SHUFFLE GAUSSIAN MECHANISM FOR DIFFERENTIAL PRIVACY

Anonymous authors
Paper under double-blind review

ABSTRACT

We study Gaussian mechanism in the shuffle model of differential privacy (DP). We present the *first* non-trivial privacy guarantee of the mechanism by showing that its Rényi differential privacy (RDP) is of the form:

$$\epsilon(\lambda) = \frac{1}{\lambda - 1} \log \left(\frac{e^{-\lambda/2\sigma^2}}{n^\lambda} \sum_{\substack{k_1+\ldots+k_n=\lambda; \\ k_1,\ldots,k_n \geq 0}} \binom{\lambda}{k_1, \ldots, k_n} e^{\sum_{i=1}^n k_i^2/2\sigma^2} \right)$$

We further prove that the RDP is strictly upper-bounded by the Gaussian RDP without shuffling. The shuffle Gaussian RDP is advantageous in composing multiple DP mechanisms, where we demonstrate its improvement over the state-of-the-art approximate DP composition theorems in privacy guarantees of the shuffle model. Our formalism also has immediate application in several problems studied in the literature, including learning with stochastic gradient descent and distributed/federated learning, of which an empirical study is presented as well to demonstrate the efficacy of learning privately with the shuffle Gaussian mechanism.

1 INTRODUCTION

The shuffle/shuffled model (Cheu et al., 2019; Erlingsson et al., 2019) has attracted attention recently as an intermediate model of trust in differential privacy (DP) (Dwork et al., 2006a;b). Within this setup, each user sends a locally differentially private (LDP) (Kasiviswanathan et al., 2011) report to a trusted shuffler, where the collected reports are anonymized/shuffled, before being forwarded to the untrusted analyzer/server. When viewed in the central model, the central DP parameter, ϵ, can be smaller than the local one, ϵ_0; a phenomenon also known as privacy amplification. This yields better utility-privacy trade-offs than the local model of DP in general, without relying on a highly trusted server as in the central model of DP. The shuffle model can be realized in practice through the use of trusted hardware (Bittau et al., 2017), mix-nets (Chaum, 1981), or peer-to-peer protocols (Liew et al., 2022b). Various aspects of the shuffle model have been investigated in the literature.

A prominent use case of the shuffle model is in the DP version of distributed learning, or federated learning (we use these two terms interchangeably in this paper) (Kairouz et al., 2021b; McMahan et al., 2017a). Non-private federated learning (FL) proceeds roughly as follows. The ochestrating server initiates the learning by sending a model to users. Then, each user calculates the gradient/model update using her own data, and sends the gradient to the server while keeping her data local. This is repeated for multiple rounds until the model converges.

In order to incorporate DP to distributed learning, it is convenient to first consider the conventional gradient-based approach of learning with DP under the centralized (where private data are curated for training) setting, i.e., differentially private stochastic gradient descent (DP-SGD) (Abadi et al., 2016; Bassily et al., 2014; Song et al., 2013). Here, the Gaussian mechanism is employed, where noise in the form of Gaussian/normal distribution is applied to a batch of clipped gradients to attain DP model updates. As learning often requires repeated interaction, privacy composition is vital when working under a pre-determined privacy budget. In DP-SGD, privacy is conventionally accounted for via the moments accountant (Abadi et al., 2016), providing much tighter DP composition, leading to a significant saving of privacy budget compared to strong composition theorems (Dwork et al., 2010; Kairouz et al., 2015). The subsampling effect (Ullman, spring 2017) is typically leveraged as well to

achieve acceptable levels of privacy. Moreover, the central-DP version of DP-SGD has been studied and adapted to federated learning (McMahan et al., 2017b).

However, perhaps due to the difficulty of handling approximate DP mechanisms (particularly in privacy accounting), such as the Gaussian mechanism, the *shuffle Gaussian mechanism*, and its applications in (distributed) learning have not been explored in depth (Koskela et al., 2021; Liew et al., 2022a). Distributed learning in the local or shuffle model is often conducted by utilizing variants of the LDP-SGD algorithm (Duchi et al., 2018; Erlingsson et al., 2020) in the literature, which is arguably more sophisticated than the Gaussian mechanism. [1] Moreover, even the centralized learning paradigm has a discrepancy in privacy accounting of DP-SGD: as noted in De et al. (2022), open-source implementations of DP-SGD typically take mini-batches of samples in a shuffled scheme, but the privacy accountant implemented often ignores the shuffling effect.

Our contributions. In this paper, we provide the *first* non-trivial privacy guarantee for the shuffle Gaussian mechanism. Our principal result is to give an exact expression of the Rényi divergence or Rényi differential privacy (RDP) for the shuffle Gaussian mechanism (see Equation 4). We also prove that the shuffle Gaussian RDP is always bounded above by the Gaussian RDP without shuffling. While limited to Gaussian distribution, this *improves* over recent analyses where privacy amplification occurs only at limited parameter regions (Erlingsson et al., 2019; Feldman et al., 2022).

To evaluate the expression of shuffle Gaussian numerically, we reduce it to a partition problem in number theory. This enables us to perform concrete calculation and achieve tight privacy composition of any Gaussian-based randomization mechanism coupled with a shuffler. Furthermore, the shuffle Gaussian RDP characterization allows us to compute the RDP of subsampled shuffle Gaussian and shuffled check-in Gaussian mechanisms, which are mechanisms tailored to SGD and distributed learning (Girgis et al., 2021a; Liew et al., 2022a), *resolving* the open problems mentioned above posed by De et al. (2022); Koskela et al. (2021). In later sections, we also perform experiments applying variants of shuffle Gaussian to centralized and distributed learning tasks to gauge the utility-privacy trade-offs.

Related work. Early studies on the shuffle model put focus on ϵ_0-local randomizer (Balle et al., 2019; Cheu et al., 2019; Erlingsson et al., 2019). In Balle et al. (2020b); Feldman et al. (2022), extension to approximate DP, i.e., (ϵ_0, δ_0)-local randomizer has been worked on. While methods for composing optimally approximate DP mechanisms are known (Kairouz et al., 2015; Murtagh & Vadhan, 2016), they are unable to compose optimally mechanisms that satisfy multiple values of (ϵ, δ) simultaneously, such as the Gaussian mechanism.

A preliminary analysis of the shuffle Gaussian mechanism using the Fourier/numerical accountant (Gopi et al., 2021; Koskela et al., 2020) has been performed in Koskela et al. (2021). This approach is known to give even tighter composition in general, but faces the curse of dimensionality in n (n being the database size): only $n \lesssim 10$ can be evaluated within reasonable accuracy and amount of computation, not suitable for evaluating tasks involving $n \gg 1$ like machine learning. We resolve the difficulty encountered in Koskela et al. (2021) through the use of RDP.

Distributed learning with differential privacy is traditionally studied under the central-DP model setting (Bonawitz et al., 2017; Kairouz et al., 2021a; McMahan et al., 2017b). Most studies of distributed learning under the local or shuffle model have investigated only the ϵ_0-local randomizer (Bhowmick et al., 2018; Erlingsson et al., 2020; Girgis et al., 2021a;c). In Liew et al. (2022a), the Gaussian mechanism is adopted for distributed learning, but the accounting of privacy presented there is essentially based on approximate DP, less optimal compared to our RDP approach (as will be shown in later sections).

Paper organization. In the next section, we provide our problem formulation and preliminaries containing previous related results of DP. Then, we present and prove our main RDP results of the shuffle Gaussian mechanism. In Section 4, the shuffle Gaussian mechanism is extended for machine learning applications. Before closing, we give the numerical results in Section 5.

[1] The LDP-SGD algorithm proceeds roughly as follows. The client-side algorithm first clips user gradient, and flips its sign with a certain probability. Then, a unit vector is sampled randomly to make an inner product with the processed gradient to yield the inner product's sign. The sign is again flipped with a certain probability, before sending it along with the unit vector to the server. The server-side algorithm includes normalizing the aggregated reports, and projecting them to a convex set before updating the model. See, e.g., Erlingsson et al. (2020) for details.

2 PRELIMINARIES

Problem formulation. Let $D = (x_1, \ldots, x_n)$ be a database of size n, where each of i, $i \in [n]$ is a data instance. x_i is a d-dimensional vector, $x_i \in \mathbb{R}^d$ and w.l.o.g., normalized to be $||x_i||_2 \in [0.1]$.

Gaussian mechanism $\mathcal{G}$ with variance σ^2 is applied to each x_i, resulting in $\tilde{x_i}$. Then, all $\tilde{x_i}$'s undergo shuffling; let $\mathcal{S}_n : \mathcal{Y}^n \to \mathcal{Y}^n$ be the shuffling operation which takes $\tilde{x_i}$'s as input and outputs a uniformly and randomly permuted $\tilde{x_i}$'s. The randomization mechanism $\mathcal{M}$ of interest is therefore

$$\mathcal{M}(D) = \mathcal{S}_n(\mathcal{G}(x_1), \ldots, \mathcal{G}(x_n)).$$

Our purpose is to characterize and quantify the DP of $\mathcal{M}$ through RDP.

Let us next give definitions related to differential privacy and other known results.

Definition 1 (Central Differential Privacy (Dwork et al., 2014))**.** A randomization mechanism, $\mathcal{M} : \mathcal{D}^n \to \mathcal{S}$ with domain $\mathcal{D}^n$ and range $\mathcal{S}$ satisfies central (ϵ, δ)-differential privacy (DP), where $\epsilon \geq 0$, $\delta \in [0, 1]$, 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:

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

In this paper, adjacent databases are referred to as databases that have one data instance replaced by another; also known as "replacement" DP in the literature. The (ϵ, δ)-DP mechanism is also said to satisfy *approximate* DP, or is (ϵ, δ)-indistinguishable. Furthermore, we simply refer to (ϵ, δ)-DP as "DP" when it is unambiguous. We also abbreviate $\mathcal{D}^n$ as $\mathcal{D}$ when the context is clear.

The randomization mechanism is known as local randomizer when the mechanism is applied to each data instance instead of the whole database. Formally, this form of mechanism is said to satisfy local DP:

Definition 2 (Local Differential Privacy (LDP) (Kasiviswanathan et al., 2011))**.** A randomization mechanism $\mathcal{A} : \mathcal{D} \to \mathcal{S}$ 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 use ϵ_0 instead of ϵ when referring to LDP. A mechanism satisfying $(\epsilon_0, 0)$-LDP is also called an ϵ_0-LDP randomizer. Let us next define Rényi differential privacy, the main privacy notion used throughout this work (see also Bun & Steinke (2016); Dwork & Rothblum (2016)).

Definition 3 (Rényi Differential Privacy (RDP) (Mironov, 2017))**.** A randomization mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{S}$ is said to satisfy (λ, ϵ)-RDP, for $\lambda \in (1, \infty)$, if for any adjacent databases D, $D' \in \mathcal{D}$, the Rényi divergence of order λ between $\mathcal{M}(D)$ and $\mathcal{M}(D')$, as defined below, is upper-bounded by ϵ:

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

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

We take a functional view of ϵ, writing it as $\epsilon(\lambda)$. We occasionally call $\epsilon(\lambda)$ the RDP, and also refer to $D_\lambda(\mathcal{M}(D)||\mathcal{M}(D'))$ as the RDP when the context is clear.

The RDP is most useful at composing DP mechanisms, where it has cleaner composition than approximate DP. The formal description is given below.

Lemma 1 (Adaptive RDP composition (Mironov, 2017))**.** *Let mechanisms $\mathcal{M}_1, \mathcal{M}_2$ taking $D \in \mathcal{D}$ as input be (λ, ϵ_1),(λ, ϵ_2)-RDP respectively, the composed mechanism, $\mathcal{M}_1 \times \mathcal{M}_2$ satisfies $(\lambda, \epsilon_1 + \epsilon_2)$-RDP.*

After accounting for the privacy with RDP, the RDP notion is often converted to the conventional and interpretable approximate DP notion. The conversion is given by the following lemma.

Lemma 2 (RDP-to-DP conversion (Balle et al., 2020a; Canonne et al., 2020))**.** *A mechanism $\mathcal{M}$ satisfying $(\lambda, \epsilon(\lambda))$-RDP also satisfies (ϵ, δ)-DP, where $1 < \delta < 0$ is arbitrary and ϵ is given by*

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

In the above definitions, we have not specified what the underlying randomization mechanisms are. We next give the definition of the core mechanism used in this work, the Gaussian mechanism.

Definition 4 (Gaussian mechanism)**.** Given $x \in \mathbb{R}^d$, the Gaussian mechanism applied to x is a mechanism with parameter σ that adds zero mean isotropic Gaussian perturbation of variance σ^2 to x, outputting $\mathcal{M}(x) = x + \mathcal{N}(0, \sigma^2 I_d)$.

3 THE RDP OF SHUFFLE GAUSSIAN

We first derive the Rényi divergence (of order λ) of the shuffle Gaussian mechanism in this section. Subsequently, an upper bound on the notion is given, followed by a description of the numerical technique used for evaluating the RDP.

The main result of this paper is given by the following theorem.

Theorem 1 (Shuffle Gaussian RDP)**.** *The Rényi divergence (of order λ) of the shuffle Gaussian mechanism with variance σ^2 for neighboring datasets with n instances is given by*

$$D_\lambda(\mathcal{M}(D)||\mathcal{M}(D')) = \frac{1}{\lambda - 1} \log \left(\frac{e^{-\lambda/2\sigma^2}}{n^\lambda} \sum_{\substack{k_1+\ldots+k_n=\lambda; \\ k_1,\ldots,k_n \geq 0}} \binom{\lambda}{k_1, \ldots, k_n} e^{\sum_{i=1}^n k_i^2/2\sigma^2} \right) \tag{4}$$

Proof sketch. We give a proof outline here. In Appendix B, the full proof complete with details of calculation are provided. We consider w.l.o.g. adjacent databases $D, D' \in \mathbb{R}^n$ with one-dimensional data instances $D = (0, \ldots, 0)$, $D' = (1, 0, \ldots, 0)$. [2] Note that the databases can also be represented in the form of n-dimensional vector: where D is simply 0, an n-dimensional vector with all elements equal to zero, and $D' = e_1$, where e_i is a unit vector with the elements in all dimensions except the n-th ($i \in [n]$) one equal to zero.

The shuffle Gaussian mechanism first applies Gaussian noise (with variance σ^2) to each data instance, and subsequently shuffle the noisy instances. The shuffled output of D distributes as $\mathcal{M}(D) \sim (\mathcal{N}(0, \sigma^2), \ldots, \mathcal{N}(0, \sigma^2))$, an n-tuple of $\mathcal{N}(0, \sigma^2)$. Intuitively, the adversary sees a anonymized set of randomized data of size n. Due to the homogeneity of D, we can write $\mathcal{M}(D) \sim \mathcal{N}(0, \sigma^2 I_n)$ using the vector notation mentioned above. [3]

On the other hand, since all data instances except one is 0 in D', the mechanism can be written as a mixture distribution: $\mathcal{M}(D') \sim \frac{1}{n}(\mathcal{N}(e_1, \sigma^2 I_n) + \ldots + \mathcal{N}(e_n, \sigma^2 I_n))$. Intuitively, each output has probability $1/n$ carrying the non-zero element, and hence the mechanism as a whole is a uniform mixture of n distributions. In summary, the neighbouring databases of interest are

$$\mathcal{M}(D) \sim \mathcal{N}(0, \sigma^2 I_n),\ \mathcal{M}(D') \sim \frac{1}{n}\left(\mathcal{N}(e_1, \sigma^2 I_n) + \ldots + \mathcal{N}(e_n, \sigma^2 I_n)\right). \tag{5}$$

Note that $\mathbb{P}(\mathcal{N}(0, \sigma^2)) = \exp[-x^2/2\sigma^2]/\sqrt{2\pi\sigma^2}$, and

$$\begin{aligned} n \cdot \left(2\pi\sigma^2\right)^{n/2} \cdot \mathbb{P}(\mathcal{M}(D')(x)) = \exp\left[-(x_1 - 1)^2/2\sigma^2 - \sum_{i=2}^{n} x_i^2/2\sigma^2\right] + \ldots \\ + \exp\left[-(x_n - 1)^2/2\sigma^2 - \sum_{i=1}^{n-1} x_i^2/2\sigma^2\right]. \end{aligned} \tag{6}$$

That is, for $j \in [n]$, the j-th component of the mixture distribution contains a term proportional to $\exp\left[-(x_j - 1)^2/2\sigma^2 - \sum_{i \neq j} x_i^2/2\sigma^2\right]$.

[2] This can be done w.l.o.g. by normalizing and performing unitary transform on the data instances. See also Appendix B.1 of Koskela et al. (2020).

[3] Note that the vector space defined here is over the dataset dimension instead of input dimension as in Definition 4.

We are concerned with calculating the following quantity, $\mathbb{E}_{x\sim\mathcal{M}(D)}\left[\left(\frac{\mathcal{M}(D')(x)}{\mathcal{M}(D)(x)}\right)^{\lambda}\right]$ for RDP, which can be simplified to:

$$\begin{aligned}
&\mathbb{E}_{x\sim\mathcal{M}(\mathcal{D})}\left[\left(\frac{\mathcal{M}(D')(x)}{\mathcal{M}(D)(x)}\right)^{\lambda}\right] \\
&= \int\left(\frac{\exp\left[-(x_1-1)^2/2\sigma^2 - \sum_{i=2}^n x_i^2/2\sigma^2\right]+\dots}{n\exp[-\sum_{i=1}^n x_i^2/2\sigma^2]}\right)^{\lambda}\exp[-\sum_{i=1}^n x_i^2/2\sigma^2]\frac{d^n x}{(2\pi\sigma^2)^{n/2}} \\
&= \int\left(\sum_{i=1}^n \exp\left[(2x_i-1)/2\sigma^2\right]\right)^{\lambda}\exp[-\sum_{i=1}^n x_i^2/2\sigma^2]\frac{d^n x}{n^{\lambda}(2\pi\sigma^2)^{n/2}}. \qquad (7)
\end{aligned}$$

Then, we expand the expression $(\dots)^{\lambda}$ using the multinomial theorem:

$$\left(\sum_{i=1}^n \exp\left[(2x_i-1)/2\sigma^2\right]\right)^{\lambda} = \sum_{\substack{k_1+\dots+k_n=\lambda;\\ k_1,\dots,k_n\geq 0}}\binom{\lambda}{k_1,\dots,k_n}\prod_{i=1}^n \exp\left[k_i(2x_i-1)/2\sigma^2\right], \qquad (8)$$

where $\binom{\lambda}{k_1,\dots,k_n} = \frac{\lambda!}{k_1!k_2!\dots k_n!}$ is the multinomial coefficient, and $k_i \in \mathbb{Z}^+$ for $i \in [n]$. Hence,

$$\begin{aligned}
&\int\left(\sum_{i=1}^n \exp\left[(2x_i-1)/2\sigma^2\right]\right)^{\lambda}\exp[-\sum_{i=1}^n x_i^2/2\sigma^2]\frac{d^n x}{n^{\lambda}(2\pi\sigma^2)^{n/2}} \\
&= \frac{1}{n^{\lambda}}\sum_{\substack{k_1+\dots+k_n=\lambda;\\ k_1,\dots,k_n\geq 0}}\binom{\lambda}{k_1,\dots,k_n}\prod_{i=1}^n \exp[(k_i^2-k_i)/2\sigma^2].
\end{aligned}$$

Noticing that $\prod_{i=1}^n \exp[-k_i] = \exp[-\sum_i^n k_i]$ and that $\sum_{i=1}^n k_i = \lambda$ under the multinomial constraint, we have

$$\begin{aligned}
&\frac{1}{n^{\lambda}}\sum_{\substack{k_1+\dots+k_n=\lambda;\\ k_1,\dots,k_n\geq 0}}\binom{\lambda}{k_1,\dots,k_n}\prod_{i=1}^n \exp[(k_i^2-k_i)/2\sigma^2] \qquad (9) \\
&= \frac{e^{-\lambda/2\sigma^2}}{n^{\lambda}}\sum_{\substack{k_1+\dots+k_n=\lambda;\\ k_1,\dots,k_n\geq 0}}\binom{\lambda}{k_1,\dots,k_n}e^{\sum_{i=1}^n k_i^2/2\sigma^2}.
\end{aligned}$$

Combining the above expression with Equation 2, we obtain the expression given in Equation 4 as desired. ■

Upper bound of shuffle Gaussian RDP. Using the above exact expression, we next give an upper bound on the shuffle Gaussian RDP.

Corollary 1 (Upper bound of shuffle Gaussian RDP)**.** The shuffle Gaussian RDP $\epsilon(\lambda)$ is upper-bounded by $\lambda/2\sigma^2$.

Proof. Applying the Cauchy-Schwartz inequality, $\sum_{i=1}^n k_i^2 \leq (\sum_{i=1}^n k_i)^2 = \lambda^2$, to Equation 9,

$$\begin{aligned}
\frac{e^{-\lambda/2\sigma^2}}{n^{\lambda}}\sum_{\substack{k_1+\dots+k_n=\lambda;\\ k_1,\dots,k_n\geq 0}}\binom{\lambda}{k_1,\dots,k_n}e^{\sum_{i=1}^n k_i^2/2\sigma^2} &\leq \frac{e^{-\lambda/2\sigma^2}}{n^{\lambda}}\sum_{\substack{k_1+\dots+k_n=\lambda;\\ k_1,\dots,k_n\geq 0}}\binom{\lambda}{k_1,\dots,k_n}e^{\lambda^2/2\sigma^2} \\
&= e^{(\lambda^2-\lambda)/2\sigma^2},
\end{aligned}$$

noting that $\sum_{\substack{k_1+\dots+k_n=\lambda;\\ k_1,\dots,k_n\geq 0}}\binom{\lambda}{k_1,\dots,k_n} = n^{\lambda}$. Substituting this to Equation 2, we get the desired expression. ■

Algorithm 1 Numerical evaluation of Equation 4.

Inputs: Database size n, RDP order λ, variance of Gaussian mechanism σ^2.
Output: RDP parameter ϵ.
Initialize: $\Gamma : \{\emptyset\}$, $\texttt{Sum} = 0$.
$\Gamma \leftarrow \texttt{GetPartition}(\lambda)$
for $\rho \in \Gamma$ **do**
    $\{\kappa_i\} \leftarrow \texttt{GetUniqueCount}(\rho)$
    $\texttt{Sum} \leftarrow \texttt{Sum} + \frac{e^{-\lambda/2\sigma^2}}{n^\lambda} \binom{\lambda}{k_1 \ldots k_n} \frac{n!}{\kappa_1! \ldots \kappa_i!} e^{\sum_{j=1}^n k_j^2}$ where $\{k_i\} \in \rho$
Return: $\epsilon = \texttt{Sum}$

This corollary states that the upper bound of the shuffle Gaussian RDP is equal to the Gaussian RDP without shuffling (Mironov, 2017). Let us emphasize that this is a strictly tight upper bound compared with previous studies. In earlier analyses of shuffling (Balle et al., 2020b; Erlingsson et al., 2019), privacy amplification occurs at limited parameter region. More concretely, the resulting ϵ after shuffling can be larger than the local one, ϵ_0, especially at the large ϵ_0 region. In Feldman et al. (2022), the shuffle ϵ is shown, only numerically, to saturate at ϵ_0. We have instead proved in the above corollary, at least for the Gaussian mechanism, that the upper bound is analytically and strictly the same as the one without shuffling, and our result is valid for all parameter region.

3.1 Numerical computation of the shuffle Gaussian RDP

Evaluating the shuffle Gaussian RDP numerically is non-trivial, as n can be very large ($\gtrsim 10^5$) in distributed learning scenarios and the order of RDP, λ to be evaluated has to be large especially when δ is required to be small. Calculating the multinomial coefficients can be prohibitively expensive as a result. Here, we describe our numerical recipes for calculating the RDP.

We exploit the permutation invariance inherent in Equation 4 to perform more efficient computation. Equation 4 contains a sum of multinomial coefficients multiplied by an expression with integer k_i ($i \in [n]$) constrained by $k_1 + \ldots + k_n = \lambda$. We first obtain all possible k_i's with the above constraint, without counting same summands differing only in the order. This is integer partition, a subset sum problem known to be NP (Kleinberg & Tardos, 2006). It is also related to partition in number theory (Andrews & Eriksson, 2004). Nevertheless, the calculation can be performed rather efficiently in practice (Kelleher & O'Sullivan, 2009).

Note that there is a one-to-one mapping between the term $e^{\sum_{i=1}^n k_i^2/2\sigma^2}$ in Equation 9 and the partition obtained above. We can therefore calculate the number of permutation corresponding to each partition, multiplied by the multinomial coefficient, and $e^{\sum_{i=1}^n k_i^2/2\sigma^2}$, and sum over all the partitions to obtain a numerical expression of Equation 9. For a partition with the number of repetition of exponents with the same degrees (or the counts of unique $k_1, \ldots, k_n$) being $\kappa_1, \ldots, \kappa_l$ ($l \leq n$), the number of permutation can be calculated to be $n!/(\kappa_1! \ldots \kappa_l!)$. See Algorithm 1 for the procedure of calculating the RDP, Appendix C for relevant details, Appendix A for a link to the source code.

4 Applications

In this section, we extend the study of the shuffle Gaussian RDP to mechanisms more attuned to gradient-based learning. Specifically, we consider mechanisms where only a subset of dataset or users participates in training in each round, typical for large-scale distributed learning for the latter.

4.1 Subsampled shuffle Gaussian mechanism

We first consider the mechanism where a fixed number of data instances, m, is sampled randomly from n data instances to participate in training. Each data instance is randomized with Gaussian noises before being shuffled. We call this overall procedure the subsampled (without replacement) shuffle (Girgis et al., 2021b) Gaussian mechanism. Let $\text{subsamp}_m^n : \mathcal{Y}^n \to \mathcal{Y}^m$ be the subsampling operation mentioned above. Then, the mechanism can be written as

$$\mathcal{M}(\mathcal{D}) = \mathcal{S}_n \circ \text{subsamp}_m^n(\mathcal{G}(x_1), \ldots, \mathcal{G}(x_n)).$$

The RDP of the subsampled shuffle Gaussian mechanism is given by the following theorem.

Theorem 2 (Subsampled Shuffle Gaussian RDP)**.** *Let n be the total number of users, m be the number of subsampled users in each round of training, and γ be the subsampling rate, $\gamma = m/n$. Also let $\epsilon_m^{\mathrm{SG}}(\lambda)$ be the shuffle Gaussian RDP given in Theorem 1. The subsampled shuffle Gaussian mechanism satisfies $(\lambda, \epsilon_{\gamma,m}^{\mathrm{SSG}}(\lambda))$-RDP, where*

$$\epsilon_{\gamma,m}^{\mathrm{SSG}}(\lambda) \leq \frac{1}{\lambda-1}\log\left(1+\gamma^2\binom{\lambda}{2}\min\left\{4(e^{\epsilon_m^{\mathrm{SG}}(2)}-1), 2e^{\epsilon_m^{\mathrm{SG}}(2)}\right\} + \sum_{j=3}^{\lambda} 2\gamma^j \binom{\lambda}{j} e^{(j-1)\epsilon_m^{\mathrm{SG}}(j)}\right) \tag{10}$$

Proof sketch. The proof is a direct application of Theorem 1 and the subsampled RDP results of Wang et al. (2019). The full proof can be found in Appendix B. ■

We remark that Theorem 2 (as well as Theorem 1) is useful for privacy accounting of popular open-source implementations of (centralized) DP-SGD (Google; Yousefpour et al., 2021). In these implementations, mini-batches of fixed size are sampled without replacement using a random shuffling scheme. However, the implemented accountant (Google) assumes that each sample is sampled with replacement with a fixed probability from the whole dataset, causing discrepancy between theory and empirical study, posed as an open problem in De et al. (2022). Our privacy accountant (accounting for shuffle DP with Gaussian perturbations) henceforth provides a solution to this discrepancy in open-source implementations, albeit using a weaker trust model, i.e., the shuffler. Additionally, the shuffle model allows one to publish the perturbed gradients of each sample anonymously, which may be useful for debugging and exploratory data analysis purposes without violating privacy.

4.2 Shuffled Check-in Gaussian mechanism

In distributed learning, it is difficult to achieve uniform subsampling fixing the number of participating users in each round (Balle et al., 2020b; Girgis et al., 2021c) in practice. It is more natural to let the user decide (with her own randomness) whether to participate in training. This is also known as the shuffled check-in mechanism in Liew et al. (2022a).

Let n be the total number of users, and γ be the check-in rate (the probability of a user participating in training) The shuffled check-in's RDP is given as follows (Liew et al., 2022a).

$$\epsilon^{\mathrm{SCI}}(\lambda) \leq \frac{1}{\lambda-1}\log\left(\sum_{k=1}^{n}\binom{n}{k}\gamma^k(1-\gamma)^{n-k}e^{(\lambda-1)\epsilon_{\gamma,k}^{\mathrm{SSG}}(\lambda)}\right). \tag{11}$$

Here, $\epsilon_{\gamma,k}^{\mathrm{SSG}}(\lambda)$ is the subsampled shuffle Gaussian RDP defined in Equation 10, with k the number of shuffled instances, γ the subsampling rate.

The direct evaluation of the above equation is cumbersome numerically, as it involves a sum over n. We opt for a more efficient computation, which requires the expression in Equation 9, that is,

$$\frac{e^{-\lambda/2\sigma^2}}{n^\lambda}\sum_{\substack{k_1+\ldots+k_n=\lambda;\\ k_1,\ldots,k_n\geq 0}}\binom{\lambda}{k_1,\ldots,k_n}e^{\sum_{i=1}^n k_i^2/2\sigma^2}, \tag{12}$$

to be a monotonically decreasing function with respect to n. Although not proven rigorously, our empirical scan over n shows that it is indeed monotonically decreasing (at least within our scope of investigation). With this assumption, we present the following theorem for calculating the shuffled check-in Gaussian RDP with efficiency.

Theorem 3 (Shuffled Check-in Gaussian RDP)**.** *Let n be the total number of users, and γ be the check-in rate. Also let $\epsilon_{\gamma,m}^{\mathrm{SSG}}(\lambda)$ be the subsampled shuffle Gaussian RDP defined in Equation 10, with k the number of shuffled instances, γ the subsampling rate. If Equation 12 is a monotonically decreasing function with respect to n, the shuffled check-in Gaussian mechanism satisfies $(\lambda, \epsilon^{\mathrm{SCI}}(\lambda))$-RDP, where*

$$\epsilon^{\mathrm{SCI}}(\lambda) \leq \frac{1}{\lambda-1}\log\left(e^{(\lambda-1)\epsilon_{\gamma,1}^{\mathrm{SSG}}(\lambda)-\Delta^2 n\gamma/2} + e^{(\lambda-1)\epsilon_{\gamma,(1-\Delta)n\gamma+1}^{\mathrm{SSG}}(\lambda)}\right). \tag{13}$$

Here, $\Delta \in [0,1]$ is arbitrary.

Table 1: the value of shuffle ϵ under composition. The clones method Feldman et al. (2022) is compared with our RDP-based method.

No. of composition	1	2	3	4	5	6	7
Clones	0.18623	0.38461	0.59516	0.79355	1.02241	1.22689	1.43138
Ours	0.22820	0.22820	0.22821	0.22821	0.22821	0.22822	0.22822

Proof sketch. The proof follows closely to the techniques used in Liew et al. (2022a), where monotonicity and the Chernoff bound are utilized to prove the theorem. The full proof can be found in Appendix B. ■

5 Numerical Results

In this section, we present the numerical evaluation results of the shuffle Gaussian and its extensions. Additional results can be found in Appendix D.

5.1 Numerical results of the shuffle Gaussian

An upper bound on the shuffle model with an (ϵ, δ)-LDP randomizer is given by the clones method (Feldman et al., 2022), which is our target of comparison. According to Feldman et al. (2022), given n instances undergone (ϵ_0, δ_0)-LDP randomization satisfying $\epsilon_0 \leq \log(\frac{n}{16\log(2\delta)})$ for any $\delta \in [0, 1]$, and being shuffled, the overall DP is $(\epsilon, \delta + (s^\epsilon + 1))(1 + e^{-\epsilon_0}/2)n\delta_0)$-DP. Here, $\epsilon = O(\epsilon_0 \frac{\sqrt{e^{\epsilon_0}\log(1/\delta)}}{\sqrt{n}})$ when $\epsilon_0 > 1$.

The Gaussian mechanism is however governed by the noise variance, σ, and there is an infinite pair of corresponding (ϵ, δ). To use the clones method, we fix σ, $\delta_0 = \delta$ and scan over δ to find parameters satisfying $\delta_{\text{final}} \leq 1/n$, which is the overall DP δ parameter. Privacy accounting for iterative application of shuffle Gaussian is performed with the strong composition theorem (Kairouz et al., 2015).

In Table 1, we compare privacy accounting using the clones method with ours (with RDP), fixing $n = 60,000$, $\sigma = 9.48$ (corresponding to $(1, 1/n)$-LDP) and global sensitivity 1. We set the maximum λ to be evaluated as 30. While the clones method gives tighter result without composition, the RDP accounting shows tighter results by a large margin with composition. For composition results with more iterations, see Appendix D.1.

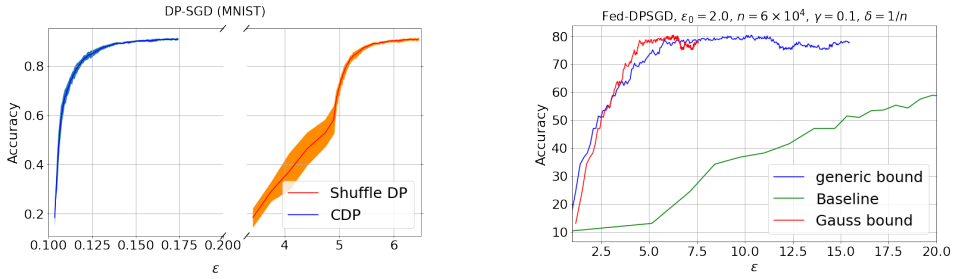
5.2 Shuffle Gaussian for machine learning tasks

Privacy accounting of DP-SGD. We first investigate the privacy-utility trade-offs of substituting the central-DP version of DP-SGD with the shuffle model to resolve the discrepancy in open-source implementations, as mentioned in Section 4.1. Using the shuffle model, a weak trust model, inevitably leads to a drop in utility compared to the highly trusted central-DP model. We show that nevertheless the shuffle model can achieve relatively satisfactory performance within a reasonable range of ϵ's.

We use the MNIST handwritten digit dataset (LeCun et al., 2010) to train a convolutional neural network. In the shuffle model, each sample's gradient (of dimension d) is clipped, $g \leftarrow g \cdot \min\left\{1, \frac{C}{|g|_2}\right\}$, and perturbed with Gaussian noise, $\tilde{g} \rightarrow g + \mathcal{N}(0, C^2\sigma^2 I_d)$, where C and σ are the clipping size and noise multiplier respectively. The aggregated gradients are then used to update the model at each iteration: $\sum_{i=1}^{B}\left\{\tilde{g}_i + \mathcal{N}(0, C^2\sigma^2 I_d)\right\} = \sum_{i=1}^{B}\tilde{g}_i + \mathcal{N}(0, BC^2\sigma^2 I_d)$, with B the batch size.

The central-DP model adds Gaussian noises with noise multiplier σ_{CDP} to the *aggregated* gradients of a batch instead: $\sum_{i=1}^{B}\tilde{g}_i + \mathcal{N}(0, C^2\sigma_{\text{CDP}}^2 I_d)$. For fair comparisons, we set $\sigma_{\text{CDP}} = \sqrt{B}\sigma$ such that the total amount of noise added is the same for both models. We use the MNIST test dataset to evaluate the training performance.

We set the parameters as follows: batch size 60, $\delta = 10^{-5}$, noise multiplier $\sigma = 0.86$ (corresponding to $(\epsilon_0, \delta_0) = (12, 10^{-5})$), clipping size $C = 0.05$, learning rate 0.05. Note that to obtain reasonable ϵ



(a) DP-SGD

(b) Distributed learning

Figure 1: On the left is the comparison of privacy-utility trade-offs of the standard DP-SGD, under the central DP (blue) and the shuffle DP (orange) trust models. On the right is the accuracy versus ϵ under the distributed learning setting, "Gauss bound" (in red) being our approach.

in the shuffle model, the noise multiplier has to be set to larger values compared to those set under the conventional central-DP assumption (Abadi et al., 2016). Figure 1a shows the trade-off of using the shuffle model, where we compare the total privacy budget spent computed using moments accountant (Google) (central DP) with our proposed method (shuffle model). It can be seen that using the shuffle model results in a trade-off of roughly a factor of $O(10)$ in ϵ. Still, optimal accuracy (90% for selected hyperparameters) is achievable within a reasonable amount of privacy budget ($\epsilon \lesssim 10$). We relegate details as well as other results (including an experiment on CIFAR10) to Appendix D.

Distributed learning. We perform an experiment under the shuffled check-in protocol with a setup similar to the one given in Liew et al. (2022a). Again, the MNIST dataset is used with each user holding one data instance from it, with a convolutional neural network (same as above) used as the target model $\theta \in \mathbb{R}^d$ under the distributed learning setting. In each round, each participating user calculates the gradient using the received model. As before, each user clips the gradient and add Gaussian noise to it before sending it to the server. The total number of user is 60,000, i.e., the size of the MNIST train dataset. The check-in rate is set to be $\gamma = 0.1$, $C = 0.01$, $\sigma = 5$. The default test dataset is used to evaluate the classification accuracy.

We compare our approach with two methods introduced in Liew et al. (2022a). The first method performs conservatively the privacy accounting of approximate DP of subsampling and shuffling (Baseline). The second method converts approximate DP of (subsampled) shuffling to RDP, and perform the privacy accounting under the shuffled check-in RDP framework (generic bound).

Figure 1b shows the accuracy-versus-ϵ plot of our experiment. Here, a total number of 5,540 rounds of training has been run. Our accounting requires less ϵ to reach optimal accuracy (80%). Note also that the generic accounting method in Liew et al. (2022a) is rather sophisticated (further requiring an inner round of optimization) and requires $O(n)$ times more computation compared to ours.

6 Conclusion

In this paper, we have analyzed the RDP of the shuffle Gaussian mechanism, and applied it to machine learning. Our results resolve the open problems posed by De et al. (2022); Koskela et al. (2021). Still, some questions remain open: Calculating the RDP at large λ appears computationally intensive. It has deep relation with number theory and looking for hints from there to calculate these values is a possible future direction. A rigorous proof of the monotonicity of Equation 12 for the shuffled check-in protocol is also needed.

We hope that our privacy accountant of shuffle Gaussian can help open up an avenue for more in-depth studies on various datasets and settings of learning/analytics under the shuffle model, using the relatively simple Gaussian mechanism, similar to DP-SGD. For example, performing shuffle Gaussian with adaptive gradient clipping (Andrew et al., 2021; Pichapati et al., 2019), adaptive learning rate (Koskela & Honkela, 2020), private SGD with momentum (McMahan et al., 2018), etc..

REFERENCES

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*, pp. 308–318. ACM, 2016.

Galen Andrew, Om Thakkar, Brendan McMahan, and Swaroop Ramaswamy. Differentially private learning with adaptive clipping. *Advances in Neural Information Processing Systems*, 34:17455–17466, 2021.

George E Andrews and Kimmo Eriksson. *Integer partitions*. Cambridge University Press, 2004.

Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. In *Annual International Cryptology Conference*, pp. 638–667. Springer, 2019.

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*, pp. 2496–2506. PMLR, 2020a.

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, 2020b.

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*, pp. 464–473. IEEE, 2014.

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.

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*, pp. 441–459, 2017.

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*, pp. 1175–1191, 2017.

Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pp. 635–658. Springer, 2016.

Clément L Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. *Advances in Neural Information Processing Systems*, 33:15676–15688, 2020.

David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

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*, pp. 375–403. Springer, 2019.

Soham De, Leonard Berrada, Jamie Hayes, Samuel L Smith, and Borja Balle. Unlocking high-accuracy differentially private image classification through scale. *arXiv preprint arXiv:2204.13650*, 2022.

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.

Cynthia Dwork and Guy N Rothblum. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*, 2016.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Eurocrypt*, volume 4004, pp. 486–503. Springer, 2006a.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pp. 265–284. Springer, 2006b.

Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pp. 51–60. IEEE, 2010.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

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*, pp. 2468–2479. SIAM, 2019.

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.

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)*, pp. 954–964. IEEE, 2022.

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, 2021a.

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*, pp. 2521–2529. PMLR, 2021b.

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)*, pp. 338–343. IEEE, 2021c.

Google. Tensorflow privacy. https://github.com/tensorflow/privacy.

Sivakanth Gopi, Yin Tat Lee, and Lukas Wutschitz. Numerical composition of differential privacy. *Advances in Neural Information Processing Systems*, 34:11631–11642, 2021.

Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *International conference on machine learning*, pp. 1376–1385. PMLR, 2015.

Peter Kairouz, Ziyu Liu, and Thomas Steinke. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *International Conference on Machine Learning*, pp. 5201–5212. PMLR, 2021a.

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, 2021b.

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.

Jerome Kelleher and Barry O'Sullivan. Generating all partitions: a comparison of two encodings. *arXiv preprint arXiv:0909.2331*, 2009.

Jon Kleinberg and Eva Tardos. *Algorithm design*. Pearson Education India, 2006.

Antti Koskela and Antti Honkela. Learning rate adaptation for differentially private learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 2465–2475. PMLR, 2020.

Antti Koskela, Joonas Jälkö, and Antti Honkela. Computing tight differential privacy guarantees using fft. In *International Conference on Artificial Intelligence and Statistics*, pp. 2560–2569. PMLR, 2020.

Antti Koskela, Mikko A Heikkilä, and Antti Honkela. Tight accounting in the shuffle model of differential privacy. *arXiv preprint arXiv:2106.00477*, 2021.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Seng Pei Liew, Satoshi Hasegawa, and Tsubasa Takahashi. Shuffled check-in: Privacy amplification towards practical distributed learning. *arXiv preprint arXiv:2206.03151*, 2022a.

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*, 2022b.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017a.

H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017b.

H Brendan McMahan, Galen Andrew, Ulfar Erlingsson, Steve Chien, Ilya Mironov, Nicolas Papernot, and Peter Kairouz. A general approach to adding differential privacy to iterative training procedures. *arXiv preprint arXiv:1812.06210*, 2018.

Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pp. 263–275. IEEE, 2017.

Jack Murtagh and Salil Vadhan. The complexity of computing the optimal composition of differential privacy. In *Theory of Cryptography Conference*, pp. 157–175. Springer, 2016.

Venkatadheeraj Pichapati, Ananda Theertha Suresh, Felix X Yu, Sashank J Reddi, and Sanjiv Kumar. Adaclip: Adaptive clipping for private sgd. *arXiv preprint arXiv:1908.07643*, 2019.

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*, pp. 245–248. IEEE, 2013.

Jonathan Ullman. Cs7880: Rigorous approaches to data privacy, spring 2017. URL `http://www.ccs.neu.edu/home/jullman/PrivacyS17/HW1sol.pdf`.

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*, pp. 1226–1235. PMLR, 2019.

Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, et al. Opacus: User-friendly differential privacy library in pytorch. *arXiv preprint arXiv:2109.12298*, 2021.