research – Brain, aguhatha@ucsc.edu.

or triggered by specific software activity; additionally, not just one, but multiple, software attributes may be monitored, and these attributes may all be correlated, as well as sensitive, and may also change in a correlated fashion. Hence, a user’s actual LDP privacy guarantees may be dramatically lower than they might appear, since LDP guarantees can be exponentially reduced by multiple correlated reports (see Tang et al. [TKB<sup>+</sup>17] for a case study). Furthermore, lower accuracy is achieved by mechanisms that defend against such privacy erosion (e.g., the memoized backstop in Google’s RAPPOR [EPK14]). Thus, to square this circle, and make good privacy/utility tradeoffs, practical deployments of privacy-preserving monitoring rely on additional assumptions—in particular, the assumption that each user’s reports are anonymous at each timestep and unlinkable over time.

In this work, we formalize how the addition of anonymity guarantees can improve differential-privacy protection. Our direct motivation is the *Encode, Shuffle, Analyze* (ESA) architecture and PROCHLO implementation of Bittau et al. [BEM<sup>+</sup>17], which relies on an explicit intermediary that processes LDP reports from users to ensure their anonymity. The ESA architecture is designed to ensure a sufficient number of reports are collected at each timestep so that any one report can “hide in the crowd” and to ensure that those reports are randomly shuffled to eliminate any signal in their order. Furthermore, ESA will also ensure that reports are disassociated and stripped of any identifying metadata (such as originating IP addresses) to prevent the linking of any two reports to a single user, whether over time or within the collection at one timestep. Intuitively, the above steps taken to preserve anonymity will greatly increase the uncertainty in the analysis of users’ reports; however, when introducing ESA, Bittau et al. [BEM<sup>+</sup>17] did not show how that uncertainty could be utilized to provide a tighter upper bound on the worst-case privacy loss.

Improving on this, this paper derives results that account for the benefits of anonymity to provide stronger differential privacy bounds. First, inspired by differential privacy under continual observation, we describe an algorithm for high-accuracy online monitoring of users’ data in the LDP model whose total privacy cost is polylogarithmic in the number of changes to each user’s value. This algorithm shows how LDP guarantees can be established in online monitoring, even when users report repeatedly, over multiple timesteps, and whether they report on the same value, highly-correlated values, or independently-drawn values.

Second, and more fundamentally, we show how—when each report is properly anonymized—any collection of LDP reports (like those at each timestep of our algorithm above) with sufficient privacy ( $\epsilon < 1$ ) is actually subject to much stronger privacy guarantees in the central model of differential privacy. This improved worst-case privacy guarantee is a direct result of the uncertainty induced by anonymity, which can prevent reports from any single user from being singled out or linked together, whether in the set of reports at each timestep, or over time.

## 1.1 Background and related work.

Differential privacy is a quantifiable measure of the stability of the output of a randomized mechanism in the face of changes to its input data—specifically, when the input from any single user is changed. (See Section 2 for a formal definition.)

**Local differential privacy (LDP).** In the local differential privacy model, formally introduced by Kaviswanathan et al. [KLN<sup>+</sup>08], the randomized mechanism’s output is the transcript of the entire interaction between a specific user and a data collector (e.g., a monitoring system). Even if a user arbitrarily changes their privately held data, local differential privacy guarantees will ensure the stability of the distribution of all possible transcripts. *Randomized response*, a disclosure control technique from the

1960s [War65], is a particularly simple technique for designing single-round LDP mechanisms. Due to their attractive trust model, LDP mechanisms have recently received significant industrial adoption for the privacy-preserving collection of heavy hitters [EPK14, App17, DKY17], as well as increased academic attention [BS15, BNST17, QYY<sup>+</sup>16, WBLJ17].

**Anonymous data collection.** As a pragmatic means for reducing privacy risks, reports are typically anonymized and often aggregated in deployments of monitoring by careful operators (e.g., RAPPOR [EPK14])—even though anonymity is no privacy panacea [DSSU17, Dez18].

To guarantee anonymity of reports, multiple mechanisms have been developed. Many, like Tor [DMS04], are based on the ideas of cryptographic onion routing or mixnets, often trading off latency to offer much stronger guarantees [vdHLZZ15, TGL<sup>+</sup>17, LGZ18]. Some, like those of PROCHLO [BEM<sup>+</sup>17], are based on oblivious shuffling, with trusted hardware and attestation used to increase assurance. Others make use of the techniques of secure multi-party computation, and can simultaneously aggregate reports and ensure their anonymity [CGB17, BIK<sup>+</sup>17]. Which of these mechanisms is best used in practice is dictated by what trust model and assumptions apply to any specific deployment.

**Central differential privacy.** The traditional, central model of differential privacy applies to a centrally-held dataset for which privacy is enforced by a trusted *curator* that mediates upon queries posed on the dataset by an untrusted *analyst*—with curators achieving differential privacy by adding uncertainty (e.g., random noise) to the answers for analysts’ queries. For differential privacy, answers to queries need only be stable with respect to changes in the data of a single user (or a few users); these may constitute only a small fraction of the whole, central dataset, which can greatly facilitate the establishment of differential privacy guarantees. Therefore, the central model can offer much better privacy/utility tradeoffs than the LDP setting. (In certain cases, the noise introduced by the curator may even be less than the uncertainty due to population sampling.)

**Longitudinal privacy.** Online monitoring with privacy was formalized by Dwork et al. as the problem of differential privacy under continual observation [DNPR10]. That work proposed a privacy-preserving mechanisms in the central model of differential privacy, later extended and applied by Chan et al. [CSS11] and Jain et al. [JKT12].

Continual observations constitute a powerful attack vector. For example, Calandrino et al. [CKN<sup>+</sup>11] describe an attack on a collaborative-based recommender system via passive measurements that effectively utilizes differencing between a sequence of updates.

In the local model, Google’s RAPPOR [EPK14] proposed a novel *memoization* approach as a backstop against privacy erosion over time: A noisy answer is memorized by the user and repeated in response to the same queries about a data value. To avoid creating a trackable identifier, RAPPOR additionally randomizes those responses, which may only improve privacy. (See Ding et al. [DKY17] for alternative approach to memoization.) Although memoization prevents a single data value from ever being fully exposed, over time the privacy guarantees will weaken if answers are given about correlated data or sequences of data values that change in a non-independent fashion.

More recently, Tang et al. [TKB<sup>+</sup>17] performed a detailed analysis of one real-world randomized response mechanisms and examined its longitudinal privacy implications.

## 1.2 Our contributions

Motivated by the gap in accuracy between central and local differential privacy under continual observations, we describe a general technique for obtaining strong central differential privacy guarantees from (relatively) weak privacy guarantees in the local model. Specifically, our main technical contribution demonstrates that random shuffling of data points ensures that the reports from any LDP protocol will also satisfy central differential privacy at a per-report privacy-cost bound that is a factor  $\sqrt{n}$  lower than the LDP privacy bound established in the local model. Here,  $n$  is the total number of reports, which can reach into the billions in practical deployments; therefore, the privacy amplification can be truly significant.

**Privacy amplification by shuffling.** An immediate corollary of our amplification result is that composing client-side local differential privacy with server-side shuffling allows one to claim strong central differential privacy guarantees *without any explicit server-side noise addition*.

For this corollary to hold, the LDP reports must be amenable to anonymization via shuffling: the reports cannot have any discriminating characteristics and must, in particular, all utilize the same local randomizer (since the distribution of random values may be revealing). However, even if this assumption holds only partially—e.g., due to faults, software mistakes, or adversarial control of some reports—the guarantees degrade gracefully. Each set of  $n'$  users for which the corollary is applicable (e.g., that utilize the same local randomizer) will still be guaranteed a factor  $\sqrt{n'}$  reduction in their worst-case privacy cost in the central model. (See Section 4.1 for a more detailed discussion.)

It is instructive to compare our technique with privacy amplification by subsampling [KLN<sup>+</sup>08]. As in the case of subsampling, we rely on the secrecy of the samples that are used in nested computations. However, unlike subsampling, shuffling, by itself, does not offer any differential privacy guarantees. Yet its combination with a locally differentially private mechanism has an effect that is essentially as strong as that achieved via known applications of subsampling [BST14, ACG<sup>+</sup>16, BBG18]. An important advantage of our reduction over subsampling is that it can include all the data reports (exactly once) and hence need not modify the underlying statistics of the dataset.

In concurrent and independent work, Cheu et al. [CSU<sup>+</sup>18] have also examined an augmented local model of differential privacy that includes an anonymous channel. In this model they demonstrate privacy amplification by the same factor for one-bit randomized response. The analysis in this case relies on a direct estimation of  $(\epsilon, \delta)$ -divergence between two binomial distributions. This simple case was also the starting point of our work but its analysis is unrelated to the general case presented here.

We also remark that another recent privacy amplification technique, via contractive iteration [FMTT18] relies on additional properties of the algorithm and is not directly comparable to results in this work.

**Lower bounds in the local model.** Our amplification result can be viewed, conversely, as giving a lower bound for the local model. Specifically, our reduction means that lower bounds in the central model translate—with a  $\Omega(\sqrt{n})$  *penalty* factor in the privacy parameter—to a large class of local protocols. In particular, this suggests that our results in the local model are near-optimal unless the corresponding results in the central model can be improved.

**LDP monitoring with longitudinal privacy guarantees.** We introduce an online monitoring protocol that guarantees longitudinal privacy to users that report over multiple timesteps, irrespective of whether their reports are about independent or correlated values. By utilizing our protocol, users need not worry

about revealing too much over time, and if they are anonymous, their reports may additionally “hide in the crowd” and benefit by amplification-by-shuffling, at least at each timestep.

As a motivating task, we can consider the collection of global statistics from users’ mobile devices, e.g., about users’ adoption of software apps or the frequency of users’ international long-distance travel. This task is a natural fit for a continual-observations protocol with LDP guarantees—since both software use and travel can be highly privacy sensitive—and can be reduced to collecting a boolean value from each user (e.g., whether they are in a country far from home). However, our protocol can be extended to the collection of multi-valued data or even data strings by building on existing techniques [EPK14, App17, BNST17].

Concretely, we consider the collection of user statistics across  $d$  time periods (e.g., for  $d$  days) with each user *changing* their underlying boolean value at most  $k$  times for some  $k \leq d$ . This is the only assumption we place on the data collection task. For software adoption and international travel, the limit on the number of changes is quite natural. New software is adopted, and then (perhaps) discarded, with only moderate frequency; similarly, speed and distance limit the change rate for even the most ardent travelers. Formally, we show the following: Under the assumption stated above, one can estimate all the  $d$  frequency statistics from  $n$  users with error at most  $O((\log d)^2 k \sqrt{n}/\epsilon)$  in the local differential privacy model.

Motivated by similar issues, in a recent work Joseph et al. [JRUW18] consider the problem of tracking a distribution that changes only a small number of times. Specifically, in their setting each user at each time step  $t$  receives a random and independent sample from a distribution  $P_t$ . It is assumed that  $P_t$  changes at most  $k$  times and they provide utility guarantees that scale logarithmically with the number of time steps. The key difference between this setting and ours is the assumption of independence of each user’s inputs across time steps. Under this assumption even a fixed unbiased coin flip would result in each user receiving values that change in most of the steps. Therefore the actual problems addressed in this work are unrelated to ours and we rely on different algorithmic techniques.

### 1.3 Organization of the paper

Section 2 introduces our notation and recalls the definition of differential privacy. Section 3 provides an algorithm for collecting statistics and proves its accuracy and privacy in the local model under continual observations. Section 4 contains the derivation of our amplification-by-shuffling result. Section 5 concludes with a discussion.

## 2 Technical Preliminaries and Background

**Notation:** For any  $n \in \mathbb{N}$ ,  $[n]$  denotes the set  $\{1, \dots, n\}$ . For a vector  $\vec{x}$ ,  $x[j]$  denotes the value of its  $j$ ’th coordinate. For an indexed sequence of elements  $a_1, a_2, \dots$  and two indices  $i, j$  we denote by  $a_{i:j}$  the subsequence  $a_i, a_{i+1}, \dots, a_j$  (or empty sequence if  $i > j$ ).  $\|\vec{x}\|_1$  denotes  $\sum |x[j]|$ , which is also the Hamming weight of  $\vec{x}$  when  $\vec{x}$  has entries in  $\{-1, 0, 1\}$ . All logarithms are meant to be base  $e$  unless stated otherwise. For a finite set  $X$ , let  $x \stackrel{r}{\leftarrow} X$  denote a sample from  $X$  drawn uniformly at random.

**Differential privacy:** The notion of differential privacy was introduced by Dwork et al. [DMNS06, Dwo06]. We are using the (standard) relaxation of the definition that allows for an additive term  $\delta$ .

**Definition 1** ( $(\epsilon, \delta)$ -DP [DKM<sup>+</sup>06]). *A randomized algorithm  $\mathcal{A}$ :  $\mathcal{D}^n \rightarrow \mathcal{S}$  satisfies  $(\epsilon, \delta)$ -differential privacy (DP) if for all  $S \subset \mathcal{S}$  and for all adjacent  $D, D' \in \mathcal{D}$  it holds that*

$$\Pr[\mathcal{A}(D) \in S] \leq e^\epsilon \Pr[\mathcal{A}(D') \in S] + \delta.$$

The notion of adjacent inputs is application-dependent, and it is typically taken to mean that  $D$  and  $D'$  differ in one of the  $n$  elements (that corresponds to the contributions of a single individual). We will also say that an algorithm satisfies differential privacy *at index* 1 if the guarantees hold only for datasets that differ in the element at index  $i$ . We assume the  $\varepsilon$  parameter to be a small constant, and  $\delta$  is set to be much smaller than  $1/|D|$ . We repeatedly use the (advanced) composition property of differential privacy.

**Theorem 2** (Advanced composition [DRV10, DR14]). *If  $\mathcal{A}_1, \dots, \mathcal{A}_k$  are randomized algorithms satisfying  $(\varepsilon, \delta)$ -DP, then their composition, defined as  $(\mathcal{A}_1(D), \dots, \mathcal{A}_k(D))$  for  $D \in \mathcal{D}$  satisfies  $(\varepsilon', k\delta + \delta')$  differential privacy where  $\varepsilon' = \varepsilon\sqrt{2k \log(1/\delta')} + k\varepsilon(\exp(\varepsilon) - 1)$ . Moreover,  $\mathcal{A}_i$  can be chosen adaptively depending on the outputs of  $\mathcal{A}_1, \dots, \mathcal{A}_{i-1}$ .*

A straightforward corollary implies that for  $\varepsilon, \varepsilon' < 1$ , a  $k$ -fold composition of  $(\varepsilon, \delta)$ -DP algorithms leads to an  $O(\sqrt{k \log(1/\delta)})$  overhead, i.e.,  $\varepsilon' = 2\varepsilon\sqrt{2k \log(1/\delta)}$ .

It will also be convenient to work with the notion of distance between distributions on which  $(\varepsilon, \delta)$ -DP is based more directly. We define it below and describe some of the properties we will use. Given two distributions  $\mu$  and  $\mu'$ , we will say that they are  $(\varepsilon, \delta)$ -DP close, denoted by  $\mu \approx_{(\varepsilon, \delta)} \mu'$ , if for all measurable  $A$ , we have

$$\exp(-\varepsilon)(\mu'(A) - \delta) \leq \mu(A) \leq \exp(\varepsilon)\mu'(A) + \delta.$$

For random variables  $X, X'$ , we write  $X \approx_{(\varepsilon, \delta)} X'$  to mean that their corresponding distributions  $\mu, \mu'$  are  $(\varepsilon, \delta)$ -DP close. We use  $X \cong X'$  to mean that the random variables are identically distributed.

For distributions  $\mu_1, \mu_2$  and  $a \in [0, 1]$ , we write  $a\mu_1 + (1 - a)\mu_2$  to denote the mixture distribution that samples from  $\mu_1$  with probability  $a$  and from  $\mu_2$  with probability  $(1 - a)$ . The following properties are well-known properties of  $(\varepsilon, \delta)$ -DP.

**Lemma 3.** *The notion of  $(\varepsilon, \delta)$ -DP satisfies the following properties:*

**Monotonicity** *Let  $\mu \approx_{(\varepsilon, \delta)} \mu'$ . Then for  $\varepsilon' \geq \varepsilon$  and  $\delta' \geq \delta$ ,  $\mu \approx_{(\varepsilon', \delta')} \mu'$ .*

**Triangle inequality** *Let  $\mu_1 \approx_{(\varepsilon_1, \delta_1)} \mu_2$  and  $\mu_2 \approx_{(\varepsilon_2, \delta_2)} \mu_3$ . Then  $\mu_1 \approx_{(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)} \mu_3$ .*

**Quasi-convexity** *Let  $\mu_1 \approx_{(\varepsilon, \delta)} \mu'_1$  and  $\mu_2 \approx_{(\varepsilon, \delta)} \mu'_2$ , then for any  $a \in [0, 1]$ , it holds that  $(1 - a)\mu_1 + a\mu_2 \approx_{(\varepsilon, \delta)} (1 - a)\mu'_1 + a\mu'_2$ .*

The following lemma is a reformulation of the standard privacy amplification-by-sampling result [KLN<sup>+</sup>08] (with the tighter analysis from [Ull17]).

**Lemma 4** ([KLN<sup>+</sup>08, Ull17]). *Let  $q < \frac{1}{2}$  and let  $\mu_0, \mu_1$  be distributions such that  $\mu_1 \approx_{(\varepsilon, \delta)} \mu_0$ . For  $\mu = (1 - q)\mu_0 + q\mu_1$ , it holds that  $\mu \approx_{(\varepsilon', q\delta)} \mu_0$ , where  $\varepsilon' = \log(q(e^\varepsilon - 1) + 1) \leq q(e^\varepsilon - 1)$ .*

### 3 Locally Private Protocol for Longitudinal Data

Recall the motivation for collecting statistics from user devices with the intent of tracking global trends. We remain in the local differential privacy model, but our protocol addresses the task of collecting reports from users to derive global statistics that are *expected* to change across time. We consider the simplified task of collecting a boolean value from each user, e.g., their device being at an international location, far from the user's home. However, our protocol can be straightforwardly extended to collecting richer data, such as strings, by building on existing techniques [EPK14, App17, BNST17].

In what follows, we consider a natural model of collecting user statistics across time periods. We make two minimal assumptions: we are given the time horizon  $d$ , or the number of time periods (or days) ahead of time, and each user changes their underlying data at most  $k \leq d$  times. The first assumption is mild: a loose upper bound on  $d$  suffices, and the error depends only polylogarithmically on the upper bound. The second assumption can be enforced at the client side to ensure privacy, while suffering some loss in accuracy.

Our approach is inspired by the work on privacy under continual observations by Dwork et al. [DNPR10], who give a (central) DP mechanism to maintain a counter that is incrementally updated in response to certain user-driven events. Its (central) differential privacy is defined in respect to a single increment, the so-called event-level privacy. The naïve solution of applying additive noise to all partial sums introduces error  $\Theta(\sqrt{d})$ , proportional to the square root of the time horizon. The key algorithmic contribution of Dwork et al. is an elegant aggregation scheme that reduces the problem of releasing partial sums to the problem of maintaining a binary tree of counters. By carefully correlating noise across updates, they reduce the error to  $O(\text{polylog } d)$ . (In related work Chan et al. [CSS11] describe a post-processing procedure that guarantees output consistency; Xiao et al. [XWG11] present a conceptually similar algorithm framed as a basis transformation.)

We adapt these techniques to the local setting by pushing the tree-of-counters to the client (Algorithm 1 below). The server aggregates clients’ reports and computes the totals (Algorithm 2).

**Setup.** We more formally define the problem of collecting global statistics based on reports from users’ devices. Given a time horizon  $d$ , we consider a population of  $n$  users reporting a boolean value about their state at each time period  $t \in [d]$ . (Without loss of generality, we assume that  $d$  is a power of 2.) Let  $\vec{\text{st}}_i = [\text{st}_i[1], \dots, \text{st}_i[d]]$  denote the states of the  $i$ ’th user across the  $d$  time periods with at most  $k$  changes. The task of collecting statistics requires the server to compute the sum  $\sum_{i \in [n]} \text{st}_i[t]$  for every time periods  $t$ .

For the reason that will become clear shortly, it is convenient to consider the setup where users report only changes to their state, i.e., a finite derivative of  $\vec{\text{st}}_i$ . Let  $\vec{x}_i = [x_i[1], \dots, x_i[d]] \in \{-1, 0, 1\}^d$  denote the changes in the  $i$ ’th user’s state between consecutive time periods. Our assumption implies that each  $\vec{x}_i$  has at most  $k$  non-zero entries. It holds that  $\text{st}_i[t] = \sum_{\ell \in [t]} x_i[\ell]$  for all  $t \in [d]$ . Let  $f_t = \sum_{i=1}^n x_i[t]$ . For the collection task at hand, it suffices to estimate “running counts” or marginal sums  $\{f_t\}_{t \in [d]}$ .

An online client-side algorithm for reporting statistics runs on each client device and produces an output for each time period. Correspondingly, the online server-side algorithm receives reports from  $n$  clients and outputs estimates for the marginal  $f_t$  at each time period  $t$ .

**Outline.** To demonstrate the key techniques in the design of our algorithm, consider a version of the data collection task with every client’s data known ahead of time. Given  $\vec{x}_i$  for user  $i$ , the client-side algorithm produces (up to)  $d$  reports and the server computes estimates of the marginal sum  $f_t$  for all  $t \in [d]$ . Our algorithm is based on the tree-based aggregation scheme [DNPR10, CSS11] used previously for releasing continual statistics over longitudinal data in the central model. Each client maintains a binary tree over the  $d$  time steps to track the (up to)  $k$  changes of their state. The binary tree ensures that each change affects only  $\log_2 d$  nodes of the tree. We extend the construction of Dwork et al. [DNPR10] in a natural manner to the local model by having each client maintain and report values in this binary tree with sub-sampling as follows.

In the beginning, the client samples uniformly from  $[k]$  the  $\kappa^*$ ’th change they would like to report on. Changes other than the  $\kappa^*$ ’th one are ignored. The client builds a tree with leaves corresponding to an index vector capturing the  $\kappa^*$ ’th change (0 everywhere except  $\pm 1$  at the change). The rest of the nodes

are populated with the sums of their respective subtrees. The client then chooses a random level of the tree to report on. Then, the client runs randomized response on each node of the selected level (with noise determined by  $\varepsilon$ ) and reports the level of the tree along with the randomized response value for each node. In actual implementations and our presentation of the protocol (Algorithm 1) the tree is never explicitly constructed. The state maintained by the client consists of just four integer values ( $\kappa^*$ , the level of the tree, and two counters).

The server accumulates reports from clients to compute an aggregate tree comprising sums of reports from all clients. To compute the marginal estimate  $\tilde{f}_t$  for the time step  $t$ , the server sums up the respective internal nodes whose subtrees form a disjoint cover of the interval  $[1, t]$  and scales it up by the appropriate factor (to compensate for the client-side sampling).

**Notation.** To simplify the description of what follows, for a given  $d$  (that is a power of two), we let  $h \in [\log_2(d) + 1]$  (and variants such as  $h_i$  and  $h^*$ ) denote the  $h$ 'th level of a balanced binary tree with  $2d$  nodes where leaves have level 1. We let  $H(h)$  denote the value  $d/2^{h-1}$ , the number of nodes at level  $h$ , and write  $H_i$  (resp.  $H^*$ ) to denote  $H(h_i)$  (resp.  $H(h^*)$ ). We let  $[h, j]$ , for  $h \in [\log_2(d) + 1]$  and  $j \in [H(h)]$  denote the  $j$ 'th node at level  $h$  of the binary tree  $T$ , and  $T[h, j]$  as the corresponding value stored at the node. Algorithm 1 describes the client-side algorithm to generate differentially-private reports and Algorithm 2 describes the server-side algorithm that collects these reports and estimates marginals  $\tilde{f}_t$  for each time period. Theorems 5 and 6 state the privacy and utility guarantees of the algorithms.

---

**Algorithm 1** ( $\mathcal{A}_{\text{client}}$ ) : Locally differentially private reports.

---

```

1: procedure SETUP( $d, k$ )
   Input: Time bound  $d$ ; bound on the number of non-zero entries in  $\vec{x}$ :  $k \geq \|\vec{x}\|_0$ .
2:   Sample  $\kappa^* \xleftarrow{r} [k]$  and  $h^* \xleftarrow{r} [\log_2(d) + 1]$ 
3:   Initialize counters  $\kappa \leftarrow 0$  and  $c \leftarrow 0$ 

4: procedure UPDATE( $t, x_t, \varepsilon$ )
   Input: Time  $t \leq d$ ,  $x_t \in \{-1, 0, 1\}$ , privacy budget  $\varepsilon$ .
   Effects: Modifies counters  $\kappa$  and  $c$ .
5:   if  $x_t \neq 0$  then
6:      $\kappa \leftarrow \kappa + 1$  ▷  $\kappa$  tracks the number of non-zeroes
7:     if  $\kappa = \kappa^*$  then
8:        $c \leftarrow x_t$ 
9:     if  $t$  is divisible by  $2^{h^*-1}$  then
10:      if  $c = 0$  then
11:         $u \xleftarrow{r} \{-1, 1\}$ 
12:      else
13:         $b \leftarrow 2 \cdot \text{B}\left(\frac{e^{\varepsilon/2}}{1+e^{\varepsilon/2}}\right) - 1$  ▷  $\text{B}(p)$ —Bernoulli r.v. with expectation  $p$ 
14:         $u \leftarrow b \cdot c$ 
15:         $c \leftarrow 0$  ▷  $c$  will never be non-zero again
16:      report  $(h^*, t, u)$ 

```

---

**Theorem 5 (Privacy).** *The sequence of  $d$  outputs of  $\mathcal{A}_{\text{client}}$  (Algorithm 1) satisfies  $\varepsilon$ -local differential privacy.*

---

**Algorithm 2** ( $\mathcal{A}_{\text{server}}$ ): Aggregation of locally differentially private reports (server side)

---

**Input:** For every  $t \in [d]$ , reports  $(h_{i,t}, t, u_{i,t})$  from the  $i$ 'th client running  $\mathcal{A}_{\text{client}}$ . (Some of these reports can be empty.) Privacy budget  $\varepsilon$ , bound  $k$ .

- 1: Create a balanced binary tree  $T_{\text{sum}}$  with  $d$  leaves.
- 2: **for**  $t$  **in**  $[d]$  **do**
- 3:     **for**  $h$  s.t.  $2^{h-1}$  divides  $t$  **do**
- 4:          $T_{\text{sum}}[h, t/2^{h-1}] \leftarrow \sum_{i: h_{i,t}=h} u_{i,t}$               $\triangleright$  Accumulate all reports from level  $h$  and time  $t$
- 5: **for**  $t$  **in**  $[d]$  **do**
- 6:     Initialize  $C \leftarrow \{[1, 1], [1, 2], \dots, [1, t]\}$ .
- 7:     **while**  $h$  and even  $i$  exist s.t.  $[h, i-1], [h, i] \in C$  **do**
- 8:         Remove  $[h, i-1], [h, i]$  from  $C$ .
- 9:         Add  $[h+1, i/2]$  to  $C$ .
- 10:      $\tilde{f}_t \leftarrow \frac{e^{\varepsilon/2}+1}{e^{\varepsilon/2}-1} k (\log_2 d) \cdot \sum_{[h,i] \in C} T_{\text{sum}}[h, i]$
- 11:     **report** privately estimated marginal  $\tilde{f}_t$

---

*Proof.* To show the local differential privacy of the outputs, we consider two aspects of the client's reports: (1) the (randomized) values at the nodes output in Step 16, and (2) the timing of the report. The latter entirely depends on the choice of  $h^*$  which is independent of the client's underlying data record and hence does not affect the privacy analysis. Furthermore,  $\kappa^*$  is also independently sampled and for the rest of the proof, we fix  $\kappa^*$  and  $h^*$  and focus only on the randomized values output in Step 16.

By construction, the client chooses only the  $\kappa^*$ 'th change to report on. This would imply that two inputs would affect at most two nodes at level  $h^*$  with each of the values changing by at most one.

This bound on the sensitivity of the report enables us to use standard arguments for randomized response mechanisms [EPK14] to Steps 13 and 14 to show that the noisy values  $T[h^*, 1], \dots, T[h^*, H^*]$  satisfy  $\varepsilon$ -local differential privacy. □

**Theorem 6** (Utility). *For  $\varepsilon \leq 1$ , with probability at least  $2/3$  over the randomness of  $\mathcal{A}_{\text{client}}$  run on  $n$  data records, the outputs  $\tilde{f}_1, \dots, \tilde{f}_d$  of  $\mathcal{A}_{\text{server}}$  satisfy:*

$$\max_{t \in [d]} |f_t - \tilde{f}_t| = O(c_\varepsilon (\log d)^2 k \sqrt{n}),$$

where  $c_\varepsilon = \frac{e^{\varepsilon/2}+1}{e^{\varepsilon/2}-1}$ .

*Proof.* The leaves of the binary tree with nodes  $T[h, t]$  in Algorithms 1 and 2 comprise events in each of the time periods  $[1, d]$ . The marginal counts  $f_t$  and  $\tilde{f}_t$  comprise the exact (resp., approximate) number of events observed by all clients in the time period  $[1, t]$ .

Consider the set  $C$  constructed in Steps 6–9 of Algorithm 2. We observe that  $C$  satisfies the following properties:

- The set  $C$  is uniquely defined, i.e., it is independent of the order in which pairs of intervals are selected in the **while** loop.
- The size of the set  $|C|$  is at most  $\log_2 d$ .
- The leaf nodes of the subtrees  $[h, i]$  in  $C$  partition  $[1, t]$ .

The marginal counts  $f_t$  totaling events in the interval  $[1, t]$  can be computed by adding the corresponding  $\log_2 d$  nodes in the tree whose subtrees partition the interval  $[1, t]$ .

It follows that largest error in any  $f_t$  is at most  $\log_2 d$  times the largest error in any of the subtree counts. We proceed to bound that error.

For a node  $[h, j]$  in the tree, let  $S[h, j]$  denote the sum  $\sum_i \sum_{t \in [h, j]} x_i[t]$ , i.e., the actual sum of  $x_i[t]$  values in the subtree at  $[h, j]$ , summed over all  $i$ . Let  $z_i^{[h, j]} = \sum_{t \in [h, j]} x_i[t]$  denote the contribution of client  $i$  to  $S[h, j]$ . We will argue that  $|S[h, j] - c_\epsilon k \log_2(d) \cdot T[h, j]|$  is small with high probability, which would then imply the bound on  $|f_t - \tilde{f}_t|$ .

Towards that goal, for a client  $i$ , and node  $[h, j]$ , let  $u_i^{[h, j]}$  be the contribution of client  $i$  to the sum  $T[h, j]$  computed by the server. Thus for any  $h \neq h_i^*$ , this value is zero, and for  $h = h_i^*$ ,  $u_i^{[h, j]}$  is a randomized response as defined in steps 10–14 of Algorithm 1. Clearly  $u_i^{[h, j]} \in \{-1, 0, 1\}$ . We next argue that  $c_\epsilon k \log_2(d) \mathbb{E}[u_i^{[h, j]}]$  is equal to  $z_i^{[h, j]}$ . Indeed, for a specific  $x_i[t]$  to have an effect on  $T[h, j]$ , we should have this be the  $\kappa^*$ 'th non-zero entry in  $x_i$  (which happens with probability  $\frac{1}{k}$ ). Further,  $h$  should equal  $h_i^*$  (which happens with probability  $\frac{1}{\log_2 d}$ ). Conditioned on these two events, the expected value of  $u_i^{[h, j]}$  is determined in step 13 as  $x_i[t]$  multiplied by  $2 \frac{e^\epsilon/2}{e^\epsilon/2+1} - 1 = \frac{1}{c_\epsilon}$ . When one of these two events fails to happen, the expected value of  $u_i^{[h, j]}$  is determined in line 11, and is zero. It follows that the expectation of  $u_i^{[h, j]}$  is as claimed and thus the expectation of  $T[h, j]$  is exactly  $\frac{S[h, j]}{c_\epsilon k \log_2(d)}$ .

To complete the proof, we note that  $T[h, j]$  is the sum of independent (for each  $i$ ) random variables that come from the range  $[-1, 1]$ . Standard Chernoff bounds then imply that  $T[h, j]$  differs from its expectation by at most  $c \sqrt{3 \mathbb{E}[T[h, j]] \log \frac{1}{\beta}}$ , except with probability  $\beta'$ . This expectation is bounded by  $O_\epsilon(n / \log_2 d)$ . Setting  $\beta' = \beta / 2d$ , and taking a union bound, we conclude that except with probability  $\beta$ :

$$\forall [h, j]: |T[h, j] - \mathbb{E}[T[h, j]]| \leq c_\epsilon \sqrt{n \log \frac{2d}{\beta} / \log_2 d}.$$

Scaling by  $c'_\epsilon k \log_2 d$ , and multiplying by  $\log_2 d$  to account for the size of  $C$ , we conclude that except with probability  $\beta$ ,

$$\forall t \in [d]: |f_t - \tilde{f}_t| \leq c_\epsilon k (\log_2 d)^{\frac{3}{2}} \sqrt{n \log \frac{2d}{\beta}}.$$

The claim follows. □

## 4 Privacy Amplification via Shuffling

Local differential privacy holds against a strictly more powerful adversary than central differential privacy, namely one that can examine all communications with a particular user. What can we say about central DP guarantees of a protocol that satisfies  $\epsilon$ -local DP? Since local DP implies central DP, this procedure satisfies  $\epsilon$ -central DP. Moreover, without making additional assumptions, we cannot make any stronger statement than that. Indeed, the central mechanism may release the entire transcripts of its communications with all its users annotated with their identities, leaving their local differential privacy as the only check on information leakage via the mechanism's outputs.

We show that, at a modest cost and with little changes to its data collection pipeline, the analyst can achieve much better central DP guarantees than those suggested by this conservative analysis. Specifically, by shuffling (applying a random permutation), the analyst may *amplify* local differential privacy by a  $\tilde{\Theta}(\sqrt{n})$  factor.

To make the discussion more formal we define the following general form of locally differentially private algorithms that allows picking local randomizers sequentially on the basis of answers from the previous randomizers (Algorithm 3).

---

**Algorithm 3** :  $\mathcal{A}_{\text{local}}$ : Local Responses

---

- Input:** Dataset  $D = x_{1:n}$ . Algorithms  $\mathcal{A}_{\text{ldp}}^{(i)} : \mathcal{S}^{(1)} \times \dots \times \mathcal{S}^{(i-1)} \times \mathcal{D} \rightarrow \mathcal{S}^{(i)}$  for  $i \in [n]$ .
- 1: **for**  $i$  **in**  $[n]$  **do**
  - 2:      $z_i \leftarrow \mathcal{A}_{\text{ldp}}^{(i)}(z_{1:i-1}; x_i)$
  - 3: **return** sequence  $z_{1:n}$
- 

Our main amplification result is for Algorithm 4 that applies the local randomizers *after* randomly permuting the elements of the dataset. For algorithms that use a single fixed local randomizer it does not matter whether the data elements are shuffled before or after the application of the randomizer. We will discuss the corollaries of this result for algorithms that shuffle the responses later in this section.

---

**Algorithm 4** :  $\mathcal{A}_{\text{sl}}$ : Local Responses with Shuffling

---

- Input:** Dataset  $D = x_{1:n}$ . Algorithms  $\mathcal{A}_{\text{ldp}}^{(i)} : \mathcal{S}^{(1)} \times \dots \times \mathcal{S}^{(i-1)} \times \mathcal{D} \rightarrow \mathcal{S}^{(i)}$  for  $i \in [n]$ .
- 1: Let  $\pi$  be a uniformly random permutation of  $[n]$ .
  - 2:  $\pi(D) \leftarrow (x_{\pi(1)}, x_{\pi(2)}, \dots, x_{\pi(n)})$
  - 3: **return**  $\mathcal{A}_{\text{local}}(\pi(D))$
- 

**Theorem 7** (Amplification by shuffling). *For a domain  $\mathcal{D}$ , let  $\mathcal{A}_{\text{ldp}}^{(i)} : \mathcal{S}^{(1)} \times \dots \times \mathcal{S}^{(i-1)} \times \mathcal{D} \rightarrow \mathcal{S}^{(i)}$  for  $i \in [n]$  (where  $\mathcal{S}^{(i)}$  is the range space of  $\mathcal{A}_{\text{ldp}}^{(i)}$ ) be a sequence of algorithms such that  $\mathcal{A}_{\text{ldp}}^{(i)}$  is  $\varepsilon_0$ -differentially private for all values of auxiliary inputs in  $\mathcal{S}^{(1)} \times \dots \times \mathcal{S}^{(i-1)}$ . Let  $\mathcal{A}_{\text{sl}} : \mathcal{D}^n \rightarrow \mathcal{S}^{(1)} \times \dots \times \mathcal{S}^{(n)}$  be the algorithm that given a dataset  $x_{1:n} \in \mathcal{D}^n$ , samples a uniform random permutation  $\pi$  over  $[n]$ , then sequentially computes  $z_i = \mathcal{A}_{\text{ldp}}(z_{1:i-1}, x_{\pi(i)})$  for  $i = 1, 2, \dots, n$  and outputs  $z_{1:n}$  (see Algorithm 4). For any  $n \geq 1000$ ,  $0 < \varepsilon_0 < 1/2$  and  $0 < \delta < 1/100$ ,  $\mathcal{A}_{\text{sl}}$  satisfies  $(\varepsilon, \delta)$ -differential privacy in the central model, where  $\varepsilon = 12\varepsilon_0 \sqrt{\frac{\log(1/\delta)}{n}}$ .*

We remark that while we state our amplification result for local randomizers operating on a single data element, the result extends immediately to arbitrary  $\varepsilon_0$ -DP algorithms that operate on disjoint batches of data elements.

A natural approach to proving this result is to use privacy amplification via subsampling (Lemma 4). At step  $i$  in the algorithm, conditioned on the values of  $\pi(1), \dots, \pi(i-1)$ , the random variable  $\pi(i)$  is uniform over the remaining  $(n-i+1)$  indices. Thus the  $i$ 'th step will be  $O((e^{\varepsilon_0} - 1)/(n-i+1))$ -DP. The obvious issue with this argument is that it gives very little amplification for values of  $i$  that are close to  $n$ , falling short of our main goal. It also unclear how to formalize the intuition behind this argument.

Instead our approach crucially relies on a reduction to analysis of the algorithm  $\mathcal{A}_{\text{swap}}$  that swaps the first element in the dataset with a uniformly sampled element in the dataset before applying the local randomizers

(Algorithm 5). We show that  $\mathcal{A}_{\text{swap}}$  has the desired privacy parameters for the first element (that is, satisfies the guarantees of differential privacy only for pairs of datasets that differ in the first element). We then show that for every index  $i^*$ ,  $\mathcal{A}_{\text{sl}}$  can be decomposed into a random permutation that maps element  $i^*$  to be the first element followed by  $\mathcal{A}_{\text{swap}}$ . This implies that the algorithm  $\mathcal{A}_{\text{sl}}$  will satisfy differential privacy at  $i^*$ .

To argue about the privacy properties of the  $\mathcal{A}_{\text{swap}}$  we decompose it into a sequence of algorithms, each producing one output symbol (given the dataset and all the previous outputs). It is not hard to see that the output distribution of the  $i$ 'th algorithm is a mixture  $\mu = (1 - p)\mu_0 + p\mu_1$ , where  $\mu_0$  does not depend on  $x_1$  (the first element of the dataset) and  $\mu_1$  is the output distribution of the  $i$ 'th local randomizer applied to  $x_1$ . We then demonstrate that the probability  $p$  is upper bounded by  $e^{2\varepsilon_0}/n$ . Hence, using amplification by subsampling, we obtain that  $\mu$  is close to  $\mu_0$ . The distribution  $\mu_0$  does not depend on  $x_1$  and therefore, by the same argument,  $\mu_0$  (and hence  $\mu$ ) is close to the output distribution of the  $i$ 'th algorithm on any dataset  $D'$  that differs from  $D$  only in the first element. Applying advanced composition to the sequence of algorithms gives the claim.

We now provide the full details of the proof starting with the description and analysis of  $\mathcal{A}_{\text{swap}}$ .

---

**Algorithm 5** :  $\mathcal{A}_{\text{swap}}$ : Local responses with one swap

---

**Input:** Dataset  $D = x_{1:n}$ . Algorithms  $\mathcal{A}_{\text{ldp}}^{(i)} : \mathcal{S}^{(1)} \times \dots \times \mathcal{S}^{(i-1)} \times \mathcal{D} \rightarrow \mathcal{S}^{(i)}$  for  $i \in [n]$ .

- 1: Sample  $I \xleftarrow{r} [n]$
  - 2: Let  $\sigma_I(D) \leftarrow (x_I, x_2, \dots, x_{I-1}, x_1, x_{I+1}, \dots, x_n)$
  - 3: **return**  $\mathcal{A}_{\text{local}}(\sigma_I(D))$
- 

**Theorem 8** (Amplification by swapping). *For a domain  $\mathcal{D}$ , let  $\mathcal{A}_{\text{ldp}}^{(i)} : \mathcal{S}^{(1)} \times \dots \times \mathcal{S}^{(i-1)} \times \mathcal{D} \rightarrow \mathcal{S}^{(i)}$  for  $i \in [n]$  (where  $\mathcal{S}^{(i)}$  is the range space of  $\mathcal{A}_{\text{ldp}}^{(i)}$ ) be a sequence of algorithms such that  $\mathcal{A}_{\text{ldp}}^{(i)}$  is  $\varepsilon_0$ -differentially private for all values of auxiliary inputs in  $\mathcal{S}^{(1)} \times \dots \times \mathcal{S}^{(i-1)}$ . Let  $\mathcal{A}_{\text{swap}} : \mathcal{D}^n \rightarrow \mathcal{S}^{(1)} \times \dots \times \mathcal{S}^{(n)}$  be the algorithm that given a dataset  $D = x_{1:n} \in \mathcal{D}^n$ , samples a uniform index  $I \in [n]$ , swaps element 1 with element  $I$  and then applies the local randomizers to the resulting dataset sequentially (see Algorithm 5). For any  $n \geq 1000$ ,  $0 < \varepsilon_0 < 1/2$  and  $0 < \delta < 1/100$ ,  $\mathcal{A}_{\text{swap}}$  satisfies  $(\varepsilon, \delta)$ -differential privacy at index 1 in the central model, where  $\varepsilon \leq 12\varepsilon_0 \sqrt{\frac{\log(1/\delta)}{n}}$ .*

*Proof.* The algorithm  $\mathcal{A}_{\text{swap}}$  defines a joint distribution between  $I$  and the corresponding output sequence of  $\mathcal{A}_{\text{swap}}$ , which we denote by  $Z_1, Z_2, \dots, Z_n$ . We first observe that  $Z_{1:n}$  can be seen as the output of a sequence of  $n$  algorithms with conditionally independent randomness:  $\mathcal{B}^{(i)} : \mathcal{S}^{(1)} \times \dots \times \mathcal{S}^{(i-1)} \times \mathcal{D}^n \rightarrow \mathcal{S}^{(i)}$  for  $i \in [n]$ . On input  $s_{1:i-1}$  and  $D$ ,  $\mathcal{B}^{(i)}$  produces a random sample from the distribution of  $Z_i$  conditioned on  $Z_{1:i-1} = s_{1:i-1}$ . The outputs of  $\mathcal{B}^{(1)}, \dots, \mathcal{B}^{(i-1)}$  are given as the input to  $\mathcal{B}^{(i)}$ . By definition, this ensures that random bits used by  $\mathcal{B}^{(i)}$  are independent of those used by  $\mathcal{B}^{(1)}, \dots, \mathcal{B}^{(i-1)}$  conditioned on the previous outputs. Therefore in order to upper bound the privacy parameters of  $\mathcal{A}_{\text{swap}}$  we can analyze the privacy parameters of  $\mathcal{B}^{(1)}, \dots, \mathcal{B}^{(n)}$  and apply the advanced composition theorem for differential privacy (Theorem 2).

Next we observe that, by the definition of  $\mathcal{A}_{\text{swap}}$ , conditioned on the value of  $I$ ,  $Z_i$  is independent of  $Z_{1:i-1}$ . In particular, for  $i \geq 2$ ,  $\mathcal{B}^{(i)}$  can equivalently be implemented as follows. First, sample an index  $T$  from the distribution of  $I$  conditioned on  $Z_{1:i-1} = s_{1:i-1}$ . Then, if  $T = i$  output  $\mathcal{A}_{\text{ldp}}^{(i)}(s_{1:i-1}; x_1)$ . Otherwise, output  $\mathcal{A}_{\text{ldp}}^{(i)}(s_{1:i-1}, x_i)$ . To implement  $\mathcal{B}^{(1)}$  we sample  $T$  uniformly from  $[n]$  and then output  $\mathcal{A}_{\text{ldp}}^{(1)}(x_T)$ .

We now prove that for every  $i \in [n]$ ,  $\mathcal{B}^{(i)}$  is  $(2e^{\varepsilon_0}(e^{\varepsilon_0} - 1)/n, 0)$ -differentially private at index 1. Let  $D = x_{1:n}$  and  $D' = (x'_1, x_{2:n})$  be two datasets that differ in the first element. Let  $s_{1:i-1}$  denote the input to  $\mathcal{B}^{(i)}$ . We denote by  $\mu$  the probability distribution of  $\mathcal{B}^{(i)}(s_{1:i-1}, D)$ , denote by  $\mu_0$  (or  $\mu_1$ ) the probability distribution of  $\mathcal{B}^{(i)}(s_{1:i-1}, D)$  conditioned on  $T \neq i$  ( $T = i$ , respectively) and by  $p$  the probability that  $T = i$  (where  $T$  is sampled from the distribution of  $I$  conditioned on  $Z_{1:i-1} = s_{1:i-1}$  as described above). We also denote by  $\mu', \mu'_0, \mu'_1$  and  $p'$  the corresponding quantities when  $\mathcal{B}^{(i)}$  is run on  $D'$ . By the definition,  $\mu = (1 - p)\mu_0 + p\mu_1$  and  $\mu' = (1 - p')\mu'_0 + p'\mu'_1$ .

For  $i = 1$ ,  $T$  is uniform over  $[n]$  and hence  $p = p' = 1/n$ . Further,  $\mu_0$  is equal to  $\mu'_0$  (since both are equal to the output distribution of  $\mathcal{A}_{\text{ldp}}^{(1)}(x_T)$  conditioned on  $T \neq 1$ ). By  $\varepsilon_0$ -local differential privacy of  $\mathcal{A}_{\text{ldp}}^{(1)}$  and quasi-convexity of  $(\varepsilon, \delta)$ -DP we obtain that  $\mu_0 \approx_{(\varepsilon_0, 0)} \mu_1$ . Therefore, privacy amplification by subsampling (Lemma 4) implies that  $\mu_0 \approx_{((e^{\varepsilon_0} - 1)/n, 0)} \mu$ . Similarly, we obtain that  $\mu'_0 \approx_{((e^{\varepsilon_0} - 1)/n, 0)} \mu'$  and therefore, by the triangle inequality,  $\mu \approx_{(2(e^{\varepsilon_0} - 1)/n, 0)} \mu'$ . In other words,  $\mathcal{B}^{(1)}$  is  $(2(e^{\varepsilon_0} - 1)/n, 0)$ -differentially private at index 1.

For  $i \geq 2$ , we again observe that  $\mu_0 = \mu'_0$  since in both cases the output is generated by  $\mathcal{A}_{\text{ldp}}^{(i)}(s_{1:i-1}; x_i)$ . Similarly,  $\varepsilon_0$ -local differential privacy of  $\mathcal{A}_{\text{ldp}}^{(i)}$  implies that  $\mu_0 \approx_{(\varepsilon_0, 0)} \mu_1$  and  $\mu'_0 \approx_{(\varepsilon_0, 0)} \mu'_1$ .

We now claim that  $p \leq e^{2\varepsilon_0}/n$ . To see this, we first observe that the condition  $Z_{1:i-1} = s_{1:i-1}$  is an event defined over the output space of  $\mathcal{A}_{\text{local}}$ . Conditioning on  $T = i$  reduces  $\mathcal{A}_{\text{swap}}$  to running  $\mathcal{A}_{\text{local}}$  on  $\sigma_i(D)$ . Note that for  $j \neq i$ ,  $\sigma_i(D)$  differs from  $\sigma_j(D)$  in at most two positions. Therefore, by  $\varepsilon_0$ -differential privacy of  $\mathcal{A}_{\text{local}}$  and group privacy (e.g. [DR14]), we obtain that

$$\frac{\Pr[Z_{1:i-1} = s_{1:i-1} \mid T = i]}{\Pr[Z_{1:i-1} = s_{1:i-1} \mid T = j]} \leq e^{2\varepsilon_0}.$$

By quasi-convexity of  $(\varepsilon, \delta)$ -DP we obtain that

$$\frac{\Pr[Z_{1:i-1} = s_{1:i-1} \mid T = i]}{\Pr[Z_{1:i-1} = s_{1:i-1}]} \leq e^{2\varepsilon_0}.$$

This immediately implies our claim since

$$\Pr[T = i \mid Z_{1:i-1} = s_{1:i-1}] = \frac{\Pr[Z_{1:i-1} = s_{1:i-1} \mid T = i] \cdot \Pr[T = i]}{\Pr[Z_{1:i-1} = s_{1:i-1}]} \leq \frac{1}{n} \cdot e^{2\varepsilon_0}.$$

Privacy amplification by sampling implies that  $\mu_0 \approx_{(e^{2\varepsilon_0}(e^{\varepsilon_0} - 1)/n, 0)} \mu$ . Applying the same argument to  $D'$  and using the triangle inequality we get that  $\mu \approx_{(2e^{2\varepsilon_0}(e^{\varepsilon_0} - 1)/n, 0)} \mu'$ .

Applying the advanced composition theorem for differential privacy with  $\varepsilon_1 = 2e^{2\varepsilon_0}(e^{\varepsilon_0} - 1)/n$  and  $n$  steps we get that  $\mathcal{A}_{\text{swap}}$  satisfies  $(\varepsilon, \delta)$ -DP at index 1 for

$$\varepsilon \leq \varepsilon_1 \sqrt{2n \log(1/\delta)} + n\varepsilon_1(e^{\varepsilon_1} - 1).$$

Note that for  $\varepsilon_0 \leq 1/2$  we get that  $\varepsilon_1 \leq 8\varepsilon_0/n$  and the first term

$$\varepsilon_1 \sqrt{2n \log(1/\delta)} \leq 8\varepsilon_0 \sqrt{\frac{2 \log(1/\delta)}{n}}.$$

Using the fact that  $n \geq 1000$  we have that  $\varepsilon_1 \leq 1/250$ . This implies that  $e^{\varepsilon_1} - 1 \leq \frac{65}{64}\varepsilon_1$  and using  $n \geq 1000$  and  $\delta \leq 1/100$ , we get the following bound on the second term

$$n\varepsilon_1(e^{\varepsilon_1} - 1) \leq \frac{65}{64}n\varepsilon_1^2 \leq \frac{65\varepsilon_0^2}{n} \leq \frac{2}{3}\varepsilon_0 \sqrt{\frac{\log(1/\delta)}{n}}.$$

Combining these terms gives the claimed bound.  $\square$

Finally, we describe a reduction of the analysis of  $\mathcal{A}_{\text{sl}}$  to the analysis of  $\mathcal{A}_{\text{swap}}$ .

*Proof of Theorem 7.* Let  $D$  and  $D'$  be two datasets of length  $n$  that differ at some index  $i^* \in [n]$ . The algorithm  $\mathcal{A}_{\text{sl}}$  can be seen as follows. We first pick a random one-to-one mapping  $\pi^*$  from  $\{2, \dots, n\} \rightarrow [n] \setminus \{i^*\}$  and let

$$\pi^*(D) = (x_{i^*}, x_{\pi^*(2)}, \dots, x_{\pi^*(n)}).$$

Namely, we move  $x_{i^*}$  to the first place and apply a random permutation to the remaining elements. In the second step we apply  $\mathcal{A}_{\text{swap}}$  to  $\pi^*(D)$ . It is easy to see that for a randomly and uniformly chosen  $\pi^*$  and uniformly chosen  $I \in [n]$  the distribution of  $\sigma_I(\pi^*(D))$  is exactly a random and uniform permutation of elements in  $D$ .

For a fixed mapping  $\pi^*$ , the datasets  $\pi^*(D)$  and  $\pi^*(D')$  differ only in the element with index 1. Therefore  $\mathcal{A}_{\text{swap}}(\pi^*(D)) \cong_{(\varepsilon, \delta)} \mathcal{A}_{\text{swap}}(\pi^*(D'))$  for  $\varepsilon$  and  $\delta$  given in Theorem 8. Using the quasi-convexity of  $(\varepsilon, \delta)$ -DP over a random choice of  $\pi^*$  we obtain that  $\mathcal{A}_{\text{sl}}(D) \cong_{(\varepsilon, \delta)} \mathcal{A}_{\text{sl}}(D')$ .  $\square$

## 4.1 Shuffling after local randomization

The proof of Theorem 7 relies crucially on shuffling the data elements before applying the local randomizers. However implementing such an algorithm in a distributed system requires trusting a remote shuffler with sensitive user data, thus negating the key advantage of the LDP model. Conversely, even if shuffling is performed on a set of already-randomized LDP responses, no additional privacy guarantees will be achieved if some attribute of each report (e.g., the choice of a randomizer) can reveal the identity of the reporting user.

Fortunately, it is possible design software systems where reports are randomized before shuffling and in which the reports coming from large groups of users are indistinguishable, e.g., because they apply the same local randomizer. In such constructions, the privacy of each user's report still have its privacy amplified, by a factor proportional to the square root of the cardinality of indistinguishable reports. This follows immediately from the fact that shuffling the responses from the same local randomizers is equivalent to first shuffling the data points and then applying the local randomizers.

We make this claim formal in the following corollary.

**Corollary 9.** For a domain  $\mathcal{D}$ , let  $\mathcal{A}_{\text{ldp}}^{(i)}: \mathcal{S}^{(1)} \times \dots \times \mathcal{S}^{(i-1)} \times \mathcal{D} \rightarrow \mathcal{S}^{(i)}$  for  $i \in [n]$  (where  $\mathcal{S}^{(i)}$  is the range space of  $\mathcal{A}_{\text{ldp}}^{(i)}$ ) be a sequence of algorithms such that  $\mathcal{A}_{\text{ldp}}^{(i)}$  is  $\varepsilon_0$ -differentially private for all values of auxiliary inputs in  $\mathcal{S}^{(1)} \times \dots \times \mathcal{S}^{(i-1)}$ . Let  $\mathcal{A}_{\text{post}}: \mathcal{D}^n \rightarrow \mathcal{S}^{(1)} \times \dots \times \mathcal{S}^{(n)}$  be the algorithm that given a dataset  $D \in \mathcal{D}^n$ , computes  $z_{1:n} = \mathcal{A}_{\text{local}}(D)$ , samples a random and uniform permutation and outputs  $z_{\pi(1)}, \dots, z_{\pi(n)}$ . Let  $S \subseteq [n]$  be any set of indices such that for all  $i, j \in S$ ,  $\mathcal{A}_{\text{ldp}}^{(i)} \equiv \mathcal{A}_{\text{ldp}}^{(j)}$ . Then for  $|S| \geq 1000$ ,  $0 < \varepsilon_0 < 1/2$  and  $0 < \delta < 1/100$  and every  $i \in S$ ,  $\mathcal{A}_{\text{post}}$  satisfies  $(\varepsilon, \delta)$ -differential privacy at index  $i$  in the central model, where  $\varepsilon = 12\varepsilon_0 \sqrt{\frac{\log(1/\delta)}{|S|}}$ .

We note that for the conclusion of this corollary to hold it is not necessary to randomly permute all the  $n$  randomized responses. It suffices to shuffle the elements of  $S$ . We also clarify that for  $i < j$ , by  $\mathcal{A}_{\text{ldp}}^{(i)} \equiv \mathcal{A}_{\text{ldp}}^{(j)}$  we mean that for all sequences  $z_{1:j-1}$  and  $x \in \mathcal{D}$ , the output distributions of  $\mathcal{A}_{\text{ldp}}^{(i)}(z_{1:i-1}, x)$  and  $\mathcal{A}_{\text{ldp}}^{(j)}(z_{1:j-1}, x)$  are identical (and, in particular, the output distribution  $\mathcal{A}_{\text{ldp}}^{(j)}$  does not depend on  $z_{i:j-1}$ ).

## 4.2 Lower Bound for Local DP Protocols

The results of this section give us a natural and powerful way to prove lower bounds for protocols in the local differential privacy model. We can apply Theorem 7 in the reverse direction to roughly state that for any

given problem, lower bounds on the error of  $\Omega(\alpha/\varepsilon)$  (for some term  $\alpha$  that might depend on the parameters of the system) of an  $\varepsilon$ -centrally differentially private protocol translate to a  $\Omega(\alpha\sqrt{n}/\varepsilon)$  lower bound on the error of any  $\varepsilon$ -locally differentially private protocol of the kind that our techniques apply to.

As an exercise, a lower bound of  $\Omega(\sqrt{k} \text{polylog}(d)/\varepsilon)$  for the problem of collecting frequency statistics from users across time in the central DP framework with privacy guarantee  $\varepsilon$  directly implies that the result in Theorem 6 is tight. We note here that the results of Dwork et al. [DNPR10] do show a lower bound of  $\Omega(\log(d)/\varepsilon)$  for the setting when  $k = 1$  in the central DP framework. This strongly suggests that our bounds might be tight, but we cannot immediately use this lower bound as it is stated only for the pure  $\varepsilon$ -differential privacy regime. It is an open problem to extend these results to the approximate differential privacy regime.

## 5 Discussion and Future Work

Our amplification-by-shuffling result is encouraging, as it demonstrates that the formal guarantees of differential privacy can encompass intuitive privacy-enhancing techniques, such as the addition of anonymity, which are typically part of existing, best-practice privacy processes. By accounting for the uncertainty induced by anonymity, in the central differential privacy model the worst-case, per-user bound on privacy cost can be dramatically lowered.

Our result implies that industrial adoption of LDP-based mechanisms may have offered much stronger privacy guarantees than previously accounted for, since anonymization of telemetry reports is standard privacy practice in industry. This is gratifying, since the direct motivation for our work was to better understand the guarantees offered by one industrial privacy-protection mechanism: the *Encode, Shuffle, Analyze* (ESA) architecture and PROCHLO implementation of Bittau et al. [BEM<sup>+</sup>17].

However, there still remain gaps between our analysis and proposed practical, real-world mechanisms, such as those of the ESA architecture. In particular, our formalization assumes the user population to be static, which undoubtedly it is not. On a related note, our analysis assumes that (most) all users send reports at each timestep and ignores the privacy implications of timing or traffic channels, although both must be considered, since reports may be triggered by privacy-sensitive events on users' devices, and it is infeasible to send all possible reports at each timestep. The ESA architecture addresses traffic channels using large-scale batching and randomized thresholding of reports, with elision, but any benefits from that mitigation are not included in our analysis.

Finally, even though it is a key aspect of the ESA architecture, our analysis does not consider how users may fragment their sensitive information and at any timestep send multiple LDP reports, one for each fragment, knowing that each will be anonymous and unlinkable. The splitting of user data into carefully constructed fragments to increase users' privacy has been explored for specific applications, e.g., by Fanti et al. [FPE16] which fragmented users' string values into overlapping  $n$ -grams, to bound sensitivity while enabling an aggregator to reconstruct popular user strings. Clearly, such fragmentation should be able to offer significantly improved privacy/utility tradeoffs, at least in the central model. However, in both the local and central models of differential privacy, the privacy implications of users' sending LDP reports about disjoint, overlapping, or equivalent fragments of their sensitive information remain to be formally understood, in general.

## References

- [ACG<sup>+</sup>16] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proc. of the 2016 ACM*

- SIGSAC Conf. on Computer and Communications Security (CCS)*, pages 308–318, 2016.
- [App17] Apple’s Differential Privacy Team. Learning with privacy at scale. *Apple Machine Learning Journal*, 1(9), December 2017.
- [BBG18] Borja Balle, Gilles Barthe, and Marco Gaboardi. Privacy amplification by subsampling: Tight analyses via couplings and divergences. *CoRR*, abs/1807.01647, 2018.
- [BEM<sup>+</sup>17] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proc. of the 26th ACM Symp. on Operating Systems Principles (SOSP)*, 2017.
- [BIK<sup>+</sup>17] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proc. of the 2017 ACM Conf. on Computer and Communications Security (CCS)*, pages 1175–1191, 2017.
- [BNST17] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Thakurta. Practical locally private heavy hitters. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2288–2296, 2017.
- [BS15] Raef Bassily and Adam Smith. Local, private, efficient protocols for succinct histograms. In *Proc. of the Forty-Seventh Annual ACM Symp. on Theory of Computing (STOC)*, pages 127–135, 2015.
- [BST14] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Proc. of the 2014 IEEE 55th Annual Symp. on Foundations of Computer Science (FOCS)*, pages 464–473, 2014.
- [CGB17] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *Proc. of the 14th USENIX Conf. on Networked Systems Design and Implementation (NSDI)*, pages 259–282, 2017.
- [CKN<sup>+</sup>11] Joseph A. Calandrino, Ann Kilzer, Arvind Narayanan, Edward W. Felten, and Vitaly Shmatikov. “You might also like:” Privacy risks of collaborative filtering. In *32nd IEEE Symp. on Security and Privacy*, pages 231–246, 2011.
- [CSS11] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *ACM Trans. on Information Systems Security*, 14(3):26:1–26:24, November 2011.
- [CSU<sup>+</sup>18] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via mixnets. *CoRR*, abs/1808.01394, 2018.
- [Dez18] Ryan Dezember. Your smartphone’s location data is worth big money to Wall Street. *The Wall Street Journal*, November 2018. <https://www.wsj.com/articles/your-smartphones-location-data-is-worth-big-money-to-wall-street-1541131260>.

- [DKM<sup>+</sup>06] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology—EUROCRYPT*, pages 486–503, 2006.
- [DKY17] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3574–3583, 2017.
- [DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proc. of the Third Conf. on Theory of Cryptography (TCC)*, pages 265–284, 2006.
- [DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *13th USENIX Security Symp.*, pages 21–21, 2004.
- [DNPR10] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *Proc. of the Forty-Second ACM Symp. on Theory of Computing (STOC)*, pages 715–724, 2010.
- [DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.
- [DRV10] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *Proc. of the 51st Annual IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 51–60, 2010.
- [DSSU17] Cynthia Dwork, Adam Smith, Thomas Steinke, and Jonathan Ullman. Exposed! A survey of attacks on private data. *Annual Review of Statistics and Its Application*, 4(1):61–84, 2017.
- [Dwo06] Cynthia Dwork. Differential privacy. In *Proc. of the 33rd International Conf. on Automata, Languages and Programming—Part II (ICALP)*, pages 1–12, 2006.
- [EPK14] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *Proc. of the 2014 ACM Conf. on Computer and Communications Security (CCS)*, pages 1054–1067, 2014.
- [FMTT18] Vitaly Feldman, Ilya Mironov, Kunal Talwar, and Abhradeep Thakurta. Privacy amplification by iteration. In *59th Annual IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 521–532, 2018.
- [FPE16] Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson. Building a RAPPOR with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proc. on Privacy Enhancing Technologies (PoPETS)*, 2016(3):41–61, 2016.
- [JKT12] Prateek Jain, Pravesh Kothari, and Abhradeep Thakurta. Differentially private online learning. In *Proc. of the 25th Annual Conf. on Learning Theory (COLT)*, volume 23, pages 24.1–24.34, 2012.
- [JR UW18] Matthew Joseph, Aaron Roth, Jonathan Ullman, and Bo Waggoner. Local differential privacy for evolving data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

- [KLN<sup>+</sup>08] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. What can we learn privately? In *49th Annual IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 531–540, 2008.
- [LGZ18] David Lazar, Yossi Gilad, and Nikolai Zeldovich. Karaoke: Distributed private messaging immune to passive traffic analysis. In *13th USENIX Symp. on Operating Systems Design and Implementation (OSDI)*, pages 711–725, 2018.
- [QYY<sup>+</sup>16] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. Heavy hitter estimation over set-valued data with local differential privacy. In *Proc. of the 2016 ACM Conf. on Computer and Communications Security (CCS)*, pages 192–203, 2016.
- [TGL<sup>+</sup>17] Nirvan Tyagi, Yossi Gilad, Derek Leung, Matei Zaharia, and Nikolai Zeldovich. Stadium: A distributed metadata-private messaging system. In *Proc. of the 26th ACM Symp. on Operating Systems Principles (SOSP)*, pages 423–440, 2017.
- [TKB<sup>+</sup>17] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and XiaoFeng Wang. Privacy loss in Apple’s implementation of differential privacy on macOS 10.12. *CoRR*, abs/1709.02753, 2017.
- [Ull17] Jonathan Ullman. CS7880. Rigorous approaches to data privacy, Spring 2017. <http://www.ccs.neu.edu/home/jullman/PrivacyS17/HW1sol.pdf>, 2017.
- [vdHLZZ15] Jelle van den Hooff, David Lazar, Matei Zaharia, and Nikolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *Proc. of the 25th ACM Symp. on Operating Systems Principles (SOSP)*, pages 137–152, 2015.
- [War65] Stanley L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *J. of the American Statistical Association*, 60(309):63–69, 1965.
- [WBLJ17] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. Locally differentially private protocols for frequency estimation. In *26th USENIX Security Symp.*, pages 729–745, 2017.
- [XWG11] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. Differential privacy via wavelet transforms. *IEEE Trans. on Knowledge and Data Engineering*, 23(8):1200–1214, August 2011.