Communication Efficient Secure and Private Multi-Party Deep Learning

Sankha Das
Microsoft Research India
t-sankhadas@microsoft.com

Sayak Ray Chowdhury
Microsoft Research India
sayak261090@gmail.com

Nishanth Chandran
Microsoft Research India
nichandr@microsoft.com

Divya Gupta
Microsoft Research India
divya.gupta@microsoft.com

Satya Lokam
Microsoft Research India
satya.lokam@microsoft.com

Rahul Sharma
Microsoft Research India
rahsha@microsoft.com

Abstract

Distributed training that enables multiple parties to jointly train a model on their respective datasets is a promising approach to address the challenges of large volumes of diverse data for training modern machine learning models. However, this approach immediately raises security and privacy concerns; both about each party wishing to protect its data from other parties during training and preventing leakage of private information from the model after training through various inference attacks. In this paper, we address both these concerns simultaneously by designing efficient Differentially Private, secure Multiparty Computation (DP-MPC) protocols for jointly training a model on data distributed among multiple parties. Our DP-MPC protocol in the two-party setting is 56-794× more communication-efficient and 16-182× faster than previous such protocols. Conceptually, our work simplifies and improves on previous attempts to combine techniques from secure multiparty computation and differential privacy, especially in the context of ML training.

Keywords

Differential Privacy, Secure Multi-Party Computation, Secure and Private Deep Learning, Discrete Gaussian Mechanism

1 Introduction

Over the last two decades, application areas of Machine Learning (ML) have exploded. At the same time, the ML community faces the bottleneck of access to ever increasing volumes of rich training data. This bottleneck is only made worse by expanding regimes of legal regulations on the use of personal data [4] and controversies about copyright violations and data ownership. One natural approach to address the issue of large volumes of diverse training data is to enable multiple parties with their own respective data sets to run a *distributed* training algorithm on the union of their data to build a common model. For example, several banks might want to use their respective transaction data sets to build a common model for fraud detection without relying on a central entity. This approach immediately raises two critical concerns.

Proceedings on Privacy Enhancing Technologies 2025(1), 169–183
© 2025 Copyright held by the owner/author(s).
https://doi.org/10.56553/popets-2025-0010

The first concern is that these parties might not trust each other and fear exposing private information from their own data set to other parties through the distributed training algorithm. In recent years, secure Multi-Party Computation (MPC) has emerged as a promising solution framework to this problem, called *secure training*, (and the complementary problem of secure inference). By now, secure training and inference protocols have been constructed for several deep learning models using techniques such as secret sharing [6, 44, 50, 51, 63, 66, 67] (and references therein) in the MPC framework and homomorphic encryption [26, 33, 47, 65].

The second concern comes from potential leakage from the model itself after it is trained on the collective data. A slew of attacks such as Membership Inference Attacks (MIA's), Attribute Inference Attacks, and Data Reconstruction Attacks [9, 17, 19, 34, 61, 74], use a deployed model to make cleverly chosen inference queries to learn sensitive information from, or even entirely reconstruct, data items from the underlying training data. Differential Privacy (DP) [24] has emerged as the *de facto* solution framework to protect against this kind of leakage. A vast amount of research shows that models trained by differentially private training algorithms, such as DP-SGD in the seminal work of Abadi *et al.* [1], are resistant to the classes of attacks mentioned above [29, 30, 52, 74]. DP gives mathematically quantifiable guarantees on dataset privacy making it a highly dependable privacy-protection paradigm and used in services of tech giants such as Google [68], Apple [64] and even in the US Census [2].

To summarize, distributed training raises two kinds of concerns: (i) security of one party's data against other parties *during* training and (ii) privacy leakage from deployed model *after* training, through inference attacks.

In this paper, we address both these concerns simultaneously by constructing Differentially Private, secure Multi-Party Computation (DP-MPC) protocols for jointly training a model on data distributed among multiple parties. We do this by combining techniques from the areas of secure MPC protocols and Differential Privacy for ML. Our protocols improve the previous state of the art in such protocols by several orders of magnitude, and make some new conceptual contributions.

1.1 Our Contributions

Our contributions are summarized below.

- *A new private neural network training algorithm*: We first present a new algorithm for training neural networks with differential privacy - at a high level, this algorithm adds multiple discrete

Gaussian noise samples to the gradient values at each iteration in a fixed-point training algorithm (as opposed to DP-SGD[1] that added a single continuous Gaussian noise sample). We prove privacy of this new algorithm and further also extend the privacy proof to the case where an adversary learns a strict subset of these noise samples. We also introduce a moments accountant for the discrete Gaussian mechanism (as opposed to the continuous accountant from [1]), which we believe to be of independent interest in DP training.

- *A new private training toolkit*: We empirically show that our training algorithm preserves accuracy relative to the DP-SGD algorithm. Along this way, we develop a toolkit for fixed-point ML training with privacy. To the best of our knowledge, this is the first DP framework that comes with an end-to-end proof of privacy of its implementation since it does not use floating-point arithmetic which has been shown to be vulnerable to precision-based attacks on DP mechanisms [36, 40, 48].
- *A communication-efficient protocol for secure and private deep learning*: We then convert the above algorithm into a *secure* and private n-party training protocol. We do this by using existing MPC protocols [44] to securely realize the fixed-point training algorithm while having each of the n parties sample a single noise sample *locally* and adding it to the *secret shares* of the gradients at each iteration. Using our above privacy proof, we prove that this protocol is private as well. We implement our protocol for secure and private training and demonstrate that for the case of 2-party training, our protocol is at least 56-794× communication-efficient and 16-182× faster than the current state-of-the-art protocols [43] for the same task.

We note that all our secure DP Training protocols naturally generalize to more than two parties.

1.2 Our Techniques

We now present a high-level overview of the techniques used in our secure and private training protocol. Let us first consider the private training algorithm DP-SGD [1]. Our training method differs from DP-SGD on two counts. First, while standard DP-SGD works over reals, we make use of fixed-point arithmetic since MPC protocols are more efficient over fixed-point arithmetic. To enable this, we sample (random) noise from the *discrete* Gaussian distribution instead of the continuous Gaussian distribution. Second, to avoid expensive (single) noise sampling via MPC, we sample n noise samples in each training iteration (looking ahead, n will be the number of parties in the multi-party training protocol). We add these n noise samples to the (clipped) gradient. Third, unlike standard use of DP-SGD, in the context of distributed training, an adversary controlling a subset of parties, will also have access to a subset of added noise samples, and privacy must be argued against such an adversary. As one of our main technical contributions, we show that the above modified training algorithm also satisfies differential privacy. This does not directly follow from prior works, e.g., the privacy analysis of [1], due to reasons we elaborate next.

First, while it is easy to see that if we sample multiple values from a continuous Gaussian distribution and take their sum, then the resulting distribution is also a continuous Gaussian distribution, this, unfortunately, *does not* hold for discrete Gaussian distributions – adding multiple discrete Gaussian samples results in a distribution that is far from a discrete Gaussian with an approximation error, which must be controlled to obtain any meaningful privacy guarantees. We control this approximation error by adapting techniques from [41]. Additionally, we extend the moments accountant [1] to the case of the discrete Gaussian mechanism. Second, DP-SGD [1] does not consider privacy in the case where n noise samples are added and the adversary is additionally also provided a strict subset of these noise samples. However, to argue privacy against an adversary controlling a subset of the parties in the multi-party training protocol, we also prove this stronger guarantee. Combining these techniques, gives us our proof of privacy.

Next, we build on this private training algorithm to build our secure and private multiparty training protocol. To do this, we use an off-the-shelf MPC protocol from the MP-SPDZ library [44] to securely emulate each functionality of our protocol. To obtain a secure protocol for noise sampling, since we add n noise samples and prove our privacy guarantee against an adversary that learns a strict subset of these, we have each party *locally* sample one noise value, using the discrete Gaussian noise sampling technique from [15]. This gives us a highly efficient secure and private training protocol.

1.3 Related Work

Secure ML training with private inputs and parameters has been long studied in the domain of MPC. Various works have tackled ML tasks such as regression [7, 39, 51, 62], neural network training and inference [21, 35, 38, 45, 49, 51, 63, 67, 69] (and references therein) and inference for transformer-based models [31, 32, 54]. While the secure training protocols preserve data-privacy *during* the training, they do not guarantee protection of the underlying datasets against membership inference attacks on the models *after* training. To protect against such inference attacks, several works [3, 42, 56, 57] attempt to combine MPC protocols for secure training with differential privacy. Unfortunately, all of them make use of continuous noise distributions and use floating point arithmetic, which could inherently lead to vulnerabilities that violate DP privacy guarantees [48]. In contrast, in our protocols, we use *discrete* noise sampling and fixed point arithmetic which are compatible with both differential privacy and MPC protocols. Previously, Kairouz *et al.* [41] studied differentially private federated learning using discrete Gaussian noise distributions. Indeed, we rely heavily on techniques from their work in our proofs. Their work, however, does not consider secure training using secret-sharing in the MPC setting, which is our focus here. The work by Keller *et al.* [43] introduced an MPC protocol for two parties to jointly sample a discrete Gaussian random variable, producing a single noise sample secret-shared between the parties. However, as we will show in Section 5, the communication costs of noise generation using their protocol is prohibitively high for applications such as DP-SGD, where a large number of noise samples need to be generated in each training iteration.

1.4 Organization

The rest of the paper is organized as follows. Section 2 discusses relevant background for differential privacy and MPC. We describe our DP training algorithm with multiple noise samples and state its privacy guarantees in Section 3. In Section 4, we convert this algorithm to an n-party training protocol using secure MPC primitives. We present experimental results for privacy-accuracy trade-offs achieved by our training protocol and compare our protocol's efficiency to related work in terms of communication and runtime in Section 5. Section 6 concludes the paper.

2 Preliminaries

We begin by setting notation used in the rest of the paper and present the necessary background on differential privacy and secure multiparty computation.

2.1 Notation

The symbols $\mathbb{Z}, \mathbb{Z}^+, \mathbb{Q}$ and $\mathbb{R}$ denote the set of integers, positive integers, rational numbers and real numbers, respectively. We represent the set $\{1, 2, \cdots, t\}$ by $[t]$ for $t \in \mathbb{Z}^+$ and λ represents the computational security parameter. The ring of w-bit integers is represented by $\mathbb{Z}_W$ where $W = 2^w$. We use $\mathbb{P}[X = x]$ to represent the probability of a random variable X taking a value x, under a given probability distribution. $\mathbf{I}_d$ denotes the $d \times d$ identity matrix. A vector is represented by $\vec{v}$ and the i-th element of the vector $\vec{v}$ is represented by $\vec{v}[i]$.

Fixed-Point Numbers. Finite-bitwidth computers have a fixed number of bits assigned to represent real numbers, with up to a certain level of precision for non-integral numbers. Real numbers which do not have a finite representation (e.g. irrational numbers) cannot be exactly represented on finite-bitwidth computers. Hence, all real numbers are represented to some degree of approximation using either a fixed-point representation [73] or a floating-point representation [53]. Fixed-point representation allows representing real numbers as integers from a ring, by scaling the real number by a given scaling factor. A real number $r \in \mathbb{R}$ can be represented as a *fixed-point number* $x \in \mathbb{Z}_W$ with bitwidth w and scale f, where $x = \lfloor r \cdot 2^f \rceil \bmod W$. The real value corresponding to a fixed-point number x with bitwidth w and scale f is $\frac{x}{2^f}$.

2.2 Differential Privacy

Differential Privacy (DP) [24, 25] is a technique by which privacy of a dataset is preserved by adding random noise to responses of queries on the dataset (e.g. gradients computed during training ML models). Training datasets often contain sensitive information pertaining to individuals' identity, health records, financial statements, etc. which must be kept private. Even if the training dataset itself is kept protected, allowing inference on the model still makes the dataset vulnerable to membership inference attacks [16, 61]. The goal of these attacks is to determine if a particular sample is present in the dataset on the basis of the output of the model on carefully chosen inference inputs. Differential privacy (amongst other things) protects against membership inference attacks by injecting random noise into the model training algorithm. We provide a formal definition of DP below.

Definition 2.1 (Adjacent datasets). Let $\mathcal{D}$ be the set of all datasets comprising of m records. Two datasets $D = \{d_1, \cdots, d_m\}$ and $D' = \{d'_1, \cdots, d'_m\}$ in $\mathcal{D}$ are said to be adjacent if they differ in exactly one record. That is, there exists exactly one index $i^* \in [m]$ such that $d_{i^*} \neq d'_{i^*}$ and $d_i = d'_i$ for all $i \neq i^*$.

Definition 2.2 (Differential Privacy). Let $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ be a randomized mechanism over $\mathcal{D}$ with range $\mathcal{R}$ – the set of all possible outcomes of $\mathcal{M}$. Let $\varepsilon > 0$ and $\delta \in (0, 1]$. Then $\mathcal{M}$ is said to satisfy (ε, δ)-differential privacy if for all adjacent datasets $D, D' \in \mathcal{D}$ and $S \subseteq \mathcal{R}$, it holds that

$$\mathbb{P}[\mathcal{M}(D) \in S] \leq e^{\varepsilon}\, \mathbb{P}[\mathcal{M}(D') \in S] + \delta. \tag{1}$$

In this work, we are interested in training ML models and hence, the mechanism corresponds to the private training algorithm. The parameter ε in Equation (1) is the privacy budget of the mechanism $\mathcal{M}$. A smaller value of ε indicates that outputs of $\mathcal{M}$ on two adjacent datasets will be close. This makes it difficult for an adversary to detect the presence of a single data or record in the training dataset only by looking at the output of the model, implying better privacy protection of an individual's sensitive data. The parameter δ specifies the maximum acceptable probability with which one can allow the above privacy to not hold, referred to as the failure probability of differential privacy. Overall, small values of ε and δ imply better privacy for the training dataset. Finally, we consider differential privacy against an adversary which has some auxiliary information about the mechanism.

Definition 2.3. Consider an adversary $\mathcal{A}$ that has some auxiliary information, Aux, about the randomized mechanism $\mathcal{M}$ (alongside the mechanism's output). We say that $\mathcal{M}$ is (ε, δ)-DP against $\mathcal{A}$ if for all adjacent datasets $D, D' \in \mathcal{D}$ and $S \subseteq \mathcal{R}$, it holds that

$$\mathbb{P}[\mathcal{M}(D) \in S | \mathsf{Aux}] \leq e^{\varepsilon}\, \mathbb{P}[\mathcal{M}(D') \in S | \mathsf{Aux}] + \delta .$$

Differential Privacy via Gaussian Mechanism. To ensure differential privacy for a dataset, a query q operating on the dataset needs to be transformed into a differentially private mechanism $\mathcal{M}$. Typically, this is done by adding noise to the query's output and this noise is sampled from some specific statistical distribution. Sampling noise from the Gaussian distribution has been shown to achieve tight privacy bounds [25]. The variance of the distribution is typically set according to the required level of privacy and the *sensitivity* of the query, i.e., how much the output of the query can potentially vary when subject to a change in the input.

Definition 2.4 (L_p-sensitivity). Let $q : \mathcal{D} \to \mathbb{R}^b$ be a query on $\mathcal{D}$. For any $p \geq 1$, the L_p-sensitivity $\Delta_p(q)$ of the query q is defined as the maximum L_p-distance between the outputs of q across all pairs of adjacent datasets D, D', i.e.

$$\Delta_p(q) = \max_{D, D' \in \mathcal{D}} ||q(D) - q(D')||_p .$$

The larger the sensitivity, the easier it is for an adversary to detect a change in the output of the query on different inputs and hence the variance is typically set proportional to the squared L_2-sensitivity of the query. Hence, the Gaussian mechanism for the query q is defined as

$$\mathcal{M}(D) = q(D) + \mathcal{N}(0, \Delta_2^2(q)\sigma^2 \mathbf{I}_b) , \tag{2}$$

where the noise multiplier σ is set greater than or equal to $\frac{\sqrt{2 \log(1.25/\delta)}}{\varepsilon}$ to ensure (ε, δ)-DP [25].

2.2.1 *DP-SGD Algorithm.*

The Gaussian mechanism is used to design a differentially private stochastic gradient descent (DP-SGD) algorithm for training (deep) neural networks [1]. The DP-SGD algorithm proceeds as follows. Consider a training dataset $D = (X, Y)$ with m records, where $X \in \mathbb{R}^{m \times h}$ contains h features for each of the m records and $Y \in \mathbb{R}^m$ denotes labels of the records. At each training iteration $\tau \in \{1, \ldots, T\}$, DP-SGD constructs a batch $\widetilde{D}_\tau$ of size ℓ_τ by including each record with probability $\gamma = \frac{\ell}{m}$, chosen i.i.d., where ℓ denotes the (expected) size of the (random) batch, i.e., $\mathbb{E}[\ell_\tau] = \ell$. For each sampled record $\widetilde{d}_{\tau,i} \in \widetilde{D}_\tau$, $1 \leq i \leq \ell_\tau$, the algorithm first computes the gradient $g_\tau(\widetilde{d}_{\tau,i})$ of a loss function[1] $\mathcal{L}$ with respect to the current model parameters $\theta_\tau \in \mathbb{R}^b$. It then clips the gradient to $\widehat{g}_\tau(\widetilde{d}_{\tau,i})$ such that its L_2-norm remains less than or equal to a threshold C (called the clipping threshold hereon). Next, the algorithm samples a vector of noise samples from the Gaussian distribution $\mathcal{N}(0, C^2\sigma^2\mathbf{I}_b)$, scales it down by the batch size ℓ, and adds the noise vector to the average clipped gradient. Note that the noise vector is of the same length as the gradient vector. Finally, DP-SGD updates the model parameters as $\theta_{\tau+1} = \theta_\tau - \zeta\widetilde{g}_\tau$, where $\widetilde{g}_\tau = \frac{1}{\ell}\left\{\sum_{i=1}^{\ell_\tau} \widehat{g}_\tau(\widetilde{d}_{\tau,i}) + \mathcal{N}(0, C^2\sigma^2\mathbf{I}_b)\right\}$ denotes the average noisy gradient calculated in the previous step. ζ denotes the learning rate.

At each training iteration, the addition of noise to the clipped gradients makes the model differentially private w.r.t. to the respective batch, according to the definition of the Gaussian mechanism. Note that the batch $\widetilde{D}_\tau$ at iteration τ is randomly sampled and DP-SGD is run for total T iterations. [1] introduces a moments accountant that ensures the model θ_{T+1} obtained after T iterations is (ε, δ)-DP for any $\varepsilon < c_1\gamma^2 T$ and $\delta \in (0, 1]$ if one chooses $\sigma \geq c_2 \frac{\gamma\sqrt{T\log(1/\delta)}}{\varepsilon}$. Here, $\gamma = \frac{\ell}{m}$ is the probability with which each datapoint in D is picked into a batch at each training iteration and c_1 and c_2 are appropriate constants.

2.2.2 *Discrete Gaussian Mechanism [15].*

In this work, we consider the setting where multiple parties hold sensitive data and we train an ML model on combined data using a secure multiparty computation (MPC) protocol (Section 2.3.2). Since MPC protocols are much more efficient over fixed-point arithmetic compared to floating-point arithmetic [58], we consider SGD and DP-SGD algorithm over fixed-point arithmetic with fixed bitwidths and scale (section 2.1). For this, we must sample noise values from a discrete Gaussian distribution instead of a continuous distribution. The discrete Gaussian distribution $\mathcal{N}_\mathbb{Z}(\mu, \sigma^2)$ with mean $\mu \in \mathbb{Z}$ and variance $\sigma^2 \in \mathbb{R}_+$ has the probability mass function

$$\forall z \in \mathbb{Z}, \quad \mathbb{P}\left[Z = z\right] = \frac{e^{-\frac{(z-\mu)^2}{2\sigma^2}}}{\sum_{y\in\mathbb{Z}} e^{-\frac{(y-\mu)^2}{2\sigma^2}}} . \tag{3}$$

Recall that the DP-SGD algorithm adds noise samples to the clipped gradients, which are typically not integers. Hence, we want to sample noise samples from a finer distribution $\mathcal{N}_{\alpha\mathbb{Z}}(0, C^2\sigma^2)$, where the support of the distribution is the set of rational numbers $\alpha\mathbb{Z}$ for some rational $\alpha \in (0, 1]$. Note that the noise samples are represented as fixed-point numbers of bitwidth w and scale f and hence belong to the ring $\mathbb{Z}_W$. Similar to equation (2), a mechanism $\mathcal{M}(D) = q(D) + \mathcal{N}_{\alpha\mathbb{Z}}(0, \Delta_2^2(q)\sigma^2\mathbf{I}_b)$ is referred to as the *discrete Gaussian mechanism.* Cannonne *et al.* [15] show that the discrete Gaussian mechanism achieves the same privacy guarantee as that of the (continuous) Gaussian mechanism.

[1]The loss function $\mathcal{L}$ is task dependent, e.g., for binary classification task it could be log-loss, for regression task it could be square-loss etc.

2.3 Cryptographic Primitives

We now review cryptographic primitives used.

2.3.1 *Secret-Sharing.*

A secret-sharing scheme splits a secret x into random "shares" that can be put together to reconstruct the secret, but individually do not reveal any information about the secret. An (n, t)-linear secret-sharing scheme splits a secret x (from some ring) into n random shares $\{[\![x]\!]_j\}_{j\in[n]}$ such that:

(1) Each share $[\![x]\!]_j$ for $j \in [n]$ is an element of the ring.
(2) *(Security)* No set of t or less shares reveals any information about x.
(3) *(Correctness)* Any set of $t + 1$ distinct shares can be used to reconstruct the private input x. Formally, $\sum_{j\in I} c_j [\![x]\!]_j = x$, where I is an index set of size $t + 1$, c_j's are constants and addition is over the ring.

Linear secret-sharing schemes such as Shamir secret-sharing [60] and additive secret-sharing [13] are extensively used in secure MPC protocols.

2.3.2 *Secure Multiparty Computation.*

Secure Multi-Party Computation (MPC) [27, 72] is a protocol that allows $n > 1$ mutually distrusting parties to compute a public function f on private inputs held by each party. Formally, consider n parties $\{\mathcal{P}_i\}_{i\in[n]}$, each holding a private input D_i (dataset comprising a bunch of records in our case) and an n-party MPC protocol secure against t corruptions. Then, to compute $f(D_1, D_2, \ldots, D_n)$ (an ML model in our case), the parties can run the MPC protocol for this function, with the security guarantee that no subset of t parties get any information about other parties' data except what is revealed from the output of f.

In our secure and private training protocol, we make use of MPC protocols which operate in the following manner. A protocol computes a function f that is represented as a circuit comprising of different gates. Each gate computes some specific functionality required to evaluate the function such as comparison, exclusive-OR, scalar addition, multiplication and so on. The inputs to the gate are supplied on the input wires and the computation of the gate is produced on its output wire. The protocol should maintain the following invariant. The parties participating in the protocol begin with linear secret-shares of the inputs to a gate. Using a secure protocol, they interact with each other and end up with linear secret-shares of the output of the gate. Using the reconstruction property of linear secret-shares (section 2.3.1), the parties learn the function output by reconstructing the output shares of the final gate in the circuit.

The above structure required from the MPC protocol is satisfied by most secret-sharing based protocols and in particular by SPDZ-style protocols [44] that we use in our implementation.

2.3.3 *Threat-Model and Security.*

We consider n parties and a semi-honest adversary $\mathcal{A}$ that corrupts at most $t < n$ parties. We prove security using the standard simulation paradigm [46]. In an Ideal world, a trusted third-party computes the target function f using private inputs from each party, and makes the output available to all parties. In contrast, the Real world eliminates any such trusted third-party and the function output evaluated using an interactive protocol executed between the parties. In particular, consider the view of the adversary $\mathsf{view}_{\mathcal{A}}$ in the real world which consists of the inputs of the corrupt parties, the random bits of the corrupt parties and the messages that each one of them receives during the protocol execution. A protocol realizing a function f is said to be secure if for every adversary $\mathcal{A}$ in the real world, there exists a simulator $\mathcal{S}$ in the ideal world which given the inputs and the function output of the corrupt parties alone, produces the views of the corrupt parties which is computationally indistinguishable from the view of $\mathcal{A}$ in real protocol execution. Intuitively, this indistinguishability implies that the views of the corrupt parties reveal no more information about the honest parties' inputs than the inputs and outputs of corrupt parties.

2.4 Distributed DP Protocols

In order to show that secure protocols realizing a differentially private mechanism also preserve privacy, the work of [10, 11] introduced the notion of distributed differentially private protocols. We briefly review them here, starting with the following definitions introduced in these works but adapted to our setting. To this end, let D denote a dataset comprising of m records which is disjointly distributed among n parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$, i.e. $D = \cup_{i=1}^{n} D_i$ and $D_i \cap D_j = \emptyset$ for all $i \neq j$. With this, we define the notion of neighbouring datasets following the literature in [10, 11].

Definition 2.5. (Ψ**-Neighbouring Datasets**). Given a positive integer $n \geq 2$ and an index set $\Psi \subseteq [n]$, two datasets $D = \cup_{i=1}^{n} D_i$ and $D' = \cup_{i=1}^{n} D'_i$ are said to be Ψ*-neighbouring* if there exists exactly one index $i^* \notin \Psi$ such that the subsets D_{i^*} and D'_{i^*} are adjacent (i.e., they differ in exactly one record, see Definition 2.1) and for all other indices $i \neq i^*$, the subsets D_i, D'_i are same.

Note that if two datasets D, D' are Ψ-neighbouring, they are also adjacent. Following prior work [10, 11], we define differential privacy for a multi-party protocol. Consider n parties, and a set of t parties indexed by Ψ which are controlled by a semi-honest adversary $\mathcal{A}$. Differential privacy with respect to the joint view of the adversary is defined as follows.

Definition 2.6. (Distributed Differential Privacy.) A protocol Π computing a function f on inputs from parties $\{\mathcal{P}_i\}_{i \in [n]}$ is said to be $(t, (\varepsilon, \delta))$-DP if for all index sets $\Psi \subset [n]$ with $|\Psi| \leq t$, for all Ψ-neighboring datasets $D, D' \in \mathcal{D}$, and for all possible views $\mathcal{V}_{\mathcal{A}}$ of the adversary, it holds that

$$\mathbb{P}[\mathsf{view}_{\mathcal{A}}(D) \in \mathcal{V}_{\mathcal{A}}] \leq e^{\varepsilon}\, \mathbb{P}[\mathsf{view}_{\mathcal{A}}(D') \in \mathcal{V}_{\mathcal{A}}] + \delta \,,$$

where the probabilities are taken over the random bits of the parties in the protocol.

The above definition can be alternatively viewed as an algorithm which tries to distinguish between the views of the adversary in the case where there is a change in one of the honest party's private dataset. For the case of computationally bounded distinguisher, prior works [10, 11] extend the above definition as follows:

Definition 2.7. (Computational Differential Privacy.) A protocol Π is said to be computationally $(t, (\varepsilon, \delta))$-DP if for every probabilistic polynomial-time distinguisher $\mathcal{B}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all Ψ-neighbouring datasets $D, D' \in \mathcal{D}$, it holds that

$$\begin{aligned}\mathbb{P}\left[\mathcal{B}(\mathsf{view}_{\mathcal{A}}(D)) = 1\right] \leq & e^{\varepsilon}\, \mathbb{P}\left[\mathcal{B}(\mathsf{view}_{\mathcal{A}}(D')) = 1\right] + \delta \\ & + (e^{\varepsilon} + 1)\mathsf{negl}(\lambda)\,,\end{aligned}$$

where the probabilities are taken over the random inputs of the parties in protocol and the randomness of $\mathcal{B}$, and λ denotes the computational security parameter.[2]

We want to establish that a protocol securely realizing a functionality that is differentially private, is also differentially private under some given assumptions. This allows us to claim that an MPC protocol realizing our DP training algorithm that is executed between multiple parties achieves differential privacy against a semi-honest adversary controlling a subset of the parties. We restate [Lemma 2.7, [12]] that proves computational differential privacy for a protocol that securely realizes an (ε, δ)-differentially private functionality.

Lemma 2.1. *[12] Let f be (ε, δ)-DP, and let Π be a protocol computing f which is secure against collusions up to t parties, then Π is computationally $(t, (\varepsilon, \delta))$-DP.*

3 DP-SGD with Multiple Discrete Noise

We now consider a version of DP-SGD algorithm that is a modification of the one presented in Section 2.2.1. Our algorithm is different in two ways: first, the training is performed over fixed-point arithmetic; second, at each training iteration, n noise samples are sampled from the discrete Gaussian distribution and the sum of all these values is added to the clipped gradients.

The complete training procedure is outlined in Algorithm 1. The training dataset D contains m records, with h attributes in each record. First, similar to DP-SGD, we randomly sample a batch $\widetilde{D}_{\tau} = \{\widetilde{d}_{\tau,i}\}_{i \in [\ell_{\tau}]}$ of size ℓ_{τ} from the train dataset D in each training iteration τ, with each example in the dataset having probability $\gamma = \frac{\ell}{m}$ of being sampled. Then, we compute the gradient $g_{\tau}(\widetilde{d}_{\tau,i}) = \nabla_{\theta_{\tau}} \mathcal{L}(\theta_{\tau}, \widetilde{d}_{\tau,i})$ of the loss function $\mathcal{L}$ w.r.t. the current model θ_{τ} for each $\widetilde{d}_{\tau,i} \in \widetilde{D}_{\tau}$, $i \in [\ell_{\tau}]$. The gradients are clipped as

$$\widehat{g}_{\tau}(\widetilde{d}_{\tau,i}) = \frac{g_{\tau}(\widetilde{d}_{\tau,i})}{\max\left\{1, \frac{||g_{\tau}(\widetilde{d}_{\tau,i})||_2}{C}\right\}} \,, \tag{4}$$

where C is the clipping threshold. Next, in contrast to DP-SGD where only one continuous noise vector is sampled at each iteration, we sample n discrete noise vectors $\vec{\eta}_{\tau,1}, \ldots, \vec{\eta}_{\tau,n}$, where each $\vec{\eta}_{\tau,j} = \langle \vec{\eta}_{\tau,j}[1], \ldots, \vec{\eta}_{\tau,j}[b] \rangle \in \mathbb{Z}^b$ and each entry $\vec{\eta}_{\tau,j}[i] \sim \mathcal{N}_{\mathbb{Z}}(0, C^2\sigma^2)$ is independent and identically distributed (i.i.d.). These n noise vectors are added to the clipped gradients and the average noisy gradient is computed as

$$\widetilde{g}_{\tau} = \frac{1}{\ell}\left(\sum_{i=1}^{\ell_{\tau}} \widehat{g}_{\tau}(\widetilde{d}_{\tau,i}) + \sum_{j=1}^{n} \vec{\eta}_{\tau,j}\right) . \tag{5}$$

Finally, similar to DP-SGD, the model is updated as $\theta_{\tau+1} = \theta_{\tau} - \zeta \widetilde{g}_{\tau}$, where ζ is the learning rate. Note here that all the computations

[2]Although the term $(e^{\varepsilon} + 1)\mathsf{negl}(\lambda)$ can be subsumed under a negligible function for practical choices of ε, we keep it explicit to be consistent with prior literature [10, 11].

Algorithm 1: DP-SGD with Multiple Discrete Noise Samples

```
1: Input: Training dataset $D = (X, Y)$, $X \in \mathbb{Z}^{m \times h}$, $Y \in \mathbb{Z}^m$,
   learning rate $\zeta$, (expected) batch size $\ell$, training iterations $T$,
   clipping threshold $C$, noise std. dev. $\sigma$, number of generated
   noise vectors per iteration $n$
2: Initialize model parameters $\theta_1 \in \mathbb{Z}^b$
3: for iteration $\tau = 1, \ldots, T$ do
4:   Sample a subset $\widetilde{D}_\tau = \{\widetilde{d}_{\tau,1}, \ldots, \widetilde{d}_{\tau,\ell_\tau}\}$ from $D$ at random
5:   for $i = 1, \ldots, \ell_\tau$ do
6:     Compute gradient $g_\tau(\widetilde{d}_{\tau,i}) = \nabla_{\theta_\tau} \mathcal{L}(\theta, \widetilde{d}_{\tau,i})$
7:     Clip gradient $\widehat{g}_\tau(\widetilde{d}_{\tau,i}) = \frac{g_\tau(\widetilde{d}_{\tau,i})}{\max\left\{1, \frac{\|g_\tau(\widetilde{d}_{\tau,i})\|_2}{C}\right\}}$
8:   end for
9:   for $j = 1, \ldots, n$ do
10:    for $i = 1, \ldots, b$ do
11:      Sample noise $\vec{\eta}_{\tau,j}[i] \sim \mathcal{N}_{\mathbb{Z}}(0, C^2\sigma^2)$
12:    end for
13:  end for
14:  $\widetilde{g}_\tau = \frac{1}{\ell}\left(\sum_{i=1}^{\ell_\tau} \widehat{g}_\tau(\widetilde{d}_{\tau,i}) + \sum_{j=1}^{n} \vec{\eta}_{\tau,j}\right)$ ▷ average noisy gradient
15:  Update model $\theta_{\tau+1} = \theta_\tau - \zeta \widetilde{g}_\tau$
16: end for
17: return trained model $\theta_{T+1}$
```

are done using fixed-point arithmetic in practice.

3.1 Privacy Guarantee

In this section, we provide the privacy guarantee of Algorithm 1 against an adversary $\mathcal{A}$ that has the following auxiliary information Aux. Before the training starts, the adversary picks a set $\Psi \subset [n]$ of t indices. At the end of the training, the adversary receives T sets of noise $\chi_1, \ldots, \chi_T$, where each set χ_τ consists of t noise vectors indexed by Ψ. Therefore, $\mathcal{A}$ has the auxiliary information Aux $= (\chi_1, \ldots, \chi_T)$. Note that the rest of the $(n-t) \geqslant 1$ noise vectors added in each training iteration are still unknown to $\mathcal{A}$. We show below that due to these $(n-t)$ "secret" sets of noise vectors, the model given to the adversary is (ε, δ) differentially private according to Definition 2.3 for values of ε, δ that we calculate below.

Theorem 3.1. *For any $\delta > 0$, the trained model θ_{T+1} returned by Algorithm 1 is (ε, δ)-DP against the adversary $\mathcal{A}$ with auxiliary information Aux $= (\chi_1, \ldots, \chi_T)$ defined above, where $\varepsilon = \gamma\varepsilon_0\sqrt{2T\log(1/\delta)} + \gamma\varepsilon_0^2\sqrt{T}/2$. Here,*

$$\varepsilon_0 = \min\left\{\sqrt{\frac{1}{(n-t)\sigma^2} + 2\varepsilon' b}, \sqrt{\frac{1}{(n-t)\sigma^2} + \frac{2\sqrt{b}\varepsilon'}{\sqrt{n-t}\sigma} + \varepsilon'^2 b}, \frac{1}{\sqrt{n-t}\sigma} + \varepsilon'\sqrt{b}\right\}, \ \varepsilon' = 10\sum_{j=1}^{n-t-1} e^{-2\pi^2 C^2\sigma^2 \frac{j}{j+1}}, \text{ and}$$

T is the total number of iterations, $\gamma = \frac{\ell}{m}$ is the probability of sampling an example into a batch at each iteration and b is the total parameters of the model.

3.1.1 *Overview of Proof Technique.*

Before we formally prove Theorem 3.1, we briefly discuss the associated challenges and how we tackle those. First note that unlike the continuous Gaussian distribution, the sum of two random variables drawn i.i.d. from a discrete Gaussian distribution is *not* distributed according to a discrete Gaussian. Hence the n discrete Gaussian noise samples that our protocol adds in each training iteration do not simply add to a discrete Gaussian with n times the variance, and thus needs to be approximated. Therefore, the standard privacy analysis of DP-SGD [1] *does not* directly apply to our protocol, and extra care is needed to control the approximation error. Using techniques from [41], we account for this error in the privacy budget ε obtained from the vanilla DP-SGD algorithm. As we will see in the proof, the contribution of this error term to ε is expressible as a negative exponential function varying with the total noise samples n (for fixed training hyperparameters), thus giving us tight privacy guarantees. Thus, we obtain a new discrete Gaussian mechanism which in contrast to the Gaussian mechanism of DP-SGD, is differentially private using $n > 1$ noise samples. Further, we will prove privacy of this new mechanism against an adversary who learns *any* strict subset of t noise samples added in the training. We do so by expressing the output distribution of the private mechanism conditioned on the event that the adversary has access to some auxiliary information (the t noise samples in this case). Specifically, we express the privacy budget as a function of the total noise samples (n) and the number of noise samples that the adversary learns (t). Finally, while Abadi *et al.* [1] employ the moments accountant for continuous Gaussians to prove privacy guarantee of DP-SGD involving multiple training iterations, we derive from scratch the moments accountant for discrete Gaussian mechanism to prove our protocol's privacy. Now, we present the proof of Theorem 3.1.

Proof. Recall that $\widetilde{D}_\tau = \{\widetilde{d}_{\tau,1}, \ldots, \widetilde{d}_{\tau,\ell_\tau}\}$ is the randomly sampled batch of size ℓ_τ from the training dataset D at step $\tau \in [T]$. Define a query function $q : \widetilde{D}_\tau \longrightarrow \mathbb{Z}^b$ as $q(\widetilde{D}_\tau) = \sum_{i=1}^{\ell_\tau} \widehat{g}_\tau(\widetilde{d}_{\tau,i})$, where $\widehat{g}_\tau(\widetilde{d}_{\tau,i})$ denotes the clipped gradient. Note that the query q has L_2 sensitivity C and L_1 sensitivity $C\sqrt{b}$. Now, we define a mechanism $\mathcal{M} : \widetilde{D}_\tau \longrightarrow \mathbb{Z}^b$ as

$$\mathcal{M}(\widetilde{D}_\tau) = q(\widetilde{D}_\tau) + \sum_{j=1}^{n} \vec{\eta}_{\tau,j},$$

where $\vec{\eta}_{\tau,j} = \langle \vec{\eta}_{\tau,j}[1], \ldots, \vec{\eta}_{\tau,j}[b] \rangle$ and each $\vec{\eta}_{\tau,j}[i] \sim \mathcal{N}_{\mathbb{Z}}(0, C^2\sigma^2)$. Without loss of generality, assume that the adversary picks $\Psi = [t]$. Then $\chi_\tau = (\vec{\eta}_{\tau,1}, \ldots, \vec{\eta}_{\tau,t})$ denotes the set of t noise vectors that is available to the adversary $\mathcal{A}$ at iteration τ. Let $\vec{\eta}_{\tau,1:t} = \sum_{j=1}^{t} \vec{\eta}_{\tau,j}$ denote the sum of these t noise vectors. With this, define another mechanism $\mathcal{M}'(\widetilde{D}_\tau) = \mathcal{M}(\widetilde{D}_\tau) - \vec{\eta}_{\tau,1:t}$. Observe that, for any $z \in \mathbb{Z}^b$, we have

$$\mathbb{P}\left[\mathcal{M}(\widetilde{D}_\tau) = z \mid \chi_\tau\right] = \mathbb{P}\left[\mathcal{M}'(\widetilde{D}_\tau) = z - \vec{\eta}_{\tau,1:t} \mid \chi_\tau\right] . \quad (6)$$

Now the privacy guarantee of $\mathcal{M}'$ is dictated by the $(n-t)$ independent noise samples which are not in χ_τ. Now, if these noise samples were coming from a continuous Gaussian distribution $\mathcal{N}(0, \sigma^2)$, one could have computed the privacy guarantee in the same way as DP-SGD as if one single noise were sampled from the distribution $\mathcal{N}(0, (n-t)\sigma^2)$. Instead, since we sample noise from a discrete

Gaussian distribution $\mathcal{N}_{\mathbb{Z}}(0,\sigma^2)$, the noise variances won't just simply add up like the continuous case, and we suffer an additional privacy expenditure ε' roughly proportional to $(n-t)e^{-\pi^2C^2\sigma^2}$. Kairouz *et al.* [41] formalizes this for sums of n i.i.d. discrete Gaussians. Now, invoking [41, Theorem 5], the mechanism $\mathcal{M}'$ satisfies $\frac{1}{2}\varepsilon_0^2$ concentrated differential privacy [14], where ε_0 is defined as above. Now, using the *conversion lemma* from concentrated DP to approximate DP [41, Lemma 4], we have, for any $\delta > 0$, that $\mathcal{M}'$ satisfies $(\widetilde{\varepsilon},\delta)$-DP, where

$$\widetilde{\varepsilon} \le \varepsilon_0\sqrt{2\log(1/\delta)} + \varepsilon_0^2/2 \ .$$

By (6) and the post-processing property of DP [25], this implies the average noisy gradient $\widetilde{g}_\tau$ at iteration τ is $(\widetilde{\varepsilon},\delta)$-DP against an adversary with auxiliary information χ_τ. Note that this DP guarantee is with respect to the dataset $\widetilde{D}_\tau$ at iteration τ. Since $\widetilde{D}_\tau$ is also a random subset of D with sampling probability γ and we have total T iterations, we can now employ a moment accountant for the discrete Gaussian mechanism to conclude that the trained model θ_{T+1} is $\left(O\left(\gamma\widetilde{\varepsilon}\sqrt{T}\right),\delta\right)$-DP against the adversary $\mathscr{A}$ with auxiliary information $\mathsf{Aux} = (\chi_1,\ldots,\chi_T)$. We describe this moments accountant in Section 3.1.2. □

Remark 3.2. Consider the case where the adversary picks $t = n-1$ indices. In this case, we can exactly recover the privacy guarantee of DP-SGD from Theorem 3.1. To see this, note that the additional privacy expenditure $\varepsilon' = 0$ in this case, and hence $\varepsilon_0 = \frac{1}{\sigma}$. This yields $\varepsilon = \frac{\gamma}{\sigma}\sqrt{2T\log(1/\delta)} + \frac{\gamma}{\sigma^2}\sqrt{T}/2$. Hence, setting $\sigma \ge c_2\frac{\gamma\sqrt{T\log(1/\delta)}}{\varepsilon}$, we can ensure that our trained model θ_{T+1} is (ε,δ)-DP for any $\varepsilon < c_1\gamma^2 T$, where c_1 and c_2 are appropriate constants. This matches the privacy guarantee of DP-SGD indicating that we prove a strictly general result. We need this to prove the privacy guarantee of our secure SGD training protocol with n parties against an adversary $\mathscr{A}$ that controls t of them (see Section 4).

3.1.2 *Moments Accountant for the Discrete Gaussian Mechanism.*

Abadi *et al.* [1] introduced a moments accountant for the continuous Gaussian mechanism to calculate privacy guarantees for training a model over multiple training iterations using the DP-SGD algorithm. The moments accountant reports the private budget (ε) by bounding $\alpha_{\mathcal{M}}(\lambda)$, which represents the logarithm of the moment generating function of a Gaussian distribution evaluated at all $\lambda \le \sigma\ln\frac{1}{q\sigma}$. Here, q represents the sampling probability and σ represents the standard deviation of the Gaussian distribution used. Since our training algorithm samples noise vectors from the discrete Gaussian distribution, we derive a moments accountant for the discrete Gaussian mechanism on the lines of the continuous counterpart. Essentially, we show that Theorem 1 in [1] which derives privacy guarantees over T training iterations and a given noise level for the continuous Gaussian mechanism, can be restated for the discrete Gaussian mechanism. The crux of the proof of [Theorem 1, [1]] lies in proving [Lemma 3, [1]] which we restate below in the context of the discrete Gaussian mechanism:

Lemma 3.3 (Abadi *et al.* [1])**.** *Suppose that $f : \mathcal{D} \to \mathbb{Z}^b$ with $||f(\cdot)||_2 \le 1$. Let $\sigma \ge 1$ and let J be a sample from $[n]$ where each $i \in [n]$ is chosen independently with probability $q < \frac{1}{16\sigma}$. Then for any positive integer $\lambda \le \sigma\ln\frac{1}{q\sigma}$, the mechanism $\mathcal{M}(d) = \sum_{i\in J} f(d_i) + \mathcal{N}_{\mathbb{Z}}(0,\sigma^2\mathbf{I}_b)$ satisfies $\alpha_{\mathcal{M}}(\lambda) \le \frac{q^2\lambda(\lambda+1)}{(1-q)\sigma^2} + O(\frac{q^3\lambda^3}{\sigma^3})$, where symbols have the usual meaning as defined in [1].*

Proof. The proof sketch of lemma 3.3 above follows exactly from the proof of [Lemma 3, [1]]. Since the proof of [Lemma 3, [1]] relies on three other facts based on the properties of the continuous Gaussian distribution, we restate and prove these facts for the discrete Gaussian distribution in the form of the lemmas below. Throughout, we use $\mu_0(z)$ and $\mu_1(z)$ to denote the probability mass function of the distribution $\mathcal{N}_{\mathbb{Z}}(0,\sigma^2)$ and $\mathcal{N}_{\mathbb{Z}}(1,\sigma^2)$, similarly as in the proof of [Lemma 3, [1]].

The first lemma states that under the discrete Gaussian distribution $\mathcal{N}_{\mathbb{Z}}(0,\sigma^2)$, the expected value of $\exp\left(\frac{2az}{2\sigma^2}\right)$ is equal to $\exp\left(\frac{a^2}{2\sigma^2}\right)$. This result is similar the case of the continuous Gaussian distribution $\mathcal{N}(0,\sigma^2)$.

Lemma 3.4. *For any $a \in \mathbb{R}$, $\mathbb{E}_{z\sim\mu_0}\left[\exp\left(\frac{2az}{2\sigma^2}\right)\right] = \exp\left(\frac{a^2}{2\sigma^2}\right)$.*

Proof. Note that

$$\begin{aligned}\mathbb{E}_{z\sim\mu_0}\left[\exp\left(\frac{2az}{2\sigma^2}\right)\right] &= \sum_{z\in\mathbb{Z}} e^{\frac{2az}{2\sigma^2}}\frac{e^{\frac{-z^2}{2\sigma^2}}}{\sum_{y\in\mathbb{Z}}e^{\frac{-y^2}{2\sigma^2}}}\\ &= \frac{\sum_{z\in\mathbb{Z}}e^{-\frac{(z-a)^2+a^2}{2\sigma^2}}}{\sum_{y\in\mathbb{Z}}e^{-\frac{y^2}{2\sigma^2}}} = \frac{e^{\frac{a^2}{2\sigma^2}}\sum_{z\in\mathbb{Z}}e^{-\frac{(z-a)^2}{2\sigma^2}}}{\sum_{y\in\mathbb{Z}}e^{-\frac{y^2}{2\sigma^2}}}\\ &= \frac{e^{\frac{a^2}{2\sigma^2}}\sum_{z\in\mathbb{Z}}e^{-\frac{z^2}{2\sigma^2}}}{\sum_{y\in\mathbb{Z}}e^{-\frac{y^2}{2\sigma^2}}} = e^{\frac{a^2}{2\sigma^2}}\ .\end{aligned}$$

□

The next lemma proves the following distance relations between the discrete Gaussian distributions $\mathcal{N}_{\mathbb{Z}}(0,\sigma^2)$ and $\mathcal{N}_{\mathbb{Z}}(1,\sigma^2)$.

Lemma 3.5. *The following relations hold for $z \in \mathbb{Z}$:*

- $\forall z < 0 : |\mu_0(z) - \mu_1(z)| \leqslant -\frac{(z-1)\mu_0(z)}{\sigma^2}$
- $\forall z > 1 : |\mu_0(z) - \mu_1(z)| \leqslant -\frac{z\mu_1(z)}{\sigma^2}$
- $\forall z \in \{0,1\} : |\mu_0(z) - \mu_1(z)| \leqslant \frac{\mu_0(z)}{\sigma^2}$

Proof. For $z \leqslant 0$,

$$\begin{aligned}\frac{|\mu_0(z)-\mu_1(z)|}{\mu_0(z)} &= \frac{\frac{e^{-\frac{z^2}{2\sigma^2}}}{\sum_{y\in\mathbb{Z}}e^{-\frac{y^2}{2\sigma^2}}} - \frac{e^{-\frac{(z-1)^2}{2\sigma^2}}}{\sum_{y\in\mathbb{Z}}e^{-\frac{(y-1)^2}{2\sigma^2}}}}{\frac{e^{-\frac{z^2}{2\sigma^2}}}{\sum_{y\in\mathbb{Z}}e^{-\frac{y^2}{2\sigma^2}}}}\\ &= 1 - e^{\frac{2z-1}{2\sigma^2}} = 1 - e^{\frac{z-1}{\sigma^2}}\cdot e^{\frac{1}{2\sigma^2}}\\ &\leqslant \frac{z-1}{\sigma^2} \qquad (\text{since } ke^{-x} \geqslant 1-x \text{ for } k>1)\ .\end{aligned}$$

For $z \geqslant 1$,

$$\frac{|\mu_0(z)-\mu_1(z)|}{\mu_1(z)} = \frac{\frac{e^{-\frac{(z-1)^2}{2\sigma^2}}}{\sum_{y\in\mathbb{Z}} e^{-\frac{(y-1)^2}{2\sigma^2}}} - \frac{e^{-\frac{z^2}{2\sigma^2}}}{\sum_{y\in\mathbb{Z}} e^{-\frac{y^2}{2\sigma^2}}}}{\frac{e^{-\frac{(z-1)^2}{2\sigma^2}}}{\sum_{y\in\mathbb{Z}} e^{-\frac{(y-1)^2}{2\sigma^2}}}}$$
$$= 1 - e^{-\frac{z}{\sigma^2}} \cdot e^{\frac{1}{2\sigma^2}} \leqslant -\frac{z}{\sigma^2} .$$

Observe that $|\mu_0(z)-\mu_1(z)|$ is symmetric around $z=\frac{1}{2}$, for $z \in \mathbb{R}$. We prove the third result for $0 \leqslant z \leqslant 1/2$, and the proof for $1/2 < z \leqslant 1$ follows from symmetry:

$$\frac{|\mu_0(z)-\mu_1(z)|}{\mu_0(z)} = 1 - e^{\frac{2z-1}{2\sigma^2}} = 1 - e^{\frac{-1}{\sigma^2}} \cdot e^{\frac{2z+1}{2\sigma^2}} \leqslant \frac{1}{\sigma^2} .$$

Therefore, for $z \in \{0,1\}$, $|\mu_0(z)-\mu_1(z)| \leqslant \frac{\mu_0(z)}{\sigma^2}$. □

Finally, lemma 3.6 below relates the t-th moment of $|z|$ (z is sampled from $\mathcal{N}_{\mathbb{Z}}(0,\sigma^2)$) to the double factorial of t.

Lemma 3.6. *$\mathbb{E}_{z\sim\mu_0}[|z|^t] \leqslant \sigma^t(t-1)!!$ where $k!!$ denotes the double factorial [70] of k.*

Proof. In the proof below, we use Fact 19 from [15] stating that $\forall\sigma \in \mathbb{R}^+$,

$$max\{1, \sigma\sqrt{2\pi}\} \leqslant \sum_{n\in\mathbb{Z}} e^{-\frac{n^2}{2\sigma^2}} \leqslant \sqrt{2\pi\sigma^2} + 1 .$$

This gives us

$$\mathbb{E}_{z\sim\mu_0}\left[|z|^t\right] = \sum_{z\in\mathbb{Z}} |z|^t \frac{e^{-\frac{z^2}{2\sigma^2}}}{\sum_{y\in\mathbb{Z}} e^{-\frac{y^2}{2\sigma^2}}} = 2\sum_{z=0}^{\infty} z^t \frac{e^{-\frac{z^2}{2\sigma^2}}}{\sum_{y\in\mathbb{Z}} e^{-\frac{y^2}{2\sigma^2}}}$$
$$\leqslant \frac{2}{max\{1,\sigma\sqrt{2\pi}\}} \sum_{z=0}^{\infty} z^t e^{-\frac{z^2}{2\sigma^2}}$$
$$\leqslant \frac{2}{max\{1,\sigma\sqrt{2\pi}\}} \int_0^{\infty} z^t e^{-\frac{z^2}{2\sigma^2}} dz .$$

By substituting $z^2 = 2\sigma^2 x$ in the integral above and using the definition of the Gamma function [59], the inequality becomes

$$\mathbb{E}_{z\sim\mu_0}\left[|z|^t\right] \leqslant \frac{\sigma^{t+1}2^{\frac{t+1}{2}}}{max\{1,\sigma\sqrt{2\pi}\}} \int_0^{\infty} x^{\frac{t-1}{2}} e^{-x} dx$$
$$\leqslant \frac{\sigma^{t+1}2^{\frac{t+1}{2}}}{max\{1,\sigma\sqrt{2\pi}\}} \Gamma\left(\frac{t+1}{2}\right) .$$

Now, we consider two cases. Case 1: $\sigma\sqrt{2\pi} \geqslant 1$: We have

$$\mathbb{E}_{z\sim\mu_0}\left[|z|^t\right] \leqslant \frac{\sigma^t \cdot 2^{\frac{t}{2}}}{\sqrt{\pi}} \cdot \Gamma\left(\frac{t+1}{2}\right) .$$

Case 2: $\sigma\sqrt{2\pi} < 1$: We have

$$\mathbb{E}_{z\sim\mu_0}\left[|z|^t\right] \leqslant \sigma\sqrt{2\pi}\left\{\frac{\sigma^t \cdot 2^{\frac{t}{2}}}{\sqrt{\pi}} \cdot \Gamma\left(\frac{t+1}{2}\right)\right\} \leqslant \frac{\sigma^t \cdot 2^{\frac{t}{2}}}{\sqrt{\pi}} \cdot \Gamma\left(\frac{t+1}{2}\right) .$$

In both cases above, $\mathbb{E}_{z\sim\mu_0}\left[|z|^t\right] \leqslant \sigma^t\left\{\frac{2^{\frac{t}{2}}}{\sqrt{\pi}}\Gamma\left(\frac{t+1}{2}\right)\right\} \leqslant \sigma^t(t-1)!!$ by the definition of double factorial. □

Lemmas 3.4-3.6 are used to prove upper bounds on the individual terms of the binomial expansion of [Equation (5), [1]], that says $\mathbb{E}_{z\sim\nu_1}\left[\left(\frac{\nu_0(z)}{\nu_1(z)}\right)^{\lambda+1}\right] = \sum_{t=0}^{t=\lambda+1}\binom{\lambda+1}{t}\mathbb{E}_{z\sim\nu_1}\left[\left(\frac{\nu_0(z)-\nu_1(z)}{\nu_1(z)}\right)^t\right]$ for $\nu_0(z) = \mu_0(z), \nu_1(z) = \mu_1(z)$. The proof of lemma 3.3 follows by substituting lemmas 3.4-3.6 in place of the corresponding fact (for the continuous Gaussian distribution) in the proof of [Lemma 3, [1]]. Note that the rest of the statements in the proof of [Lemma 3, [1]] do not depend on any specific property of the (continuous) Gaussian distribution. Hence, our proof for lemma 3.3 follows from the proof of [Lemma 3, [1]] and lemmas 3.4-3.6. □

Since the proof of the privacy guarantees reported by the moments accountant [Theorem 1, [1]] again does not depend on any specific property of the Gaussian distribution, we can apply the lemma 3.3 in conjunction with the proof of Theorem 1 in [1], thereby obtaining the proof of privacy guarantees reported by the moments accountant for the discrete Gaussian mechanism.

4 Secure and Private Training

We now present our protocol for secure and private n-party training. At a high level, for an n-party protocol tolerating t corruptions, we use a standard secure multi-party computation protocol to realize the SGD algorithm. Through this, for every iteration of the training, parties will begin with shares of all the values (inputs, gradients etc.) and end with shares of updated gradient values. To make the protocol differentially private, each of the parties will locally pick one noise sample from the discrete Gaussian distribution. These noise samples will then be (n,t) secret shared with all the n parties who will add them to the shares of the gradients that they had previously obtained to obtain secret shares of differentially private gradients (this is possible since we are working with an MPC protocol that operates over linear secret sharing schemes). The parties are then in a position to continue the next iteration of SGD training through MPC. For the special case of $(n,n-1)$ secure and private training, the parties need not even communicate shares of their individual noise samples and can simply treat their noise samples as $(n,n-1)$ additive shares of the final noise sample to be added. We provide more details below.

4.1 Secure DP Training with Multiple Parties

Consider a set of $n > 1$ parties $P = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n\}$ who wish to jointly train a model on private datasets D_i held by party P_i. We assume a static, semi-honest adversary $\mathcal{A}$ which controls a set of t parties ($t < n$), fixed before protocol execution begins.

Protocol Initialization. The training protocol begins with each party secret-sharing their (private) dataset with the other parties. The parties start with shares of initial model parameters θ_1 (which is a publicly agreed upon value). The noise distribution parameters (variance of the discrete Gaussian) and training hyperparameters (e.g. number of epochs, batch size, learning rate, clipping factor) are also public, and thus known to all parties. This initialization step does not require involvement of any central entity.

Model Training. In each training iteration τ, the protocol executes the following steps. The protocol first invokes the MPC protocol for the gates that compute the gradient of the loss function and clipping of the gradients by the clipping threshold. This protocol maintains the invariant specified in Section 2.3.2, i.e., that parties begin with (n, t) linear secret shares of the input and end the protocol with (n, t) linear secret shares of the output. At the end of this step, all parties in P obtain (n, t) linear secret-shares of the average clipped gradients $\widehat{g}$. In the next step, each party $\mathcal{P}_i$ samples a vector of noise samples $\vec{\eta}_i$ from the discrete Gaussian distribution *locally*. Observe that this local noise sampling step involves no interaction between the parties. Each party then scales down the noise samples by the batch size and forms (n, t) linear secret-shares of each noise sample. Finally, each party obtains a linear secret-share of the noise vector $\vec{\eta}_k$ for each $k \in [n]$. Each party locally adds these n shares of noise vectors to their respective shares of the average clipped gradient, obtaining linear secret-shares of the noisy average gradient $\widetilde{g}$ (while also maintaining the invariant as desired). The training iteration ends by invoking the gate for calculating the updated parameters according to the parameter update step (step 15) in algorithm 1, and each party obtaining shares of the updated model parameters. At the end of T training iterations, the parties reconstruct the final model parameters.

Note that our protocol incurs bulk of the communication only in the MPC steps corresponding to computing the conventional SGD algorithm (namely, batch sampling, forward propagation, gradient computation and parameter update) with an additional gradient clipping operation. The noise sampling comes for free, albeit with a minimal cost for secret-sharing the noise vectors which consist of w-bit ring elements.

Remark 4.1. In the case where the adversary controls $t = n - 1$ parties, we use an $(n, n - 1)$ additive secret-sharing scheme in our secure and private training protocol. In this case, each of the n parties sample noise in the training protocol. Each party adds their noise vector to the gradient share, without secret-sharing the noise vector with the other parties. Secret-sharing of the noise vectors is not required since each party implicitly holds a share of $\left(g + \sum_{k=1}^{k=n} \vec{\eta}_k\right)$, where g is the clipped gradient.

4.2 Privacy for n-Party Training

The following theorem summarizes the privacy for the n-party secure and private training protocol.

Theorem 4.2. *The training protocol described above is computationally (t, ε, δ)-differentially private.*

Proof. Observe that the adversary in the training protocol described above is identical to the adversary in Theorem 3.1. This follows from MPC security; since the above protocol is a secure MPC protocol [44], the view of the adversary in the above protocol can be completely simulated given only the model output. Now, according to Theorem 3.1, Algorithm 1 satisfies (ε, δ)-DP against the adversary $\mathcal{A}$ for values of ε and δ as calculated in the theorem. Therefore, invoking Lemma 2.1, the protocol above is computationally $(t, (\varepsilon, \delta))$-DP. □

4.2.1 Relation between Privacy Budget and Adversarial Setting. We study how the privacy budget ε varies with the number of corruptions in the multi-party training setup. Observe from Theorem 3.1 that ε depends on the number of parties n, number of corruptions t (i.e., noise samples given to the adversary), number of training iterations T, sampling probability γ, clipping factor C, noise multiplier σ, dimensions of the model b, and failure probability δ. In order to illustrate how ε varies with t, we consider the following setting with $n = 10$ parties. We train model A from table 2 and fix (hyper)parameters as follows: $T = 1000$, $\gamma = 0.01$, $C = 4$, $\sigma = 2$, $b = 79510$, $\delta = 10^{-5}$. As an example, for $t = 5$ corruptions, using the expressions in Theorem 3.1, we first calculate ε' by plugging in the required values to obtain $\varepsilon' \approx 9.12 \times 10^{-274}$. Next, we plug ε' into the expression for ε_0 along with n, t and σ to get $\varepsilon_0 = 0.224$ (ignoring lower order terms). Finally, using this value of ε_0 in the expression for ε, we get the overall privacy budget $\varepsilon = 0.232$. Similarly, we calculate the privacy budget for other values of $t = 1$ and 9 (presented in Table 1). As expected and as can be seen, the privacy budget is lower when there are fewer corrupt parties.

Number of corrupt parties (t)	**1**	**5**	**9**
Privacy Budget (ε)	0.172	0.232	0.540

Table 1: Relation of privacy budget and adversarial setting

5 Evaluation

In this section, we justify two claims. First, the accuracy of our secure and private training protocol matches the accuracy of standard DP-SGD [1] (i.e., in the single party case without security) on different benchmarks. Second, for the case of 2−party secure and private training (where prior works exist), our protocol is up to two orders of magnitude more efficient, both in terms of communication as well as runtime than prior state-of-the-art [43]. We first begin by describing the datasets and models we work with, in Section 5.1. Next, in Section 5.2, we show that our secure and private training protocol preserves accuracy (over standard DP-SGD) in the 2-party setting and has only marginally lower accuracy even when several parties contribute data. Finally, in Section 5.3, we show that in the 2-party setting, our protocol is orders of magnitude more efficient (both in terms of communication and runtime) than prior state-of-the-art [43]. Since, our protocol for making a secure SGD protocol also private (via differential privacy) incurs practically no communication when compared with any MPC protocol to compute the SGD algorithm itself, our improvements translate to the n-party case as well (when compared with any MPC protocol that could be used to sample noise).

5.1 Datasets and Models

We use two widely employed datasets for image classification, namely MNIST [23] and Fashion MNIST [71]. Each dataset contains 60000 training images and 10000 test images, each image being a 28×28 matrix of pixel values between 0 and 255. Additionally, each dataset contains labels for each test image, which denotes the class of the image by a digit from the set $\{0, 1, \ldots, 9\}$. The MNIST dataset contains images of handwritten digits from 0 to 9, while the Fashion MNIST dataset contains images of 10 clothing accessories such as sweaters, socks, shoes, etc. As a pre-processing step for each dataset, we normalize each image in the dataset by descaling the pixel value from [0, 255] to the range [0, 1]. In addition, we evaluate our results on a tabular dataset called the “CDC Diabetes Health Indicators” dataset taken from the UCI Repository [8]. This

Model Name	Reference	Architecture	No. of Parameters
A	[1]	784-100-10	79510
B	[51]	784-128-128-10	118282
C	[1]	784-500-10	397510
D	[8]	21-128-2	3074

Table 2: Neural Networks used in our Experiments

is a classification dataset consisting of health records of 253680 patients, each consisting of 21 features. The target attribute is a binary variable indicating whether the patient suffers from diabetes or not. We split the dataset into 200000 data points for training and the rest of the 53680 data points for the test. All the attributes are in numerical format and we do not perform any dataset pre-processing. We will refer to this dataset as the “Diabetes” dataset.

We use 4 different neural network [37] architectures in our experiments described in Table 2. The model architecture is denoted by a string "$x - y - z$" where x denotes the number of input features, y denotes the number of neurons in each hidden layer and z denotes the number of output neurons. The last column in Table 2 specifies the number of trainable parameters in the network which is the sum of weights and biases for each layer. Models A-C are used in experiments for the MNIST and Fashion MNIST datasets. We note that each image in the datasets above is a 2D matrix of 784 elements, which we flatten to a vector of 784 elements before inputting it to the neural network. Each hidden layer is implicitly followed by a ReLU activation layer [5]. Model D is used for evaluating our training protocol on the Diabetes dataset. In all models, the output layer is implicitly followed by a softmax activation [28] layer in each model which converts the output to a one-hot vector representing the class with the highest probability as predicted by the model. We use the cross-entropy loss function [76] as the optimizing function to train the networks. Note that we do not use any additional techniques in training such as adding dropout or normalization layers.

Hyperparameter	Dataset	
	MNIST/Fashion MNIST	Diabetes
Learning Rate (ζ)	0.1	0.1
Batch Size (ℓ)	500	400
Clipping Factor (C)	4	4
No. of Epochs (E)	10	5

Table 3: Hyperparameters used in each Experiment

Several works [18, 22, 55] have studied the influence of hyperparameters like the learning rate (ζ), batch size (ℓ), clipping factor (C) and noise multiplier (σ) along with their interdependence on each other on getting accuracy-privacy tradeoffs in differentially private training. These works are orthogonal to our contributions and hence in our experiments with different datasets, models and noise levels, we fix all hyperparameters, apart from the noise multiplier, across all experiments to the values in listed in table 3.

5.2 Accuracy v/s Privacy Tradeoff

We compare the accuracy of our protocol with the DP-SGD algorithm on the three fully connected neural network (FCNN) architectures listed in Table 2 over the two datasets. Since the MPC protocol that we use computes the cleartext functionality in a *bit-wise equivalent* manner, it suffices to run accuracy experiments with the cleartext code (with differential privacy). To illustrate the accuracy obtained in different settings (with or without privacy and/or security), we consider five different settings and find the accuracy of each model on the two datasets for all these settings. Table 4 summarises the features of each experimental setting.

Setting I represents the accuracy of the cleartext training algorithm with no privacy and no security. This training algorithm operates over floating-point arithmetic and is implemented in the Opacus [75] framework. Setting II represents the accuracy of a non-private but secure training protocol obtained through the use of MPC. As described earlier, such an algorithm, for efficiency reasons, must operate over fixed-point arithmetic. We use the MP-SPDZ [44] to generate the fixed-point cleartext training algorithm (which is bitwise equivalent to the output of the MPC protocol). Setting III reflects the accuracy of a private (but insecure) training algorithm - i.e., the accuracy of standard DP-SGD [1]. The training is over floating-point arithmetic and noise is sampled from the continuous Gaussian noise distribution, implemented via the the **`torch.normal`** function in Opacus [75]. Setting IV represents the accuracy of private and secure training as obtained through prior state-of-the-art [43]. Here the training is performed over fixed-point arithmetic and noise is sampled from the discrete Gaussian distribution [15] (via the MPC protocol, **`MDGauss`** in [43]). Here, note that a single noise sample is used per model parameter per iteration. We remark here that while the other subroutines used in the SGD algorithm were taken off-the-shelf from MP-SPDZ, we implemented a new subroutine for clipping the parameter gradients using the primitives available in the framework. Finally, setting V represents the accuracy of our private and secure training protocol in the 2-party setting. Here, the training is over fixed-point arithmetic and since every party locally adds a noise sample from the discrete Gaussian distribution (implemented using **`SampleDGauss`** [Algorithm 3, [15]]), 2 noise samples are added at every iteration to all the gradients.

Table 5 presents accuracy results for training in settings I-V for models A-D using the hyperparameters in Table 3 and using noise multiplier $\sigma = 2$ (the total noise variance is thus $C^2\sigma^2 = 64$). We obtain privacy guarantees as $(0.59, 10^{-5})$ for MNIST/Fashion MNIST and $(0.24, 10^{-6})$ for Diabetes dataset. We make two observations. First, the accuracy of our protocol is within a margin of 0.7% of the accuracy of the state-of-the-art secure and private training protocol [43]. Second, the accuracy of our protocol is at most $0.08 - 3.23\%$ lower than the accuracy of standard DP-SGD [1]. Next, in Figure 1, we show the trade-off between privacy and accuracy of our training protocol for varying values of the privacy budget ε (ranging from 0.2 to 1.8). As can be seen from table 5, this trade-off is acceptable.

5.2.1 *Training in the $(n, n-1)$ Setting.* We now show the accuracy of our protocol in the case when n parties perform the training (where $n-1$ of these parties are controlled by an adversary). As we discussed in Section 4, it is required that all n parties sample and add noise in the training protocol. Since, now, we are adding more

Setting	Description	Arithmetic Mode	Differential Privacy	Gaussian Noise Distribution	No. of Noise Samples/Iter.
I	Non-Private Non-Secure Training	Floating-Point	✗	-	0
II	Non-Private Secure Training	Fixed-Point	✗	-	0
III	Private Non-Secure Training	Floating-Point	✓	Continuous	1
IV	Private Secure Training ([43])	Fixed-Point	✓	Discrete	1
V	Private Secure Training (Our Protocol)	Fixed-Point	✓	Discrete	2

Table 4: Settings for Accuracy-Privacy Experiments

Dataset	Model Name	Test Accuracy in Different Settings (in %)				
		Setting I	Setting II	Setting III	Setting IV	Setting V
MNIST	A	93.50	94.11	90.70	91.23	90.62
	B	94.12	95.45	90.21	91.16	90.62
	C	93.86	94.71	90.99	91.58	91.06
Fashion MNIST	A	83.16	83.97	80.55	81.21	81.10
	B	82.98	84.08	79.37	81.05	80.90
	C	83.41	84.45	80.92	81.42	81.36
Diabetes	D	86.20	85.99	85.96	82.03	82.73

Table 5: Accuracy for different models after training in setting I-V

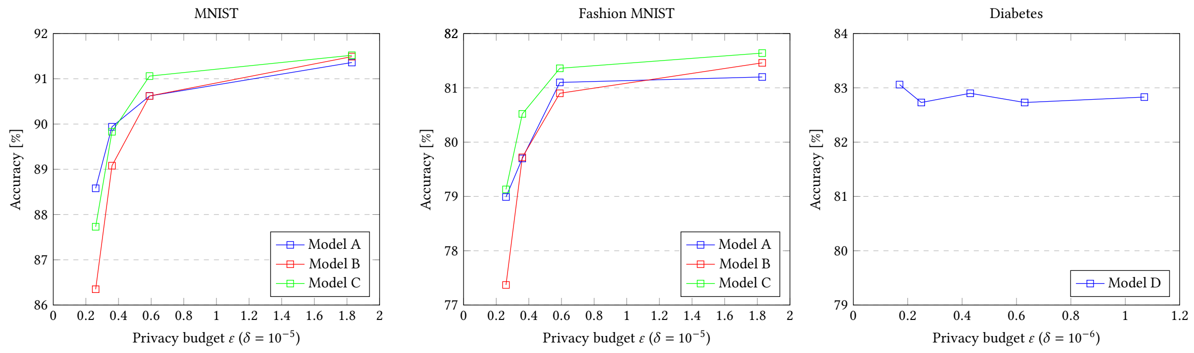

Figure 1: Accuracy for different levels of privacy for different models and datasets trained using our 2PC DP Protocol

noise samples to our training protocol, we investigate the resulting degradation in the accuracy that our models as we vary the number of parties. We present our accuracy results for training with multiple parties in Table 6 for models A-D in the $(3, 2)$, $(5, 4)$ and $(10, 9)$ adversarial settings where each party draws noise samples from the discrete Gaussian distribution by setting $\sigma = 2$, as in the previous experiments. Note that the other training hyperparameters are fixed as per Table 3 as before. As one can observe from the table, the accuracy of our protocol as we move from the 2-party setting (with 2 noise samples) to the 10−party setting (with 10 noise samples), degrades only by $\approx 5\%$. The privacy budget in these settings and chosen noise level remains under the acceptable budget of $\varepsilon = 2$.

(n,t)	MNIST			Fashion MNIST			Diabetes
	A	B	C	A	B	C	D
(2, 1)	90.62	90.62	91.06	81.10	80.90	81.36	82.73
(3, 2)	90.46	90.18	90.63	80.30	80.45	81.00	83.26
(5, 4)	89.76	88.82	89.77	80.12	79.50	80.19	82.76
(10, 9)	87.84	85.29	86.88	78.72	76.00	78.24	82.51

Table 6: Accuracy for setting of n parties with t corruptions.

Model Information			SGD Comm. (TB)	Communication for Noise Generation (TB)		Total Communication (TB) [SGD + Noise Generation]		Improvement Factor
Name	Total Parameters	Batch Size		Baseline*	Our Protocol	Baseline	Our Protocol	
A	79510	100	0.1	84.2	0	84.3	0.1	766×
		500	0.5			84.7	0.5	161×
		1000	1.0			85.2	1.0	81×
B	118282	100	0.2	125.3	0	125.5	0.2	724×
		500	0.8			126.1	0.8	151×
		1000	1.7			127.0	1.7	76×
C	397510	100	0.5	421.2	0	421.7	0.5	794×
		500	2.5			423.7	2.5	167×
		1000	5.0			426.2	5.0	84×
D	3074	100	0.006	3.3	0	3.306	0.006	539×
		500	0.030			3.330	0.030	111×
		1000	0.060			3.360	0.060	56×

*This is an estimate based on numbers reported for generating one noise sample in [43]

Table 7: Communication Comparison between Π_{Baseline} and Our 2PC Protocol (Π_{Us}) (1 training iteration)

Model Information			SGD Runtime (seconds)	Runtime for Noise Generation (sec)		Total Runtime (sec) [SGD + Noise Generation]		Improvement Factor
Name	Total Parameters	Batch Size		Baseline*	Our Protocol	Baseline	Our Protocol	
A	79510	100	596	86268	5	86864	601	144×
		500	1747			88015	1752	50×
		1000	3233			89501	3238	27×
B	118282	100	875	128335	7	129210	882	146×
		500	2703			131038	2709	48×
		1000	5077			133412	5084	26×
C	397510	100	2333	431298	38	433631	2371	182×
		500	6302			437600	6340	69×
		1000	11205			442503	11243	39×
D	3074	100	26	3339	1	3365	27	125×
		500	105			3444	106	32×
		1000	212			3551	213	16×

*This is a lower-bound based on just communication time

Table 8: Runtime Comparison of Π_{Baseline} and Our 2PC Protocol (Π_{Us}) in a 1GB/s 1ms RTT LAN Setting (1 training iteration)

5.3 Performance

The work of Keller *et al.* [43] shows how to sample noise samples from the discrete Gaussian distribution using a 2PC protocol. Their protocol can be used in conjunction with an MPC protocol for SGD to obtain a secure and private training protocol. This forms our baseline 2PC protocol Π_{Baseline} with which we compare our protocol. This protocol executes two subprotocols: 1) Π_{MDGauss} from [43] to sample noise from a discrete Gaussian distribution and secret-share the noise samples between the two parties, and 2) Π_{SGD} which carries out SGD training while implementing the functionality required in DP-SGD such as gradient clipping and adding the noise samples generated by Π_{MDGauss} to the gradients. More concretely, Π_{Baseline} executes DP-SGD between two training parties (each holding secret-shares of a dataset) while adding shares of Gaussian noise into the shares of the gradients for each party, for a given number of training iterations, where the noise itself is generated using Π_{MDGauss}.

In contrast, our protocol Π_{Us} just consists of one subprotocol, namely Π_{SGD} which in our case emulates the DP-SGD algorithm and adds noise sampled locally by each party to the gradient shares of the respective party. More concretely, Π_{Us} executes DP-SGD between two training parties (each holding secret-shares of a dataset) while adding *independent* Gaussian noise samples into the shares of the gradients for each party, for a given number of training iterations. We now do a comparative cost analysis between the two protocols Π_{Baseline} and Π_{Us} in terms of communication and runtime. Note that the performance of SGD training depends only on model architecture and batch size, and is identical for MNIST and Fashion MNIST due to identical structures of the two datasets.

5.3.1 Communication. Suppose protocol Π_{SGD} incurs a total communication of X GB in one training iteration for a given model and hyperparameter combination. Note that the only hyperparameter that Π_{SGD}'s communication depends on, is the batch size. Suppose the protocol Π_{MDGauss} incurs a total communication of Y GB to generate noise samples required in one training iteration. Y is given by simply multiplying the communication cost of generating one noise sample (roughly 1.085 GB as per [43]) by b (total number of trainable parameters in the model), since one noise sample needs to be added to the gradient of each parameter in the model in a training iteration. Therefore, the total communication incurred by protocol Π_{Baseline} in one training iteration is given by $X + Y$. On the other hand, the total communication incurred by our protocol Π_{Us} is just X, since noise sampling in our protocol is local and does not involve any communication. It is therefore evident that our end-to-end training protocol enjoys an improvement factor of $\left(\frac{X+Y}{X}\right)$ over Π_{Baseline}. We present this improvement factor concretely in Table 7 for training models A-C, each with three different batch sizes. As can be seen from the table, our protocol is $56 - 794\times$ more communication efficient than prior state-of-the-art.

5.3.2 Runtime. We run our experiments for neural network training on two F16s v2 Azure instances, each equipped with 16 CPU cores and 32 GB of RAM, with a bandwidth of 1 GB/s and 1 ms RTT operating in LAN setting. Since the code of [43] is not available, we conservatively estimate a lower bound on the runtime of their noise sampling protocol, dividing communication by bandwidth.

Now, suppose Π_{SGD} takes a total of t seconds to execute one iteration of training for a given model and hyperparameter combination. Note that t depends on the batch size used and is higher for higher batch size. We also obtain a lower bound on the runtime for generating one batch of noise samples by protocol Π_{MDGauss} [43] as $\frac{Y}{B}$, where Y is the communication incurred calculated in Section 5.3.1 and B represents the network bandwidth. Finally, we empirically measure the time t' taken by our algorithm **SampleDGauss** to (locally) generate one batch of noise samples for each party for one training iteration. The total runtime for one iteration of end-to-end training taken by our protocol is given by $t + t'$ seconds, while the value $\left(t + \frac{Y}{B}\right)$ represents a lower bound on the total runtime for one iteration of end-to-end training using protocol Π_{Baseline}. As we show in Table 8, our protocol is 16-182$\times$ faster than prior state-of-the-art [43]. From the tables, we see that the cost (runtime and communication) of SGD via MPC scales (almost) linearly with batch size. Also, the cost of noise sampling depends on the number of model parameters alone and is independent of the batch size. Hence, our improvements over prior work are higher for lower batch sizes. For batch size 500, our protocol has $111 - 167\times$ lower communication and $32 - 69\times$ lower runtime.

Remark 5.1. In Theorem 3.1, we analyze the privacy guarantee of our training algorithm (Algorithm 1), which similar to Abadi *et al.* [1], constructs batches by sampling each record i.i.d. with a certain probability (this process is called Poisson sampling) and thus enjoys an amplification of privacy. This, however, leads to variable-sized batches, which is technically challenging to handle in practice. To get around this issue in our implementation, we randomly shuffle all the records in the dataset and construct fixed-sized batches by going over the shuffled dataset sequentially, similar to the approach in almost all practical private deep learning frameworks. Chua *et al.* [20] observe that there exists a gap between the privacy proofs in literature using Poisson sampling and the practical implementations of private training algorithms. We remark that, as evident from [Figure 5, [20]], this subtle difference in the accounted value of privacy level ε is negligible for our choices of noise standard deviation σ, training iterations T and failure probability δ.

6 Conclusion

In this work, we introduced a new protocol for the secure and private multi-party training of machine learning models that is orders of magnitude more performant than prior works. The protocol allows n parties, t of which may be controlled by a semi-honest adversary, to securely train a neural network on private datasets held by each party, such that the trained model is differentially private against the adversary. As a stepping stone, we showed that adding n noise samples to DP-SGD (instead of 1 as is done in standard DP-SGD) preserves differential privacy even when the adversary learns $t < n$ out of these noise samples. This theorem may be of independent interest. Interestingly, we show that adding multiple noise samples to DP-SGD does not degrade the accuracy of the final trained model significantly. Our distributed noise generation method is compatible with existing secure MPC protocols and since the method comes with practically zero overhead, it can be used with any secure training algorithm to also make it private.

Acknowledgments

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318. https://arxiv.org/pdf/1607.00133

[2] John M Abowd. 2018. The US Census Bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2867–2867.

[3] Abbas Acar, Z Berkay Celik, Hidayet Aksu, A Selcuk Uluagac, and Patrick McDaniel. 2017. Achieving secure and differentially private computations in multiparty settings. In *2017 IEEE Symposium on Privacy-Aware Computing (PAC)*. IEEE, 49–59.

[4] Accountability Act. 1996. Health insurance portability and accountability act of 1996. *Public law* 104 (1996), 191.

[5] Abien Fred Agarap. 2018. Deep Learning using Rectified Linear Units (ReLU). *CoRR* abs/1803.08375 (2018). arXiv:1803.08375 http://arxiv.org/abs/1803.08375

[6] Nitin Agrawal, Ali Shahin Shamsabadi, Matt J Kusner, and Adrià Gascón. 2019. QUOTIENT: two-party secure neural network training and prediction. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1231–1247.

[7] Yoshinori Aono, Takuya Hayashi, Le Trieu Phong, and Lihua Wang. 2016. Scalable and secure logistic regression via homomorphic encryption. In *Proceedings of the sixth ACM conference on data and application security and privacy*. 142–144.

[8] Arthur Asuncion, David Newman, et al. 2007. UCI machine learning repository.

[9] Borja Balle, Giovanni Cherubin, and Jamie Hayes. 2022. Reconstructing Training Data with Informed Adversaries. In *2022 IEEE Symposium on Security and Privacy (SP)*. 1138–1156. https://doi.org/10.1109/SP46214.2022.9833677

[10] Amos Beimel, Iftach Haitner, Kobbi Nissim, and Uri Stemmer. 2020. On the round complexity of the shuffle model. In *Theory of Cryptography Conference*. Springer, 683–712.

[11] Amos Beimel, Kobbi Nissim, and Eran Omri. 2008. Distributed private data analysis: Simultaneously solving how and what. In *Advances in Cryptology–CRYPTO 2008: 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings 28*. Springer, 451–468.

[12] Amos Beimel, Kobbi Nissim, and Eran Omri. 2008. Distributed private data analysis: Simultaneously solving how and what. In *Advances in Cryptology–CRYPTO 2008: 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings 28*. Springer, 451–468.

[13] George Robert Blakley. 1979. Safeguarding cryptographic keys. In *Managing requirements knowledge, international workshop on*. IEEE Computer Society, 313–313.

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

[15] Clément L Canonne, Gautam Kamath, and Thomas Steinke. 2020. The discrete gaussian for differential privacy. *Advances in Neural Information Processing Systems* 33 (2020), 15676–15688. https://arxiv.org/abs/2004.00010

[16] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. 2022. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1897–1914.

[17] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*. 2633–2650.

[18] Yannis Cattan, Christopher A Choquette-Choo, Nicolas Papernot, and Abhradeep Thakurta. 2022. Fine-tuning with differential privacy necessitates an additional hyperparameter search. *arXiv preprint arXiv:2210.02156* (2022).

[19] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. 2021. Label-only membership inference attacks. In *International conference on machine learning*. PMLR, 1964–1974.

[20] Lynn Chua, Badih Ghazi, Pritish Kamath, Ravi Kumar, Pasin Manurangsi, Amer Sinha, and Chiyuan Zhang. 2024. How Private are DP-SGD Implementations?. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 235)*, Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (Eds.). PMLR, 8904–8918. https://proceedings.mlr.press/v235/chua24a.html

[21] Anders Dalskov, Daniel Escudero, and Marcel Keller. 2019. Secure evaluation of quantized neural networks. *arXiv preprint arXiv:1910.12435* (2019).

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

[23] Li Deng. 2012. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.

[24] Cynthia Dwork. 2006. Differential privacy. In *International colloquium on automata, languages, and programming*. Springer, 1–12.

[25] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[26] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy. In *Proceedings of The 33rd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 48)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.). PMLR, New York, New York, USA, 201–210. https://proceedings.mlr.press/v48/gilad-bachrach16.html

[27] Oded Goldreich, Silvio Micali, and Avi Wigderson. 2019. How to play any mental game, or a completeness theorem for protocols with honest majority. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*. 307–328.

[28] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[29] Ganesh Del Grosso, Georg Pichler, Pablo Piantanida, and Catuscia Palamidessi. 2021. Formalizing Attribute and Membership Inference Attacks on Machine Learning Models. *privacy Prsereving Machine Learning, ACM CCS Workshop, November 2021* (2021). https://ppml-workshop.github.io/ppml21/pdfs/ppml21-final48.pdf

[30] Chuan Guo, Brian Karrer, Kamalika Chaudhuri, and Laurens van der Maaten. 2022. Bounding Training Data Reconstruction in Private (Deep) Learning. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 8056–8071. https://proceedings.mlr.press/v162/guo22c.html

[31] Kanav Gupta, Neha Jawalkar, Ananta Mukherjee, Nishanth Chandran, Divya Gupta, Ashish Panwar, and Rahul Sharma. 2024. SIGMA: Secure GPT Inference with Function Secret Sharing. *Proceedings on Privacy Enhancing Technologies* (2024).

[32] Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, Guowen Xu, and Tianwei Zhang. 2022. Iron: Private inference on transformers. *Advances in neural information processing systems* 35 (2022), 15718–15731.

[33] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. 2017. Cryptodl: Deep neural networks over encrypted data. *arXiv preprint arXiv:1711.05189* (2017).

[34] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. 2022. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)* 54, 11s (2022), 1–37.

[35] Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiansheng Ding. 2022. Cheetah: Lean and fast secure {Two-Party} deep neural network inference. In *31st USENIX Security Symposium (USENIX Security 22)*. 809–826.

[36] Christina Ilvento. 2020. Implementing the exponential mechanism with base-2 differential privacy. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 717–742.

[37] Peter A Jansson. 1991. Neural networks: An overview. *Analytical chemistry* 63, 6 (1991), 357A–362A.

[38] Neha Jawalkar, Kanav Gupta, Arkaprava Basu, Nishanth Chandran, Divya Gupta, and Rahul Sharma. 2024. Orca: FSS-based Secure Training and Inference with GPUs. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 66–66. https://doi.org/10.1109/SP54263.2024.00063

[39] Yichen Jiang, Jenny Hamer, Chenghong Wang, Xiaoqian Jiang, Miran Kim, Yongsoo Song, Yuhou Xia, Noman Mohammed, Md Nazmus Sadat, and Shuang Wang. 2018. SecureLR: Secure logistic regression model via a hybrid cryptographic protocol. *IEEE/ACM transactions on computational biology and bioinformatics* 16, 1 (2018), 113–123.

[40] Jiankai Jin, Eleanor McMurtry, Benjamin IP Rubinstein, and Olga Ohrimenko. 2022. Are we there yet? timing and floating-point attacks on differential privacy systems. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 473–488.

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

[42] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. 2015. Secure multi-party differential privacy. *Advances in neural information processing systems* 28 (2015).

[43] Hannah Keller, Helen Möllering, Thomas Schneider, Oleksandr Tkachenko, and Liang Zhao. 2023. Secure Noise Sampling for DP in MPC with Finite Precision. Cryptology ePrint Archive, Paper 2023/1594. https://eprint.iacr.org/2023/1594 https://eprint.iacr.org/2023/1594.

[44] Marcel Keller. 2020. MP-SPDZ: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*. 1575–1590.

[45] Nishant Kumar, Mayank Rathee, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. 2020. Cryptflow: Secure tensorflow inference. In *2020*

IEEE Symposium on Security and Privacy (SP). IEEE, 336–353.

[46] Yehuda Lindell. 2020. Secure multiparty computation. *Commun. ACM* 64, 1 (2020), 86–96.

[47] Kentaro Mihara, Ryohei Yamaguchi, Miguel Mitsuishi, and Yusuke Maruyama. 2020. Neural network training with homomorphic encryption. *arXiv preprint arXiv:2012.13552* (2020).

[48] Ilya Mironov. 2012. On significance of the least significant bits for differential privacy. In *Proceedings of the 2012 ACM conference on Computer and communications security*. 650–661.

[49] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. 2020. Delphi: A cryptographic inference system for neural networks. In *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*. 27–30.

[50] Payman Mohassel and Peter Rindal. 2018. ABY3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 35–52.

[51] Payman Mohassel and Yupeng Zhang. 2017. SecureML: A System for Scalable Privacy-Preserving Machine Learning. In *2017 IEEE Symposium on Security and Privacy (SP)*. 19–38. https://doi.org/10.1109/SP.2017.12

[52] Prateeti Mukherjee and Satya Lokam. 2024. On the Query Complexity of Training Data Reconstruction in Private Learning. arXiv:2303.16372 [cs.LG]

[53] Jean-Michel Muller, Nicolas Brisebarre, Florent De Dinechin, Claude-Pierre Jeannerod, Vincent Lefevre, Guillaume Melquiond, Nathalie Revol, Damien Stehlé, Serge Torres, et al. 2018. *Handbook of floating-point arithmetic*. Springer.

[54] Q. Pang, J. Zhu, H. Möllering, W. Zheng, and T. Schneider. 2024. BOLT: Privacy-Preserving, Accurate and Efficient Inference for Transformers. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 133–133. https://doi.org/10.1109/SP54263.2024.00130

[55] Nicolas Papernot and Thomas Steinke. 2022. Hyperparameter Tuning with Renyi Differential Privacy. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

[56] Sikha Pentyala, Davis Railsback, Ricardo Maia, Rafael Dowsley, David Melanson, Anderson Nascimento, and Martine De Cock. 2022. Training differentially private models with secure multiparty computation. *arXiv preprint arXiv:2202.02625* (2022).

[57] Martin Pettai and Peeter Laud. 2015. Combining differential privacy and secure multiparty computation. In *Proceedings of the 31st annual computer security applications conference*. 421–430.

[58] Deevashwer Rathee, Anwesh Bhattacharya, Divya Gupta, Rahul Sharma, and Dawn Song. 2023. Secure Floating-Point Training. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 6329–6346. https://www.usenix.org/conference/usenixsecurity23/presentation/rathee

[59] Pascal Sebah and Xavier Gourdon. 2002. Introduction to the gamma function. *American Journal of Scientific Research* (2002), 2–18.

[60] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.

[61] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 3–18.

[62] Aleksandra B Slavkovic, Yuval Nardi, and Matthew M Tibbits. 2007. " Secure" Logistic Regression of Horizontally and Vertically Partitioned Distributed Databases. In *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*. IEEE, 723–728.

[63] Sijun Tan, Brian Knott, Yuan Tian, and David J Wu. 2021. CryptGPU: Fast privacy-preserving machine learning on the GPU. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1021–1038.

[64] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and Xiaofeng Wang. 2017. Privacy loss in apple's implementation of differential privacy on macos 10.12. *arXiv preprint arXiv:1709.02753* (2017).

[65] Tim van Elsloo, Giorgio Patrini, and Hamish Ivey-Law. 2019. SEALion: A framework for neural network inference on encrypted data. *arXiv preprint arXiv:1904.12840* (2019).

[66] Sameer Wagh, Divya Gupta, and Nishanth Chandran. 2019. SecureNN: 3-Party Secure Computation for Neural Network Training. *Proceedings on Privacy Enhancing Technologies* 2019 (07 2019), 26–49. https://doi.org/10.2478/popets-2019-0035

[67] Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. 2021. Falcon: Honest-Majority Maliciously Secure Framework for Private Deep Learning. *Proceedings on Privacy Enhancing Technologies* 2021 (01 2021), 188–208. https://doi.org/10.2478/popets-2021-0011

[68] Jinyan Wang, Zhou Tan, Xianxian Li, Yuhang Hu, et al. 2020. Differential privacy preservation in interpretable feedforward-designed convolutional neural networks. In *2020 IEEE 19th international conference on trust, security and privacy in computing and communications (TrustCom)*. IEEE, 631–638.

[69] Jean-Luc Watson, Sameer Wagh, and Raluca Ada Popa. 2022. Piranha: A {GPU} platform for secure computation. In *31st USENIX Security Symposium (USENIX Security 22)*. 827–844.

[70] Eric W Weisstein. 2002. Double factorial. *https://mathworld. wolfram. com/* (2002).

[71] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).

[72] Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In *27th annual symposium on foundations of computer science (Sfcs 1986)*. IEEE, 162–167.

[73] Randy Yates. 2009. Fixed-point arithmetic: An introduction. *Digital Signal Labs* 81, 83 (2009), 198.

[74] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*. IEEE, 268–282.

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

[76] Zhilu Zhang and Mert Sabuncu. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems* 31 (2018).