

Shuffled Check-in: Privacy Amplification towards Practical Distributed Learning

Seng Pei Liew *

Satoshi Hasegawa *

Tsubasa Takahashi *

Abstract

Recent studies of distributed computation with formal privacy guarantees, such as differentially private (DP) federated learning, leverage random sampling of clients in each round (privacy amplification by subsampling) to achieve satisfactory levels of privacy. Achieving this however requires strong assumptions which may not hold in practice, including precise and uniform subsampling of clients, and a highly trusted aggregator to process clients' data. In this paper, we explore a more practical protocol, shuffled check-in, to resolve the aforementioned issues. The protocol relies on client making independent and random decision to participate in the computation, freeing the requirement of server-initiated subsampling, and enabling robust modelling of client dropouts. Moreover, a weaker trust model known as the shuffle model is employed instead of using a trusted aggregator. To this end, we introduce new tools to characterize the Rényi differential privacy (RDP) of shuffled check-in. We show that our new techniques improve at least three times in privacy guarantee over those using approximate DP's strong composition at various parameter regimes. Furthermore, we provide a numerical approach to track the privacy of generic shuffled check-in mechanism including distributed stochastic gradient descent (SGD) with Gaussian mechanism. To the best of our knowledge, this is also the first evaluation of Gaussian mechanism within the local/shuffle model under the distributed setting in the literature, which can be of independent interest.

1 Introduction

Cross-device federated learning (FL) has attracted attention lately as a scalable and privacy-friendly computational framework of large-scale machine learning. In this framework, clients send model updates or gradients to the governing server keeping their data decentralized, while the server aggregates the gradients to update the model, and sends the updated model back to the clients to initiate next round of training [27, 33]. There have been efforts combining FL with differential privacy (DP) [16, 17] to achieve rigorous privacy guarantees [34]. Typically, a subsampling procedure is taken, where in each round of training, only clients sampled in a uniform and random manner by the server participate in the training. Once all sampled clients finish training and sending out the gradients, the server aggregates the gradients to update the model, assuming that the server does not leak information (such as the subsampled clients) other than the model to adversaries (server is trustworthy). This randomness of subsampling leads to privacy amplification, critical at achieving acceptable levels of utility under meaningful DP guarantees [1, 7, 29, 38]. That there is a trustworthy server collecting raw data and adding noises to the aggregated gradients achieves "central" DP for the system.

Such a system may not be easily realizable in practice however. As noted in [6, 26, 27], the server-initiated sampling is "impossible in practice". Clients can fail to follow the server's command or drop out due to issues such as network disconnection or battery outage. Typically, the server also discards

*LINE Corporation, {sengpei.liew, satoshi.hasegawa, tsubasa.takahashi}@linecorp.com

stragglers, or clients that take a much longer time to complete the training due to reasons such as hardware capabilities, as they affect the time of completing a round of training [11]. Subsequently, the number of participating in each round becomes a *variable*, an effect not taken into account when quantifying the privacy of the system using the standard approach.

In this paper, we propose a novel protocol of distributed computation, **shuffled check-in**, to tackle the aforementioned issues. First, our protocol lets clients utilize their own randomness to decide (with a certain probability) whether to participate in the server-initiated training, without being specifically indexed by the server. This consequently reduces the reliance on the orchestrating server. As explained later, our protocol also allows the modelling of drop-out as a random (probabilistic) event.

Next, instead of relying on a highly trusted aggregating server, we propose to use a weaker trust model, the *shuffler*, or the shuffle model, that is responsible for data anonymization [14, 20]. Deploying a shuffler in practice is simpler implementation-wise and requires fewer trust assumptions compared to an aggregating server. For example, the Prochlo [10] implementation leverages the Trusted Execution Environment (TEE) to perform shuffling. Raw data are not exposed to the shuffler, in contrast to the aggregating server, as its sole responsibility is to mask the data origin (shuffling can be performed on encrypted data by, e.g., removing only metadata or identifiers, such that sensitive contents are not exposed to the shuffler). Other realizations of the shuffle model include the utilization of mix-nets [13, 14] and peer-to-peer protocols [32], which also do not expose raw data to other entities.

More concretely, our contributions are as follows.

- We propose shuffled check-in, a privacy amplification technique to address practical issues of distributed systems, leveraging clients’ randomness and the recently proposed shuffle model.
- We give a detailed privacy analysis of our proposal, particularly utilizing Rényi differential privacy (RDP) to account for the composition of privacy loss. Analytical upper and lower bounds on the RDP parameters for shuffled check-in with discrete ϵ_0 -local DP randomizer are provided. Furthermore, we propose a numerical approach of calculating the RDP bound of shuffled check-in with generic local DP randomizer.
- We evaluate our proposal to demonstrate its performance with machine learning tasks under the distributed setting. We experiment with a standard LDP-SGD [15, 19] algorithm, and a modified version of DP-SGD [1, 7, 36], Federated DP-SGD. Note that to our best knowledge, Federated DP-SGD which utilizes the popular Gaussian mechanism has not been evaluated within the local/shuffle privacy model before. Our evaluation can henceforth be of independent interest.

Related work. [24] has studied a similar protocol, but only an order approximation of the privacy accounting based on advanced composition is given. We give a precise and analytical result based on RDP that leads to tighter accounting instead. Our protocol is also different from the subsampled shuffle mechanism studied in [22], which assumes that the number of subsampled users is fixed; our protocol requires different privacy accounting techniques as a result. That client making independent decision to participate in distributed computation has also been considered in [6], but their scheme relies on a highly trusted aggregator/orchestrating server, in contrast to our intermediate-trust shuffle model. We finally note that various aspects of the shuffle model as an intermediate trust model are studied in the literature [5, 14, 19–22, 25, 30].

2 Preliminaries and Problem Setup

We first present important definitions and known results of differential privacy.

Definition 1 (Central Differential Privacy [18]). Given $\epsilon \geq 0$ and $\delta \geq 0$, a randomization mechanism, $\mathcal{M} : \mathcal{D}^n \rightarrow \mathcal{S}$ with domain \mathcal{D}^n and range \mathcal{S} satisfies central (ϵ, δ) -differential privacy (DP) if for any two adjacent databases $D, D' \in \mathcal{D}^n$ with n data instances and for any subset of outputs $S \subseteq \mathcal{S}$, the following holds:

$$\Pr[\mathcal{M}(D) \in S] \leq e^\epsilon \cdot \Pr[\mathcal{M}(D') \in S] + \delta. \quad (1)$$

We say that an (ϵ, δ) -DP mechanism satisfies *approximate* DP, or is (ϵ, δ) -indistinguishable. (ϵ, δ) -DP is also simply referred to as “DP” when the context is clear. Moreover, we note that we work with the

"replacement" version of DP, where adjacent databases have one data instance replaced by another data instance.

When D consists of only a single element, the mechanism, also known as a local randomizer, is said to be satisfying local DP:

Definition 2 (Local Differential Privacy (LDP) [29]). A randomization mechanism $\mathcal{A} : \mathcal{D} \rightarrow \mathcal{S}$ satisfies local (ϵ, δ) -DP if for all pairs $x, x' \in \mathcal{D}$, $\mathcal{A}(x)$ and $\mathcal{A}(x')$ are (ϵ, δ) -indistinguishable.

We often refer to a mechanism satisfying $(\epsilon, 0)$ LDP as an ϵ -LDP randomizer.

We next introduce Rényi differential privacy, the main privacy notion used in this paper.

Definition 3 (Rényi Differential Privacy (RDP) [35]). A randomization mechanism $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{S}$ is ϵ -Rényi differential privacy of order $\lambda \in (1, \infty)$ (or (λ, ϵ) -RDP), if for any adjacent databases $D, D' \in \mathcal{D}$, the Rényi divergence of order λ between $\mathcal{M}(D)$ and $\mathcal{M}(D')$ is upper-bounded by ϵ :

$$D_\lambda(\mathcal{M}(D) || \mathcal{M}(D')) = \frac{1}{\lambda - 1} \log \left(\mathbb{E}_{\phi \sim \mathcal{M}(D')} \left[\left(\frac{\mathcal{M}(D)(\phi)}{\mathcal{M}(D')(\phi)} \right)^\lambda \right] \right) \leq \epsilon, \quad (2)$$

where $\mathcal{M}(\mathcal{D})(\phi)$ denotes \mathcal{M} taking D as input to output ϕ with certain probability.

We sometimes write ϵ as $\epsilon(\lambda)$ to indicate that it is a function of λ . The main strength of RDP lies in its composition property, which is cleaner than approximate DP. The formal description is as follows.

Lemma 1 (Adaptive composition of RDP [35]). *Given two mechanisms $\mathcal{M}_1, \mathcal{M}_2$ taking $D \in \mathcal{D}$ as input that are $(\lambda, \epsilon_1), (\lambda, \epsilon_2)$ -RDP respectively, the composition of \mathcal{M}_1 and \mathcal{M}_2 satisfies $(\lambda, \epsilon_1 + \epsilon_2)$ -RDP.*

While the privacy accounting involving composition is preferably done in terms of RDP, we often need to convert the RDP notion back to the approximate DP notion in the final step. This is achieved with the following.

Lemma 2 (RDP-to-DP conversion [4, 12]). *If a mechanism \mathcal{M} is $(\lambda, \epsilon(\lambda))$ -RDP, \mathcal{M} is also (ϵ, δ) -DP, where $1 < \delta < 0$ is arbitrary and ϵ is given by*

$$\epsilon = \min_{\lambda} \left(\epsilon(\lambda) + \frac{\log(1/\delta) + (\lambda - 1) \log(1 - 1/\lambda) - \log(\lambda)}{\lambda - 1} \right). \quad (3)$$

Given an approximate DP mechanism, we may wish to convert it to the RDP to obtain tighter composition-based privacy accounting. The conversion is performed with the following lemma.

Lemma 3 (DP-to-RDP conversion [2]). *If a mechanism \mathcal{M} is (ϵ, δ) -DP, \mathcal{M} is also $(\lambda, \epsilon(\lambda))$ -RDP, where $\lambda > 1$ and $\epsilon(\lambda)$ is given by*

$$\epsilon(\lambda) = \min_{r \in (\delta, 1)} \left(r^\lambda (r - \delta)^{1-\lambda} + (1 - r)^\lambda (e^\epsilon - r + \delta)^{1-\lambda} \right). \quad (4)$$

2.1 Problem setup

We consider the task of learning under the distributed setting,² where there are n clients each holding a data record x_i for $i \in [n]$. The whole decentralized dataset is denoted by $D = (x_1, \dots, x_n)$.

Clients (assumed to be non-malicious) are connected to the trusted *shuffler* via secure communication channels, and the trusted shuffler also communicates with an untrusted *aggregator*. The purpose of the system is to train a model with parameter $\theta \in \Theta$ by minimizing a certain loss function $l : \mathcal{D}^n \times \Theta \rightarrow \mathbb{R}_+$ via stochastic gradient descent (SGD), while providing clients with formal privacy guarantees.

3 Shuffled Check-in

3.1 The protocol

We begin with describing the protocol of shuffled check-in. In each round t , a message is broadcast to all clients to ask for participation in the learning. Each client flips a biased coin to decide whether

²While our protocol is applicable to any distributed computation, we focus on distributed learning (FL) as a concrete use case.

Server-side protocol:

parameters: Number of rounds T
 Initialize model θ_1
for $t \in [T]$ **do**
 Broadcast model θ_t and request users to participate in training
 Shuffler receives client responses and outputs the shuffled messages (\tilde{g}_t) to aggregator
 $\theta_{t+1} \leftarrow \text{ModelUpdate}(\theta_t; \tilde{g}_t)$
 Output θ_{t+1}

Client-side protocol for User i :

parameters: probability p , loss function l , local randomizer \mathcal{A} , Number of rounds T
private data: $x_i \in \mathcal{D}$
for $t \in [T]$ **do**
 Receive request to participate in training
 if a p -biased coin returns head **then**
 Download θ_t
 $\tilde{g}_t \leftarrow \mathcal{A}(\nabla_{\theta} l(\theta_t; x_i))$
 Send \tilde{g}_t to shuffler

Algorithm 1: The distributed protocol of shuffled check-in.

to participate in the training. The probability of a client participating in the training successfully is modeled by a parameter γ ($0 \leq \gamma \leq 1$), which we call the *check-in rate*. The participating probability follows the Bernoulli distribution, $\text{Bern}(\gamma)$. In Section 3.3, we discuss how to determine γ in practice.

Clients decided to participate download the model θ_t , calculate the gradient, apply local randomizer to it (e.g., clip the gradient norm and add Gaussian noise), and send it (encrypted) to the trusted shuffler. The shuffler permutes randomly all the received messages before forwarding them to the untrusted aggregator. Then, the aggregator updates the model parameters to θ_{t+1} using the shuffled messages. The system protocol is described in Algorithm 1.

More formally, let shuff be the shuffling mechanism that randomly permutes any number of received messages and outputs them. The shuffled check-in mechanism can be written as

$$\mathcal{M}(D) := \text{shuff}(\{\mathcal{A}(x_i) | \sigma_i = 1, i \in [n]\}), \quad \sigma_i \sim \text{Bern}(\gamma). \quad (5)$$

3.2 RDP for shuffled check-in

Under the setting given in the previous subsection, we derive the RDP of the shuffled check-in mechanism, which is summarized by the following theorem.

Theorem 1 (RDP of shuffled check-in). *Let D, D' be adjacent databases consisting of n clients, and γ the (effective) check-in rate. The RDP of order λ of the shuffled check-in mechanism is bounded by*

$$\epsilon(\lambda) \leq \frac{1}{\lambda - 1} \log \left(\sum_{k=0}^n \binom{n}{k} \gamma^k (1 - \gamma)^{n-k} \mathbb{E}_{\phi_k \sim q} \left[\left(\frac{p(\phi_k)}{q(\phi_k)} \right)^\lambda \right] \right). \quad (6)$$

Here, $p(\phi_k)$ is the output probability distribution of the mechanism $\mathcal{P}_k : \mathcal{D}^n \rightarrow S^{(1)} \times \dots \times S^{(k)}$, which is a subsampled without replacement (of k data instances), shuffle mechanism. $q(\phi_k)$ is the output probability distribution induced by the same mechanism but on D' .

Proof. We first note that the shuffled check-in mechanism \mathcal{M} can be written as $\mathcal{M} : \mathcal{D}^n \rightarrow 2^n \times \mathcal{S}^n$, as it outputs a subset of shuffled data processed by a LDP randomizer $\mathcal{A} : \mathcal{D} \rightarrow \mathcal{S}$. Notice that there are two variables in \mathcal{M} , $k \in [n]$, the number of data instances, and $\phi \in \mathcal{S}$, the randomized content of each data instance.

As clients check in independently with a probability of γ , the total number of clients checking in distributes as a binomial distribution. Hence, the mechanisms distribute as

$$\begin{aligned} \mathcal{M}(\mathcal{D})(k, \phi) &\sim \binom{n}{k} \gamma^k (1 - \gamma)^{n-k} p(\phi_k) \\ \mathcal{M}(\mathcal{D}')(k, \phi) &\sim \binom{n}{k} \gamma^k (1 - \gamma)^{n-k} q(\phi_k). \end{aligned}$$

To complete the proof, we need to show that $p(\phi_k)$ is the probability distribution of a subsampled (without replacement) shuffle mechanism. For brevity, we write $p(\phi_k)$ and $q(\phi_k)$ as p_k and q_k respectively. We note that both p_k and q_k are mixture distributions consisting of distribution with (and without) the differing data instance. More precisely, let the differing data instance be the n -th

item of D, D' . Let E be the subset containing the n -th item (E^c the subset with the complementary item). Subsequently,

$$p_k = (1 - \gamma)p_k(\cdot|E^c) + \gamma p_k(\cdot|E), \quad q_k = (1 - \gamma)q_k(\cdot|E^c) + \gamma q_k(\cdot|E)$$

and $p_k(\cdot|E^c) = q_k(\cdot|E^c)$. This expression is by definition the distribution corresponding to the subsampling without replacement scenario [38]. Plugging the above expressions to Equation 2, we obtain Equation 6 as desired. ■

Theorem 1 states effectively that the shuffled check-in's RDP is composed of subsampled shuffle mechanism's RDP weighted by a binomial distribution. This allows us to leverage known behaviors of subsampling and shuffling to calculate the RDP of shuffled check-in, as described in the next section.

3.3 Practical considerations

In our protocol, each client carries a p -biased coin such that she would decide to participate in the training only when head is returned. Even after a client decides to participate, she may *drop out* due to various reasons as discussed in Introduction. Assuming that clients have a certain constant and independent probability of dropping out, p' , the *effective* check-in rate is then $\gamma = p(1 - p')$. Hence, a practitioner should monitor the number of client checking in in each round and determine the empirical γ instead of simply using p as the check-in rate to model dropouts robustly. Moreover, instead of treating p' as a constant, one could further better model the dropout rate as, e.g., a time-dependent function, as in reliability engineering [8], but we leave this system-dependent and slightly orthogonal consideration for future work.

4 RDP Bounds on Shuffled Check-in

Theorem 1 formalizes the RDP of the shuffled check-in mechanism. Based on the theorem, in this section, we characterize the RDP by first presenting upper and lower RDP bounds of shuffled check-in with discrete ϵ_0 -LDP randomizer, which can be expressed in a clean and straightforward way. Then, we present the procedures for calculating the generic RDP bounds, which, although computationally heavy, have wider applications.

4.1 Upper and lower bounds with discrete ϵ_0 -LDP randomizer

A variety of discrete ϵ_0 -LDP SGD methods has been applied to distributed learning [9, 19, 23]. Roughly speaking, the following procedures are taken in these algorithms: clients clip the l_p -norm ($p \in [1, \infty]$) of the gradient, apply an ϵ_0 -LDP mechanism to randomize and represent the clipped gradient with a finite number of bits, and send it to the server. We provide the RDP bounds on these types of mechanism, beginning with the upper bound.

Theorem 2 (Upper Bound). *For any $\epsilon_0 \geq 0$, $n \in \mathbb{N}$, $l \leq n$ such that $n\gamma = l$ ($0 \leq \gamma \leq 1$), and any integer $\lambda \geq 2$, the RDP of the shuffled check-in mechanism (Equation 1) is upper-bounded by*

$$\epsilon(\lambda) \leq \frac{1}{\lambda-1} \log \left(\mathbb{E} \left[\left(\frac{\mathcal{M}(D)}{\mathcal{M}(D')} \right)^\lambda \right] \right), \text{ where}$$

$$\begin{aligned} \mathbb{E} \left[\left(\frac{\mathcal{M}(D)}{\mathcal{M}(D')} \right)^\lambda \right] &\leq 1 + 4 \binom{\lambda}{2} \gamma^2 (e^{\epsilon_0} - 1)^2 \left(e^{-\epsilon_0 - \Delta^2 n \gamma / 2} + e^{-\epsilon_0} / \tilde{l} \right) \\ &\quad + \sum_{j=3}^{\lambda} \binom{\lambda}{j} \gamma^j j \Gamma(j/2) \left(\frac{2(e^{2\epsilon_0} - 1)^2}{e^{2\epsilon_0}} \right)^{j/2} \left(e^{-\Delta^2 n \gamma / 2} + \tilde{l}^{-j/2} \right) \\ &\quad + \Upsilon_1 e^{-\Delta^2 n \gamma / 2} + \Upsilon_{(1-\Delta)n\gamma+1}. \end{aligned} \quad (7)$$

Here, $0 \leq \Delta \leq 1$ is arbitrary, $\tilde{l} = \lfloor \frac{(1-\Delta)n\gamma}{2\epsilon_0} \rfloor + 1$, $\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx$ is the Gamma function, Υ_k is given by $\Upsilon_k = \left(\left(1 + \gamma \frac{e^{2\epsilon_0} - 1}{e^{\epsilon_0}} \right)^\lambda - 1 - \lambda \gamma \frac{e^{2\epsilon_0} - 1}{e^{\epsilon_0}} \right) e^{-\frac{k-1}{8\epsilon_0}}$.

Proof sketch. Here, we give an outline of the proof. We notice from Equation 6 that inside the logarithm is a summation over k of the expected moments of a subsampled without replacement (with k data instances), shuffle mechanism. We first obtain a bound on the latter using the results from [22]. Then, the summation over k is bounded using the properties of the expected moments and the Chernoff bound. The full proof is available in Appendix A. ■

While Equation 7 may seem complicated, it is necessary as we attempt to obtain a precise expression (instead of an order approximation). This is particularly critical when considering deployment in real-life private systems, as constants matter [38]. Some numerical issues related to its evaluation will be discussed in later remarks and sections.

We next provide the corresponding lower bound.

Theorem 3 (Lower Bound). *For any $\epsilon_0 \geq 0$, $n \in \mathbb{N}$, $l \leq n$ such that $0 \leq \gamma \leq 1$, and any integer $\lambda \geq 2$, the RDP of the shuffled check-in mechanism (Equation 1) is lower-bounded by*

$$\epsilon(\lambda) \geq \frac{1}{\lambda - 1} \log \left(1 + (1 - e^{-\Delta^2 n \gamma / (2 + \Delta)}) \binom{\lambda}{2} \gamma^2 \frac{(e^{\epsilon_0} - 1)^2}{(1 + \Delta) n \gamma e^{\epsilon_0}} \right), \quad (8)$$

where $0 \leq \Delta \leq 1$ is arbitrary.

Proof sketch. The lower bound is calculated assuming that the underlying ϵ_0 -LDP discrete mechanism is a binary randomized response. Again, we first obtain the expression of the subsampled without replacement (with k data instances), shuffle binary randomized response. Using the properties of the expression as well as the Chernoff bound, we place a lower bound on the RDP. For the full proof, see Appendix A. ■

Several remarks on the bounds are in line.

Remark 1. The value Δ comes from the Chernoff bound. While we can optimize this value to provide tighter bounds, we typically use $\Delta = 1/2$ as it gives reasonably good results, and simplifies the evaluation.

Remark 2. Converting the RDP bounds to approximate DP requires the use of Lemma 2, which is a quasi-convex optimization problem given $\epsilon(\lambda)$, solvable in logarithmic time. As the computational cost of Equation 7 is $O(\lambda)$ (assuming that we fix a maximum λ), the overall computational cost is $O(\lambda \log(\lambda))$.

Remark 3. One may further close the gap between the upper and lower bounds under the following conditions. A tighter upper bound may be obtained by considering specific subsampled shuffle mechanism. Meanwhile, a tighter lower bound may be obtained by carefully bounding the binomial distribution central moments (see Appendix A), at the sacrifice of the expression's (Equation 8) simplicity.

4.2 Numerical bounds with generic LDP randomizer

The bounds given in Section 4.1 apply only to shuffled check-in with discrete ϵ_0 -LDP randomizer. In this subsection, we extend our consideration to generic, (ϵ_0, δ_0) -LDP randomizer not limited to discrete ϵ_0 -LDP mechanism. One important application of such a consideration is the FL variant of Differentially Private SGD (DP-SGD) [1, 34], where continuous and isotropic Gaussian noises are added to the clipped gradients.

Given an (ϵ_0, δ_0) -LDP randomizer, our strategy is to first convert it to the corresponding RDP parameters using Lemma 3. The RDP parameters are then plugged into Equation 6 to calculate the resulting $\epsilon(\lambda)$. Note that the subsampling and shuffling mechanisms' approximate DP properties are relatively well studied. By converting them into RDP and utilizing Theorem 1, one can calculate the RDP of shuffled check-in in a rather straightforward way. Note also that Equation 4 of Lemma 3 involves a optimization problem that cannot be written in a closed form. Our procedure is henceforth mainly numerical (in contrast to the analytical bounds given in the previous subsection). In the following, we propose two approaches of tackling the problem.

Shuffling conversion (Approach 1). Our first approach proceeds as follows:

Table 1: Our approaches to bounding shuffled check-in mechanism with a generic (ϵ_0, δ_0) -LDP randomizer. The conversion from DP to RDP (using Lemma 3) can be performed in two different steps leading to different time complexities as shown in the Table. Dependence of the time complexity on other factors is suppressed.

	Shuffling	Subsampling	Convert to RDP from	Time complexity
Approach 1	Use [21] (DP)	Use [38] (RDP)	[21]	$O(n\lambda \log \lambda)$
Approach 2	Use [21] (DP)	Use [3] (DP)	[21]+[3]	$O(n \log \lambda)$

1. Convert shuffle DP to shuffle RDP.
2. Use shuffle RDP to evaluate subsampled shuffle RDP
3. Substitute it into Equation 6 to evaluate the composition of RDP.

To calculate the shuffle DP with (ϵ_0, δ_0) -LDP randomizer, we use Theorem 3.8 given by [21].³ The subsampled RDP can be calculated using Theorem 9 of [38].

Note that Step 2 is a calculation of $O(\lambda)$ in terms of time complexity [38]. Step 3 involves evaluating the summation with respect to n as in Equation 6, which is of $O(n)$. One also needs to convert the RDP notion back to approximate DP using Lemma 2, which is an $O(\log \lambda)$ operation, as explained in Remark 2. Overall, the shuffling conversion approach is of time complexity $O(n\lambda \log \lambda)$.⁴

Subsampled shuffling conversion (Approach 2). Our second approach converts the subsampled shuffle mechanism’s DP to RDP:

1. Evaluate subsampled shuffle DP.
2. Convert subsampled shuffle DP to subsampled shuffle RDP.
3. Substitute it into Equation 6 to evaluate the composition of RDP.

The approximate DP of subsampled shuffling can be calculated using the result of [21] for shuffling and Theorem 9 of [3] for subsampling, with both having $O(1)$ time complexity. Hence, in terms of time complexity with respect to λ , the overall approach takes $O(n \log \lambda)$. Table 1 summarizes the two approaches mentioned above. Although the subsampling shuffling conversion approach has lower time complexity, we expect the RDP bound to be looser than the shuffling conversion approach as the conversion to RDP occurs one step later.

5 Numerical Results

In this section, we present numerical evaluation results of our proposal to demonstrate the performance of the RDP bounds. We also apply our techniques to training machine learning models under the distributed setting.

Before doing so, let us comment on several numerical tricks we have used in our evaluation. We often need to evaluate large binomial coefficients especially when evaluating RDP at large λ . Stirling’s approximation is employed in such a case.⁵ As mentioned in [38], the log-sum-exp trick is also useful when evaluating the logarithmic term of Equation 6. Moreover, the bisection method is employed to solve efficiently (quasi-) convex optimization problems such as Equations 3 and 4.

³ The result of privacy amplification by shuffling by [21] is valid only when $\epsilon_0 \leq \log(n/16 \log(2/\delta))$. For parameters violating this condition, we assume that no amplification occurs, i.e., $\epsilon = \epsilon_0$. This is well corroborated by the numerical experiments performed in [21].

⁴ Here, we do not include the time complexity of optimizing Equation 4 for convenience as it is irrelevant to the subsequent discussions. We note however that Equation 4 is convex optimization problem which can be solved efficiently.

⁵ More precisely, $\binom{n}{k} \approx \frac{n!}{k!}$ for $k \lesssim \sqrt{n}$ and $\binom{n}{k} \approx \frac{n!}{k!(n-k)!}$ otherwise.

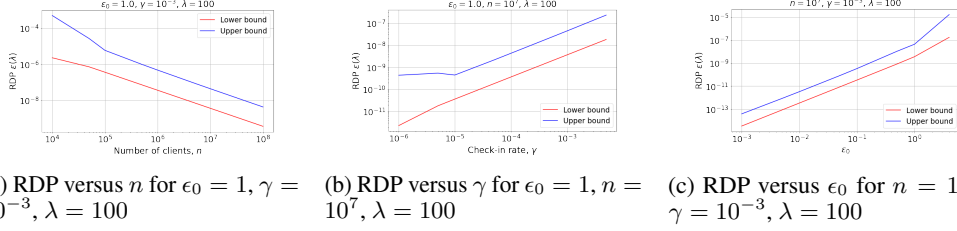


Figure 1: Upper (blue) and lower (red) bounds on the RDP of the shuffled check-in mechanism.

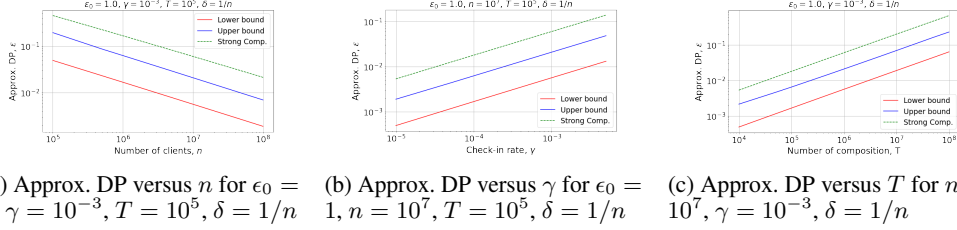


Figure 2: Comparison of several bounds on the approximate DP of the shuffled check-in mechanism. We compare the upper (blue) and lower (red) bound of our formulation (RDP) with the bound based on strong composition [24] (green dashed).

5.1 RDP and approximate DP of shuffled check-in

In Figure 1, we plot the RDP upper and lower bounds varying several parameters of interest. One can observe from Figure 1a that, as n increases, privacy is amplified due to a larger anonymous population (shuffling). Increasing γ lowers the subsampling effect as in Figure 1b. Significant amplification can be observed at low ϵ_0 as in Figure 1c.

In Figure 2, we plot the approximate DP bounds of shuffled check-in in various parameter regimes. The approximate DP is calculated by solving for the optimal λ in Equation 3. We mainly compare with [24], which uses strong composition for privacy accounting. As can be seen in the figure, our upper bound is approximately 3 times tighter than that of accounting using strong composition, and our lower bound is 10 times tighter.

5.2 Private distributed learning with shuffled check-in

We perform experiments of distributed machine learning tasks to evaluate our proposal’s empirical performance. The MNIST handwritten digit dataset [31] is used, assuming that each client is holding one data instance. A convolutional neural network with architecture similar to the one used in [19] is employed (see Appendix B) as the model to be trained privately. Two different SGD algorithms are used for training: LDP-SGD and Federated DP-SGD. The privacy guarantees of LDP-SGD (Federated DP-SGD) can be bounded applying results from Section 4.1 (Section 4.2).

LDP-SGD. Our first experiment of distributed learning utilizes the LDP-SGD algorithm [15, 19] where the underlying LDP discrete randomizer satisfies ϵ_0 -LDP. We set the number of clients to be 60,000, each holding a data instance from the MNIST train dataset. The classification accuracy of the trained model is evaluated with the MNIST test dataset (containing 10,000 instances). The check-in rate γ is set to be 0.1, $\epsilon_0 = 2$, and the clipping size is $C = 0.1$.

We train the network until the test accuracy reaches and stabilizes at around 90% (taking around 6,800 rounds). The accumulated budget is then calculated using our RDP upper bound (Equation 7) and [24]. As can be observed from Figure 3a, a privacy budget of $\epsilon \simeq 1$ is sufficient to train the model to reach 90% of accuracy using our RDP approach, while the strong composition-based approach requires $\epsilon \simeq 3$. This 1 : 3 ratio is also consistent with the observation made in Figure 2.

Federated DP-SGD. We next adapt DP-SGD [1, 7, 36] to distributed learning. Here, the Gaussian mechanism is applied to each client’s clipped gradient to achieve (ϵ_0, δ_0) -LDP. Let us remark that, despite its wide usage (under the central DP setting) and simplicity, we do not realize any existing

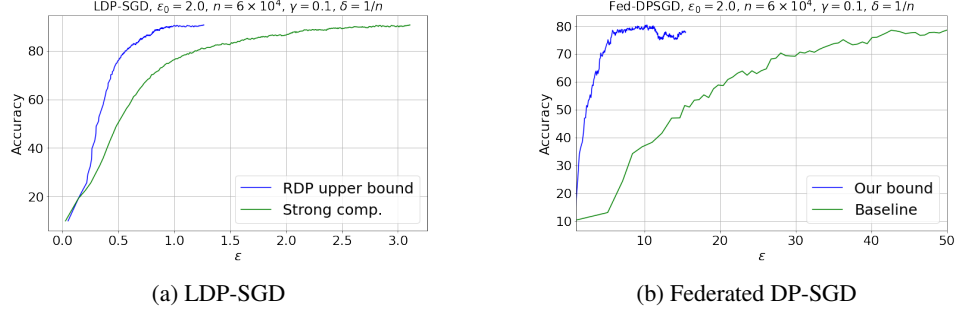


Figure 3: Privacy-utility trade-off of private distributed learning with the LDP-SGD and Federated DP-SGD algorithms on the MNIST dataset. Our approaches (blue) achieve better accuracies at lower ϵ 's compared to baseline approaches (green), thus achieving tighter accounting.

work using the Gaussian mechanism in the LDP/shuffle model literature, possibly due to the difficulty of evaluating an (ϵ_0, δ_0) -LDP randomizer. Hence, we believe that our evaluation is of independent interest as well. One fact worth mentioning for the adaptation to the distributed setting is that as we are working with replacement DP, the sensitivity should be set to twice of those given in [1], which is defined in terms of addition/removal DP [17, 37].

Baseline method. Before proceeding to present the experiment, we devise a baseline method of privacy accounting without using Theorem 1 to make comparisons with our proposal. To do so, we modify existing techniques of subsampling and shuffling to perform the accounting (we give a sketch of our accounting method here. See Appendix B for the full description). We first get a high probability bound $(1 - \delta')$ on the number of clients l checking in in each round. Then, with probability $1 - \delta'$, we obtain a conservative subsampled shuffle privacy amplification using the pre-determined l ; denote the resulting DP by (ϵ_1, δ_1) . We finally absorb δ' to the resulting DP to obtain $(\epsilon_1, \delta_1 + \delta')$ -DP, and use RDP composition over all rounds.

Experimental details and results. As evaluating the RDP of shuffled check-in with Approach 1 in Section 4.2 is quite prohibitive computationally (Table 1), we evaluate the resulting DP using only Approach 2 (which is still an $O(n)$ operation) and the above baseline. We perform two experiments. In the first experiment, the parameters are set to be similar to the one in the evaluation using LDP-SGD: $\epsilon_0 = 2$, $n = 60,000$, $\gamma = 0.1$, $C = 0.05$ and the experiment is run for 5,540 rounds. The result is shown in Figure 3b, showing that our proposed accounting is much tighter. In the second experiment, we set the parameters of the experiment as follows: $\epsilon_0 = 8$, $n = 10^7$ (by bootstrapping from the train dataset), $\gamma = 10^{-4}$, $C = 0.05$. Under this setting, we reach around 90% of test accuracy after running the training for 2,000 rounds. An accuracy-round curve and other details can be found in Appendix B. We obtain the final ϵ as 0.67 with our approach, in contrast with accounting with the baseline method presented above, which yields $\epsilon = 0.80$. This again demonstrates the effectiveness of our approach. As a side note, we find that in general Federated DP-SGD performs better at large ϵ_0 .

6 Conclusion

As a conclusion, we remark that, while we are making a step towards privacy amplification under a more practical setting, our protocol has by no means resolved all problems in privacy amplification applied to practical distributed learning. We hope that this paper can spur the study of privacy amplification with practical distributed protocols in mind within the research community.

Societal impact. This paper addresses privacy issues in distributed learning. While employing our protocol guarantees rigorous privacy, we have not discussed other possible negative impacts, such as fairness due to biased data and the impact of malicious clients on distributed learning. Incorporating these considerations in our protocol is an important future direction.

References

- [1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.
- [2] Shahab Asoodeh, Jiachun Liao, Flavio P Calmon, Oliver Kosut, and Lalitha Sankar. Three variants of differential privacy: Lossless conversion and applications. *IEEE Journal on Selected Areas in Information Theory*, 2(1):208–222, 2021.
- [3] Borja Balle, Gilles Barthe, and Marco Gaboardi. Privacy amplification by subsampling: Tight analyses via couplings and divergences. *Advances in Neural Information Processing Systems*, 31, 2018.
- [4] Borja Balle, Gilles Barthe, Marco Gaboardi, Justin Hsu, and Tetsuya Sato. Hypothesis testing interpretations and renyi differential privacy. In *International Conference on Artificial Intelligence and Statistics*, pages 2496–2506. PMLR, 2020.
- [5] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. In *Annual International Cryptology Conference*, pages 638–667. Springer, 2019.
- [6] Borja Balle, Peter Kairouz, Brendan McMahan, Om Thakkar, and Abhradeep Guha Thakurta. Privacy amplification via random check-ins. *Advances in Neural Information Processing Systems*, 33:4623–4634, 2020.
- [7] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473. IEEE, 2014.
- [8] Igor Bazovsky. *Reliability theory and practice*. Courier Corporation, 2004.
- [9] Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. Protection against reconstruction and its applications in private federated learning. *arXiv preprint arXiv:1812.00984*, 2018.
- [10] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 441–459, 2017.
- [11] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.
- [12] Clément L Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. *Advances in Neural Information Processing Systems*, 33:15676–15688, 2020.
- [13] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.
- [14] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 375–403. Springer, 2019.
- [15] John C Duchi, Michael I Jordan, and Martin J Wainwright. Minimax optimal procedures for locally private estimation. *Journal of the American Statistical Association*, 113(521):182–201, 2018.
- [16] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Eurocrypt*, volume 4004, pages 486–503. Springer, 2006.

- [17] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.
- [18] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.
- [19] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Shuang Song, Kunal Talwar, and Abhradeep Thakurta. Encode, shuffle, analyze privacy revisited: Formalizations and empirical evaluation. *arXiv preprint arXiv:2001.03618*, 2020.
- [20] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep 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.
- [21] Vitaly Feldman, Audra McMillan, and Kunal Talwar. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 954–964. IEEE, 2022.
- [22] Antonious Girgis, Deepesh Data, and Suhas Diggavi. Renyi differential privacy of the subsampled shuffle model in distributed learning. *Advances in Neural Information Processing Systems*, 34, 2021.
- [23] Antonious Girgis, Deepesh Data, Suhas Diggavi, Peter Kairouz, and Ananda Theertha Suresh. Shuffled model of differential privacy in federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2521–2529. PMLR, 2021.
- [24] Antonious M Girgis, Deepesh Data, and Suhas Diggavi. Differentially private federated learning with shuffling and client self-sampling. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 338–343. IEEE, 2021.
- [25] Antonious M Girgis, Deepesh Data, Suhas Diggavi, Ananda Theertha Suresh, and Peter Kairouz. On the renyi differential privacy of the shuffle model. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2321–2341, 2021.
- [26] Peter Kairouz, Brendan McMahan, Shuang Song, Om Thakkar, Abhradeep Thakurta, and Zheng Xu. Practical and private (deep) learning without sampling or shuffling. In *International Conference on Machine Learning*, pages 5213–5225. PMLR, 2021.
- [27] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.
- [28] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *International conference on machine learning*, pages 1376–1385. PMLR, 2015.
- [29] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.
- [30] Antti Koskela, Mikko A Heikkilä, and Antti Honkela. Tight accounting in the shuffle model of differential privacy. *arXiv preprint arXiv:2106.00477*, 2021.
- [31] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [32] Seng Pei Liew, Tsubasa Takahashi, Shun Takagi, Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. Network shuffling: Privacy amplification via random walks. *arXiv preprint arXiv:2204.03919*, 2022.
- [33] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

- [34] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.
- [35] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017.
- [36] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248. IEEE, 2013.
- [37] Salil Vadhan. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*, pages 347–450. Springer, 2017.
- [38] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. Subsampled rényi differential privacy and analytical moments accountant. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1226–1235. PMLR, 2019.

Supplementary Material

A Proofs

A.1 Proof of Theorem 2 (upper bound)

We first recite the result from [22] for the upper bound of a subsampled shuffle mechanism.

Lemma 4 (RDP upper bound of subsampled shuffle mechanism with discrete ϵ_0 -LDP randomizer [22]). *For any $n \in \mathbb{N}$, $k \leq n$, $\epsilon_0 \geq 0$, and any integer $\lambda \geq 2$, the RDP of the subsampled shuffle mechanism \mathcal{M} is upper-bounded by*

$$\epsilon(\lambda) \leq \frac{1}{\lambda - 1} \log \left(1 + 4 \binom{\lambda}{2} \gamma^2 \frac{(e^{\epsilon_0} - 1)^2}{\bar{k} e^{\epsilon_0}} + \sum_{j=3}^{\lambda} \binom{\lambda}{j} \gamma^j j \Gamma(j/2) \left(\frac{2(e^{2\epsilon_0} - 1)^2}{\bar{k} e^{2\epsilon_0}} \right)^{j/2} + \Upsilon_k \right), \quad (9)$$

where $\bar{k} = \lfloor \frac{k-1}{2e^{\epsilon_0}} \rfloor + 1$, $\gamma = \frac{k}{n}$, and $\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx$ is the Gamma function. The term Υ is given by $\Upsilon_k = \left(\left(1 + \gamma \frac{e^{2\epsilon_0} - 1}{e^{\epsilon_0}} \right)^\lambda - 1 - \lambda \gamma \frac{e^{2\epsilon_0} - 1}{e^{\epsilon_0}} \right) e^{-\frac{k-1}{8e^{\epsilon_0}}}$.

We also make the following straightforward observation.

Lemma 5. *Equation 9 is a monotonically decreasing function with respect to k .*

Then, we apply the Chernoff bound: $\mathbb{P}[X \leq (1 - \Delta)\mu] \leq e^{-\Delta^2 \mu/2}$ for all $0 < \Delta < 1$.

Assume that Δ is chosen such that $(1 - \Delta)n\gamma$ is an integer. We split the summation over k to two parts: those equal or less than $(1 - \Delta)n\gamma$, and those equal or larger than $(1 - \Delta)n\gamma + 1$. We then have from Equation 9,

$$\begin{aligned} \mathbb{E} \left[\left(\frac{\mathcal{M}(\mathcal{D})}{\mathcal{M}(\mathcal{D}')} \right)^\lambda \right] &\leq 1 + 4 \binom{\lambda}{2} \gamma^2 (e^{\epsilon_0} - 1)^2 \left(e^{-\epsilon_0 - \Delta^2 n\gamma/2} + e^{-\epsilon_0} / \tilde{k} \right) \\ &\quad + \sum_{j=3}^{\lambda} \binom{\lambda}{j} \gamma^j j \Gamma(j/2) \left(\frac{2(e^{2\epsilon_0} - 1)^2}{e^{2\epsilon_0}} \right)^{j/2} \left(e^{-\Delta^2 n\gamma/2} + \tilde{k}^{-j/2} \right) \\ &\quad + \Upsilon_1 e^{-\Delta^2 n\gamma/2} + \Upsilon_{(1-\Delta)n\gamma+1}, \end{aligned} \quad (10)$$

where $\tilde{k} = \lfloor \frac{(1-\Delta)n\gamma}{2e^{\epsilon_0}} \rfloor + 1$, as desired (Equation 7).

A.2 Proof of Theorem 3 (lower bound)

We first consider binary randomized response within the (subsampled) shuffle model, which has been studied before in [14, 21, 25, 30]. Here, we give the full treatment for completeness.

First, we prove the following lemma.

Lemma 6. *For binary randomized response of a subsampled shuffling mechanism, the follow lower bound holds.*

$$\mathbb{E}_{q_k} \left[\left(\frac{p_k}{q_k} \right)^\lambda \right] \geq 1 + \binom{\lambda}{2} \gamma^2 \frac{(e^{\epsilon_0} - 1)^2}{k e^{\epsilon_0}}. \quad (11)$$

Proof. The binary randomized response satisfying ϵ_0 -LDP takes input $x \in \{0, 1\}$ and outputs with probability $\Pr[\mathcal{A}_{bin}(x) = x] = \frac{e^{\epsilon_0}}{e^{\epsilon_0} + 1}$. Let $\rho = \frac{1}{e^{\epsilon_0} + 1}$. We consider adjacent databases $D, D' \in \{0, 1\}^k$ with instances $D = (0, \dots, 0), D' = (1, 0, \dots, 0)$. Let the output number of one's

of the shuffler be m ($m \in [k]$). The distributions of m for D and D' are respectively,

$$\begin{aligned}\mu_0(m) &= \binom{k}{m} \rho^m (1-\rho)^{k-m}, \\ \mu_1(m) &= (1-\rho) \binom{k-1}{m-1} \rho^{m-1} (1-\rho)^{k-m} + \rho \binom{k-1}{m} \rho^m (1-\rho)^{k-m-1}.\end{aligned}\tag{12}$$

Hence, a subsampled shuffle mechanism with binary randomized response can be represented by

$$\begin{aligned}q_k(m) &= \mu_0(m) \\ p_k(m) &= (1-\gamma)\mu_0(m) + \gamma\mu_1(m)\end{aligned}$$

Next, we obtain

$$\begin{aligned}\frac{\mu_1(m)}{\mu_0(m)} &= \frac{(1-\rho) \binom{k-1}{m-1} \rho^{m-1} (1-\rho)^{k-m} + \rho \binom{k-1}{m} \rho^m (1-\rho)^{k-m-1}}{\binom{k}{m} \rho^m (1-\rho)^{k-m}} \\ &= \frac{m}{k} \frac{(1-\rho)}{\rho} + \frac{k-m}{k} \frac{\rho}{1-\rho} \\ &= \frac{m}{k} e^{\epsilon_0} + \frac{k-m}{k} e^{-\epsilon_0}.\end{aligned}$$

Using the above result, we calculate

$$\begin{aligned}\frac{p_k}{q_k} - 1 &= (1-\gamma) + \gamma \frac{\mu_1(m)}{\mu_0(m)} - 1 \\ &= \gamma \left(\frac{m}{k} [e^{\epsilon_0} - e^{-\epsilon_0}] + e^{-\epsilon_0} - 1 \right) \\ &= \gamma \frac{e^{2\epsilon_0} - 1}{k e^{\epsilon_0}} \left(m - \frac{k(e^{\epsilon_0} - 1)}{e^{2\epsilon_0} - 1} \right) \\ &= \gamma \frac{e^{2\epsilon_0} - 1}{k e^{\epsilon_0}} \left(m - \frac{k}{e^{\epsilon_0} + 1} \right),\end{aligned}$$

where in the last step, we have used $e^{2\epsilon_0} - 1 = (e^{\epsilon_0} - 1)(e^{\epsilon_0} + 1)$.

We can now calculate the moment by binomial expansion (with respect to λ),

$$\begin{aligned}\mathbb{E}_{q_k} \left[\left(\frac{p_k}{q_k} \right)^\lambda \right] &= \mathbb{E}_{q_k} \left[\left(1 + \frac{p_k}{q_k} - 1 \right)^\lambda \right] \\ &= 1 + \binom{\lambda}{2} \gamma^2 \frac{(e^{\epsilon_0} - 1)^2}{k e^{\epsilon_0}} + \sum_{j=3}^{\lambda} \binom{\lambda}{j} \gamma^j \left(\frac{e^{2\epsilon_0} - 1}{k e^{\epsilon_0}} \right)^j \mathbb{E} \left(m - \frac{k}{e^{\epsilon_0} + 1} \right)^j\end{aligned}\tag{13}$$

where the expectation is over $\text{Bin}(k, \rho)$. We have also used $\mathbb{E} \left(m - \frac{k}{e^{\epsilon_0} + 1} \right)^2 = k\rho(1-\rho)$ in the second term. Note that $\mathbb{E} \left(m - \frac{k}{e^{\epsilon_0} + 1} \right)^j$ is the j -th *central* moment of the binomial distribution.

The above equation may further be simplified using Jensen's inequality, $\mathbb{E} \left(m - \frac{k}{e^{\epsilon_0} + 1} \right)^j \geq \left[\mathbb{E} \left(m - \frac{k}{e^{\epsilon_0} + 1} \right) \right]^j = 0$:

$$\begin{aligned}\mathbb{E}_{q_k} \left[\left(\frac{p_k}{q_k} \right)^\lambda \right] &= 1 + \binom{\lambda}{2} \gamma^2 \frac{(e^{\epsilon_0} - 1)^2}{k e^{\epsilon_0}} + \sum_{j=3}^{\lambda} \binom{\lambda}{j} \gamma^j \left(\frac{e^{2\epsilon_0} - 1}{k e^{\epsilon_0}} \right)^j \mathbb{E} \left(m - \frac{k}{e^{\epsilon_0} + 1} \right)^j \\ &\geq 1 + \binom{\lambda}{2} \gamma^2 \frac{(e^{\epsilon_0} - 1)^2}{k e^{\epsilon_0}}.\end{aligned}$$

■

Table 2: Neural network architecture used in our experiment.

Layer	Parameters
Convolution	16 filters of 8×8 , strides 2
Max-pooling	2×2
Convolution	32 filters of 4×4 , strides 2
Max-pooling	2×2
Linear	32 units
Softmax	10 units

We finally move to proving Equation 8. We use the fact that binomial distribution is concentrated at its mean μ , and with the Chernoff bound: $\mathbb{P}[X \geq (1 + \Delta)\mu] \leq e^{-\Delta^2\mu/(2+\Delta)}$ for all Δ .

Note that the term depending on k in Equation 11 is a monotonically decreasing function with respect to k . Hence,

$$\begin{aligned}
\mathbb{E}_{i,k \sim \mathcal{M}(\mathcal{D}')} \left[\left(\frac{\mathcal{M}(\mathcal{D})}{\mathcal{M}(\mathcal{D}')} \right)^\lambda \right] &\geq \sum_{k=1}^n \binom{n}{k} \gamma^k (1 - \gamma)^{n-k} \cdot \left(1 + \binom{\lambda}{2} \gamma^2 \frac{(e^{\epsilon_0} - 1)^2}{k e^{\epsilon_0}} \right) \\
&= 1 + \sum_{k=1}^n \binom{n}{k} \gamma^k (1 - \gamma)^{n-k} \cdot \binom{\lambda}{2} \gamma^2 \frac{(e^{\epsilon_0} - 1)^2}{k e^{\epsilon_0}} \\
&\geq 1 + (1 - e^{-\Delta^2 n \gamma / (2 + \Delta)}) \binom{\lambda}{2} \gamma^2 \frac{(e^{\epsilon_0} - 1)^2}{(1 + \Delta) n \gamma e^{\epsilon_0}},
\end{aligned}$$

because $\mathbb{P}[X \leq (1 + \Delta)\mu] \geq 1 - e^{-\Delta^2\mu/(2+\Delta)}$. We have henceforth obtained the desired result.

B Experimental Details

Network architecture. The neural network we use in the experiments presented in Section 5 is as in Table 2.

Baseline for privacy accounting of shuffled check-in with generic LDP mechanism. We here describe in more detail our method of shuffled check-in with generic LDP mechanism without using Theorem 1.

Our shuffled check-in protocol has an issue that cannot be dealt with using standard subsampling/shuffling techniques: the number of clients checking in is a variable, following a probability distribution equal to those followed by k in Equation 6.

To resolve this, we devise the following method. Given $0 \leq \delta' \leq 1$, we first find l such that, $\mathbb{P}[k \leq l] = 1 - \delta'$ for k and the corresponding binomial probability distribution defined in Equation 6. Fixing the number of clients to be l , we use the following to *conservatively* bound the approximate DP of subsampled shuffling:

- No privacy amplification occurs ($\epsilon = \epsilon_0$) due to shuffling (see Footnote 3).
- Privacy amplification by subsampling is accounted for by assuming conservatively that l clients are sampled. More precisely, let $\gamma' = l/n$, privacy amplification by subsampling leads to $\epsilon_1 = \log(1 + \gamma'(e^{\epsilon_0} - 1))$ and $\delta_1 = \gamma'\delta_0$ [3].

Consequently, the approximate DP of shuffled check-in for a single round is $(\epsilon_1, \delta_1 + \delta')$ by absorbing δ' to the final DP. To compose the mechanism, we convert the approximate DP to RDP as done in Section 4.2 and use Lemma 1.⁶

Federated DP-SGD. Given ϵ_0 , we fix $\delta_0 = 1/n^{1.5}$ to calculate the noise multiplier z of the Gaussian mechanism, which adds Gaussian noise, $N(0, z^2 C^2 I)$ to the gradient [1].

⁶We find that composition using the strong composition theorem [28] leads to a much looser privacy accounting.

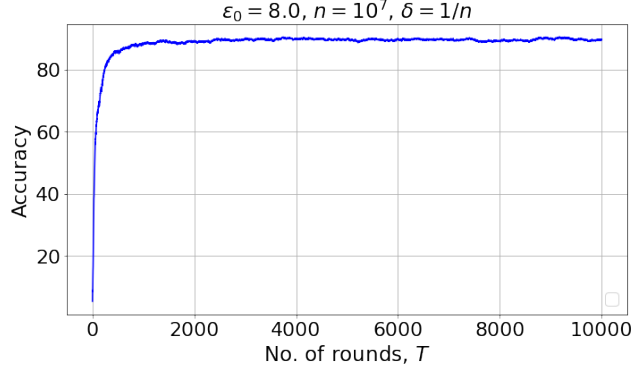


Figure 4: Federated DP-SGD.

In Figure 4, we show the accuracy-round curve of our Federated DP-SGD experiment with the following parameters: $\epsilon_0 = 8$, $n = 10^7$ (by bootstrapping from the train dataset), $\gamma = 10^{-4}$, clipping size $C' = 0.05$. It should be noted that the average number of clients checking in is 1,000, around the same order as the ones used in the settings given in Section 5. It can then be seen that the accuracy can reach the optimal 90%, and converges faster with a larger ϵ_0 , noting that n mainly affects the final ϵ , not the convergence. Note also that we have not evaluated the accuracy with respect to all ranges of ϵ in this experiment as it is computationally too expensive.