Privacy Amplification via Random Check-Ins

Borja Balle* Peter Kairouz† H. Brendan McMahan† Om Thakkar†
Abhradeep Thakurta‡

July 30, 2020

Abstract

Differentially Private Stochastic Gradient Descent (DP-SGD) forms a fundamental building block in many applications for learning over sensitive data. Two standard approaches, privacy amplification by subsampling, and privacy amplification by shuffling, permit adding lower noise in DP-SGD than via naïve schemes. A key assumption in both these approaches is that the elements in the data set can be uniformly sampled, or be uniformly permuted — constraints that may become prohibitive when the data is processed in a decentralized or distributed fashion. In this paper, we focus on conducting iterative methods like DP-SGD in the setting of federated learning (FL) wherein the data is distributed among many devices (clients). Our main contribution is the *random check-in* distributed protocol, which crucially relies only on randomized participation decisions made locally and independently by each client. It has privacy/accuracy trade-offs similar to privacy amplification by subsampling/shuffling. However, our method does not require server-initiated communication, or even knowledge of the population size. To our knowledge, this is the first privacy amplification tailored for a distributed learning framework, and it may have broader applicability beyond FL. Along the way, we improve the privacy guarantees of amplification by shuffling and show that, in practical regimes, this improvement allows for similar privacy and utility using data from an order of magnitude fewer users.

1 Introduction

Modern mobile devices and web services benefit significantly from large-scale machine learning, often involving training on user (client) data. When such data is sensitive, steps must be taken to ensure privacy, and a formal guarantee of differential privacy (DP) [15, 16] is the gold standard. For this reason, DP has been adopted by companies including Google [9, 18, 20], Apple [2], Microsoft [13], and LinkedIn [31], as well as the US Census Bureau [26].

Other privacy-enhancing techniques can be combined with DP to obtain additional benefits. In particular, cross-device federated learning (FL) [27] allows model training while keeping client data decentralized (each participating device keeps its own local dataset, and only sends model updates or gradients to the coordinating server). However, existing approaches to combining FL and DP make a number of assumptions that are unrealistic in real-world FL deployments such as [10]. To highlight these challenges, we must first review the state-of-the-art in centralized DP training, where differentially private stochastic gradient descent (DP-SGD) [1, 8, 34] is ubiquitous. It achieves optimal error for convex problems [8], and can also be applied to non-convex problems, including deep learning, where the privacy amplification offered by randomly subsampling data to form batches is critical for obtaining meaningful DP guarantees [1, 5, 8, 25, 37].

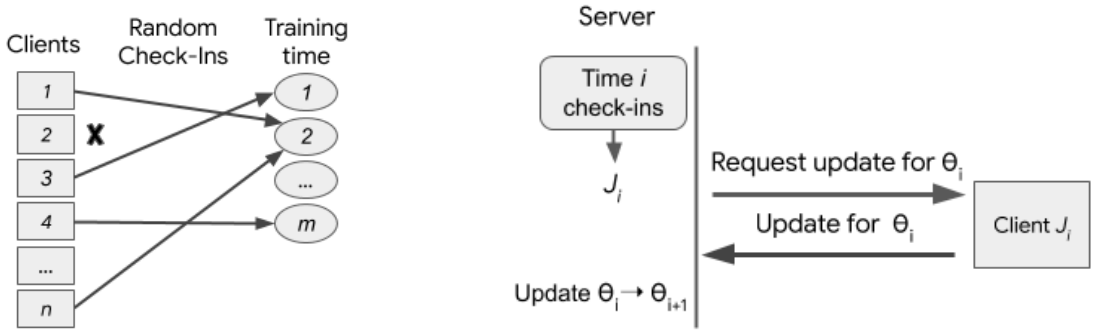
Attempts to combine FL and the above lines of DP research have been made previously; notably, [3, 28] extended the approach of [1] to FL and user-level DP. However, these works and others in the area sidestep a critical issue: the DP guarantees require very specific sampling or shuffling schemes assuming, for example, that each client participates in each iteration with a fixed probability. While possible in theory, such schemes are incompatible with the practical constraints and design goals of cross-device FL protocols [10]; to quote [23], a comprehensive recent FL survey, *"such*

*DeepMind. bballe@google.com
†Google. {kairouz, mcmahan, omthkkr}@google.com
‡Google Research - Brain. {athakurta}@google.com

a sampling procedure is nearly impossible in practice."[1] The fundamental challenge is that clients decide when they will be available for training and when they will check in to the server, and by design the server cannot index specific clients. In fact, it may not even know the size of the participating population.

Our work targets these challenges. Our primary goal is to provide strong central DP guarantees for the final model released by FL-like protocols, under the assumption of a trusted[2] orchestrating server. This is accomplished by building upon recent work on amplification by shuffling [6, 12, 18, 19, 22] and combining it with new analysis techniques targeting FL-specific challenges (e.g., client-initiated communications, non-addressable global population, and constrained client availability).

We propose the first privacy amplification analysis specifically tailored for distributed learning frameworks. At the heart of our result is a novel technique, called *random check-in*, that relies only on randomness independently generated by each individual client participating in the training procedure. We show that distributed learning protocols based on random check-ins can attain privacy gains similar to privacy amplification by subsampling/shuffling (see Table 1 for a comparison), while requiring minimal coordination from the server. While we restrict our exposition to distributed DP-SGD within the FL framework for clarity and concreteness (see Figure 1 for a schematic of one of our protocols), we note that the techniques used in our analyses are broadly applicable to any distributed iterative method and might be of interest in other applications[3].



(a) (b)

Figure 1: A schematic of the Random Check-ins protocol with Fixed Windows (Section 3.1) for Distributed DP-SGD (Algorithm 1). For the central DP guarantee, all solid arrows represent communication over privileged channels not accessible to any external adversary. (a) n clients performing random check-ins with a fixed window of m time steps. 'X' denotes that the client randomly chose to abstain from participating. (b) A time step at the server, where for training time $i \in [m]$, the server selects a client j from those who checked-in for time i, requests an update for model θ_i, and then updates the model to θ_{i+1} (or gradient accumulator if using minibatches).

Contributions The main contributions of this paper can be summarized as follows:

1. We propose *random check-ins*, the first privacy amplification technique for distributed systems with minimal server-side overhead. We also instantiate three distributed learning protocols that use random check-ins, each addressing different natural constraints that arise in applications.

2. We provide formal privacy guarantees for our protocols, and show that random check-ins attain similar rates of privacy amplification as subsampling and shuffling while reducing the need for server-side orchestration. We also

[1] In cross-silo FL applications [23], an enumerated set of addressable institutions or data-silos participate in FL, and so explicit server-mediated subsampling or shuffling using existing techniques may be feasible.

[2] Notably, our guarantees are obtained by amplifying the privacy provided by local DP randomizers; we treat this use of local DP as an implementation detail in accomplishing the primary goal of central DP. As a byproduct, our approach offers (weaker) local DP guarantees even in the presence of an untrusted server.

[3] In particular, the Federated Averaging [27] algorithm, which computes an update based on multiple local SGD steps rather than a single gradient, can immediately be plugged into our framework.

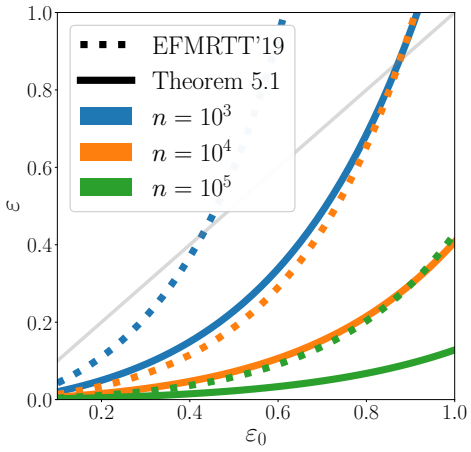


Figure 2: Values of ε (for $\delta = 10^{-6}$) after amplification by shuffling of ε_0-DP local randomizers obtained from: Theorem 5.1 (solid lines) and [19, Theorem 7] (dotted lines). The grey line represents the threshold of no amplification ($\varepsilon = \varepsilon_0$); after crossing the line amplification bounds become vacuous. Observe that our bounds with $n = 10^3$ and $n = 10^4$ are similar to the bounds from [19] with $n = 10^4$ and $n = 10^5$, respectively.

provide utility guarantees for one of our protocols in the convex case that match the optimal privacy/accuracy trade-offs for DP-SGD in the central setting [7].

3. As a byproduct of our analysis, we improve privacy amplification by shuffling [19] on two fronts. For the case of ε_0-DP local randomizers, we improve the dependency of the final central DP ε by a factor of $O(e^{0.5\varepsilon_0})$. Figure 2 provides a numerical comparison of the bound from [19] with our bound; for typical parameter values this improvement allows us to provide similar privacy guarantees while reducing the number of required users by one order of magnitude. We also extend the analysis to the case of $(\varepsilon_0, \delta_0)$-DP local randomizers, including Gaussian randomizers that are widely used in practice.

Related work Our work considers the paradigm of federated learning as a stylized example throughout the paper. We refer the reader to [23] for an excellent overview of the state-of-the-art in federated learning, along with a suite of interesting open problems. There is a rich literature on studying differentially private ERM via DP-SGD [1, 8, 30, 34, 35, 39]. However, constraints such as limited availability in distributed settings restrict direct applications of existing techniques. There is also a growing line of works on privacy amplification by shuffling [4, 6, 9, 12, 18, 19, 22] that focus on various ways in which protocols can be designed using trusted shuffling primitives. Lastly, privacy amplification by iteration [21] is another recent advancement that can be applied in an iterative distributed setting, but it is limited to convex objectives.

2 Background and Problem Formulation

Differential Privacy To formally introduce our notion of privacy, we first define neighboring data sets. We will refer to a pair of data sets $D, D' \in \mathcal{D}^n$ as neighbors if D' can be obtained from D by modifying one sample $d_i \in D$ for some $i \in [n]$.

Definition 2.1 (Differential privacy [15, 16])**.** *A randomized algorithm $\mathcal{A} : \mathcal{D}^n \to \mathcal{S}$ is (ε, δ)-differentially private if, for any pair of neighboring data sets $D, D' \in \mathcal{D}^n$, and for all events $S \subseteq \mathcal{S}$ in the output range of $\mathcal{A}$, we have* $\mathbf{Pr}[\mathcal{A}(D) \in S] \leq e^{\varepsilon} \cdot \mathbf{Pr}[\mathcal{A}(D') \in S] + \delta$.

For meaningful *central DP* guarantees (i.e., when $n > 1$), ε is assumed to be a small constant, and $\delta \ll 1/n$. The case $\delta = 0$ is often referred to as *pure DP* (in which case, we just write ε-DP). We shall also use the term *approximate DP* when $\delta > 0$.

Adaptive differentially private mechanisms occur naturally when constructing complex DP algorithms, for e.g., DP-SGD. In addition to the dataset D, adaptive mechanisms also receive as input the output of other differentially private mechanisms. Formally, we say that an adaptive mechanism $\mathcal{A} : \mathcal{S}' \times \mathcal{D}^n \to \mathcal{S}$ is (ε, δ)-DP if the mechanism $\mathcal{A}(s', \bullet)$ is (ε, δ)-DP for every $s' \in \mathcal{S}'$.

Specializing Definition 2.1 to the case $n = 1$ gives what we call a *local randomizer*, which provides a *local DP* guarantee. Local randomizers are the typical building blocks of local DP protocols where individuals privatize their data before sending it to an aggregator for analysis [25].

Problem Setup The distributed learning setup we consider in this paper involves n clients, where each client $j \in [n]$ holds a data record[4] $d_j \in \mathcal{D}$, $j \in [n]$, forming a distributed data set $D = (d_1, \ldots, d_n)$. We assume a coordinating server wants to train the parameters $\theta \in \Theta$ of a model by using the dataset D to perform stochastic gradient descent steps according to some loss function $\ell : \mathcal{D} \times \Theta \to \mathbb{R}_+$. The server's goal is to protect the privacy of all the individuals in D by providing strong DP guarantees against an adversary that can observe the final trained model as well as all the intermediate model parameters. We assume the server is trusted, all devices adhere to the prescribed protocol (i.e., there are no malicious users), and all server-client communications are privileged (i.e., they cannot be detected or eavesdropped by an external adversary).

The server starts with model parameters θ_1 and over a sequence of m time slots produces a sequence of model parameters $\theta_2, \ldots, \theta_{m+1}$. Our random check-ins technique allows clients to independently decide when to offer their contributions for a model update. If and when a client's contribution is accepted by the server, she uses the current parameters θ and her data d to send a privatized gradient of the form $\mathcal{A}_{ldp}(\nabla_\theta \ell(d, \theta))$ to the server, where $\mathcal{A}_{ldp}$ is a DP local randomizer (e.g., performing gradient clipping and adding Gaussian noise [1]).

Our results consider three different setups inspired by practical applications [10]: (1) The server uses $m \ll n$ time slots, where at most one user's update is used in each slot, for a total of m/b minibatch SGD iterations. It is assumed all n users are available for the duration of the protocol, but the server does not have enough bandwidth to process updates from every user (Section 3.1); (2) The server uses $m \approx n/b$ time slots, and all n users are available for the duration of the protocol (Section 4.1). On average, b users contribute updates to each time slot, and so, we take m minibatch SGD steps; (3) As with (2), but each user is only available during a small window of time relative to the duration of the protocol (Section 4.2).

3 Distributed Learning with Random Check-Ins

This section presents the *random check-ins* technique for privacy amplification in the context of distributed learning. We formally define the random check-ins procedure, describe a fully distributed DP-SGD protocol with random check-ins, and analyze its privacy and utility guarantees.

3.1 Random Check-Ins with a Fixed Window

Consider the distributed learning setup described in Section 2 where each client is willing to participate in the training procedure as long as their data remains private. To boost the privacy guarantees provided by the local randomizer $\mathcal{A}_{ldp}$, we will let clients volunteer their updates at a *random* time slot of their choosing. This randomization has a similar effect on the uncertainty about the use of an individual's data on a particular update as the one provided by uniform subsampling or shuffling. We formalize this concept using the notion of random check-in, which can be informally expressed as a client in a distributed iterative learning framework randomizing their instant of participation, and determining with some probability whether to participate in the process at all.

Definition 3.1 (Random check-in)**.** *Let $\mathcal{A}$ be a distributed learning protocol with m check-in time slots. For a set $R_j \subseteq [m]$ and probability $p_j \in [0, 1]$, client j performs an (R_j, p_j)-check-in in the protocol if with probability p_j she requests the server to participate in $\mathcal{A}$ at time step $I \xleftarrow{u.a.r.} R_j$, and otherwise abstains from participating. If $p_j = 1$, we alternatively denote it as an R_j-check-in.*

Our first distributed learning protocol based on random check-ins is presented in Algorithm 1. Client j independently decides in which of the possible time steps (if any) she is willing to participate by performing an (R_j, p_j)-check-in. We set $R_j = [m]$ for all $j \in [n]$, and assume[5] all n clients are available throughout the duration of the

[4]Each client is identified as a user. In a general FL setting, each d_j can correspond to a local data set [10].

[5]We make this assumption only for utility; the privacy guarantees are independent of this assumption.

Server-side protocol:

parameters: local randomizer $\mathcal{A}_{ldp}$, number of steps m

Initialize model $\theta_1 \in \Theta$
Initialize gradient accumulator $g_1 \leftarrow 0^p$
for $i \in [m]$ **do**
  $S_i \leftarrow \{j : \text{User}(j) \text{ checked-in at time } i\}$
  if S_i is empty **then**
    $\tilde{g}_i \leftarrow \mathcal{A}_{ldp}(0^p)$ // Dummy gradient
  else
    Sample $J_i \xleftarrow{u.a.r.} S_i$
    Request User(J_i) for update to model θ_i
    Receive $\tilde{g}_i$ from User(J_i)
  $(\theta_{i+1}, g_{i+1}) \leftarrow \text{ModelUpdate}(\theta_i, g_i + \tilde{g}_i, i)$
  Output θ_{i+1}

Client-side protocol for User(j)**:**

parameters: check-in window R_j, check-in probability p_j, loss function ℓ, local randomizer $\mathcal{A}_{ldp}$
private inputs: datapoint $d_j \in \mathcal{D}$

if a p_j-biased coin returns heads **then**
  Check-in with the server at time $I \xleftarrow{u.a.r.} R_j$
  if receive request for update to model θ_I **then**
    $\tilde{g}_I \leftarrow \mathcal{A}_{ldp}(\nabla_\theta \ell(d_j, \theta_I))$
    Send $\tilde{g}_I$ to server

ModelUpdate(θ, g, i)**:**

parameters: batch size b, learning rate η

if $i \mod b = 0$ **then**
  return $\left(\theta - \frac{\eta}{b} g, 0^p\right)$ // Gradient descent step
else
  return (θ, g) // Skip update

Algorithm 1: $\mathcal{A}_{fix}$ – Distributed DP-SGD with random check-ins (fixed window).

protocol. On the server side, at each time step $i \in [m]$, a random client J_i among all the ones that checked-in at time i is queried: this client receives the current model θ_i, locally computes a gradient update $\nabla_\theta \ell(d_{J_i}, \theta_i)$ using their data d_{J_i}, and returns to the server a privatized version of the gradient obtained using a local randomizer $\mathcal{A}_{ldp}$. Clients checked-in at time i that are not selected do not participate in the training procedure. If at time i no client is available, the server adds a "dummy" gradient to update the model.

3.2 Privacy Analysis

From a privacy standpoint, Algorithm 1 shares an important pattern with DP-SGD: each model update uses noisy gradients obtained from a random subset of the population. However, there exist two key factors that make the privacy analysis of our protocol more challenging than the existing analysis based on subsampling and shuffling. First, unlike in the case of uniform sampling where the randomness in each update is independent, here there is a correlation induced by the fact that clients that check-in into one step cannot check-in into a different step. Second, in shuffling there is also a similar correlation between updates, but there we can ensure each update uses the same number of datapoints, while here the server does not control the number of clients that will check-in into each individual step. Nonetheless, the following result shows that random check-ins provides a factor of privacy amplification comparable to these techniques.

Theorem 3.2 (Amplification via random check-ins into a fixed window)**.** *Suppose $\mathcal{A}_{ldp}$ is an ε_0-DP local randomizer. Let $\mathcal{A}_{fix} : \mathcal{D}^n \to \Theta^m$ be the protocol from Algorithm 1 with check-in probability $p_j = p_0$ and check-in window $R_j = [m]$ for each client $j \in [n]$. For any $\delta \in (0, 1)$, algorithm $\mathcal{A}_{fix}$ is (ε, δ)-DP with $\varepsilon = p_0(e^{\varepsilon_0} - 1)\sqrt{\frac{2e^{\varepsilon_0}\log(1/\delta)}{m}} + \frac{p_0^2 e^{\varepsilon_0}(e^{\varepsilon_0}-1)^2}{2m}$. In particular, for $\varepsilon_0 \leq 1$ and $\delta \leq 1/100$, we get $\varepsilon \leq 7p_0\varepsilon_0\sqrt{\frac{\log(1/\delta)}{m}}$. Furthermore, if $\mathcal{A}_{ldp}$ is $(\varepsilon_0, \delta_0)$-DP with $\delta_0 \leq \frac{(1-e^{-\varepsilon_0})\delta_1}{4e^{\varepsilon_0}\left(2+\frac{\ln(2/\delta_1)}{\ln(1/(1-e^{-5\varepsilon_0}))}\right)}$, then $\mathcal{A}_{fix}$ is (ε', δ')-DP with $\varepsilon' = \frac{p_0^2 e^{8\varepsilon_0}(e^{8\varepsilon_0}-1)^2}{2m} + p_0(e^{8\varepsilon_0} - 1)\sqrt{\frac{2e^{8\varepsilon_0}\log(1/\delta)}{m}}$ and $\delta' = \delta + m(e^{\varepsilon'} + 1)\delta_1$.*

Remark 1 We can always increase privacy in the above statement by decreasing p_0. However, this will also increase the number of dummy updates, which suggests choosing $p_0 = \Theta(m/n)$. With such a choice, we obtain an amplification factor of $\sqrt{m}/n$. Critically, however, exact knowledge of the population size is *not* required to have a precise DP guarantee above.

Remark 2 At first look, the amplification factor of $\sqrt{m}/n$ may appear stronger than the typical $1/\sqrt{n}$ factor obtained via uniform subsampling/shuffling. Note that one run of our technique provides m updates (as opposed to n updates via the other methods). When the server has sufficient capacity, we can set $m = n$ to recover a $1/\sqrt{n}$ amplification. The primary advantage of our approach is that we can benefit from amplification in terms of n even if only a much smaller number of updates are actually processed. We can also extend our approach to recover the $1/\sqrt{n}$ amplification even when the server is rate limited ($p_0 = m/n$), by repeating the protocol $\mathcal{A}_{fix}$ adaptively n/m times to get Corollary 3.3 from Theorem 3.2 and applying advanced composition for DP [17].

Corollary 3.3. *For algorithm $\mathcal{A}_{fix} : \mathcal{D}^n \to \Theta^m$ described in Theorem 3.2, suppose $\mathcal{A}_{ldp}$ is an ε_0-DP local randomizer s.t. $\varepsilon_0 \leq \frac{2\log\left(n/8\sqrt{m}\right)}{3}$, and $n \geq (e^{\varepsilon_0}-1)^2 e^{\varepsilon_0}\sqrt{m}\log(1/\beta)$. Setting $p_0 = \frac{m}{n}$, and running $\frac{n}{m}$ repetitions of $\mathcal{A}_{fix}$ results in a total of n updates, along with an overall central (ε, δ)-DP guarantee with $\varepsilon = \tilde{O}\left(e^{1.5\varepsilon_0}/\sqrt{n}\right)$ and $\delta \in (0,1)$, where $\tilde{O}(\cdot)$ hides polylog factors in $1/\beta$ and $1/\delta$.*

Comparison to Existing Privacy Amplification Techniques Table 1 provides a comparison of the bound in Corollary 3.3 to other existing techniques, for performing one epoch of training (i.e., use one update from each client). Note that for this comparison, we assume that $\varepsilon_0 > 1$, since for $\varepsilon_0 \leq 1$ all the shown amplification bounds can be written as $O\left(\varepsilon_0/\sqrt{n}\right)$. "None" denotes a naïve scheme (with no privacy amplification) where each client is used exactly once in any arbitrary order. Also, note that in general, the guarantees via privacy amplification by subsampling/shuffling apply only under the assumption of complete participation availability[6] of each client. Thus, they define the upper limits of achieving such amplifications. Also, note that even though the bound in Corollary 3.3 appears better than amplification via shuffling, our technique does include dummy updates which do not occur in the other techniques. For linear optimization problems, it is easy to see that our technique will add a factor of e more noise as compared to the other two privacy amplification techniques at the same privacy level.

Source of Privacy Amplification	ε **for Central DP**
None [14, 33]	ε_0
Uniform subsampling [1, 8, 25]	$O\left(e^{\varepsilon_0}/\sqrt{n}\right)$
Shuffling [19]	$O\left(e^{3\varepsilon_0}/\sqrt{n}\right)$
Shuffling (Theorem 5.1, This paper)	$O\left(e^{2.5\varepsilon_0}/\sqrt{n}\right)$
Random check-ins with a fixed window (Theorem 3.2, This paper)	$O\left(e^{1.5\varepsilon_0}/\sqrt{n}\right)$

Table 1: Comparison with existing amplification techniques for a data set of size n, running n iterations of DP-SGD with batch size of 1 and ε_0-DP local randomizers. For ease of exposition, we assume $(e^{\varepsilon_0}-1) \approx \varepsilon_0$, and hide polylog factors in n and $1/\delta$.

Proof Sketch for Theorem 3.2 Here, we provide a summary of the argument[7] used to prove Theorem 3.2 in the case $\delta_0 = 0$. First, note that it is enough to argue about the privacy of the sequence of noisy gradients $\tilde{g}_{1:m}$ by post-processing. Also, the role each client plays in the protocol is symmetric, so w.l.o.g. we can consider two datasets D, D' differing in the first position. Next, we imagine that the last $n-1$ clients make the same random check-in choices in $\mathcal{A}_{fix}(D)$ and $\mathcal{A}_{fix}(D')$. Letting c_i denote the number of such clients that check-in into step $i \in [n]$, we model these choices by a pair of sequences $F = (\bar{d}_{1:m}, w_{1:m})$ where $\bar{d}_i \in \mathcal{D} \cup \{\perp\}$ is the data record of an arbitrary client who checked-in into step i (with $\perp$ representing a "dummy" data record if no client checked-in), and $w_i = 1/(c_i+1)$ represents the probability that client 1's data will be picked to participate in the protocol at step i if she checks-in in step i. Conditioned on these choices, the noisy gradients $\tilde{g}_{1:m}$ produced by $\mathcal{A}_{fix}(D)$ can be obtained by: (1) initializing a dataset $\tilde{D} = (\bar{d}_{1:m})$; (2) sampling $I \xleftarrow{u.a.r.} [m]$, and replacing $\bar{d}_I$ with d_1 in $\tilde{D}$ w.p. $p_0 w_I$; (3) producing the outputs $\tilde{g}_{1:m}$ by applying a sequence of ε_0-DP adaptive local randomizers to $\tilde{D} = (\tilde{d}_{1:m})$ by setting $\tilde{g}_i = \mathcal{A}^{(i)}(\tilde{d}_i, \tilde{g}_{1:i-1})$. Here each of the $\mathcal{A}^{(i)}$ uses all past gradients to compute the model θ_i and return $\tilde{g}_i = \mathcal{A}_{ldp}(\nabla_\theta \ell(\tilde{d}_i, \theta_i))$.

[6] By a complete participation availability for a client, we mean that the client should be available to participate when requested by the server for any time step(s) of training.

[7] Full proofs for every result in the paper are provided in Appendix A.

The final step involves a variant of the amplification by swapping technique [19, Theorem 8] which we call amplification by probable replacement. The key idea is to reformulate the composition of the $\mathcal{A}^{(i)}$ applied to the random dataset $\tilde{D}$, to a composition of mechanisms of the form $\tilde{g}_i = \mathcal{B}^{(i)}(d_1, F, \tilde{g}_{1:i-1})$. Mechanism $\mathcal{B}^{(i)}$ uses the gradient history to compute $q_i = \mathbf{Pr}[I = i|\tilde{g}_{1:i-1}]$ and returns $\mathcal{A}^{(i)}(d_1, \tilde{g}_{1:i-1})$ with probability $p_0 w_i q_i$, and $\mathcal{A}^{(i)}(\bar{d}_i, \tilde{g}_{1:i-1})$ otherwise. Note that before the process begins, we have $\mathbf{Pr}[I = i] = 1/m$ for every i; our analysis shows that the posterior probability after observing the first $i-1$ gradients is not too far from the prior: $q_i \leq \frac{e^{\varepsilon_0}}{me^{\varepsilon_0} - (e^{\varepsilon_0}-1)(i-1)}$. The desired bound is then obtained by using the overlapping mixtures technique [5] to show that $\mathcal{B}^{(i)}$ is $\log(1 + p_0 q_i (e^{\varepsilon_0} - 1))$-DP with respect to changes on d_1, and heterogeneous advanced composition [24] to compute the final ε of composing the $\mathcal{B}^{(i)}$ adaptively.

3.3 Utility Analysis

Proposition 3.4 (Dummy updates in random check-ins with a fixed window)**.** *For algorithm $\mathcal{A}_{fix} : \mathcal{D}^n \to \Theta^m$ described in Theorem 3.2, the expected number of dummy updates performed by the server is at most $\left(m\left(1 - \frac{p_0}{m}\right)^n\right)$. For $c > 0$ if $p_0 = \frac{cm}{n}$, we get at most $\frac{m}{e^c}$ expected dummy updates.*

Utility for Convex ERMs We now instantiate our amplification theorem (Theorem 3.2) in the context of differentially private empirical risk minimization (ERM). For convex ERMs, we will show that DP-SGD [1, 8, 34] in conjunction with our privacy amplification theorem (Theorem 3.2) is capable of achieving the optimal privacy/accuracy trade-offs [8].

Theorem 3.5 (Utility guarantee)**.** *Suppose in algorithm $\mathcal{A}_{fix} : \mathcal{D}^n \to \Theta^m$ described in Theorem 3.2 the loss $\ell : \mathcal{D} \times \Theta \to \mathbb{R}_+$ is L-Lipschitz and convex in its second parameter and the model space Θ has dimension p and diameter R, i.e., $\sup_{\theta,\theta' \in \Theta} \|\theta - \theta'\| \leq R$. Furthermore, let $\mathscr{D}$ be a distribution on $\mathcal{D}$, define the population risk $\mathscr{L}(\mathscr{D}; \theta) = \mathbb{E}_{d \sim \mathscr{D}}[\ell(d; \theta)]$, and let $\theta^* = \arg\min_{\theta \in \Theta} \mathscr{L}(\mathscr{D}; \theta)$. If $\mathcal{A}_{ldp}$ is a local randomizer that adds Gaussian noise with variance σ^2, and the learning rate for a model update at step $i \in [m]$ is set to be $\eta_i = \frac{R\left(1 - 2e^{-np_0/m}\right)}{\sqrt{(p\sigma^2 + L^2)i}}$, then the output θ_m of $\mathcal{A}_{fix}(D)$ on a dataset D containing n i.i.d. samples from $\mathscr{D}$ satisfies*[8]

$$\mathbb{E}_{D,\theta_m}\left[\mathscr{L}(\mathscr{D}; \theta_m)\right] - \mathscr{L}(\mathscr{D}; \theta^*) = \widetilde{O}\left(\frac{\sqrt{p\sigma^2 + L^2} \cdot R}{\left(1 - 2e^{-np_0/m}\right)\sqrt{m}}\right).$$

Remark 3 Note that as $m \to n$, it is easy to see for $p_0 = \Omega\left(\frac{m}{n}\right)$ that Theorem 3.5 achieves the optimal population risk trade-off [7, 8].

4 Variations: Thrifty Updates, and Sliding Windows

This section presents two variants of the main protocol from the previous section. The first variant makes a better use of the updates provided by each user at the expense of a small increase in the privacy cost. The second variant allows users to check-in into a sliding window to model the case where different users might be available during different time windows.

4.1 Leveraging Updates from Multiple Users

Now, we present a variant of Algorithm 1 which, at the expense of a mild increase in the privacy cost, removes the need for dummy updates, and for discarding all but one of the clients checked-in at every time step. The server-side protocol of this version is given in Algorithm 2 (the client-side protocol is identical as Algorithm 1). Note that here, if no client checked-in at some step $i \in [m]$, the server simply skips the update. Furthermore, if at some step multiple

[8] Here, $\widetilde{O}$ hides a polylog factor in m.

Server-side protocol:
parameters: total update steps m

Initialize model $\theta_1 \in \mathbb{R}^p$
for $i \in [m]$ **do**
  $S_i \leftarrow \{j : \text{User}(j) \text{ checks-in for index } i\}$
  if S_i is empty **then**
    $\theta_{i+1} \leftarrow \theta_i$
  else
    $\tilde{g}_i \leftarrow 0$
    for $j \in S_i$ **do**
      Request User(j) for update to model θ_i
      Receive $\tilde{g}_{i,j}$ from User(j)
      $\tilde{g}_i \leftarrow \tilde{g}_i + \tilde{g}_{i,j}$
    $\theta_{i+1} \leftarrow \theta_i - \frac{\eta}{|S_i|}\tilde{g}_i$
  Output θ_{i+1}

Algorithm 2: $\mathcal{A}_{avg}$ - Distributed DP-SGD with random check-ins (averaged updates).

clients checked in, the server requests gradients from all the clients, and performs a model update using the average of the submitted noisy gradients.

These changes have the obvious advantage of reducing the noise in the model coming from dummy updates, and increasing the algorithm's data efficiency by utilizing gradients provided by all available clients. The corresponding privacy analysis becomes more challenging because (1) the adversary gains information about the time steps where no clients checked-in, and (2) the server uses the potentially non-private count $|S_i|$ of clients checked-in at time i when performing the model update. Nonetheless, we show that the privacy guarantees of Algorithm 2 are similar to those of Algorithm 1 with an additional $O(e^{3\varepsilon_0/2})$ factor, and the restriction of non-collusion among the participating clients. For simplicity, we only analyze the case where each client has check-in probability $p_j = 1$.

Theorem 4.1 (Amplification via random check-ins with averaged updates)**.** *Suppose $\mathcal{A}_{ldp}$ is an ε_0-DP local randomizer. Let $\mathcal{A}_{avg} : \mathcal{D}^n \to \Theta^m$ be the protocol from Algorithm 2 performing m averaged model updates with check-in probability $p_j = 1$ and check-in window $R_j = [m]$ for each user $j \in [n]$. Algorithm $\mathcal{A}_{avg}$ is $(\varepsilon, \delta + \delta_2)$-DP with*

$$\varepsilon = \frac{e^{4\varepsilon_0}(e^{\varepsilon_0}-1)^2\varepsilon_1^2}{2} + e^{2\varepsilon_0}(e^{\varepsilon_0}-1)\varepsilon_1\sqrt{2\log(1/\delta)} \ ,$$

where $\varepsilon_1 = \sqrt{\frac{1}{n}+\frac{1}{m}} + \sqrt{\frac{\log(1/\delta_2)}{n}}$. In particular, for $\varepsilon_0 \le 1$ we get $\varepsilon = O(\varepsilon_0/\sqrt{m})$. Furthermore, if $\mathcal{A}_{ldp}$ is $(\varepsilon_0, \delta_0)$-DP with $\delta_0 \le \frac{(1-e^{-\varepsilon_0})\delta_1}{4e^{\varepsilon_0}\left(2+\frac{\ln(2/\delta_1)}{\ln(1/(1-e^{-5\varepsilon_0}))}\right)}$, then $\mathcal{A}_{avg}$ is (ε', δ')-DP with $\varepsilon' = \frac{e^{32\varepsilon_0}(e^{8\varepsilon_0}-1)^2\varepsilon_1^2}{2} + e^{16\varepsilon_0}(e^{8\varepsilon_0}-1)\varepsilon_1\sqrt{2\log(1/\delta)}$ and $\delta' = \delta + \delta_2 + m(e^{\varepsilon'}+1)\delta_1$.

Next, we provide a utility guarantee for $\mathcal{A}_{avg}$ in terms of the excess population risk for convex ERMs (similar to Theorem 3.5).

Theorem 4.2 (Utility guarantee)**.** *Suppose in algorithm $\mathcal{A}_{avg} : \mathcal{D}^n \to \Theta^m$ described in Theorem 4.1 the loss $\ell : \mathcal{D} \times \Theta \to \mathbb{R}_+$ is L-Lipschitz and convex in its second parameter and the model space Θ has dimension p and diameter R, i.e., $\sup_{\theta,\theta' \in \Theta} \|\theta - \theta'\| \le R$. Furthermore, let $\mathscr{D}$ be a distribution on $\mathcal{D}$, define the population risk $\mathscr{L}(\mathscr{D};\theta) = \mathbb{E}_{d\sim\mathscr{D}}[\ell(d;\theta)]$, and let $\theta^* = \arg\min_{\theta\in\Theta}\mathscr{L}(\mathscr{D};\theta)$. If $\mathcal{A}_{ldp}$ is a local randomizer that adds Gaussian noise with variance σ^2, and the learning rate for a model update at step $i \in [m]$ is set to be $\eta_i = \frac{R\sqrt{n}}{\sqrt{(mp\sigma^2+nL^2)i}}$, then*

the output θ_m of $\mathcal{A}_{avg}(D)$ on a dataset D containing n i.i.d. samples from $\mathscr{D}$ satisfies

$$\mathbb{E}_{D,\theta_m}\left[\mathscr{L}(\mathscr{D};\theta_m)\right] - \mathscr{L}(\mathscr{D};\theta^*) = \widetilde{O}\left(\frac{R\sqrt{mp\sigma^2 + nL^2}}{\sqrt{mn}}\right).$$

Furthermore, if the loss ℓ is β-smooth in its second parameter and we set the step-size $\eta_i = \frac{R\sqrt{n}}{\beta R\sqrt{n} + m\sqrt{L^2+p\sigma^2}}$, then we have

$$\mathbb{E}_{D,\theta_1,\ldots,\theta_m}\left[\mathscr{L}\left(\mathscr{D};\frac{1}{m}\sum_{i=1}^{m}\theta_i\right)\right] - \mathscr{L}(\mathscr{D};\theta^*) = \widetilde{O}\left(R\sqrt{\frac{L^2+p\sigma^2}{n}} + \frac{\beta R^2}{m}\right).$$

Comparison to Algorithm 1 in Section 3: Recall that in $\mathcal{A}_{fix}$ we can achieve a small fixed ε by taking $p_0 = m/n$ and $\sigma = \tilde{O}(\frac{p_0}{\varepsilon\sqrt{m}})$, in which case the excess risk bound in Theorem 3.5 becomes $\tilde{O}\left(\sqrt{\frac{L^2}{m} + \frac{p}{\varepsilon^2 n^2}}\right)$. On the other hand, in $\mathcal{A}_{avg}$ we can obtain a fixed small ε by taking $\sigma = \tilde{O}\left(\frac{1}{\varepsilon\sqrt{m}}\right)$. In this case the excess risks in Theorem 4.2 are bounded by $\tilde{O}\left(\sqrt{\frac{L^2}{m} + \frac{p}{\varepsilon^2 nm}}\right)$ in the convex case, and by $\tilde{O}\left(\sqrt{\frac{L^2}{n} + \frac{p}{\varepsilon^2 nm}} + \frac{1}{m}\right)$ in the convex and smooth case. Thus, we observe that all the bounds recover the optimal population risk trade-offs from [7, 8] as $m \to n$, and for $m \ll n$ and non-smooth loss $\mathcal{A}_{avg}$ provides a better trade-off than $\mathcal{A}_{fix}$, while on smooth losses $\mathcal{A}_{avg}$ and $\mathcal{A}_{fix}$ are incomparable. Note that $\mathcal{A}_{fix}$ (with $b = 1$) will not attain a better bound on smooth losses because each update is based on a single data-point. Setting $b > 1$ will reduce the number of updates to m/b for $\mathcal{A}_{fix}$, whereas to get an excess risk bound for $\mathcal{A}_{fix}$ for smooth losses where more than one data point is sampled at each time step will require extending the privacy analysis to incorporate the change, which is beyond the scope of this paper.

4.2 Random Check-Ins with a Sliding Window

The second variant we consider removes the need for all clients to be available throughout the training period. Instead, we assume that the training period comprises of n time steps, and each client $j \in [n]$ is only available during a window of m time steps. Clients perform a random check-in to provide the server with an update during their window of availability. For simplicity, we assume clients wake up in order, one every time step, so client $j \in [n]$ will perform a random check-in within the window $R_j = \{j, \ldots, j+m-1\}$. The server will perform $n - m + 1$ updates starting at time m to provide a warm-up period where the first m clients perform their random check-ins.

Theorem 4.3 (Amplification via random check-ins with sliding windows)**.** *Suppose $\mathcal{A}_{ldp}$ is an ε_0-DP local randomizer. Let $\mathcal{A}_{sldw} : \mathcal{D}^n \to \Theta^{n-m+1}$ be the distributed algorithm performing n model updates with check-in probability $p_j = 1$ and check-in window $R_j = \{j, \ldots, j+m-1\}$ for each user $j \in [n]$. For any $m \in [n]$, algorithm $\mathcal{A}_{sldw}$ is (ε, δ)-DP with $\varepsilon = \frac{e^{\varepsilon_0}(e^{\varepsilon_0}-1)^2}{2m} + (e^{\varepsilon_0}-1)\sqrt{\frac{2e^{\varepsilon_0}\log(1/\delta)}{m}}$. For $\varepsilon_0 \le 1$ and $\delta \le 1/100$, we get $\varepsilon \le 7\varepsilon_0\sqrt{\frac{\log(1/\delta)}{m}}$. Furthermore, if $\mathcal{A}_{ldp}$ is $(\varepsilon_0, \delta_0)$-DP with $\delta_0 \le \frac{(1-e^{-\varepsilon_0})\delta_1}{4e^{\varepsilon_0}\left(2+\frac{\ln(2/\delta_1)}{\ln(1/(1-e^{-5\varepsilon_0}))}\right)}$, then $\mathcal{A}_{sldw}$ is (ε', δ')-DP with $\varepsilon' = \frac{e^{8\varepsilon_0}(e^{8\varepsilon_0}-1)^2}{2m} + (e^{8\varepsilon_0}-1)\sqrt{\frac{2e^{8\varepsilon_0}\log(1/\delta)}{m}}$ and $\delta' = \delta + m(e^{\varepsilon'}+1)\delta_1$.*

Remark 4 We can always increase privacy in the statement above by increasing m. However, that also increases the number of clients who do not participate in training because their scheduled check-in time is before the process begins, or after it terminates. Moreover, the number of empty slots where the server introduces dummy updates will also increase, which we would want to minimize for good accuracy. Thus, m introduces a trade-off between accuracy and privacy.

Proposition 4.4 (Dummy updates in random check-ins with sliding windows)**.** *For algorithm $\mathcal{A}_{sldw} : \mathcal{D}^n \to \Theta^{n-m+1}$ described in Theorem 4.3, the expected number of dummy gradient updates performed by the server is at most $(n - m + 1)/e$.*

5 Improvements to Amplification via Shuffling

Here, we provide an improvement on privacy amplification by shuffling. This is obtained using two technical lemmas (deferred to the supplementary material) to tighten the analysis of amplification by swapping, a central component in the analysis of amplification by shuffling given in [19].

Theorem 5.1 (Amplification via Shuffling)**.** *Let $\mathcal{A}^{(i)} : \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(i-1)} \times \mathcal{D} \to \mathcal{S}^{(i)}$, $i \in [n]$, be a sequence of adaptive ε_0-DP local randomizers. Let $\mathcal{A}_{sl} : \mathcal{D}^n \to \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(n)}$ be the algorithm that given a dataset $D = (d_1, \ldots, d_n) \in \mathcal{D}^n$ samples a uniform random permutation π over $[n]$, sequentially computes $s_i = \mathcal{A}^{(i)}(s_{1:i-1}, d_{\pi(i)})$ and outputs $s_{1:n}$. For any $\delta \in (0,1)$, algorithm $\mathcal{A}_{sl}$ satisfies (ε, δ)-DP with $\varepsilon = \frac{e^{3\varepsilon_0}(e^{\varepsilon_0}-1)^2}{2n} + e^{3\varepsilon_0/2}(e^{\varepsilon_0}-1)\sqrt{\frac{2\log(1/\delta)}{n}}$. Furthermore, if $\mathcal{A}^{(i)}$, $i \in [n]$, is $(\varepsilon_0, \delta_0)$-DP with $\delta_0 \leq \frac{(1-e^{-\varepsilon_0})\delta_1}{4e^{\varepsilon_0}\left(2+\frac{\ln(2/\delta_1)}{\ln(1/(1-e^{-\varepsilon_0}))}\right)}$, then $\mathcal{A}_{sl}$ satisfies (ε', δ')-DP with $\varepsilon' = \frac{e^{24\varepsilon_0}(e^{8\varepsilon_0}-1)^2}{2n} + e^{12\varepsilon_0}(e^{8\varepsilon_0}-1)\sqrt{\frac{2\log(1/\delta)}{n}}$ and $\delta' = \delta + n(e^{\varepsilon'}+1)\delta_1$.*

For comparison, the guarantee in [19, Theorem 7] in the case $\delta_0 = 0$ results in

$$\varepsilon = 2e^{2\varepsilon_0}(e^{\varepsilon_0}-1)(e^{\frac{2\exp(2\varepsilon_0)(e^{\varepsilon_0}-1)}{n}}-1) + 2e^{2\varepsilon_0}(e^{\varepsilon_0}-1)\sqrt{\frac{2\log(1/\delta)}{n}}.$$

6 Conclusion

Our work highlights the fact that proving DP guarantees for distributed or decentralized systems can be substantially more challenging than for centralized systems, because in a distributed setting it becomes much harder to precisely control and characterize the randomness in the system, and this precise characterization and control of randomness is at the heart of DP guarantees. Specifically, production FL systems do not satisfy the assumptions that are typically made under state-of-the-art privacy accounting schemes, such as privacy amplification via subsampling. Without such accounting schemes, service providers cannot provide DP statements with small ε's. This work, though largely theoretical in nature, proposes a method shaped by the practical constraints of distributed systems that allows for rigorous privacy statements under realistic assumptions.

Nevertheless, there is more to do. Our theorems are sharpest in the high-privacy regime (small ε's), which may be too conservative to provide sufficient utility for some applications. While significantly relaxed from previous work, our assumptions will still not hold in all real-world systems. Thus, we hope this work encourages further collaboration between distributed systems and DP theory researchers in establishing protocols that address the full range of possible systems constraints as well as improving the full breadth of the privacy vs. utility Pareto frontier.

Acknowledgements

The authors would like to thank Vitaly Feldman for suggesting the idea of privacy accounting in DP-SGD via shuffling, and for help in identifying and fixing a mistake in the way a previous version of this paper handled $(\varepsilon_0, \delta_0)$-DP local randomizers.

References

[1] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security (CCS'16)*, pages 308–318, 2016.

[2] D. P. T. Apple. Learning with privacy at scale, 2017.

[3] S. Augenstein, H. B. McMahan, D. Ramage, S. Ramaswamy, P. Kairouz, M. Chen, R. Mathews, et al. Generative models for effective ml on private, decentralized datasets. *arXiv preprint arXiv:1911.06679*, 2019.

[4] V. Balcer and A. Cheu. Separating local & shuffled differential privacy via histograms. *CoRR*, abs/1911.06879, 2019.

[5] B. Balle, G. Barthe, and M. Gaboardi. Privacy amplification by subsampling: Tight analyses via couplings and divergences. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 6280–6290, 2018.

[6] B. Balle, J. Bell, A. Gascon, and K. Nissim. The privacy blanket of the shuffle model. In *Advances in Cryptology—CRYPTO*, 2019.

[7] R. Bassily, V. Feldman, K. Talwar, and A. G. Thakurta. Private stochastic convex optimization with optimal rates. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 11279–11288, 2019.

[8] R. Bassily, A. Smith, and A. 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.

[9] A. Bittau, Ú. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, M. Rudominer, U. Kode, J. Tinnés, and B. Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 441–459. ACM, 2017.

[10] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[11] S. Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[12] A. Cheu, A. Smith, J. Ullman, D. Zeber, and M. Zhilyaev. Distributed differential privacy via mixnets. *CoRR*, abs/1808.01394, 2018.

[13] B. Ding, J. Kulkarni, and S. Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 3571–3580, 2017.

[14] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 429–438. IEEE Computer Society, 2013.

[15] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology—EUROCRYPT*, pages 486–503, 2006.

[16] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proc. of the Third Conf. on Theory of Cryptography (TCC)*, pages 265–284, 2006.

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

[18] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, S. Song, K. Talwar, and A. Thakurta. Encode, shuffle, analyze privacy revisited: Formalizations and empirical evaluation. *CoRR*, abs/2001.03618, 2020.

[19] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2468–2479. SIAM, 2019.

[20] Ú. Erlingsson, V. Pihur, and A. Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *Proc. of the 2014 ACM Conf. on Computer and Communications Security (CCS'14)*, pages 1054–1067. ACM, 2014.

[21] V. Feldman, I. Mironov, K. Talwar, and A. Thakurta. Privacy amplification by iteration. In *59th Annual IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 521–532, 2018.

[22] B. Ghazi, R. Pagh, and A. Velingker. Scalable and differentially private distributed aggregation in the shuffled model. *CoRR*, abs/1906.08320, 2019.

[23] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao. Advances and open problems in federated learning. *CoRR*, abs/1912.04977, 2019.

[24] P. Kairouz, S. Oh, and P. Viswanath. The composition theorem for differential privacy. *IEEE Trans. Inf. Theory*, 63(6):4037–4049, 2017.

[25] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. D. Smith. What can we learn privately? In *49th Annual IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 531–540, 2008.

[26] Y. Kuo, C. Chiu, D. Kifer, M. Hay, and A. Machanavajjhala. Differentially private hierarchical count-of-counts histograms. *PVLDB*, 11(11):1509–1521, 2018.

[27] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, pages 1273–1282, 2017.

[28] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private language models without losing accuracy. *CoRR*, abs/1710.06963, 2017.

[29] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[30] V. Pichapati, A. T. Suresh, F. X. Yu, S. J. Reddi, and S. Kumar. Adaclip: Adaptive clipping for private sgd. *arXiv preprint arXiv:1908.07643*, 2019.

[31] R. Rogers, S. Subramaniam, S. Peng, D. Durfee, S. Lee, S. K. Kancha, S. Sahay, and P. Ahammad. Linkedin's audience engagements api: A privacy preserving data analytics system at scale, 2020.

[32] O. Shamir and T. Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International Conference on Machine Learning*, pages 71–79, 2013.

[33] A. Smith, A. Thakurta, and J. Upadhyay. Is interaction necessary for distributed private learning? In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 58–77. IEEE, 2017.

[34] S. Song, K. Chaudhuri, and A. D. Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248. IEEE, 2013.

[35] O. Thakkar, G. Andrew, and H. B. McMahan. Differentially private learning with adaptive clipping. *CoRR*, abs/1905.03871, 2019.

[36] J. S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):3757, Mar. 1985.

[37] Y. Wang, B. Balle, and S. P. Kasiviswanathan. Subsampled renyi differential privacy and analytical moments accountant. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, pages 1226–1235, 2019.

[38] Y.-X. Wang, S. E. Fienberg, and A. J. Smola. Privacy for free: Posterior sampling and stochastic gradient monte carlo. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML15, page 24932502. JMLR.org, 2015.

[39] X. Wu, F. Li, A. Kumar, K. Chaudhuri, S. Jha, and J. F. Naughton. Bolt-on differential privacy for scalable stochastic gradient descent-based analytics. In S. Salihoglu, W. Zhou, R. Chirkova, J. Yang, and D. Suciu, editors, *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD*, 2017.

A Omitted Results and Proofs

Lemma A.1. *Let $\mathcal{A}_{ldp} : \mathcal{D} \to \mathcal{S}$ be an ε_0-DP local randomizer. For $D = (d_1, \ldots, d_m) \in \mathcal{D}^m, q \in (0,1)$, and $k \in [m]$, define $\mathsf{BiasedSampling}_q(D,k)$ to return d_k with probability q, and a sample from an arbitrary distribution over $D \setminus \{d_k\}$ with probability $1-q$. For any $k \in [m]$ and any set of outcomes $S \subseteq \mathcal{S}$, we have*

$$\frac{\mathbf{Pr}\left[\mathcal{A}_{ldp}(d_k) \in S\right]}{\mathbf{Pr}\left[\mathcal{A}_{ldp}\left(\mathsf{BiasedSampling}_q(D,k)\right) \in S\right]} \leq \frac{e^{\varepsilon_0}}{1+q(e^{\varepsilon_0}-1)}.$$

Proof. Fix a set of outcomes $S \subseteq \mathcal{S}$. By ε_0-LDP of $\mathcal{A}_{ldp}$, for any $d, d' \in \mathcal{D}$, we get

$$\frac{\mathbf{Pr}[\mathcal{A}_{ldp}(d) \in S]}{\mathbf{Pr}[\mathcal{A}_{ldp}(d') \in S]} \leq e^{\varepsilon_0} \tag{1}$$

Now, for dataset $D = (d_1, \ldots, d_m) \in \mathcal{D}^n$ and $k \in [m]$, we have:

$$\begin{aligned}
\frac{\mathbf{Pr}\left[\mathcal{A}_{ldp}(d_k) \in S\right]}{\mathbf{Pr}\left[\mathcal{A}_{ldp}(\mathsf{BiasedSampling}_q(D,k)) \in S\right]} &= \frac{\mathbf{Pr}\left[\mathcal{A}_{ldp}(d_k) \in S\right]}{\sum_{j=1}^{m} \mathbf{Pr}[\mathcal{A}_{ldp}(d') \in S]\,\mathbf{Pr}[d' = d_j]} \\
&= \frac{1}{\sum_{j=1}^{m} \frac{\mathbf{Pr}[\mathcal{A}_{ldp}(d') \in S]}{\mathbf{Pr}\left[\mathcal{A}_{ldp}(d_k) \in S\right]}\,\mathbf{Pr}[d' = d_j]} \\
&= \frac{1}{q + \sum_{j \neq k} \frac{\mathbf{Pr}[\mathcal{A}_{ldp}(d') \in S]}{\mathbf{Pr}\left[\mathcal{A}_{ldp}(d_k) \in S\right]}\,\mathbf{Pr}[d' = d_j]} \\
&\leq \frac{1}{q + e^{-\varepsilon_0} \sum_{j \neq k} \mathbf{Pr}[d' = d_j]} \\
&= \frac{1}{q + (1-q)e^{-\varepsilon_0}} = \frac{e^{\varepsilon_0}}{1+q(e^{\varepsilon_0}-1)}
\end{aligned}$$

where the third equality follows as $\mathbf{Pr}[d = d_k] = q$, and the first inequality follows using inequality 1, and the fourth equality follows as $\sum_{j \neq k} \mathbf{Pr}[d = d_j] = 1 - q$. □

Lemma A.2. *Let $\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(k)}$ be mechanisms of the form $\mathcal{A}^{(i)} : \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(i-1)} \times \mathcal{D} \to \mathcal{S}^{(i)}$. Suppose there exist constants $a > 0$ and $b \in (0,1)$ such that each $\mathcal{A}^{(i)}$ is ε_i-DP with $\varepsilon_i \leq \log\left(1 + \frac{a}{k-b(i-1)}\right)$. Then, for any $\delta \in (0,1)$, the k-fold adaptive composition of $\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(k)}$ is (ε, δ)-DP with $\varepsilon = \frac{a^2}{2k(1-b)} + \sqrt{\frac{2a^2 \log(1/\delta)}{k(1-b)}}$.*

Proof. We start by applying the heterogeneous advanced composition for DP [24] for the sequence of mechanisms $\mathcal{A}_1, \ldots, \mathcal{A}_k$ to get (ε, δ)-DP for the composition, where

$$\varepsilon = \sum_{i \in [k]} \frac{(e^{\varepsilon_i} - 1)\varepsilon_i}{e^{\varepsilon_i} + 1} + \sqrt{2 \log \frac{1}{\delta} \sum_{i \in [k]} \varepsilon_i^2} \tag{2}$$

Let us start by bounding the second term in equation 2. First, observe that:

$$\sum_{i \in [k]} \varepsilon_i^2 = \sum_{i \in [k]} \left(\log\left(1 + \frac{a}{k - b(i-1)}\right)\right)^2 \leq \sum_{i \in [k]} \frac{a^2}{(k - b(i-1))^2} \tag{3}$$

where the first inequality follows from $\log(1+x) \leq x$.

Now, we have:

$$\begin{aligned}\sum_{i\in[k]} \frac{a^2}{(k-b(i-1))^2} &= \sum_{i=0}^{k-1} \frac{a^2}{(k-ib)^2} \leq a^2 \int_0^k \frac{1}{(k-xb)^2} dx \\ &= a^2 \left(\frac{1}{kb-b^2k} - \frac{1}{kb}\right) = \frac{a^2}{kb}\left(\frac{1}{1-b} - 1\right) \\ &= \frac{a^2}{k(1-b)} \end{aligned} \tag{4}$$

where the second equality follows as we have $\int \frac{1}{(c-dx)^2} dx = \frac{1}{cd-d^2x}$.

Next, we bound the first term in equation 2 as follows:

$$\begin{aligned}\sum_{i\in[k]} \frac{(e^{\varepsilon_i}-1)\varepsilon_i}{e^{\varepsilon_i}+1} &= \sum_{i\in[k]} \frac{\left(\frac{a}{k-b(i-1)}\right)\left(\log\left(1+\frac{a}{k-b(i-1)}\right)\right)}{2+\frac{a}{k-b(i-1)}} \leq \sum_{i\in[k]} \frac{\left(\frac{a}{k-b(i-1)}\right)^2}{2+\frac{a}{k-b(i-1)}} \\ &\leq \sum_{i\in[k]} \frac{a^2}{2\left(k-b(i-1)\right)^2} \leq \frac{a^2}{2k(1-b)} \end{aligned} \tag{5}$$

where the first inequality follows from $\log(1+x) \leq x$, and the last inequality follows from inequality 4.

Using inequalities 3, 4 and 5 in equation 2, we get that the k-fold adaptive composition of $\mathcal{A}_1, \ldots, \mathcal{A}_k$ satisfies (ε, δ)-DP, for $\varepsilon = \frac{a^2}{2k(1-b)} + \sqrt{\frac{2a^2\log(1/\delta)}{k(1-b)}}$. □

Lemma A.3. *Suppose $\mathcal{A} : \mathcal{D} \to \mathcal{S}$ is an $(\varepsilon_0, \delta_0)$-DP local randomizer with $\delta_0 \leq \frac{(1-e^{-\varepsilon_0})\delta_1}{4e^{\varepsilon_0}\left(2+\frac{\ln(2/\delta_1)}{\ln(1/(1-e^{-5\varepsilon_0}))}\right)}$. Then there exists an $8\varepsilon_0$-DP local randomizer $\tilde{\mathcal{A}} : \mathcal{D} \to \mathcal{S}$ such that for any $d \in \mathcal{D}$ we have $TV(\mathcal{A}(d), \tilde{\mathcal{A}}(d)) \leq \delta_1$.*

Proof. The proof is a direct application of results by Cheu et al. [12]. First we recall that from [12, Claims D.2 and D.5] (applied with $n = 1$ in their notation) it follows that given $\mathcal{A}$ there exist randomizers $\tilde{\mathcal{A}}_{k,T}$ which are $8\varepsilon_0$-DP and satisfy

$$TV(\mathcal{A}(d), \tilde{\mathcal{A}}_{k,T}(d)) \leq \left(1 - \frac{ke^{-2\varepsilon_0}}{2}\right)^T + (T+2)\frac{2\delta_0 e^{\varepsilon_0}}{1-e^{-\varepsilon_0}}$$

for any $k \in (0, 2e^{-2\varepsilon_0})$ and $T \in \mathbb{N}$ as long as $\delta_0 < \frac{1-e^{-\varepsilon_0}}{4e^{\varepsilon_0}}$. The result follows from taking $k = 2e^{-3\varepsilon_0}$, $T = \ln(2/\delta_1)/\ln(1/(1-e^{5\varepsilon_0}))$ and noting these choices imply the desired condition on the total variation distance under our assumption on δ_0. □

Proof of Corollary 3.3. Setting $p_0 = \frac{m}{n}$ in $\mathcal{A}_{fix}$, we get from Theorem 3.2 that $\beta \in (0,1)$, algorithm $\mathcal{A}_{fix}$ satisfies (ε_1, β)-DP for

$$\begin{aligned}\varepsilon_1 &= \frac{(e^{\varepsilon_0}-1)\sqrt{2me^{\varepsilon_0}\log(1/\beta)}}{n} + \frac{me^{\varepsilon_0}(e^{\varepsilon_0}-1)^2}{2n^2} \\ &\leq \frac{2(e^{\varepsilon_0}-1)\sqrt{2me^{\varepsilon_0}\log(1/\beta)}}{n} \end{aligned} \tag{6}$$

where the inequality follows since $n \geq (e^{\varepsilon_0}-1)\sqrt{me^{\varepsilon_0}}$.

Now, using inequality 6 and applying advanced composition to $\frac{n}{m}$ repetitions of $\mathcal{A}_{fix}$, we get $\left(\varepsilon, \frac{n\beta}{m} + \delta\right)$-DP, for

$$\varepsilon \leq \varepsilon_1\sqrt{\frac{2n}{m}\log(1/\delta)} + \frac{n}{m}\varepsilon_1(e^{\varepsilon_1}-1) \tag{7}$$

Since $\varepsilon_0 \leq \frac{2\log\left(n/8\sqrt{m}\right)}{3}$, we have that $\varepsilon_1 \leq \frac{1}{2}$, and thus, $(e^{\varepsilon_1}-1) \leq \frac{3\varepsilon_1}{2}$. Therefore, we get from inequality 7 that

$$\begin{aligned}
\varepsilon &\leq \varepsilon_1\sqrt{\frac{2n}{m}\log\left(1/\beta\right)} + \frac{3n}{2m}\varepsilon_1^2 \\
&\leq 4(e^{\varepsilon_0}-1)\sqrt{\frac{e^{\varepsilon_0}\log\left(1/\beta\right)\log\left(1/\delta\right)}{n}} + \frac{12(e^{\varepsilon_0}-1)^2 e^{\varepsilon_0}\log\left(1/\beta\right)}{n} \\
&= \widetilde{O}\left(\frac{e^{1.5\varepsilon_0}}{\sqrt{n}}\right)
\end{aligned}$$

where the equality holds since $n \geq (e^{\varepsilon_0}-1)^2 e^{\varepsilon_0}\sqrt{m}\log\left(1/\beta\right)$, and $\tilde{O}(\cdot)$ hides polylog factors in $1/\beta$ and $1/\delta$. □

Proof of Proposition 3.4. In Algorithm 1, for $i \in [m]$, we have

$$S_i = \{j : \text{User}(j) \text{ checks-in for index } i\}$$

For $i \in [m]$, define an indicator random variable E_i that indicates if S_i is empty. Note that the server performs a dummy gradient update for instance $i \in [n]$ if and only if S_i is empty (or, in other words, $E_i = 1$). Next, for $j \in [n]$, let I_j denote the index that user j in Algorithm $\mathcal{A}_{fix}$ performs her (R_j, p_j)-check-in into, where $R_j = [m]$ and $p_j = p_0$. Thus, for index $i \in [m]$, we have

$$\begin{aligned}
\mathbf{Pr}\left[E_i = 1\right] &= \mathbf{Pr}\left[\bigcap_{j\in[n]}\Big((\text{User } j \text{ abstains})\bigcup(\text{User } j \text{ participates} \wedge I_j \neq i)\Big)\right] \\
&= \prod_{j\in[n]}\left((1-p_0) + \mathbf{Pr}\left[I_j \neq i\right]\cdot p_0\right) = \left((1-p_0) + \left(1-\frac{1}{m}\right)\cdot p_0\right)^n \\
&= \left(1-\frac{p_0}{m}\right)^n
\end{aligned}$$

where the second equality follows since the check-ins for each user are independent of the others, and each user abstains from participating w.p. $(1-p_0)$.

Thus, for the expected number of dummy gradient updates, we have:

$$\mathbb{E}(E_{1:m}) = \sum_{i\in[m]}\mathbf{Pr}\left[E_i = 1\right] = m\left(1-\frac{p_0}{m}\right)^n \tag{8}$$

If $p_0 = \frac{cm}{n}$ for $c > 0$, from equation 8 we get

$$\mathbb{E}(E_{1:m}) = m\left(1-\frac{c}{n}\right)^n \leq \frac{m}{e^c}$$

where the inequality follows as $\left(1-\frac{a}{b}\right)^b \leq e^{-a}$ for $b > 1, |a| \leq b$. □

Proof of Theorem 3.5. To be able to directly apply [32, Theorem 2], our technique $\mathcal{A}_{fix}$ needs to satisfy two conditions: i) each model update should be an unbiased estimate of the gradient, and ii) a bound on the expected L_2-norm of the gradient. Notice that in $\mathcal{A}_{fix}$, every client $j \in [n]$ performs a $([m], p_0)$-check-in. This is analogous to a bins-and-balls setting where n balls are thrown, each with probability p_0, into m bins. Thus, for each update step $i \in [m]$, the number of clients checking-in for this step (i.e., $|S_i|$ in the notation of Algorithm 1) can be approximated by an independent Poisson random variable Y_i with mean np_0/m, using Poisson approximation [29], as follows:

$$\mathbf{Pr}[|S_i| = 0] \leq 2\,\mathbf{Pr}[Y_i = 0] = 2e^{-np_0/m} := p'$$

Now, we know that there exists a probability $p_b \leq p'$ with which the gradient update g_i is 0^p. Thus, to make the gradient update unbiased, each participating user can multiply their update by $\frac{1}{1-p_b} \leq \frac{1}{1-p'} = \frac{1}{1-2e^{-np_0/m}}$.

Consequently, the Lipschitz-constant of the loss ℓ, and the variance of the noise added to the update, increases by a factor of at most $\frac{1}{\left(1-2e^{-np_0/m}\right)^2}$. Thus, we get $\mathbb{E}\left[||\widetilde{g}_i||^2\right] \leq \frac{p\sigma^2+L^2}{1-2e^{-np_0/m}}$. With this, our technique will satisfy both the conditions required to apply the result in [32] for learning rate $\eta_i = \frac{c}{\sqrt{i}}$ as follows:

$$\mathbb{E}_{D,\theta_m}\left[\mathscr{L}(\mathscr{D};\theta_m)\right] - \mathscr{L}(\mathscr{D};\theta^*) \leq \left(\frac{R^2}{c} + \frac{c(p\sigma^2+L^2)}{1-2e^{-np_0/m}}\right)\left(\frac{2+\log(m)}{\sqrt{m}}\right)$$

Optimizing the learning rate to be $\eta_i = \frac{R\left(1-2e^{-np_0/m}\right)}{\sqrt{(p\sigma^2+L^2)i}}$ gives the statement of the theorem. □

Proof of Theorem 4.2. We prove the first bound on the line of the proof of Theorem 3.5. Since $\mathcal{A}_{avg}$ skips an update for time step $i \in [m]$ if no client checks-in at step i, and otherwise makes an update of the average of the noisy gradients received by checked-in clients, each update of the algorithm is unbiased. Now, notice that in $\mathcal{A}_{avg}$, every client $j \in [n]$ checks into $[m]$ u.a.r. Thus, each update step $i \in [m]$ will have n/m checked-in clients in expectation. As a result, for an averaged update $\widetilde{h}_i = \frac{\widetilde{g}_i}{|S_i|}$, we get $\mathbb{E}\left[\|\widetilde{h}_i\|^2\right] \leq \frac{mp\sigma^2}{n} + L^2$. With this, our technique will satisfy both the conditions required to apply [32, Theorem 2] for learning rate $\eta_i = \frac{c}{\sqrt{i}}$, giving:

$$\mathbb{E}_{D,\theta_m}\left[\mathscr{L}(\mathscr{D};\theta_m)\right] - \mathscr{L}(\mathscr{D};\theta^*) \leq \left(\frac{R^2}{c} + c\left(\frac{mp\sigma^2}{n} + L^2\right)\right)\left(\frac{2+\log(m)}{\sqrt{m}}\right)$$

Optimizing the learning rate to be $\eta_i = \frac{R\sqrt{n}}{\sqrt{(mp\sigma^2+nL^2)i}}$ gives the statement of the theorem.

When in addition the loss is β-smooth we can obtain an improved bound on the expected risk – in this case, for the average parameter vector $\frac{1}{m}\sum_i \theta_i$ – by applying [11, Theorem 6.3]. Let $h_i = \nabla_\theta \mathscr{L}(\mathscr{D};\theta_i)$ be the true gradient on the population loss at each iteration. The cited result says that after m iterations with learning rate $\eta_i = \frac{1}{\beta + \frac{\kappa\sqrt{t}}{\sqrt{2}R}}$ with $\kappa^2 \geq \mathbb{E}[\|h_i - \tilde{h}_i\|^2]$ we get

$$\mathbb{E}_{D,\theta_1,\ldots,\theta_m}\left[\mathscr{L}\left(\mathscr{D};\frac{1}{m}\sum_{i=1}^{m}\theta_i\right)\right] - \mathscr{L}(\mathscr{D};\theta^*) \leq R\kappa\sqrt{\frac{2}{m}} + \frac{\beta R^2}{m}$$

The result now follows from observing that

$$\mathbb{E}[\|h_i - \tilde{h}_i\|^2] \leq \mathbb{E}_{S\sim\mathrm{Bin}(n,1/m)}\left[\frac{1}{S}(L^2+p\sigma^2)\middle| S>0\right] = O\left(\frac{m}{n}(L^2+p\sigma^2)\right)$$

□

Proof of Proposition 4.4. In Algorithm $\mathcal{A}_{sldw}$, for $i \in [n-m+1]$, we have

$$S_i = \{j : \text{User}(j) \text{ checks-in for index } i\}$$

For $i \in [n]$, define an indicator random variable E_i that indicates if S_i is empty. Note that the server performs a dummy gradient update for instance $i \in [n]$ if and only if S_i is empty (or, in other words, $E_i = 1$). Next, for $j \in [m]$, let I_j denote the index that user j in Algorithm $\mathcal{A}_{fix}$ performs her R_j-check-in into, where $R_j = \{j, \ldots, j+m-1\}$. Thus, for index $i \in \{m, \ldots, n-m+1\}$, we have

$$\mathbf{Pr}\left[E_i = 1\right] = \mathbf{Pr}\left[\bigcap_{j\in[i-m+1,i]} I_j \neq i\right] = \prod_{j\in[i-m+1,i]} \mathbf{Pr}\left[I_j \neq i\right] = \left(1-\frac{1}{m}\right)^m \leq \frac{1}{e} \tag{9}$$

where the second equality follows since the check-ins for each user are independent of the others, and the inequality follows as $\left(1-\frac{a}{b}\right)^b \leq e^{-a}$ for $b > 1, |a| \leq b$.

Thus, for the expected number of dummy gradient updates, we have:

$$\mathbb{E}(E_{1:n}) = \sum_{i\in\{m,\ldots,n-m+1\}} \mathbf{Pr}\left[E_i = 1\right] \leq \frac{n-m+1}{e}$$

where the inequality follows from inequality 9. □

A.1 Proof of Theorems 3.2 and 4.3

We will first prove the privacy guarantee of $\mathcal{A}_{fix}$ (Algorithm 1) by reducing it to algorithm $\mathcal{A}_{rep}$ (Algorithm 3) that starts by swapping the first element in the dataset by a given replacement element, randomly chooses a position in the dataset to get replaced by the original first element with a given probability, and then carries out DP-SGD with the local randomizer. W.l.o.g., for simplicity we will define $\mathcal{A}_{rep}$ to update the model for 1-sized minibatches (i.e., update at every time step). It is easy to extend to b-sized minibatch updates by accumulating the gradient updates for every b steps and then updating the model.

For the proofs that follow, it will be convenient to define additional notation for denoting distance between distributions. Given 2 distributions μ and μ', we denote them as $\mu \approxeq_{(\varepsilon,\delta)} \mu'$ if they are (ε, δ)-DP close, i.e., if for all measurable outcomes S, we have

$$e^{-\varepsilon}\left(\mu'(S) - \delta\right) \leq \mu(S) \leq e^{\varepsilon}\mu'(S) + \delta$$

Algorithm 3 $\mathcal{A}_{rep}$: DP-SGD with One Random Replacement

Input: Dataset $D = d_{1:m}$, local randomizer $\mathcal{A}_{ldp}$.

Parameters: Initial model $\theta_1 \in \mathbb{R}^p$, weights $w_{1:m}$ where $w_i \in [0, w_{max}]$ for $i \in [m]$, replacement element d_r

Sample $I \xleftarrow{u.a.r.} [m]$

Let $G \leftarrow (d_r, d_{2:m})$

Let $\sigma_I(D) \leftarrow (G_{1:I-1}, z_I, G_{I+1:m})$, where $z_I = \begin{cases} d_1 & \text{with probability } w_I \\ G[I] & \text{otherwise} \end{cases}$

for $i \in [m]$ **do**

  $\tilde{g}_i \leftarrow \mathcal{A}_{ldp}(\theta_i; \sigma_I(D)[i])$

  $\theta_{i+1} \leftarrow \theta_i - \eta\tilde{g}_i$

  Output θ_{i+1}

Theorem A.4 (Amplification via random replacement)**.** *Suppose $\mathcal{A}_{ldp}$ is an ε_0-DP local randomizer. Let $\mathcal{A}_{rep} : \mathcal{D}^m \to \Theta^m$ be the protocol from Algorithm 3. For any $\delta \in (0,1)$, algorithm $\mathcal{A}_{rep}$ is (ε, δ)-DP at index 1 in the central model, where $\varepsilon = \frac{w_{max}^2 e^{\varepsilon_0}(e^{\varepsilon_0}-1)^2}{2m} + w_{max}(e^{\varepsilon_0}-1)\sqrt{\frac{2e^{\varepsilon_0}\log(1/\delta)}{m}}$. In particular, for $\varepsilon_0 \leq 1$ and $\delta \leq 1/100$, we get $\varepsilon \leq 7w_{max}\varepsilon_0\sqrt{\frac{\log(1/\delta)}{m}}$. Here, initial model $\theta_1 \in \mathbb{R}^p$, weights $w_{max} \in [0,1]$, $w_i \in [0, w_{max}]$ for every $i \in [m]$, and replacement element $d_r \in [0,1]$ are parameters to $\mathcal{A}_{rep}$. Furthermore, if $\mathcal{A}_{ldp}$ is an $(\varepsilon_0, \delta_0)$-DP local randomizer with $\delta_0 \leq \frac{(1-e^{-\varepsilon_0})\delta_1}{4e^{\varepsilon_0}\left(2+\frac{\ln(2/\delta_1)}{\ln(1/(1-e^{-5\varepsilon_0}))}\right)}$, then algorithm $\mathcal{A}_{rep}$ is (ε', δ')-DP at index 1 in the central model, where $\varepsilon' = \frac{w_{max}^2 e^{8\varepsilon_0}(e^{8\varepsilon_0}-1)^2}{2m} + w_{max}(e^{8\varepsilon_0}-1)\sqrt{\frac{2e^{8\varepsilon_0}\log(1/\delta)}{m}}$ and $\delta' = \delta + m(e^{\varepsilon'}+1)\delta_1$.*

Proof. We start by proving the privacy guarantee of $\mathcal{A}_{rep}$ for the case where the local randomizer $\mathcal{A}_{ldp}$ is ε_0-DP, i.e., for the case where $\delta_0 = 0$. Let us denote the output sequence of $\mathcal{A}_{rep}$ by $Z_2, Z_3, \ldots, Z_{m+1}$. Note that $Z_{2:m+1}$ can be seen as the output of a sequence of m algorithms with conditionally independent randomness: $\mathcal{B}^{(i)}$ for $i \in [m]$ as follows. On input $\theta_{2:i}$ and D, $\mathcal{B}^{(i)}$ outputs a random sample from the distribution of $Z_{i+1}|Z_{2:i} = \theta_{2:i}$. The outputs of $\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(i-1)}$ are given as input to $\mathcal{B}^{(i)}$. Therefore, in order to upper bound the privacy parameters of $\mathcal{A}_{rep}$, we analyze the privacy parameters of $\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(m)}$ and apply the heterogeneous advanced composition for DP [24].

Next, observe that conditioned on the value of I, Z_{i+1} is the output of $\mathcal{A}_{ldp}^{(i)}(\theta_i; d)$ with its internal randomness independent of $Z_{2:i}$. In particular, for $i \geq 2$, one can implement $\mathcal{B}^{(i)}$ as follows. First, sample an index T from the distribution of $I|Z_{2:i} = \theta_{2:i}$. Assign $\tilde{g}_i = \mathcal{A}_{ldp}(\theta_i; d_1)$ w.p. w_i if $T = i$, otherwise let $\tilde{g}_i = \mathcal{A}_{ldp}(\theta_i; d_i)$. For $\mathcal{B}^{(1)}$, we first sample T u.a.r. from $[m]$, and let $\tilde{g}_1 = \mathcal{A}_{ldp}(\theta_1; d_1)$ w.p. w_1 if $T = 1$, otherwise let $\tilde{g}_1 = \mathcal{A}_{ldp}(\theta_1; d_r)$. For each $i \in [m]$, algorithm $\mathcal{B}^{(i)}$ outputs $\theta_{i+1} = \theta_i - \eta\tilde{g}_i$.

We now prove that for each $i \in [m]$, $\mathcal{B}^{(i)}$ is $\left(\log\left(1 + \frac{w_{max}e^{\varepsilon_0}(e^{\varepsilon_0}-1)}{i-1+e^{\varepsilon_0}(m-i+1)}\right), 0\right)$-DP at index 1. Let $D = d_{1:m}$ and $D' = (d_1', d_{2:m})$ be 2 datasets differing in the first element. Let $\theta_{2:i}$ denote the input to $\mathcal{B}^{(i)}$. Let μ be the probability

distribution of $\mathcal{B}^{(i)}(\theta_{2:i}; D)$. Let μ_1 be the distribution of $\mathcal{B}^{(i)}(\theta_{2:i}; D)$ conditioned on $\tilde{g}_i = \mathcal{A}_{ldp}(\theta_{2:i}; d_1)$, and μ_0 be the distribution of $\mathcal{B}^{(i)}(\theta_{2:i}; D)$ conditioned on $\tilde{g}_i = \mathcal{A}_{ldp}(\theta_{2:i}; d_r)$ for $i = 1$, and $\tilde{g}_i = \mathcal{A}_{ldp}(\theta_{2:i}; d_i)$ for $i \geq 2$. Also, denote by μ', μ'_0, and μ'_1 the corresponding quantities when $\mathcal{B}^{(i)}$ is run on D'. Let q_i be the probability that $T = i$ (sampled from $I|Z_{2:i} = \theta_{2:i}$). By definition, $\mu = (1 - q_i w_i)\mu_0 + q_i w_i \mu_1$, as $\mathcal{B}^{(i)}(s_{1:i-1}; D)$ generates output using $\mathcal{A}_{ldp}(\theta_{2:i}; d_1)$ w.p. w_i if $T = i$. Similarly, $\mu' = (1 - q'_i w_i)\mu'_0 + q'_i w_i \mu'_1$ when the input dataset is D'.

For $i \in [m]$, we observe that $\mu_0 = \mu'_0$, since in both cases the output is generated by $\mathcal{A}_{ldp}(\theta_1, d_r)$ for $i = 1$, and $\mathcal{A}_{ldp}(\theta_{2:i}; d_i)$ for $i \geq 2$. W.l.o.g. assume that $q_i \geq q'_i$. Thus, we can shift $(q_i - q'_i)w_i$ mass from the first component of the mixture in μ' to the second component to obtain

$$\mu' = (1 - q_i w_i)\mu_0 + q_i w_i \left(\frac{q'_i}{q_i} \mu'_1 + \left(1 - \frac{q'_i}{q_i}\right) \mu_0 \right) = (1 - q_i w_i)\mu_0 + q_i w_i \mu''_1$$

This shows that μ and μ' are overlapping mixtures [5]. Now, ε_0-LDP of $\mathcal{A}_{ldp}$ implies $\mu_0 \approxeq_{(\varepsilon_0, 0)} \mu_1$ and $\mu'_0 \approxeq_{(\varepsilon_0, 0)} \mu'_1$. Moreover, ε_0-LDP of $\mathcal{A}_{ldp}$ also implies $\mu_1 \approxeq_{(\varepsilon_0, 0)} \mu'_1$, so by the joint convexity of the relation $\approxeq_{(\varepsilon_0, 0)}$ we also have $\mu_1 \approxeq_{(\varepsilon_0, 0)} \mu''_1$. Thus, we can apply Advanced Joint Convexity of overlapping mixtures (Theorem 2 in [5]) to get that

$$\mu \approxeq_{(\log(1 + q_i w_i (e^{\varepsilon_0} - 1)), 0)} \mu' \tag{10}$$

We now claim that $q_i \leq \frac{e^{\varepsilon_0}}{i - 1 + e^{\varepsilon_0}(m - i + 1)}$. Observe that for each $D^* \in \{D, D'\}$, conditioning on $T = i$ reduces $\mathcal{A}_{rep}$ to running $\mathcal{A}_{ldp}$ on $\sigma_i(D^*)$. Note that for $j < i$, we have that $\sigma_i(D^*)[1 : i - 1]$ differs from $\sigma_j(D^*)[1 : i - 1]$ in at most 1 position, and for $j > i$, we have $\sigma_i(D^*)[1 : i - 1] = \sigma_j(D^*)[1 : i - 1]$. Since $\mathbf{Pr}[j \geq i] = \frac{m - i + 1}{m}$, by setting $q = \frac{m - i + 1}{m}$ in Lemma A.1, we get that

$$\frac{\mathbf{Pr}[Z_{2:i} = \theta_{2:i} | T = i]}{\mathbf{Pr}[Z_{2:i} = \theta_{2:i}]} \leq \frac{e^{\varepsilon_0}}{1 + \frac{(m - i + 1)}{m}(e^{\varepsilon_0} - 1)} = \frac{m e^{\varepsilon_0}}{i - 1 + e^{\varepsilon_0}(m - i + 1)} \tag{11}$$

This immediately implies our claim, since we have

$$\begin{aligned} q_i = \mathbf{Pr}[T = i | Z_{2:i} = \theta_{2:i}] &= \frac{\mathbf{Pr}[Z_{2:i} = \theta_{2:i} | T = i] \cdot \mathbf{Pr}[t = i]}{\mathbf{Pr}[Z_{2:i} = \theta_{2:i}]} \\ &\leq \frac{e^{\varepsilon_0}}{i - 1 + e^{\varepsilon_0}(m - i + 1)} \end{aligned}$$

where the inequality follows from inequality 11, and as $\mathbf{Pr}[T = i] = \frac{1}{m}$.

Substituting the value of q_i in equation 10, and using the fact that $w_i \leq w_{max}$, we get that for each $i \in [m]$, algorithm $\mathcal{B}^{(i)}$ is $(\varepsilon_i, 0)$-DP at index 1, where $\varepsilon_i = \log\left(1 + \frac{w_{max} e^{\varepsilon_0}(e^{\varepsilon_0} - 1)}{i - 1 + e^{\varepsilon_0}(m - i + 1)}\right)$. This can alternatively be written as $\varepsilon_i = \log\left(1 + \frac{w_{max}(e^{\varepsilon_0} - 1)}{m - (i - 1)\frac{e^{\varepsilon_0} - 1}{e^{\varepsilon_0}}}\right)$, and using Lemma A.2 for the sequence of mechanisms $\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(m)}$ by setting $a = w_{max}(e^{\varepsilon_0} - 1)$, $b = \frac{e^{\varepsilon_0} - 1}{e^{\varepsilon_0}}$, and $k = m$, we get that algorithm $\mathcal{A}_{rep}$ satisfies (ε, δ)-DP at index 1, for $\varepsilon = \frac{w_{max}^2 e^{\varepsilon_0}(e^{\varepsilon_0} - 1)^2}{2m} + w_{max}(e^{\varepsilon_0} - 1)\sqrt{\frac{2e^{\varepsilon_0} \log(1/\delta)}{m}}$.

Now, for the above bound, if $\varepsilon_0 \leq 1$ and $\delta \leq 1/4$, we get that

$$\begin{aligned} \varepsilon &= \frac{w_{max}^2 e^{\varepsilon_0}(e^{\varepsilon_0} - 1)^2}{2m} + w_{max}(e^{\varepsilon_0} - 1)\sqrt{\frac{2e^{\varepsilon_0} \log(1/\delta)}{m}} \\ &= \frac{w_{max} e^{0.5\varepsilon_0}(e^{\varepsilon_0} - 1)}{\sqrt{m}} \left(\frac{w_{max} e^{0.5\varepsilon_0}(e^{\varepsilon_0} - 1)}{2\sqrt{m}} + \sqrt{2 \log(1/\delta)} \right) \\ &\leq \frac{3 w_{max} \varepsilon_0}{\sqrt{m}} \left(\frac{3 w_{max} \varepsilon_0}{2\sqrt{m}} + \sqrt{2 \log(1/\delta)} \right) \\ &\leq \frac{3 w_{max} \varepsilon_0}{\sqrt{m}} \left(\left(\sqrt{2} + \sqrt{1/2}\right) \sqrt{\log(1/\delta)} \right) \leq 7 w_{max} \varepsilon_0 \sqrt{\frac{\log(1/\delta)}{m}} \end{aligned}$$

where the first inequality follows since $e^{0.5\varepsilon_0}(e^{\varepsilon_0}-1) \leq 3\varepsilon_0$ for $\varepsilon_0 \leq 1$, and the second inequality follows since $\frac{3w_{max}\varepsilon_0}{2\sqrt{m}} \leq \sqrt{\frac{\log\frac{1}{\delta}}{2}}$ for $\delta \leq 1/100$.

Now, we prove the privacy guarantee of $\mathcal{A}_{rep}$ for the more general case where for each $i \in [m]$, the local randomizer $\mathcal{A}_{ldp}$ is $(\varepsilon_0, \delta_0)$-DP. To upper bound the privacy parameters of $\mathcal{A}_{rep}$, we modify the local randomizer to satisfy pure DP, apply the previous analysis, and then account for the difference between the protocols with original and modified randomizers using the total variation distance.

Since $\mathcal{A}_{ldp}$ is $(\varepsilon_0, \delta_0)$-DP with $\delta_0 \leq \frac{(1-e^{-\varepsilon_0})\delta_1}{4e^{\varepsilon_0}\left(2+\frac{\ln(2/\delta_1)}{\ln(1/(1-e^{-5\varepsilon_0}))}\right)}$, we get from Lemma A.3 that there exists a randomizer $\tilde{\mathcal{A}}_{ldp}$ that is $8\varepsilon_0$-DP, and for any data record d and parameter vector θ satisfies $TV\left(\mathcal{A}_{ldp}(d;\theta), \tilde{\mathcal{A}}_{ldp}(d;\theta)\right) \leq \delta_1$. After replacing every instance of $\mathcal{A}_{ldp}$ in $\mathcal{A}_{rep}$ with $\tilde{\mathcal{A}}_{ldp}$ to obtain $\tilde{\mathcal{A}}_{rep}$, a union bound gives:

$$TV\left(\mathcal{A}_{rep}(D); \tilde{\mathcal{A}}_{rep}(D)\right) \leq m\delta_1 \tag{12}$$

Now, proceeding in a similar manner as in the case of ε_0-DP local randomizers above to see that $\tilde{\mathcal{A}}_{rep}$ using the $8\varepsilon_0$-DP local randomizers $\tilde{\mathcal{A}}_{ldp}$ satisfies (ε', δ)-DP at index 1 with $\varepsilon' = \frac{w_{max}^2 e^{8\varepsilon_0}(e^{8\varepsilon_0}-1)^2}{2m} + w_{max}(e^{8\varepsilon_0}-1)\sqrt{\frac{2e^{8\varepsilon_0}\log(1/\delta)}{m}}$. Thus, using Proposition 3 from [38] and inequality 12, we get that $\mathcal{A}_{rep}$ satisfies (ε', δ')-DP at index 1 with $\delta' = \delta + m(e^{\varepsilon'}+1)\delta_1$. □

Now, we are ready to prove Theorems 3.2 and 4.3.

Proof of Theorem 3.2. Let D and D' be 2 datasets of n users that differ in a user at some index $i^* \in [n]$. Algorithm $\mathcal{A}_{fix}$ can be alternatively seen as follows. The server starts by initializing $F = [0^p]^m$, weights $W = [1]^m$, and for $i \in [m]$, set $S_i = \phi$. For each user $j \in [n]$ s.t. $j \neq i^*$, user j performs a random check-in along with some additional operations. She first samples I_j u.a.r. from $[m]$, and w.p. p_0 does the following: she requests the server for model at index I_j (and gets inserted into set S_{I_j} at the server). She also updates $F[I_j] = d_j$ with probability $W[I_j]$, and sets $W[I_j] = \frac{W[I_j]}{W[I_j]+1}$. Next, the server runs $\mathcal{A}_{rep}$ on input dataset $\pi^*(D) = (d_{i^*}, F[2:m])$, with the replacement element $F[1]$, initial model θ_1, and weight parameters set to $W'[1:m]$, where $W'[i] = W[i] \cdot p_0$.

First, notice that in the alternative strategy above, for each of the weights $W[i], i \in [m]$, it always holds that $W[i] = \frac{1}{|S_i|}$. Thus, each weight $W[i], i \in [m]$ is updated to simulate reservoir sampling [36] of size 1 in slot $F[i]$. In other words, updating $F[i] = d$ with probability $W[i]$ for an element d is equivalent to $F[j] \xleftarrow{u.a.r.} S_i$, where S_i is the set containing d and all the elements previously considered for updating S_i. As a result, since the first element in $\mathcal{A}_{rep}$ performs a random replacement with weights set to $W'[1:m]$ for its input dataset, it is easy to see that performing a concurrent random check-in for user i^* (as in Algorithm 1) is equivalent to performing a random replacement for her after the check-ins of all the other users.

From our construction, we know that datasets $\pi^*(D)$ and $\pi^*(D')$, which are each of length m, differ only in the element with index 1. Moreover, in the alternative strategy above, note that the weights $W'[1:m]$ and the replacement element $F[1]$ input to $\mathcal{A}_{rep}$ are independent of the data of user i^* in the original dataset. Therefore, in the case $\delta_0 = 0$, using Theorem A.4 and setting $w_{max} = p_0$, we get $\mathcal{A}_{rep}(\pi^*(D)) \approxeq_{\varepsilon,\delta} \mathcal{A}_{rep}(\pi^*(D'))$ at index 1, for $\varepsilon = \frac{p^2 e^{\varepsilon_0}(e^{\varepsilon_0}-1)^2}{2m} + \frac{p(e^{\varepsilon_0}-1)\sqrt{2e^{\varepsilon_0}\log(1/\delta)}}{m}$, which implies $\mathcal{A}_{dist}(D) \approxeq_{\varepsilon,\delta} \mathcal{A}_{dist}(D')$. Consequently, it implies $\varepsilon \leq 7p_0\varepsilon_0\sqrt{\frac{\log(1/\delta)}{m}}$ for $\varepsilon_0 \leq 1$ and $\delta \leq 1/100$.

The case $\delta_0 > 0$ follows from the same reduction using the corresponding setting of Theorem A.4. □

Proof of Theorem 4.3. We proceed similar to the proof of Theorem 3.2. Let D and D' be 2 datasets of n users that differ in a user at some index $i^* \in [n]$. Algorithm $\mathcal{A}_{sldw}$ can be alternatively seen as follows. The server starts by initializing $F = [0^p]^{n-m+1}$, weights $W = [1]^{n-m+1}$, and for $j \in \{m, \ldots, n\}$, set $S_j = \phi$. For each user $j \in [n]$ s.t. $j \neq i^*$, user j performs a random check-in along with some additional operations. She first samples I_j u.a.r. from $\{j, \ldots, j+m-1\}$, requests the server for model at index I_j (and gets inserted into set S_{I_j} at the server). She also updates $F[I_j] = d_j$ with probability $W[I_j]$, and sets $W[I_j] = \frac{W[I_j]}{W[I_j]+1}$.

Now, the server runs its loop until it releases $i^* - 1$ outputs. Next, the server runs $\mathcal{A}_{rep}$ on input dataset $\pi^*(D) = (d_{i^*}, F[i^*+1 : i^*+m])$, with weight parameters set to $W[i^* : i^*+m]$, initializing model θ_{i^*}, and the replacement element $F[i^*]$. Lastly, the server releases the last $(n - (i^*+m) + 1)$ outputs of $\mathcal{A}_{sldw}$ using $F[i^*+m+1 : n]$ and the local randomizer $\mathcal{A}_{ldp}$.

First, notice that in the alternative strategy above, for each of the weights $W[i], i \in [n]$, it always holds that $W[i] = \frac{1}{|S_i|}$. Thus, each weight $W[i], i \in [n]$ is updated to simulate reservoir sampling [36] of size 1 in slot $F[i]$. In other words, updating $F[i] = d$ with probability $W[i]$ for an element d is equivalent to $F[i] \xleftarrow{u.a.r.} S_i$, where S_i is the set containing z and all the elements previously considered for updating S_i. As a result, since the first element in $\mathcal{A}_{rep}$ performs a random replacement for its input dataset (which doesn't include $F_{1:i^*-1} \bigcup F_{i^*+m+1:n}$ in the alternative strategy above), it is easy to see that sequentially performing a random check-in for user i^* (as in Algorithm 1) is equivalent to performing a random replacement for her after the check-ins of all the other users and releasing the first $i^* - 1$ outputs of $\mathcal{A}_{sldw}$.

From our construction, we know that datasets $\pi^*(D)$ and $\pi^*(D')$, which are each of length m, differ only in the element with index 1. Moreover, in the alternative strategy above, note that the weights $W[i^* : i^*+m]$, initializing model θ_{i^*} and the replacement element $F[i^*]$ input to $\mathcal{A}_{rep}$ are independent of the data of user i^* in the original dataset. Therefore, using Theorem A.4 and setting $w_{max} = 1$, we get $\mathcal{A}_{rep}(\pi^*(D)) \approxeq_{\varepsilon, \delta+m\delta_0} \mathcal{A}_{rep}(\pi^*(D'))$ at index 1, for $\varepsilon = \frac{e^{\varepsilon_0}(e^{\varepsilon_0}-1)^2}{2m} + (e^{\varepsilon_0}-1)\sqrt{\frac{2e^{\varepsilon_0}\log(1/\delta)}{m}}$, which implies $\mathcal{A}_{rc}(D) \approxeq_{\varepsilon, \delta+m\delta_0} \mathcal{A}_{rc}(D')$. Consequently, it implies $\varepsilon \leq 7\varepsilon_0\sqrt{\frac{\log(1/\delta)}{m}}$ for $\varepsilon_0 \leq 1$ and $\delta \leq 1/100$.

The case $\delta_0 > 0$ follows from the same reduction using the corresponding setting of Theorem A.4. □

A.2 Proof of Theorem 4.1

Let $L = (L_1, \ldots, L_m)$ represent the number of users contributing to each of the update steps, i.e., $L_i = |S_i|$ for $i \in [m]$. We start by considering the output distribution of $\mathcal{A}_{avg}(D)$ conditioned on $L = \ell$ for some $\ell \in [n]^m$ s.t. $\sum_i \ell_i = n$. This distribution is the same as the one produced by Algorithm 4 with bin sizes ℓ on a random permutation $\pi(D)$ of the original dataset D. To analyze the privacy of $\mathcal{A}_{bin}(\pi(D), \ell)$ we use the reduction from shuffling to swapping [19] . This reduction says it suffices to analyze the privacy of $D \mapsto \mathcal{A}_{bin}(\sigma(D), \ell)$ on a pair of datasets D and D' differing in the first record, where $\sigma(D)$ randomly swaps d_1 with d_I for I uniformly sampled from $[n]$.

Algorithm 4 $\mathcal{A}_{bin}$: DP-SGD with Bins

Input: Dataset $D = d_{1:n}$, bin sizes $\ell \in [n]^m$ with $\sum_i \ell_i = n$, local randomizer $\mathcal{A}_{ldp}$
Initialize model $\theta_1 \in \mathbb{R}^p$
$j \leftarrow 1$
for $i \in [m]$ **do**
  if $\ell_i = 0$ **then**
    $\theta_{i+1} \leftarrow \theta_i$
  else
    $\tilde{g}_i \leftarrow 0$
    for $k \in \{j, \ldots, j + \ell_i - 1\}$ **do**
      $\tilde{g}_i \leftarrow \tilde{g}_i + \mathcal{A}_{ldp}(d_k, \theta_i)$
    $j \leftarrow j + \ell_i$
    $\theta_{i+1} \leftarrow \text{ModelUpdate}(\theta_i; \tilde{g}_i/\ell_i)$
return sequence $\theta_{2:m+1}$

Theorem A.5. *Suppose $\mathcal{A}_{ldp} : \mathcal{D} \times \Theta \to \Theta$ is an ε_0-DP local randomizer. Let $\ell \in [m]^n$ with $\sum_i \ell_i = n$. Also, for any dataset $D = \{d_1, \ldots, d_n\}$, define $\sigma(D)$ be the operation that randomly swaps d_1 with d_I for I uniformly sampled from $[n]$. For any $\delta \in (0, 1)$, the mechanism $M(D) = \mathcal{A}_{bin}(\sigma(D), \ell)$ is (ε, δ)-DP at index 1 with $\varepsilon = \frac{\|\ell\|_2^2 e^{4\varepsilon_0}(e^{\varepsilon_0}-1)^2}{2n^2} + \frac{\|\ell\|_2 e^{2\varepsilon_0}(e^{\varepsilon_0}-1)}{n}\sqrt{2\log(1/\delta)}$. Furthermore, if $\mathcal{A}_{ldp}$ is $(\varepsilon_0, \delta_0)$-DP with $\delta_0 \leq$*

$\frac{(1-e^{-\varepsilon_0})\delta_1}{4e^{\varepsilon_0}\left(2+\frac{\ln(2/\delta_1)}{\ln(1/(1-e^{-5\varepsilon_0}))}\right)}$, *then* M *is* (ε', δ')*-DP with* $\varepsilon' = \frac{\|\ell\|_2^2 e^{32\varepsilon_0}(e^{8\varepsilon_0}-1)^2}{2n^2} + \frac{\|\ell\|_2 e^{16\varepsilon_0}(e^{8\varepsilon_0}-1)}{n}\sqrt{2\log(1/\delta)}$ *and* $\delta' = \delta + m(e^{\varepsilon'}+1)\delta_1$.

Proof. Let $\sigma(D) = (\tilde{d}_1, \ldots, \tilde{d}_n)$ denote the dataset after the swap operation. Using the bin sizes ℓ, we split this dataset into m_0 disjoint datasets $\tilde{D}_1, \ldots, \tilde{D}_{m_0}$ of sizes $|\tilde{D}_i| = \ell_i$ with $\tilde{D}_1 = (\tilde{d}_1, \ldots, \tilde{d}_{\ell_1})$, and so on. Note that each of the outputs is obtained as $\theta_{i+1} \leftarrow \mathcal{A}^{(i)}(\theta_i; \tilde{D}_i)$ with

$$\mathcal{A}^{(i)}(\theta_i; \tilde{D}_i) = \text{ModelUpdate}\left(\theta_i; \frac{1}{\ell_i}\sum_{\tilde{d}\in\tilde{D}_i} \mathcal{A}_{ldp}(\tilde{d}, \theta_i)\right)$$

By post-processing, each of the $\mathcal{A}^{(i)}$ is $(\varepsilon_0, \delta_0)$-DP.

The next step is to modify these mechanisms to reduce the analysis to a question about adaptive composition. Thus, we introduce mechanisms $\mathcal{B}^{(i)}$ for $i \in [m_0]$ that take as input the whole dataset D and the outputs $\theta_{1:i} = (\theta_1, \ldots, \theta_i)$ of the previous mechanisms. Mechanism $\mathcal{B}^{(i)}$ starts by splitting the dataset D into m_0 disjoint datasets $D_1, \ldots, D_{m_0}$ of sizes $|D_i| = \ell_i$ as above. Then, it returns $\mathcal{A}^{(i)}(\theta_i; \bar{D}_i)$ for a dataset $\bar{D}_i$ of size ℓ_i constructed as follows: with probability $p_i = \mathbf{Pr}[d_1 \in \tilde{D}_i | \theta_{1:i}]$ it takes $\bar{D}_i$ to be the dataset obtained by replacing a random element from D_i with d_1, and with probability $1 - p_i$ it takes $\bar{D}_i = D_i$. Note this construction preserves the output distribution since for any θ we have

$$\begin{aligned}\mathbf{Pr}[\mathcal{A}^{(i)}(\theta_i; \tilde{D}_i) = \theta | \theta_{1:i}] &= (1-p_i)\,\mathbf{Pr}[\mathcal{A}^{(i)}(\theta_i; D_i) = \theta | \theta_{1:i}, d_1 \notin \tilde{D}_i] \\ &\quad + \frac{p_i}{\ell_i}\sum_{d\in D_i} \mathbf{Pr}[\mathcal{A}^{(i)}(\theta_i; D_i \cup \{d_1\} \setminus \{d\}) = \theta | \theta_{1:i}, d_1 \in \tilde{D}_i] \\ &= \mathbf{Pr}[\mathcal{B}^{(i)}(\theta_{1:i}; D) = \theta]\end{aligned}$$

To bound the probabilities p_i we write:

$$\begin{aligned}p_i &= \mathbf{Pr}[d_1 \in \tilde{D}_i | \theta_{1:i}] \\ &= \frac{\mathbf{Pr}[\theta_{1:i} | d_1 \in \tilde{D}_i]\,\mathbf{Pr}[d_1 \in \tilde{D}_i]}{\mathbf{Pr}[\theta_{1:i}]} \\ &= \frac{\ell_i}{n}\frac{\mathbf{Pr}[\theta_{1:i} | d_1 \in \tilde{D}_i]}{\sum_{k\in[m_0]}\mathbf{Pr}[\theta_{1:i} | d_1 \in \tilde{D}_k]\,\mathbf{Pr}[d_1 \in \tilde{D}_k]} \\ &= \frac{\ell_i}{\sum_{k\in[m_0]}\ell_k\frac{\mathbf{Pr}[\theta_{1:i} | d_1 \in \tilde{D}_k]}{\mathbf{Pr}[\theta_{1:i} | d_1 \in \tilde{D}_i]}}\end{aligned}$$

To proceed, we assume $\delta_0 = 0$. If that is not the case, then the same argument based on Lemma A.3 used in the proof of Theorem A.4 allows us to reduce the analysis to the case $\delta_0 = 0$ and modify the final ε and δ accordingly. When the local randomizers satisfy pure DP, we have

$$\begin{aligned}\sum_{k\in[m_0]}\ell_k\frac{\mathbf{Pr}[\theta_{1:i} | d_1 \in \tilde{D}_k]}{\mathbf{Pr}[\theta_{1:i} | d_1 \in \tilde{D}_i]} &\geq \ell_i + e^{-2\varepsilon_0}\sum_{k<i}\ell_k + e^{-\varepsilon_0}\sum_{k>i}\ell_k \\ &\geq e^{-2\varepsilon_0}n\end{aligned}$$

Thus we obtain $p_i \leq e^{2\varepsilon_0}\ell_i/n$. Now, the overlapping mixtures argument used in the proof of Theorem A.4 (see [5]) shows that $\mathcal{B}^{(i)}$ is ε_i-DP with $\varepsilon_i \leq \log(1 + e^{2\varepsilon_0}(e^{\varepsilon_0}-1)\ell_i/n)$. Furthermore, the heterogenous advanced composition theorem [24] implies that the composition of $\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(m_0)}$ satisfies (ε, δ)-DP with

$$\begin{aligned}\varepsilon &= \sum_{i\in[k]}\frac{(e^{\varepsilon_i}-1)\varepsilon_i}{e^{\varepsilon_i}+1} + \sqrt{2\log\frac{1}{\delta}\sum_{i\in[k]}\varepsilon_i^2} \\ &\leq \frac{(e^{\varepsilon_0}-1)^2 e^{4\varepsilon_0}\,\|\ell\|_2^2}{2n^2} + \sqrt{\frac{2(e^{\varepsilon_0}-1)^2 e^{4\varepsilon_0}\,\|\ell\|_2^2}{n^2}\log\frac{1}{\delta}}\end{aligned}$$

□

To conclude the proof of Theorem 4.1, we provide a high probability bound for $\|L\|_2$ for random L representing the loads of m bins when n balls are thrown uniformly and independently.

Lemma A.6. *Let $L = (L_1, \ldots, L_m)$ denote the number of users checked in into each of m update slots in the protocol from Figure 2. With probability at least $1 - \delta$, we have*

$$\|L\|_2 \leq \sqrt{n + \frac{n^2}{m}} + \sqrt{n \log(1/\delta)}.$$

Proof. The proof is a standard application of McDiarmid's inequality. First note that $\|L\|_2$ is a function of n i.i.d. random variables indicating the bin where each ball is allocated. Since changing the assignment of one ball can only change $\|L\|_2$ by $\sqrt{2}$, we have

$$\|L\|_2 \leq \mathbb{E}\,\|L\|_2 + \sqrt{n \log(1/\delta)}$$

with probability at least $1 - \delta$. Finally, we use Jensen's inequality to obtain

$$\begin{aligned}\mathbb{E}\left[\|L\|_2\right] &\leq \sqrt{\mathbb{E}\left[\|L\|_2^2\right]} = \sqrt{\sum_{i \in [m]} \mathbb{E}\left[L_i^2\right]} = \sqrt{m \mathbb{E}\left[\mathrm{Bin}(n, 1/m)^2\right]} \\ &= \sqrt{m\left(\frac{n}{m}\left(1 - \frac{1}{m}\right) + \frac{n^2}{m^2}\right)} \leq \sqrt{n + \frac{n^2}{m}}\end{aligned}$$

□

The privacy claim in Theorem 4.1 follows from using Lemma A.6 to condition with probability at least $1 - \delta_2$ to the case where L is such that

$$\frac{\|L\|_2}{n} \leq \sqrt{\frac{1}{n} + \frac{1}{m}} + \sqrt{\frac{\log(1/\delta_2)}{n}}\ ,$$

and for each individual event $L = \ell$ satisfying this condition, applying the analysis from Theorem A.5 after the reduction from shuffling to averaging (see, e.g., the proof of Theorem 5.1 below).

A.3 Proof of Theorem 5.1

Algorithm 5 $\mathcal{A}_{sl}$: Local responses with shuffling

Input: Dataset $D = d_{1:n}$, algorithms $\mathcal{A}_{ldp}^{(i)} : \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(i-1)} \times \mathcal{D} \to \mathcal{S}^{(i)}$ for $i \in [n]$.

1: Let π be a uniformly random permutation of $[n]$
2: **for** $i \in [n]$ **do**
3: $\quad s_i \leftarrow \mathcal{A}_{ldp}^{(i)}(s_{1:i-1}; d_{\pi(i)})$
4: **return** sequence $s_{1:n}$

We will prove the privacy guarantee of $\mathcal{A}_{sl}$ (Algorithm 5) in a similar manner as in the proof of Theorem 7 in [19]: by reducing $\mathcal{A}_{sl}$ to $\mathcal{A}_{swap}$ that starts by swapping the first element with a u.a.r. sample in the dataset, and then applies the local randomizers (Algorithm 6). They key difference between our proof and the one in [19] is that we provide tighter, position-dependent privacy guarantees for each of the outputs of $\mathcal{A}_{swap}$, and then use an *heterogeneous* adaptive composition theorem from [24] to compute the final privacy parameters.

Algorithm 6 $\mathcal{A}_{swap}$: Local responses with one swap

Input: Dataset $D = d_{1:n}$, algorithms $\mathcal{A}_{ldp}^{(i)} : \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(i-1)} \times \mathcal{D} \to \mathcal{S}^{(i)}$ for $i \in [n]$.

Sample $I \xleftarrow{u.a.r.} [n]$

Let $\sigma_I(D) \leftarrow (d_I, d_2, \ldots, d_{I-1}, d_1, d_{I+1}, \ldots, d_n)$

for $i \in [c]$ **do**

  $s_i \leftarrow \mathcal{A}_{ldp}^{(i)}(s_{1:i-1}; \sigma_I(D)[i])$

return sequence $s_{1:n}$

Theorem A.7. *(Amplification by swapping) For a domain $\mathcal{D}$, let $\mathcal{A}_{ldp}^{(i)} : \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(i-1)} \times \mathcal{D} \to \mathcal{S}^{(i)}$ for $i \in [n]$ (where $\mathcal{S}^{(i)}$ is the range space of $\mathcal{A}_{ldp}^{(i)}$) be a sequence of algorithms s.t. $\mathcal{A}_{ldp}^{(i)}$ is ε_0-DP for all values of auxiliary inputs in $\mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(i-1)}$. Let $\mathcal{A}_{swap} : \mathcal{D}^n \to \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(n)}$ be the algorithm that given a dataset $D = d_{1:n} \in \mathcal{D}^n$, swaps the first element in D with an element sampled u.a.r. in D, and then applies the local randomizers to the resulting dataset sequentially (see Algorithm 6). $\mathcal{A}_{swap}$ satisfies (ε, δ)-DP at index 1 in the central model, for $\varepsilon = \frac{e^{3\varepsilon_0}(e^{\varepsilon_0}-1)^2}{2n} + e^{3\varepsilon_0/2}(e^{\varepsilon_0}-1)\sqrt{\frac{2\log(1/\delta)}{n}}$. Furthermore, if the $\mathcal{A}^{(i)}$ are $(\varepsilon_0, \delta_0)$-DP with $\delta_0 \leq \frac{(1-e^{-\varepsilon_0})\delta_1}{4e^{\varepsilon_0}\left(2+\frac{\ln(2/\delta_1)}{\ln(1/(1-e^{-\delta\varepsilon_0}))}\right)}$, then $\mathcal{A}_{swap}$ is (ε', δ')-DP with $\varepsilon' = \frac{e^{24\varepsilon_0}(e^{8\varepsilon_0}-1)^2}{2n} + e^{12\varepsilon_0}(e^{8\varepsilon_0}-1)\sqrt{\frac{2\log(1/\delta)}{n}}$ and $\delta' = \delta + m(e^{\varepsilon'}+1)\delta_1$.*

Proof. We start by proving the privacy guarantee of $\mathcal{A}_{swap}$ for the case where for each $i \in [c]$, the local randomizer $\mathcal{A}_{ldp}^{(i)}$ is ε_0-DP, i.e., for the case where $\delta_0 = 0$. Let us denote the output sequence of $\mathcal{A}_{swap}$ by $Z_1, Z_2, \ldots, Z_n$. Note that $Z_{1:n}$ can be seen as the output of a sequence of n algorithms with conditionally independent randomness: $\mathcal{B}^{(i)} : \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(i-1)} \times \mathcal{D}^n \to \mathcal{S}^{(i)}$ for $i \in [n]$. On input $s_{1:i-1}$ and D, $\mathcal{B}^{(i)}$ outputs a random sample from the distribution of $Z_i | Z_{1:i-1} = s_{1:i-1}$. The outputs of $\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(i-1)}$ are given as input to $\mathcal{B}^{(i)}$. Therefore, in order to upper bound the privacy parameters of $\mathcal{A}_{swap}$, we analyze the privacy parameters of $\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(n)}$ and apply the heterogeneous advanced composition for DP [24].

Next, observe that conditioned on the value of I, Z_i is the output of $\mathcal{A}_{ldp}^{(i)}(s_{1:i-1}; d)$ with its internal randomness independent of $Z_{1:i-1}$. In particular, for $i \geq 2$, one can implement $\mathcal{B}^{(i)}$ as follows. First, sample an index T from the distribution of $I | Z_{1:i-1} = s_{1:i-1}$. Output $\mathcal{A}_{ldp}^{(i)}(s_{1:i-1}; d_1)$ if $T = i$, otherwise output $\mathcal{A}_{ldp}^{(i)}(s_{1:i-1}; d_i)$. For $\mathcal{B}^{(1)}$, we first sample T u.a.r. from $[n]$, and then output $\mathcal{A}_{ldp}^{(1)}(d_T)$.

We now prove that for each $i \in [c]$, $\mathcal{B}^{(i)}$ is $\left(\log\left(1 + \frac{e^{2\varepsilon_0}(e^{\varepsilon_0}-1)}{e^{2\varepsilon_0}+(i-1)+(n-i)e^{\varepsilon_0}}\right), 0\right)$-DP at index 1. Let $D = d_{1:n}$ and $D' = (d_1', d_{2:n})$ be 2 datasets differing in the first element. Let $s_{1:i-1}$ denote the input to $\mathcal{B}^{(i)}$. Let μ be the probability distribution of $\mathcal{B}^{(i)}(s_{1:i-1}; D)$, and let μ_0 (resp. μ_1) be the distribution of $\mathcal{B}^{(i)}(s_{1:i-1}; D)$ conditioned on $T \neq i$ (resp. $T = i$). Let q_i be the probability that $T = i$ (sampled from $I | Z_{1:i-1} = s_{1:i-1}$). By definition, $\mu = (1-q_i)\mu_0 + q_i\mu_1$. Also, denote by μ', μ_0', μ_1', and q_i' the corresponding quantities when $\mathcal{B}^{(i)}$ is run on D'. Thus, we get $\mu' = (1-q_i')\mu_0' + q_i'\mu_1'$.

For $i \in [n]$, we observe that $\mu_0 = \mu_0'$, since in both cases the output is generated by $\mathcal{A}_{ldp}^{(i)}(d_T)$ conditioned on $T \neq 1$ for $i = 1$, and $\mathcal{A}_{ldp}^{(i)}(s_{1:i-1}; d_i)$ for $i \geq 2$. W.l.o.g. assume that $q_i \geq q_i'$. Thus, we can shift $q_i - q_i'$ mass from the first component of the mixture in μ' to the second component to obtain

$$\mu' = (1-q_i)\mu_0 + q_i\left(\frac{q_i'}{q_i}\mu_1' + \left(1 - \frac{q_i'}{q_i}\right)\mu_0\right) = (1-q_i)\mu_0 + q_i\mu_1''$$

This shows that μ and μ' are overlapping mixtures [5]. Now, ε_0-LDP of $\mathcal{A}_{ldp}^{(i)}$ implies $\mu_0 \approx_{(\varepsilon_0,0)} \mu_1$ and $\mu_0 \approx_{(\varepsilon_0,0)} \mu_1'$. Moreover, ε_0-LDP of $\mathcal{A}_{ldp}^{(i)}$ also implies $\mu_1 \approx_{(\varepsilon_0,0)} \mu_1'$, so by the joint convexity of the relation $\approx_{(\varepsilon_0,0)}$ we also have $\mu_1 \approx_{(\varepsilon_0,0)} \mu_1''$. Thus, we can apply Advanced Joint Convexity of overlapping mixtures (Theorem 2 in [5]) to get that

$$\mu \approx_{(\log(1+q_i(e^{\varepsilon_0}-1)),0)} \mu' \tag{13}$$

We now claim that $q_i \leq \frac{e^{2\varepsilon_0}}{e^{2\varepsilon_0}+(i-1)+(n-i)e^{\varepsilon_0}}$. Observe that for each $D^* \in \{D, D'\}$, conditioning on $T = i$ reduces $\mathcal{A}_{swap}$ to running $\mathcal{A}_{ldp}^{(k)}, k \in [n]$ on $\sigma_i(D^*)$. Note that $\sigma_i(D^*)[1:i-1]$ differs from $\sigma_j(D^*)[1:i-1]$ in at most 2 positions for $j < i$, and at most 1 position for $j > i$. By ε_0-LDP of $\mathcal{A}_{ldp}^{(k)}, k \in [n]$, we get that

$$\frac{\mathbf{Pr}[Z_{1:i-1} = s_{1:i-1}|T=i]}{\mathbf{Pr}[Z_{1:i-1} = s_{1:i-1}|T=j]} \leq e^{2\varepsilon_0} \text{ for } j < i \quad \text{and} \quad \frac{\mathbf{Pr}[Z_{1:i-1} = s_{1:i-1}|T=i]}{\mathbf{Pr}[Z_{1:i-1} = s_{1:i-1}|T=j]} \leq e^{\varepsilon_0} \text{ for } j > i \tag{14}$$

Now, on the lines of the proof of Lemma A.1, we have:

$$\begin{aligned}
&\frac{\mathbf{Pr}[Z_{1:i-1} = s_{1:i-1}|T=i]}{\mathbf{Pr}[Z_{1:i-1} = s_{1:i-1}]} \\
&= \frac{\mathbf{Pr}[Z_{1:i-1} = s_{1:i-1}|t=i]}{\sum_{j=1}^{n} \mathbf{Pr}[Z_{1:i-1} = s_{1:i-1}|T=j]\,\mathbf{Pr}[T=j]} \\
&= \frac{1}{\sum_{j=1}^{n} \frac{\mathbf{Pr}[Z_{1:i-1}=s_{1:i-1}|t=j]}{\mathbf{Pr}[Z_{1:i-1}=s_{1:i-1}|t=i]}\,\mathbf{Pr}[T=j]} \\
&= \frac{n}{1+(i-1)\sum_{j<i} \frac{\mathbf{Pr}[Z_{1:i-1}=s_{1:i-1}|T=j]}{\mathbf{Pr}[Z_{1:i-1}=s_{1:i-1}|T=i]} + (n-i)\sum_{k>i} \frac{\mathbf{Pr}[Z_{1:i-1}=s_{1:i-1}|T=k]}{\mathbf{Pr}[Z_{1:i-1}=s_{1:i-1}|T=i]}} \\
&\leq \frac{n}{1+(i-1)e^{-2\varepsilon_0}+(n-i)e^{-\varepsilon_0}} = \frac{ne^{2\varepsilon_0}}{e^{2\varepsilon_0}+(i-1)+(n-i)e^{\varepsilon_0}}
\end{aligned}$$

where the third equality follows as for every $j \in [n], \mathbf{Pr}[T=j] = \frac{1}{n}$, and the first inequality follows from inequality 14.

This immediately implies our claim, since

$$\begin{aligned}
q_i = \mathbf{Pr}[T=i|Z_{1:i-1} = s_{1:i-1}] &= \frac{\mathbf{Pr}[Z_{1:i-1} = s_{1:i-1}|T=i] \cdot \mathbf{Pr}[T=i]}{\mathbf{Pr}[Z_{1:i-1} = s_{1:i-1}]} \\
&\leq \frac{e^{2\varepsilon_0}}{e^{2\varepsilon_0}+(i-1)+(n-i)e^{\varepsilon_0}}
\end{aligned}$$

where the inequality follows from (11), and as $\mathbf{Pr}[T=i] = \frac{1}{n}$. Substituting the value of q_i in (13), we get that for each $i \in [n]$, algorithm $\mathcal{B}^{(i)}$ is $(\varepsilon_i, 0)$-DP at index 1, where $\varepsilon_i = \log\left(1 + \frac{e^{2\varepsilon_0}(e^{\varepsilon_0}-1)}{e^{2\varepsilon_0}+(i-1)+(n-i)e^{\varepsilon_0}}\right)$. This results in $\varepsilon_i \leq \log\left(1 + \frac{e^{\varepsilon_0}(e^{\varepsilon_0}-1)}{n-(i-1)\left(1-\frac{1}{e^{\varepsilon_0}}\right)}\right)$, and using Lemma A.2 for the sequence of mechanisms $\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(n)}$ by setting $a = e^{\varepsilon_0}(e^{\varepsilon_0}-1)$, $b = 1 - \frac{1}{e^{\varepsilon_0}}$, and $k = n$, we get that algorithm $\mathcal{A}_{swap}$ satisfies (ε, δ)-DP at index 1, for $\varepsilon = \frac{e^{3\varepsilon_0}(e^{\varepsilon_0}-1)^2}{2n} + e^{3\varepsilon_0/2}(e^{\varepsilon_0}-1)\sqrt{\frac{2\log(1/\delta)}{n}}$.

The case $\delta_0 > 0$ uses the same argument based on Lemma A.3 used in the proof of Theorem A.4. This arguments allows us to reduce the analysis to the case $\delta_0 = 0$ and modify the final ε and δ accordingly.

□

Now, we are ready to prove Theorem 5.1.

Proof of Theorem 5.1. This proof proceeds in a similar manner as the proof of Theorem 7 in [19]. Let D and D' be 2 datasets of length n that differ at some index $i^* \in [n]$. Algorithm $\mathcal{A}_{sl}$ can be alternatively seen as follows. Pick a random one-to-one mapping π^* from $\{2, \ldots, n\} \to [n] \setminus \{i^*\}$ and let $\pi^*(D) = (d_{i^*}, d_{\pi^*(2)}, \ldots, d_{\pi^*(n)})$. Next, apply $\mathcal{A}_{swap}$ to $\pi^*(D)$. It is easy to see that for a u.a.r. chosen π^* and u.a.r. $I \in [n]$, the distribution of $\sigma_I(\pi^*(D))$ is a uniformly random permutation of elements in D.

For a fixed π^*, we know that $\pi^*(D)$ and $\pi^*(D')$ differ only in the element with index 1. Therefore, in the case $\delta_0 = 0$, from Theorem A.7, we get $\mathcal{A}_{swap}(\pi^*(D)) \approxeq_{\varepsilon,\delta} \mathcal{A}_{swap}(\pi^*(D'))$ at index 1, for $\varepsilon = \frac{e^{3\varepsilon_0}(e^{\varepsilon_0}-1)^2}{2n} + e^{3\varepsilon_0/2}(e^{\varepsilon_0}-1)\sqrt{\frac{2\log(1/\delta)}{n}}$, which implies $\mathcal{A}_{sl}(D) \approxeq_{\varepsilon,\delta} \mathcal{A}_{sl}(D')$.

The case $\delta_0 > 0$ follows similarly from the corresponding setting of Theorem A.7. □