Efficient Differentially Private Secure Aggregation for Federated Learning via Hardness of Learning with Errors

Timothy Stevens
University of Vermont

Christian Skalka
University of Vermont

Christelle Vincent
University of Vermont

John Ring
MassMutual

Samuel Clark
Raytheon

Joseph Near
University of Vermont

Abstract

Federated machine learning leverages edge computing to develop models from network user data, but privacy in federated learning remains a major challenge. Techniques using differential privacy have been proposed to address this, but bring their own challenges. Many techniques require a trusted third party or else add too much noise to produce useful models. Recent advances in *secure aggregation* using multiparty computation eliminate the need for a third party, but are computationally expensive especially at scale. We present a new federated learning protocol that leverages a novel differentially private, malicious secure aggregation protocol based on techniques from Learning With Errors. Our protocol outperforms current state-of-the art techniques, and empirical results show that it scales to a large number of parties, with optimal accuracy for any differentially private federated learning scheme.

1 Introduction

Mobile phones and embedded devices are ubiquitous and allow massive quantities of data to be collected from users. The recent explosion in data collection for *deep learning* has led to significant new capabilities, from image recognition to natural language processing. But collection of private data from phones and devices remains a major and growing concern. Even if user data is not directly disclosed, recent results show that trained models themselves can leak information about user training data [42, 50].

Private data for training deep learning models is typically collected from individual users at a central location, by a party we call the *server*. But this approach creates a significant computational burden on data centers, and requires complete trust in the server. Many data owners are rightfully skeptical of this arrangement, and this can impact model accuracy, since privacy-conscious individuals are likely to withhold some or even all of their data.

A significant amount of existing research aims to address these issues. *Federated learning* [30] is a family of decentralized training algorithms for machine learning that allow individuals to collaboratively train a model without collecting the training data in a central location. This addresses computational burden in data centers by shifting training computation to the edge. However, federated learning does not necessarily protect the privacy of clients, since the updates received by the server may reveal information about the client's training data [42, 50].

Combining *secure aggregation* [8, 12] with *differential privacy* [21, 29] ensures end-to-end privacy in federated learning systems. In principle, secure aggregation allows user updates to be combined without viewing any single update in isolation. Methods based on differential privacy add noise to updates to ensure that trained models do not expose information about training data. However, secure aggregation protocols are expensive, in terms of both computation and communication. The state-of-the-art protocol for aggregating large vectors (as in federated deep learning) is due to Bonawitz et al. [12]. This protocol has a communications expansion factor of more than 2x when aggregating 500 vectors of length 20,000 (i.e. it doubles the communication required for each client), and requires several minutes of computation time for the server.

In this paper we propose a new protocol, called FLDP, that supports scalable, efficient, and accurate federated learning with differential privacy, and that does not require a trusted server. A main technical contribution of our work is a novel method for *differentially private secure aggregation*. This method significantly reduces computational overhead as compared to state-of-the-art– our protocol reduces communications expansion factor from 2x to 1.7x for 500 vectors of length 20,000, and reduces computation time for the server to just a few seconds. The security of this method is based on the learning with errors (LWE) problem [36]– intuitively, the noise added for differential privacy *also* serves as the noise term in LWE.

To obtain computational differential privacy [33] FLDP uses the distributed discrete Gaussian mechanism [29] and gradient clipping, with secure aggregation accomplished efficiently via our new method. The accuracy of our approach is comparable to that achieved by the *central model* of differential privacy, while providing better efficiency and thus scalability of previous distributed approaches. We implement our approach and evaluate it empirically on neural network architectures for MNIST and CIFAR-10, measuring both accuracy and scalability of the training procedure. In terms of accuracy, our results are comparable with central-model approaches for differentially private deep learning (on MNIST: 95% accuracy for $\varepsilon \leq 2$; on CIFAR-10: 70% accuracy for $\varepsilon \leq 4$).

1.1 Contributions

In summary, our contributions are:

1. A novel malicious-secure aggregation protocol that outperforms previous approaches to gradient aggregation with differential privacy.
2. A new end-to-end protocol (FLDP) for privacy-preserving federated learning setting that uses our secure aggregation protocol to provide differential privacy even in the presence of malicious clients and a malicious server.
3. Analytic and empirical results that support our scalability claims, and that show our protocol achieves nearly the same accuracy as *central-model* approaches for differentially private deep learning on practical models for MNIST and CIFAR-10.

2 Overview

We study the problem of *distributed differentially private deep learning without a trusted data curator*. Our setting includes a set of *clients* (or data owners), each of whom holds some sensitive data, and a *server* that aggregates gradients generated by clients to obtain a model for the entire federation. The goal is to obtain a differentially private model, without revealing any private data to either the server or other clients.

2.1 Background: General Problem Setting

Deep learning. Deep learning [25] attempts to train a neural network architecture $\mathcal{F}(\theta, \cdot)$ by training its *parameters* (or *weights*) θ in order to minimize the value of a *loss function* $\mathcal{L}(\theta, \cdot)$ on the training data. Advances in deep learning have lead to significant gains in machine learning capabilities in recent years. Neural networks are typically trained via *gradient descent*: each iteration of training calculates the gradient of the loss on a subset of the training data called a *batch*, and the model parameters are updated based on the negation of the gradient.

Traditional deep learning techniques assume the training data is collected centrally; moreover, recent results suggest that trained models tend to memorize training data, and training examples can later be extracted from the trained model via *membership inference attacks* [15, 28, 42, 50]. When sensitive data is used to train the model, both factors represent significant privacy risks to data owners.

Federated learning. Federated learning is a family of techniques for training deep neural networks without collecting the training data centrally. In the simplest form of federated learning (also called *distributed SGD*), each client computes a gradient locally and sends the gradient (instead of the training data) to the server. The server averages the gradients and updates the model. More advanced approaches compute gradients in parallel to reduce communication costs; Kairouz et al. [30] provide a survey.

Differentially private deep learning. Differential privacy [21] is a rigorous privacy framework that provides a solution to the problem of privacy attacks on deep learning models. Achieving differential privacy typically involves adding noise to results to ensure privacy. Abadi et al. [2] introduced DP-SGD, an algorithm for training deep neural networks with differential privacy. DP-SGD adds noise to gradients before each model update. Subsequent work has shown that this approach provides strong privacy protection, effectively preventing membership inference attacks [15, 28, 50].

DP-SGD works in the *central model* of differential privacy—it requires the training data to be collected centrally (i.e. on a single server). The participant that holds the data and runs the training algorithm is often called the *data curator* or *server*, and in the central model, the server must be trusted. Central-model algorithms offer the best accuracy of known approaches, at the expense of requiring a trusted server.

Federated learning with local differential privacy. The classical method to eliminate a trusted server is *local differential privacy* [21], in which each client adds noise to their own data before sending it to the server. Local differential privacy algorithms for gradient descent have been proposed, but for deep neural networks, this approach introduces too much noise to train useful models [10]. The major strength of local differential privacy is the threat model: privacy is assured for each client, *even if every other client and the server act maliciously*. The local model of differential privacy has also been relaxed to the *shuffle model* [16, 22], which lies between the local and central models but which has seem limited use in distributed machine learning.

Secure aggregation. The difference in accuracy between the central and local models raises the question: can cryptography help us obtain the benefits of both, simultaneously? Several *secure aggregation protocols* have been proposed in the context of federated learning to answer this question in the affirmative. These approaches yield the *accuracy of the central model*, but *without a trusted server*.

Secure aggregation protocols allow a group of clients—some of whom may be controlled by a malicious adversary—

Setting	*Bonawitz*	*Bell*	FLDP
Client Communication	$O(k+m)$	$O(\log k+m)$	$O(m+k+n)$
Client Computation	$O(k^2+km)$	$O(\log^2 k+m\log k)$	$O(mn+k\log k)$
Server Communication	$O(k^2+km)$	$O(k\log k+km)$	$O(mk+n)$
Server Computation	$O(mk^2)$	$O(k\log^2 k+km\log k)$	$O(mk+mn+k\log k)$

Table 1: Communication and computation complexities of FLDP compared with the state of the art.

to compute the *sum* of the clients' privately-held vectors (e.g. gradients, in federated learning), without revealing individual vectors. The state-of-the-art protocol is due to Bonawitz et al. [12]. For k clients and length-m vectors, this protocol requires $O(k^2+mk)$ computation and $O(k+m)$ communication per client, and $O(mk^2)$ computation and $O(m^2+mk)$ communication for the untrusted server. Bell et al. [8] improve these to $O(\log^2 k+m\log n)$ computation and $O(\log k+m)$ communication (client) and $O(k\log^2 k+km\log k)$ computation and $O(k\log k+km)$ communication (server). These complexity classifications are summarized in Table 1.

2.2 Efficient Secure Aggregation in the Differential Privacy Setting

We present a new protocol for secure aggregation (detailed in Section 4) specifically for the setting of differentially private computations. Our protocol reduces client communications complexity to $O(m+k)$ and server communications complexity to $O(mk)$, where as above we have k parties aggregating vectors of length m, and demonstrates excellent concrete performance in our empirical evaluation (Section 5). These analytic results are summarized in Table 1 for easy comparison with previous work.

Threat model. Like previous work, we target both the semi-honest setting (in which all clients and the server correctly execute the protocol) and the malicious setting (in which the server and some fraction of the clients may act maliciously). These threat models are standard in the MPC literature [23], and match the ones targeted by Bonawitz et al. [12] and Bell et al. [8]. In the semi-honest version, we assume that the server is honest-but-curious, and that the clients have a corrupted honest-but-curious subset with an honest majority. In the malicious version, we assume that the server is malicious, and that the clients have a corrupted malicious subset with an honest majority. We present both versions in Section 4 (note that the results in Table 1 are for semi-honest protocol versions in all cases).

Data poisoning & other threats. As with other secure aggregation protocols, our threat model does not prevent the adversary from submitting maliciously-crafted data. The ability to submit malicious data can enable attacks on the resulting trained machine learning models, such as data poisoning [44], property inference [34] and model inversion [26]. Like previous approaches for secure aggregation [8, 12], our protocol does not prevent these attacks.

2.3 Paper Roadmap

The rest of the paper is organized as follows. In Section 3 we describe the *ideal* but insecure functionality of our main protocol that assumes a trusted server, along with our threat model. The trusted server assumption is removed in Section 4 where we present novel techniques for lightweight malicious-secure aggregation based on LWE. In that Section we also describe the threat model and state formal security results for the protocol, and analyze its algorithmic complexity. In Section 5 we discuss methods and results for two experiments-one that further evaluates scalability and other performance parameters, and another that evaluates the accuracy of the models using our protocol. Section 6 reviews the relevant related work. We conclude with a summary and remarks on open related problems in Section 7.

3 Differentially Private Federated Learning

Abadi et al. [2] describe a differentially private algorithm for stochastic gradient descent in the central model of differential privacy. The algorithm assumes that the training data is collected centrally by a trusted curator, and training takes place on a server controlled by the curator. For details of the algorithm the reader is referred to [2]

The primary challenge in differentially private deep learning is in bounding the sensitivity of the gradient computation. Abadi et al. [2] use the approach of computing *per-example gradients*—one for each example in the minibatch—then *clipping* each gradient to have L_2 norm bounded by the *clipping parameter* C (line 6). The summation of the clipped gradients (line 7) has global L_2 sensitivity bounded by C.

Our privacy analysis of this algorithm uses Rényi differential privacy (RDP) [32] (rather than the moments accountant) for convenience and leverages parallel composition over the minibatches in each epoch (rather than privacy amplification by subsampling). Otherwise, it is similar to that of Abadi et al. By the definition of the Gaussian mechanism for Rényi differential privacy [32], the Gaussian noise added in line 7 is sufficient to satisfy $\left(\alpha, \frac{C^2\alpha}{2\sigma^2}\right)$-RDP. By RDP's sequential composition theorem, training for E epochs satisfies

Protocol 1: FLDP Protocol

Runs on the ***untrusted server***

Input : Set of clients P, noise parameter σ, minibatch size b, learning rate η, clipping parameter C, number of epochs E.

Output : Noisy model θ.

Privacy guarantee: satisfies $\left(\alpha, \frac{EC^2\alpha}{\sigma^2}\right)$-RDP for $\alpha \geq 1$, assuming honest majority of clients in each batch

$\theta \leftarrow$ random initialization
for *E epochs* **do**
  for *each batch of clients $P_b \in P$ of size b* **do**
    $G \leftarrow \texttt{NoisyBatchGradient}(P_b, \sigma, C, \theta)$
    $\theta := \theta - \frac{1}{b}\eta G$ *update model*
return θ

$\left(\alpha, \frac{EC^2\alpha}{2\sigma^2}\right)$-RDP. Slightly tighter privacy analyses have been developed [6, 14, 20] that also apply to our work. We present the RDP analysis for simplicity, since our focus is not on improving central-model accuracy.

3.1 FLDP: Distributed DP SGD

We now extend the central-model approach to the distributed setting. The following describes a macro-level protocol for realizing differentially private distributed SGD when a trusted third party is present. Functionality 2 (`NoisyBatchGradient`) assumes the existence of a trusted third party to aggregate the noisy gradients associated with a single batch. Section 4 will describe our MPC protocol that implements Functionality 2 without a trusted third party.

Together, Protocol 1 and Functionality 2 define a differentially private distributed SGD algorithm suitable for the trusted server setting.The distributed computation follows the framework of McMahon et al. [12], in which each client computes a gradient locally (Functionality 2, line 2). To satisfy differential privacy, our adaptation clips each gradient and adds noise (lines 3-4).

Under the assumption that a trusted third party is available to compute Functionality 2, Protocol 1 satisfies differential privacy. Each execution of Functionality 2 calculates a sum of noisy gradients, each with Gaussian noise of scale $\frac{\sigma}{b}$. The final sum is:

$$\hat{G} = \sum_{i=1}^{b} \hat{g}_i = \sum_{i=1}^{b} \left(\bar{g}_i + \mathcal{N}(0, \frac{\sigma^2}{b}\mathbf{I})\right) = \left(\sum_{i=1}^{b} \bar{g}_i\right) + \mathcal{N}(0, \sigma^2\mathbf{I}), \tag{1}$$

which is exactly the same as the central model algorithm [2]. The last step of the derivation follows by the sum of Gaussian random variables. Note that the noise added by each client is *not sufficient* for a meaningful privacy guarantee (it is only $\frac{1}{b}$ of the noise required). The privacy guarantee relies on

Functionality 2: Distributed `NoisyBatchGradient`

Runs on a ***trusted third party***

Input : Batch of clients P_b of size b, noise parameter σ, clipping parameter C, current model θ.

Output : Noisy gradient $\hat{G}$.

Privacy guarantee: satisfies $\left(\alpha, \frac{C^2\alpha}{\sigma^2}\right)$-RDP for $\alpha \geq 1$, assuming honest majority of clients

Part 1: each client $p_i \in P_b$ computes a noisy gradient and sends it to the functionality $\mathcal{F}$.

1 **for** *each client $p_i \in P_b$* **do**
2   $g_i \leftarrow \nabla \mathcal{L}(\theta, \mathsf{dataOf}(p_i))$ *compute gradient*
3   $\bar{g}_i \leftarrow g_i / \max(1, \frac{\|g_i\|_2}{C})$ *clip gradient*
4   $\hat{g}_i \leftarrow \bar{g}_i + \mathcal{N}(0, \frac{\sigma^2}{b}\mathbf{I})$ *add noise*
5   p_i sends $\hat{g}_i$ to $\mathcal{F}$

Part 2: $\mathcal{F}$ computes the sum of noisy gradients and releases it to the server.

6 $\hat{G} \leftarrow \sum_{i=1}^{b} \hat{g}_i$ *sum individual gradients*
7 $\mathcal{F}$ sends $\hat{G}$ to the **untrusted server**

the noise samples being correctly summed along with the gradients. This is a major difference between Functionality 2 and approaches based on local differential privacy [10], in which *each* client adds sufficient noise for privacy.

The privacy analysis for Functionality 2 and Protocol 1 are standard, based on the conclusion of Equation (1). The L_2 sensitivity of $\left(\sum_{i=1}^{b} \bar{g}_i\right)$ is C, since at most one element of the summation may change, and it may change by at most C. By the definition of the Gaussian mechanism for Rényi differential privacy, the noisy gradient sum satisfies $\left(\alpha, \frac{C^2\alpha}{2\sigma^2}\right)$-RDP. The batches are disjoint, so over E epochs of training, each individual in the dataset incurs a total privacy loss of $\left(\alpha, \frac{EC^2\alpha}{2\sigma^2}\right)$-RDP.

3.2 Security & Privacy Risks of FLDP

Protocol 1 satisfies differential privacy when a trusted third party is available to execute Functionality 2. The server may be untrusted, since the server only receives differentially private gradients.

Malicious clients. Functionality 2 is secure against semi-honest clients (in part 1), since each client only sees their own data and the (differentially private) model θ. However, actively malicious clients may break privacy for *other* clients. Each client is required to add noise to their own gradient (line 4); malicious clients may add no noise at all.

If 50% of the clients add no noise, then the variance of the noise in the aggregated gradient $\hat{G}$ (line 6) will be $\frac{\sigma^2}{2}$ instead of σ^2, yielding $\left(\alpha, \frac{EC^2\alpha}{\sigma^2}\right)$-RDP (a weaker guarantee than given above). As the fraction of malicious clients grows, the privacy guarantee gets weaker. As discussed earlier, we assume an **honest majority of clients** and relax our privacy guarantee to this weaker form.

No trusted third party. The larger problem is with the requirement for a trusted third party to compute Part 2 of Functionality 2. Even an honest-but-curious server breaks the privacy guarantee for this part: the server receives each individual gradient separately, and each one has only a small amount of noise added. This small amount of noise is insufficient for a meaningful privacy guarantee. Section 4 describes an MPC protocol that securely implements Functionality 2 in the presence of an actively malicious server and an honest majority of clients.

Privacy analysis. The protocols we describe in Section 4 work for finite field elements, so the floating-point numbers making up noisy gradients will need to be converted to field elements. Our privacy analysis of Protocol 1 relies on a property of the sum of Gaussian random variables; as Kairouz et al. [29] describe, this property does *not* hold for discrete Gaussians. We amend the privacy analysis to address this issue in Section 4.8.

4 LWE-Based Secure Aggregation

In this Section we address the security problem described in the last Section, i.e., that state-of-the-art federated learning with differential privacy requires a trusted third-party server for aggregating gradients. Instead, we propose to use *secure aggregation* between the clients of the protocol, eliminating the need for a trusted third-party server. This allows us to keep both client inputs and gradients confidential for the calculation of a differentially private aggregate gradient. Our solution is a secure aggregation protocol that securely realizes Functionality 2 as part of Protocol 1.

Our approach is to build a LWE-based masking protocol that substantially reduces the communication complexity required to add large vectors. Rather than applying traditional secure multiparty computation (MPC) protocols to the entire vector, we generate masks that obscure the secret vectors based on the learning with errors problem. The masked vectors are safe to publish to the central server for aggregation in the clear. The sum of all vector masks can be obtained through MPC among the clients in the federation. Since the individual vector masks cannot be perfectly reconstructed from the sum of all of the masks, the security of the learning with errors problem safeguards the encryption of the masked vectors.

Due to the nature of the learning with errors problem, the individual vector masks cannot be perfectly reconstructed with the sum of all the masks. The "errors" remain in the aggregated vector sum, and are sufficient to satisfy (ε, δ)-differential privacy.

4.1 Background: Learning with Errors

To reduce the dimension of the vectors that are to be summed using MPC, we use a technique whose security relies on the difficulty of the Learning With Errors (LWE) problems [36]. These computational problems are usually posed in the following manner: Let $\mathbb{F}_q$ be the finite field of prime size q, which is sometimes denoted $GF(q)$, and fix a secret vector $s \in \mathbb{F}_q^n$. An LWE sample is a pair (a, b), where $a \in \mathbb{F}_q^n$ is chosen uniformly at random, and

$$b = a \cdot s + e \in \mathbb{F}_q,$$

where $a \cdot s$ denotes the usual dot product, and e is a so-called "error," chosen from a suitable error distribution χ on $\mathbb{F}_q$. Then the LWE (search) problem consists of retrieving the secret s given a polynomial number of LWE samples (a, b).

For our purposes we will also need the hardness of the LWE decision problem, which is the problem of distinguishing a set of pairs (a, b) with each pair chosen uniformly at random from $\mathbb{F}_q^n \times \mathbb{F}_q$ from a set of pairs that are LWE samples. In [36], Regev shows that when q is a prime of size polynomial in n and for χ any error distribution on $\mathbb{F}_q$, the LWE decision problem is at least as hard as the LWE search problem. Since the reduction from the LWE decision to the LWE search problem is trivial, in those cases the two problems are equivalent.

4.2 Background: Multiparty Computation

Secure Multiparty Computation, abbreviated MPC, refers to distributed protocols where independent data owners use cryptography to compute a shared function output without revealing their private inputs to each other or a third party [23]. In our setting, the ideal functionality computed by these clients is gradient aggregation, which as discussed in Section 3 is differentially private with regard to user inputs. Thus MPC serves to replace a trusted third party in secure function evaluation.

Security properties of Secure Aggregation protocols are categorized based on assumptions about the power of an adversary. *Semi-Honest* adversaries perform the protocol as intended, while attempting to gain information about the private inputs of the protocol. *Malicious* adversaries may exhibit arbitrary behaviors to affect the security, correctness, or fairness of an MPC protocol. Furthermore, MPC protocols must assume that some proportion of the involved clients are honest. FLDP assumes an honest majority against a malicious adversary. For a group of size k, we assume that $\frac{k}{2} + 1$ clients are honest, and make no assumption about the behavior of the rest.

FLDP requires the realization of secure vector aggregation in order to add the secret keys each participant uses to mask their larger dimension vectors. Several secure vector aggregation protocols already exist, especially for smaller sized vectors [8, 12, 41]. For the sake of consistent security and complexity analysis, we implement a secure vector aggregation protocol using Packed Shamir secret sharing [24]:

A (t,k,n) threshold secret sharing scheme will break k secret values into n shares, and require at least $t+k$ shares to recover the secret. Provided that an adversary has access to fewer than t shares, packed Shamir sharing maintains the same perfect security as traditional Shamir sharing [11]. To ensure perfect security, we choose parameter settings that ensure the adversary will never have access to t or more shares. Our secure vector aggregation protocol additionally requires that the scheme have an additive homomorphic property. That is to say if `[a]` and `[b]` are secret shares of values a and b, and c is a constant. Using `[a]`, `[b]`, and c, a party must be able to calculate `[a + b]`, `[ac]`, and `[a + c]` without communication among the other clients.

4.3 LWE-Based Masking of Input Vectors

We now describe our novel masking protocol, which allows us to reduce client communication. A high-level summary of the protocol is the following:

1. Each client generates a one-time-pad that is the same size as their gradient, masks their gradient, and sends the encrypted gradient to the server.
2. Clients add their masks together using MPC and send the aggregate mask to the server.

Through this protocol the server can recover the true sum of the gradients by adding the masked gradients and subtracting the aggregate mask. Moreover, the aggregate mask reveals nothing about any individual gradients or their masks.

We begin by assuming that all clients share a public set of m vectors chosen uniformly at random from $\mathbb{F}_q^n$, and we arrange these vectors as the rows of an $m \times n$ matrix $A \in \mathbb{F}_q^{m\times n}$. Then each client generates a secret vector $s \in \mathbb{F}_q^n$, with each entry of the vector drawn from the distribution χ, and an error vector $e \in \mathbb{F}_q^m$, with each entry of the vector also drawn from the same distribution χ, and computes the vector

$$b = As + e \in \mathbb{F}_q^m.$$

We can then think of the pair (A,b) as a set of m LWE samples, where each row of A constitutes the first entry of a sample as described in Section 4.1, and each entry of b constitutes the second entry of the sample. The hardness of the LWE decision problem tells us that the vector b is indistinguishable from a vector whose entries are chosen uniformly at random from $\mathbb{F}_q$, so b can serve as a one-time pad to encrypt the vector $v \in \mathbb{F}_q^m$:

$$h = v + b,$$

where here h is used to denote the encrypted v. Note that according to Regev [36], there is no loss in security in having all clients share the same matrix A to perform this part of the protocol.

Now suppose that h_i, v_i, b_i, s_i, and e_i are the h, v, b, s, and e vectors of client i. Additionally, suppose h_{sum}, v_{sum}, b_{sum}, s_{sum} and e_{sum} are the sum of all b_i, s_i, and e_i for clients $0,\ldots,k-1$ where k is the number of clients.

By the definition of one-time pads, each client can send h_i to the server without revealing anything about v_i. The server can obtain h_{sum} through simple vector addition. By the definition of each h_i, we further know that:

$$h_{sum} = v_{sum} + b_{sum},$$

and by the definition of each b_i and the distributive property, we obtain:

$$h_{sum} = v_{sum} + As_{sum} + e_{sum},$$

where As_{sum} denotes the usual matrix-vector multiplication. To obtain s_{sum} we assume the federation has access to a secure aggregation protocol that realizes functionality $\texttt{Sagg}(x_0,\ldots x_k,t)$. `Sagg` returns the sum of vectors $x_0,\ldots,x_k$, while not revealing any information about any inputs to any subset of parties of size smaller than t. Because they utilize `Sagg`, this reveals nothing about their individual s_i values. In the case of dropouts, `Sagg` also returns the subset of parties that participated in the aggregation. Using s_{sum}, the server can compute the following value:

$$v_{sum} + e_{sum}$$

Of course, the clients do not share their individual error vector e_i values because this would invalidate the LWE assumption that ensures b_i is a one-time pad. Therefore, we realize the ideal functionality of calculating v_{sum} by returning a noisy answer. Fortunately, each entry in e_{sum} is the sum of at most k discretized Gaussians. Therefore we can use the noise added by e_{sum} to satisfy (ε,δ)-DP.

Protocol 3 reduces the client communication complexity from $O(\log(q)mk)$ to $O(\log(q)(m+n+k))$ by requiring clients to securely aggregate only a small vector of size n. The addition of n and k can be attributed to the possible use of packed secret sharing. Each client shares their length-m vector once with the server, and then uses a packed secret sharing scheme on their length-n vector. The total number of shares required in the packed scheme is $O(n+k)$

4.4 Vector Aggregation

To add the secret vectors $s_0 \ldots s_{k-1}$, we can use any secure aggregation protocol. In our use cases, each s_i is typically of small dimension ($m \leq 800$), so we use a packed Shamir secret sharing protocol outlined in Protocol 4.

Protocol 4 is secure against semi-honest adversaries based on the security of packed secret sharing. A malicious adversary could broadcast an incorrect sum in Round 2 of the protocol, and the final result would be calculated incorrectly by the other clients. Traditionally, the `reconstruct` function has no ability to catch this kind of cheating; in many cases *all* of the shares are needed to reach the threshold during reconstruction, so corruption of a single one will change the result.

Protocol 3: Masking Aggregation

Input : Set U of k clients, each client i has a vector $v_i \in \mathbb{F}_q^m$, the secret length n, an error distribution χ, and a common matrix $A \in \mathbb{F}_q^{m \times n}$.

Output : The sum of all vectors $v_0 \dots v_{k-1}$, V

Round 1: Each client i:

1. generates a vector $s_i \in \mathbb{F}_q^n$, with each entry drawn at random from χ, using a secret seed.
2. generates $e_i \in \mathbb{F}^m$ with each entry drawn at random from χ.
3. $b_i \leftarrow As_i + e_i$
4. $h_i \leftarrow v_i + b_i$
5. sends h_i to the server.

Round 2: The server:

1. receives h_i from each non-dropped out client
2. the server sends each party the set of clients who sent an h. Call this set U_1.

Round 3: Each client i:

1. Obtains $s \leftarrow \sum_{i \in U_1} s_i$. Using `Sagg`$(\{s_i | i \in U_1\}, t)$ and U_2, the set of clients that participated in `Sagg`.
2. sends s, U_2 to server.

Round 4: The server:

1. $H \leftarrow \sum_{i \in U_2} h_i$
2. $V \leftarrow H - As$

Protocol 4: Secure Vector Addition

Input : k vectors $v_i \in \mathbb{F}_q^n$, one from each client P_i, a secret sharing threshold t, a packing threshold $p < k - t - 1$.

Output : vector sum $V \in \mathbb{F}_q^n$

Round 1: Each client j:

1. partitions v_j into a set of length-p vectors R_j
2. Generates a set of $(t - p + 1, t + 1, p, k)$-packed secret sharing called S_j with one sharing for each vector in R_j.
3. Distributes the shares of each sharing in S_j to clients $P_0 \dots P_{k-1}$. For a given sharing s in S_j, P_i receives s_{ij}.

Round 2: Each client j:

1. Receives shares $s_{j0}, \dots s_{jk-1}$ from $P_0, \dots P_{k-1}$ for all sharings in S.
2. $sum_j \leftarrow \sum s_{j0}, \dots, s_{jk-1}$ for each sharing in S.
3. Broadcasts each sum_j to every client.

Round 3: Each client j:

1. Receives $sum_0 \dots sum_{k-1}$ for each sharing in S.
2. Runs `reconstruct` on each element $sum_0 \dots sum_{k-1}$ to obtain a set of length-p vectors R_{sum}.
3. if (2) fails, broadcast `ABORT`.
4. concatenate the vectors in R_{sum} to obtain V.

4.5 Malicious-Secure Vector Aggregation

We now extend Protocol 4 to be secure against malicious clients by applying a variation of Benaloh's verifiable secret scheme [9]. The key insight behind this modification comes from the observation that in our protocol each client receives k shares from the other clients in Round 3, but only t shares are actually required for reconstruction. Our modified reconstruction procedure uses the remaining shares to catch cheating clients.

Algorithm 5: Shamir Reconstruction with Verification

Input : Let $[a]$ be a (t,k)-Shamir sharing of secret a. Assume one client has access to at least $t+1$ shares of $[a]$.

Output : a or `ABORT`

```
A ⊂ B ⊆ [a] where |A| = t and |B| = t + 1.
a' → reconstruct(A)
b → reconstruct(B)
if a' = b then
    return a'
else
    return ABORT
```

We propose the following reconstruction method for verifying that clients have behaved honestly. Requiring that each client has at least $t+1$ shares, we have each honest client take two subsets of the shares, one of size t and one of size $t+1$. The clients perform the traditional reconstruction technique on both subsets. If the values returned by both reconstructions are equivalent, they accept the result as correct. Otherwise, they abort. The modified reconstruction procedure appears in Algorithm 5. Replacing the call to `reconstruct` in Protocol 4 with a call to this modified reconstruction procedure yields a malicious-secure protocol.

Note that Algorithm 5 does not require communication with other clients. General-purpose malicious-secure protocols based on the same principle require interaction between the clients to check for cheating (e.g., the protocol of Chida et al. [17]) because they use the "extra" shares to perform multiplication. Since our application does not require multiplication, we can use these shares to catch cheating instead.

Algorithm 5 can be extended to the packed Shamir variant by requiring that each client has access to $t+k+1$ shares. The number of shares to which access is required must be increased because the reconstruction threshold is increased in the packed variant. Protocol 4 and Algorithm 5 realize the ideal functionality `Sagg` in the malicious adversary threat model.

4.6 Security Analysis

Here we analyze the security of Protocol 3, which we will denote as π.

Suppose the ideal functionality of noisy vector addition as F, an adversary A. Let v_i and x_i be input and view of client i respectively. Let x_s be the view of the server. n is the LWE security parameter. Suppose a maliciously secure aggregation protocol $\texttt{Sagg}(X,t)$. Let V be the output of π.

Let U be the set of clients, and $C \subset U \cup \{S\}$ be the set of corrupt parties.

In the malicious model, we consider dropping out an adversarial behavior without loss of generality.

Suppose the simulator has access to an oracle $\texttt{IDEAL}(t, v_u)_{u \in U \setminus C}$ where:

$$\texttt{IDEAL}(t, v_u)_{u \in U \setminus C} = \begin{cases} \Sigma_{u \in U \setminus C} v_u & |U \setminus C| > t \\ \perp & otherwise \end{cases}$$

Let $\texttt{REAL}^{U}_{\pi,C} = \{x_i | i \in C\}, V$.

Theorem 1 *There exists a PPT simulator* $\mathtt{SIM}$ *such that for all t, U, C*

$$REAL^{U}_{\pi,C}(n,t;v_{U \setminus C}) \equiv SIM_C^{U,IDEAL(t,v_u)}(n,t;x_C)$$

The proof full proof of this theorem can be found in Appendix A.

4.6.1 LWE parameters

The security of an LWE instance is parameterized by the tuple (n, q, β) where n is the width of the matrix A (or equivalently the dimension of the secret s), q is the field size, and β is such that βq is the width of the error distribution χ (so that the standard deviation is $\sigma = \frac{\beta q}{\sqrt{2\pi}}$; this quantity is denoted α in the LWE literature, but we choose β here so as to not conflict with the notation for Rényi divergence). We used the LWE estimator [4] to calculate the security of each parameter tuple. Table 2 displays a series of LWE parameters for different potential aggregation scenarios, each with at least 128 bits of security.

The different parameter settings are driven by different sizes of q, which would enable more precision in the aggregate values. A larger field size also allows more clients to be involved in the aggregation. Field sizes picked here may also utilize fast Fourier transform secret sharing. For this reason we consider q fixed by the application of the protocol. Since we also use a fixed valued of $\beta = \frac{3.2}{q}$, the security offered by the LWE problem depends on the variable n (the length of the secret s), which we call the *security parameter*.

4.7 Encoding and Decoding Gradients

In order to manipulate gradients with MPC, we require that they can be encoded as a vector of finite field elements. First we flatten the tensors that compose each gradient into a vector of floating point numbers. The aggregation operation of gradients is element wise. Therefore, we simplify the encoding problem to encoding a floating point number as a finite field element. Gradient elements are clipped, and encoded as fixed point numbers. We chose 16 bit numbers with 4 digits of precision after the decimal. This precision was sufficient for model conversion on the MNIST and CFAR-10 problems.

The integers are converted to unsigned integers using an offset, and the unsigned integer result can be encoded into any field larger than 2^{16}. The fields used in our experiment are outlined in Table 2.

Protocol 6: Malicious-Secure `NoisyBatchGradient`

Input : Batch of clients P_k of size k, noise parameter σ, clipping parameter C, current model θ.

Output : Noisy gradient $\hat{G}$.

Privacy guarantee: satisfies $\left(\alpha, \frac{C^2\alpha}{\sigma^2}\right)$-RDP for $\alpha \geq 1$, assuming honest majority of clients

for *each client* $p_i \in P_k$ **do**
- $g_i \leftarrow \nabla \mathcal{L}(\theta, \mathsf{dataOf}(p_i))$ — *compute gradient*
- $\bar{g}_i \leftarrow g_i / \max(1, \frac{\|g_i\|_2}{C})$ — *clip gradient*
- $\hat{g}_i \leftarrow \bar{g}_i + \mathcal{N}(0, \frac{\sigma^2}{k}\mathbf{I})$ — *add noise*
- $v_i \leftarrow \texttt{EncodeGradient}(\hat{g}_i)$ — *encode gradient*

Client $p_i \in P_k$ provides v_i as input to Secure Vector Addition (Protocol 4). Together, the clients compute $\hat{G} = \sum_{i=1}^{k} \hat{g}_i$. $\hat{G}$ is released to the **untrusted server**.

4.8 Malicious Secure FLDP

We now have all the MPC operations necessary to implement our ideal functionality from Protocol 1 as a secure multiparty computation. Protocol 6 securely implements Functionality 2, and can replace it directly to implement FLDP. This version of `NoisyBatchGradient` computes the gradient and adds noise to it in the same way as the ideal functionality, but invokes Protocol 4 to sum the vectors. This requires encoding each noisy gradient as a vector of field elements, as described in the last section.

Privacy analysis. The privacy analysis of Protocol 1 relies on the fact that the sum of Gaussian random variables is itself a Gaussian random variable. However, as Kairouz et al. [29] point out, this property does not hold for *discrete* Gaussians—and since `EncodeGradient` uses a fixed-point representation for noisy gradients, we cannot rely on the summation property. Instead, our privacy analysis proceeds based on Proposition 14 of Kairouz et al. [29]:

Proposition 1 (from Kairouz et al. [29]) *Let* $\sigma \geq \frac{1}{2}$. *Let* $X_{i,j} \leftarrow \mathcal{N}_{\mathbb{Z}}(0, \sigma^2)$ *independently for each i and j. Let* $X_i = (X_{i,1}, \ldots, X_{i,d}) \in \mathbb{Z}^d$. *Let* $Z_n = \sum_{i=1}^{n} X_i \in \mathbb{Z}^d$. *Then, for all*

$\Delta \in \mathbb{Z}^d$ *and all* $\alpha \in [1,\infty)$*,*

$$D_\alpha(Z_n||Z_n+\Delta) \leq \frac{\alpha||\Delta||_2^2}{2n\sigma^2} + \tau d$$

where $\tau := 10 \cdot \sum_{k=1}^{n-1} e^{-2\pi^2\sigma^2 \frac{k}{k+1}}$*.*

Proposition 1 provides a bound on Rényi divergence, D_α, for noise generated as the sum of discrete Gaussians, which directly implies Rényi differential privacy. In our setting, Proposition 1 yields almost identical results to the privacy analysis of Protocol 1 (which assumes continuous Gaussians). Note that the first term of the bound from Proposition 1 is identical to the bound given in our earlier privacy analysis, when n is equal to the batch size b and $||\Delta||_2^2$ is equal to C^2 (where C is the L_2 clipping parameter).

As the fixed-point representation of noisy gradients becomes more precise, the second term of the bound (τd) becomes extremely small. The `EncodeGradient` function uses 4 places of precision past the decimal point, meaning that the effective values of σ^2 and $||\Delta||_2^2$ are 10,000 times their "original" values. Each additional place of precision adds another factor of 10 to both values. This has the effect of reducing the value of τ to extremely close to zero.

We have implemented both the original analysis (which incorrectly assumes continuous Gaussians) and Proposition 1. The results reported in Section 5 use Proposition 1, but the two methods yield values of ε so close together that the resulting graphs are indistinguishable.

4.9 Algorithmic Complexity

Client computation is comprised of three tasks. Generating a random vector s of length n, generating a random vector e of length m, multiplying s by $m \times n$ matrix A, and generating secret shares for s. Random vector generation is an $O(m+n)$ operation where n is length of secret vector s and m is the length of e. This can be reduced to $O(m)$ because m will be larger than n in any practical use of FLDP. Matrix multiplication by a vector is an $O(mn)$ operation where m is the vector size; each matrix element is considered exactly once. Finally, secret share generation is done using the packed FFT method [19], and therefore has a complexity of $O(k\log(k))$ where k is the number of clients. In sum, this gives us a runtime of $O(mn+k\log(k))$ for client computation with respect to our vector size m and s length n.

In order to assume the difficulty of the LWE decision problem, we require that q be polynomial in n. Though the field size does affect the precision of the values to be aggregated and the possible number of parties to the aggregation scheme, it is customary to think of q as a constant, and therefore n is constant here too in our complexity analysis. However, in practice it is possible to choose n quite small relative to q.

Server complexity consists of adding k masked vectors, reconstructing the packed secret sharing, and multiplying an $m \times n$ matrix by a length n vector. The vector addition and matrix multiplication have complexity $O(mk+m\log(k))$. Reconstructing the packed secret shares takes time $O(k\log(k))$ in the semi-honest case with no dropouts using the Fast Fourier Transform method. In the case of malicious security and dropouts, we use Lagrange interpolation to obtain a runtime of $O(k^2)$. The number of dropouts does not affect runtime complexity as long as there are more than 0 dropouts. In total, the server runtime complexity is $O(mk+mn+k\log(k))$ in the no dropout scenario, and $O(mk+mn+k^2)$ in case of dropouts or malicious adversaries.

5 Evaluation

Our empirical evaluation aims to answer two research questions:

- **RQ1**: How does the concrete performance of FLDP compare to state-of-the-art secure aggregation?
- **RQ2**: Is FLDP capable of training accurate models?

We conduct two experiments to answer both questions in the affirmative. We first describe our experiment setup and the datasets used in our evaluation. Section 5.1 describes our scalability experiment; the results show that FLDP scales to realistic batch sizes, and that model updates take only seconds. Section 5.2 describes our accuracy experiment; the results demonstrate that FLDP trains models with comparable accuracy to central-model differentially private training algorithms.

Experiment setup. Our experiments take place in two phases. First, the model is trained in a single process with privacy preserving noise added to each gradient. As model training occurs, gradients for each training sample are written to file. The second phase involves the MPC simulation. Clients read their noisy gradients from file, and aggregate them using FLDP.

This experimental setup is necessary for the implementation of local experiments with batch sizes of 128. Reading each gradient from file sidesteps the need for each client to have their own TensorFlow instance, substantially reducing our memory consumption footprint.

Running these two separate experiments ensures that the MPC results reflect the performance of FLDP without considering the overhead of training 128 separate neural networks in parallel.

The memory consumption issue described here is created by simulating many clients on the same machine. In a true federated learning instance, each client would have their own independent resources, and therefore would not run into this same issue.

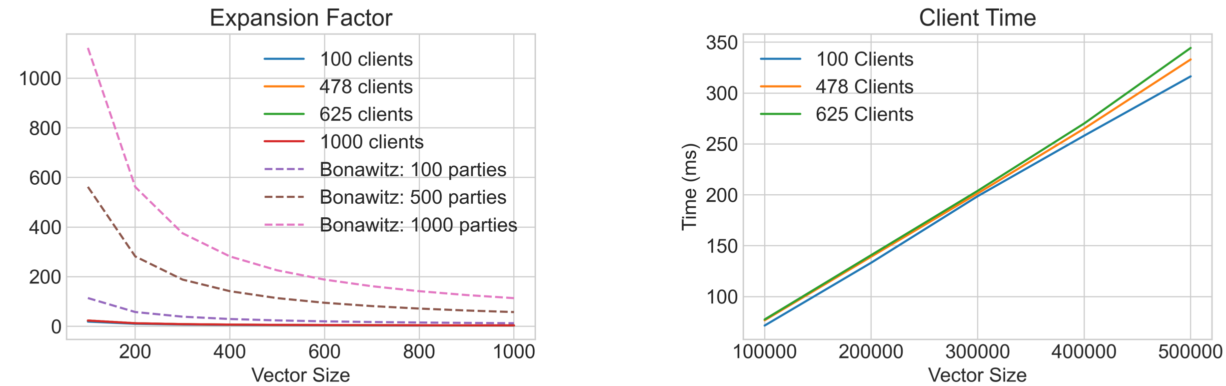

Figure 1: The left figure displays the expansion factor for using our protocol with various vector sizes and numbers of clients, comparing our approach (solid lines) against the secure aggregation protocol of Bonawitz et al. [12] (dashed lines). The right figure includes cost of a single client's computation. The client timing results are identical regardless of the dropout so only the dropout situation is plotted.

5.1 Experiment 1: Masking Scalability

This section strives to answer **RQ1**. We implemented the masking protocol in single threaded python and evaluated various federation configurations. Experiments were run on an AWS z1d2xlarge instance with a 4.0Ghz Intel Xeon processor and 64 Gb of RAM [1]. Concrete timing and expansion results for protocol computation are included in Figures 1, 2, and Table 2. We assume semi-honest behavior from the adversary and consider the scenario with no dropouts as well as a 25% dropout rate. In all experiments, β is assumed to be $3.2/q$. We assume a single aggregation server, and we assume that clients broadcast the sum of shares to the server rather than performing Shamir reconstruction themselves.

5.1.1 Experimental Performance

Figures 1 and 2 presents our concrete performance results. We see a significant improvement in client and server computation time over the concrete performance results of Bonawitz et al. [12]. Client computation takes less than half a second for all configurations tested, and is dictated by a linear relationship with the vector size.

Server computation time has a linear relationship with vector size and a quadratic relationship with the number of clients. In the case with no dropouts, server computation is quick, taking less than 5 seconds for all configurations tested. In the dropout scenario, server computation is significantly slower, but still much faster than the state of the art [12].

Server time is dominated by the Lagrange interpolation process used for reconstruction. The no dropout case has no lagrange interpolation, the dropout model uses lagrange interpolation once, and the malicious protocol performs two rounds of lagrange interpolation. Figure 2 shows that the malicious protocol requires about twice as much time as the dropout friendly protocol. This is because of lagrange interpolation. Performance can be improved with faster interpolation algorithms [51].

Recall the quantity β from Section 4.6.1, which given q the size of the field gives us the standard deviation of the noise. We observe that changing β has no effect on the runtime. We note that changing β can require different values for n and q to guarantee a certain amount of security, but this is only necessary if β is decreased. For our timing experiments we chose $\beta = 3.2/q$ to accommodate a wide variety of privacy budgets for relatively small fixed precision. Because our values are fixed precision with 4 decimal places, the chosen value of β adds noise with standard deviation .0409 to our aggregated vectors assuming 128 clients. This is far less than the minimum amount of DP-noise we added in our accuracy experiments, which had a standard deviation of 1.

5.2 Experiment 2: Model Accuracy

This section strives to answer **RQ2**. We implement our models in TensorFlow. To preserve privacy, we add noise scaled to a constant σ to each example's gradient, which results in the batch gradient described by Equation 1. Each gradient is clipped, by a constant $C = 5$ such that batch gradient sensitivity is bounded by $C/batch\_size$. These two modifications to a traditional neural network training loop ensure that our models satisfy differential privacy. Adding noise in this way also accurately reflects the process that would be used by a federation member using FLDP. Gradient updates for individual samples are saved during training for use during the MPC experiments.

We evaluate the accuracy and scalability of FLDP with the standard MNIST and CIFAR-10 datasets. Both datasets, and the models we train with them are listed in Table 3.

For both the MNIST and CIFAR-10 models, we utilize cate-

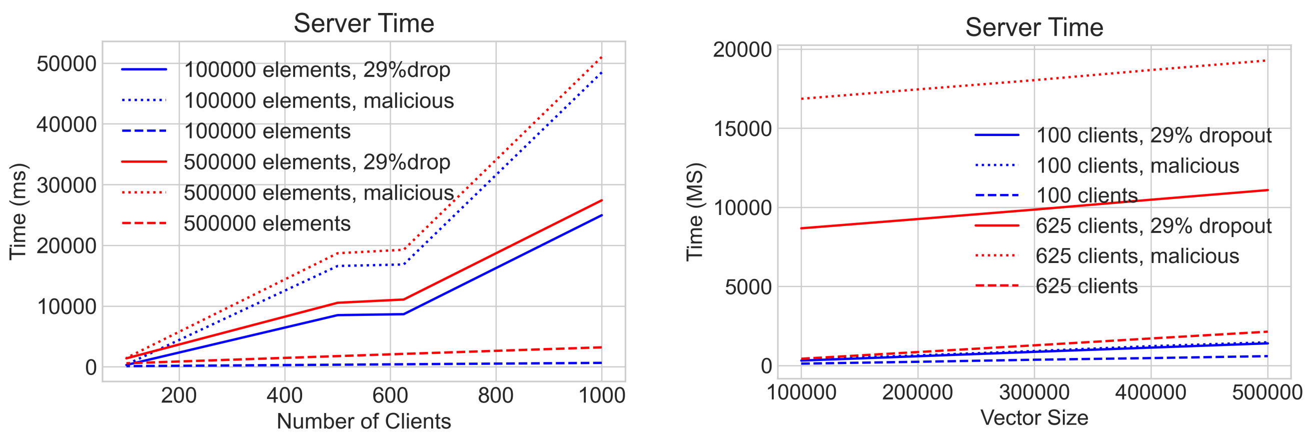

Figure 2: The effects of different federation size and different vector size on server computation time malicious, semi-honest, and dropout tolerant threat models. The malicious threat model tolerates dropouts.

Clients	q	n	% Dropout	Server	Client	Bonawitz Server	Bonawitz Client
478	31352833	710	0	310	80	2018	849
625	41057281	730	0	447	84	2018	849
1000	71663617	750	0	668	88	4887	1699
478	31352833	710	29	496	91	143389	849
625	41057281	730	29	375	93	143389	849
1000	71663617	750	29	21931	99	413767	1699

Table 2: Client and server times for various LWE configurations. Vector size is fixed at 100,000 and $\beta q = 3.2$. Times are in milliseconds. Results from Bonawitz et al. [12] for 500 and 1000 parties with 0 and 30% dropout are included for comparison.

Property	*MNIST*	*CIFAR-10*
Train Set Size	60,000	50,000
Test Set Size	10,000	10,000
# Conv layers	2	6
# Parameters	26,000	550,000
Batch Sizes	16, 32, 64, 128	16, 32, 64, 128
σ	0, 1, 2, 4, 8	0, 1, 2, 4, 8, 16

Table 3: Datasets and model configurations used in our experiments

gorical cross entropy for our loss function, stochastic gradient descent with a learning rate of 0.01 and momentum of 0.9 for our optimizer and a clipping parameter $C = 5$ for all trials.

We run a series of trials for each dataset with each pair of batch size and σ listed in Table 3. All accuracy results are the per epoch average of 4 trials with the given model configuration. ε is calculated post hoc as a function of $\sigma, C, batch\_size, epochs$. All ε values are calculated from the corresponding Rényi differential privacy guarantee by picking α to minimize the RDP ε parameter, then converting this guarantee into (ε, δ)-differential privacy with $\delta = 10^{-5}$. We see selected accuracy results reported for differing values of ε in Figure 3.

5.2.1 MNIST

The Modified National Institute of Standards and Technology database is an often used image recognition benchmark consisting of 60,000 training samples and 10,000 testing samples; each sample is a 28×28 gray scale image of a handwritten digit. We train a classifier containing 2 ReLU-activated convolution layers, max pooling following each of them, and a ReLU activated dense layer with 32 nodes. Finally, classifications are done with a softmax layer. This model has about 26,000 trainable parameters in total.

After training for 275 epochs, our private MNIST models are able to attain a maximum 98.7% mean validation accuracy over 4 trials. This is a slight decrease in accuracy from the no noise baseline accuracy of 99.2%, however the private model still generalizes very well. Figure 4 shows how different privacy budgets affect accuracy for our sample batch sizes. Models trained with all batch sizes see improved accuracy as ε increases, however larger batch sizes tend to produce more accurate models, especially for small values of ε. Improved accuracy for larger batch sizes can be seen as an effect of the private average, where the sensitivity of the gradient average is inversely proportional to the batch size. Therefore, larger batches require less noise added for a given privacy budget, resulting in a more accurate model.

5.2.2 CIFAR-10

The Canadian Institute for Advanced Research 10 dataset consists of 60,000 colored images equally partitioned into 10 classes. Each image is 32×32 with 3 channel RGB colored pixels. We separated the dataset into 50,000 training examples and 10,000 test samples for our experiment. Our trained model contains three pairs of ReLU-activated convolution layers with batch normalization after each layer, and max pooling after each pair. We also include one ReLU activated dense layer with 128 nodes, and a softmax activated output layer. This model contains 550,000 parameters.

With a batch size of 64, we achieve a maximum accuracy of 70.0% mean validation accuracy over 4 trials on CIFAR-10. This is a sizeable drop in accuracy compared to the 77.4% mean accuracy of our architecture trained without differential privacy, however it is in line with differentially private model performance in the central model [2].

Figure 4 demonstrates the correlation between larger batch size and greater accuracy when controlling for a specific privacy budget. As with MNIST, the greater accuracy with larger batch sizes likely stems from gradient sensitivity being dependent on the batch size itself. That said, for $\varepsilon < 10$, we achieve our most accurate model with a batch size of 64 ($\varepsilon = 3.67$), which is well within the scalable limits of FLDP as defined in Section 5.1.

5.2.3 Comparison With Centralized Differential Privacy

Our approach produces models with accuracy highly comparable to those achieved by Abadi et. al. [2]. Table 4 shows that for a given privacy budget, our approach is able to produce an output within 3% of the equivalent central-model accuracy. It is worth noting that we report the average of 4 trials in this table, and that we observe the same, or better, decrease in accuracy with respect to the no-noise baseline for each model.

Method	*Abadi et. al. [2]*	FLDP
MNIST ($\varepsilon \leq 2$)	95%	95%
MNIST ($\varepsilon \leq 8$)	97%	99%
CIFAR-10 ($\varepsilon \leq 4$)	70%	70%
CIFAR-10 ($\varepsilon \leq 8$)	73%	70%

Table 4: A comparison of our private model accuracy with a central-model differentially private training algorithm. For all models, $\delta = 10^{-5}$. For all of our models, batch size is 64.

These comparable accuracy results demonstrate the usability of FLDP for privacy preserving federated learning.

6 Related Work

Secure multiparty computation. Secure multiparty computation (MPC) [23] is a family of techniques that enable mutually distrustful parties to collaboratively compute a function of their distributed inputs without revealing those inputs. MPC techniques include *garbled circuits* [49] (which is most easily applied in the two-party case) and approaches based on *secret sharing* [41] (which naturally apply in the *n*-party case). MPC approaches have seen rapid improvement over the past 20 years, but scalability remains a challenge for practical deployments. In particular, most MPC protocols work best when the number of parties is small (e.g., 2 or 3), and costs grow at least quadratically with the number of parties. State-of-the-art protocols support significantly more parties: Wang et al. [47] reach 128 parties using a garbled circuits approach, and Chida et al. [17] reach 110 parties using a secret sharing approach.

MPC for differentially private deep learning. MPC techniques have been previously applied to the problem of differentially private deep learning, but these approaches require either a semi-honest data curator [45] or two non-colluding data curators [27]. Secure aggregation protocols [8, 12] (detailed in Section 2) are themselves MPC protocols, specifically designed for the many-client setting. Kairouz et al. [29] present a general framework for differentially private federated learning that leverages existing secure aggregation protocols.

Security for distributed differential privacy. Outside of deep learning, several systems have been proposed for computing differentially private results from distributed data. Honeycrisp [37] and Orchard [38] are most related to our work, and use a distributed protocol similar to secure aggregation to compute the results of database-style queries. ShrinkWrap [7] and CRYPTε [39] leverage existing MPC frameworks to implement differentially private database queries.

Secure and Private Federated Learning. MPC and Differential privacy may also be applied to the problem of Federated Learning separately. Several local differential privacy approaches have been proposed [46, 48]. These approaches

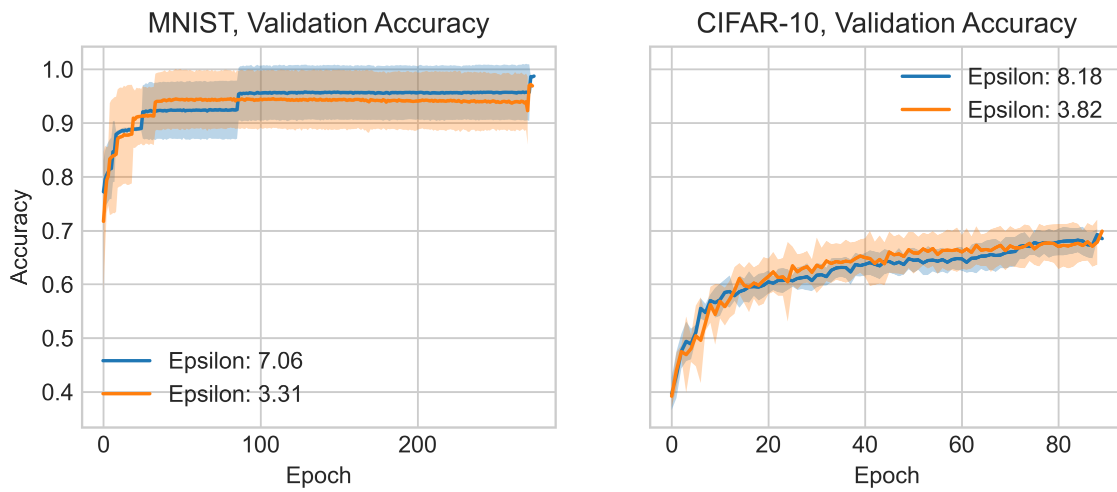

Figure 3: Validation accuracy progression over training runs on MNIST and CIFAR for various values of ε ($\delta = 10^{-5}$). All accuracy values are the average of 4 trials. Batch size is restricted to 64.

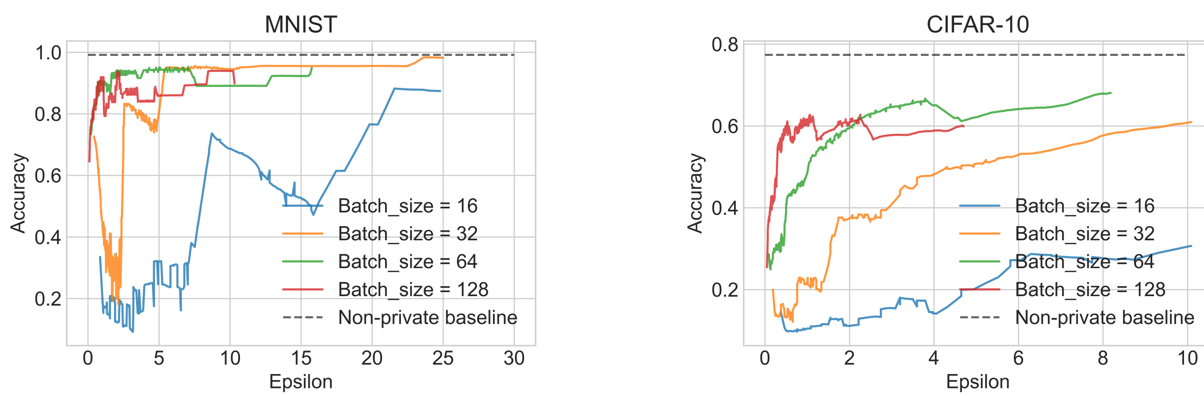

Figure 4: The effects of privacy budget and batch size on validation accuracy ($\delta = 10^{-5}$). Each solid line is a moving average of Accuracy as epsilon increases for a given batch size. The dotted line is the maximum accuracy achieved by our model with no noise added during training. Private federated learning is able to approach non-private accuracy for several batch sizes on MNIST. On CIFAR-10 e see that private models tend to be more accurate with larger batch sizes, while the opposite is true for non-private models.

add sufficient noise to a client's gradient before performing aggregation in the clear, thus circumventing the need for MPC. In this way LDP-based approaches scale to large federations quite well, but typically require very large privacy budgets for model convergence.

MPC techniques can be applied to federated learning to satisfy a variety of properties. POSEIDON [40] and So et. al [43] use MPC for substantially more robust threat models, $N-1$ adversaries and Byzantine resistance respectively, but do not utilize differential privacy, and focus on smaller federations (< 50 parties) than FLDP. EIFFeL [18] on the other hand, employs non-interactive zero-knowledge proofs to ensure the integrity of masked inputs. These approaches can prevent some additional attacks we do not consider (e.g. data poisoning), but are designed for smaller federations.

Learning with Errors. As noted in Sections 4.6.1 and 5.1.1, in this work we fix $\beta q = 3.2$. We note that the security reductions that ensure that the LWE search problem is difficult do not apply in this case: In [36], Regev shows that if q is chosen to be polynomial in n, and χ is a certain discretization of a Gaussian distribution on $\mathbb{F}_q$ with standard deviation $\frac{\beta q}{\sqrt{2\pi}}$ for $0 < \beta < 1$ and $\beta q > 2\sqrt{n}$, then solving the LWE search problem can be quantumly reduced to an algorithm that approximately solves the Shortest Vector Problem and the Shortest Independent Vectors problem. In [35], Peikert shows a classical reduction to the (slightly easier) GapSVP problem.

While as far as we know there are no security reductions for small fixed βq, at the same time we do not currently know of an attack that takes advantage of a small constant standard deviation. Accordingly, our choice is similar to the choice made in the current FrodoKEM algorithm specifications (submission to Round 3 of the NIST PQC challenge) [5, 13] and consistent with the recommendation of [3].

7 Conclusion

In the past decade, an explosion in data collection has led to huge strides forward in machine learning, but the use of sensitive personal data in machine learning also represents a serious privacy concern. We present an approach based on a new protocol called FLDP that ensures differential privacy for the trained model, *without* the need for a trusted data aggregator. Using FLDP allows a highly accurate model to be trained in a federated (distributed) manner while guaranteeing the privacy of data owners, even against powerful and colluding adversaries. Our empirical results show that these accurate models are trainable within a feasible time frame for practical applications, especially when accuracy and low trust burdens are critical.

The promising results presented in our evaluation also suggest directions for future research. For example, gradient compression techniques can substantially reduce in-communication overhead for distributed training [31]. Paired with FLDP, these techniques could further reduce the time per batch for larger models, and potentially improve our scalability with respect to model complexity. Moreover, we apply FLDP to the very specific case of privacy preserving federated learning, but additional research could consider how these techniques scale with simpler, yet important, data problems. For example, the core noise addition and secure aggregation methods described in this paper could be adapted to privacy-preserving database queries, while eliminating the need for a central database.

Acknowledgments

The authors would like to thank Connor Wagaman, Ammie Wang, Aleksei Bingham, Christopher Murphy, and Minseok Kwon for their contributions in the preliminary stages of this work. Approved for Public Release. Distribution is Unlimited. This material is based upon work supported by the Broad Agency Announcement Program and the Cold Regions Research and Engineering Laboratory (ERDC-CRREL) under Contract No. W913E521C0003, and by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00112090106. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA.

References

[1] Amazon EC2 z1d instances, 2021.

[2] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 308–318. ACM, 2016.

[3] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic encryption security standard. Technical report, HomomorphicEncryption.org, Toronto, Canada, November 2018.

[4] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, 9(3):169–203, 2015.

[5] Erdem Alkim, Joppe W. Bos, Léo Ducas, Patrick Longa, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM: Learning With Errors Key Encapsulation. Technical report, NIST PQC challenge, third round, June 2021. `https://frodokem.org/files/FrodoKEM-specification-20210604.pdf`.

[6] Shahab Asoodeh, Jiachun Liao, Flavio P Calmon, Oliver Kosut, and Lalitha Sankar. A better bound gives a hundred rounds: Enhanced privacy guarantees via f-divergences. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 920–925. IEEE, 2020.

[7] Johes Bater, Xi He, William Ehrich, Ashwin Machanavajjhala, and Jennie Rogers. Shrinkwrap: efficient sql query processing in differentially private data federations. *Proceedings of the VLDB Endowment*, 12(3), 2018.

[8] James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1253–1269, 2020.

[9] Josh Cohen Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret (extended abstract). In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 251–260, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[10] Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. Protection against reconstruction and its applications in private federated learning. *arXiv preprint arXiv:1812.00984*, 2018.

[11] G R Blakley and Catherine Meadows. Security of ramp schemes. In *Proceedings of CRYPTO 84 on Advances in Cryptology*, page 242–268, Berlin, Heidelberg, 1985. Springer-Verlag.

[12] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery.

[13] Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1006–1018, 2016.

[14] Zhiqi Bu, Jinshuo Dong, Qi Long, and Weijie J Su. Deep learning with gaussian differential privacy. *Harvard data science review*, 2020(23), 2020.

[15] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284, 2019.

[16] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 375–403. Springer, 2019.

[17] Koji Chida, Daniel Genkin, Koki Hamada, Dai Ikarashi, Ryo Kikuchi, Yehuda Lindell, and Ariel Nof. Fast large-scale honest-majority mpc for malicious adversaries. In *Annual International Cryptology Conference*, pages 34–64. Springer, 2018.

[18] Amrita Roy Chowdhury, Chuan Guo, Somesh Jha, and Laurens van der Maaten. Eiffel: Ensuring integrity for federated learning. *CoRR*, abs/2112.12727, 2021.

[19] Morten Dahl. Secret sharing, part 2 efficient sharing with the fast fourier transform, Jun 2017.

[20] Jinshuo Dong, Aaron Roth, and Weijie J Su. Gaussian differential privacy. *arXiv preprint arXiv:1905.02383*, 2019.

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

[22] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2468–2479. SIAM, 2019.

[23] David Evans, Vladimir Kolesnikov, and Mike Rosulek. A pragmatic introduction to secure multi-party computation. *Foundations and Trends® in Privacy and Security*, 2(2-3), 2017.

[24] Matthew Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '92, page 699–710, New York, NY, USA, 1992. Association for Computing Machinery.

[25] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[26] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: Information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 603–618, New York, NY, USA, 2017. Association for Computing Machinery.

[27] Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. Distributed learning without distress: Privacy-preserving empirical risk minimization. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6346–6357, 2018.

[28] Bargav Jayaraman, Lingxiao Wang, Katherine Knipmeyer, Quanquan Gu, and David Evans. Revisiting membership inference under realistic assumptions, 2020.

[29] Peter Kairouz, Ziyu Liu, and Thomas Steinke. The distributed discrete gaussian mechanism for federated learning with secure aggregation. *arXiv preprint arXiv:2102.06387*, 2021.

[30] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[31] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *CoRR*, abs/1712.01887, 2017.

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

[33] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. Computational differential privacy. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, pages 126–142, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[34] Mohammad Naseri, Jamie Hayes, and Emiliano De Cristofaro. Toward robustness and privacy in federated learning: Experimenting with local and central differential privacy. *ArXiv*, abs/2009.03561, 2020.

[35] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *STOC'09—Proceedings of the 2009 ACM International Symposium on Theory of Computing*, pages 333–342. ACM, New York, 2009.

[36] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM, New York, 2005.

[37] Edo Roth, Daniel Noble, Brett Hemenway Falk, and Andreas Haeberlen. Honeycrisp: large-scale differentially private aggregation without a trusted core. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pages 196–210, 2019.

[38] Edo Roth, Hengchu Zhang, Andreas Haeberlen, and Benjamin C Pierce. Orchard: Differentially private analytics at scale. In *14th USENIX Symposium on Operating Systems Design and Implementation ({OSDI} 20)*, pages 1065–1081, 2020.

[39] Amrita Roy Chowdhury, Chenghong Wang, Xi He, Ashwin Machanavajjhala, and Somesh Jha. Cryptε: Crypto-assisted differential privacy on untrusted servers. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 603–619, 2020.

[40] Sinem Sav, Apostolos Pyrgelis, Juan Ramón Troncoso-Pastoriza, David Froelicher, Jean-Philippe Bossuat, Joao Sa Sousa, and Jean-Pierre Hubaux. Poseidon: Privacy-preserving federated neural network learning. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. The Internet Society, 2021.

[41] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[42] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[43] Jinhyun So, Başak Güler, and A. Salman Avestimehr. Byzantine-resilient secure federated learning. *IEEE Journal on Selected Areas in Communications*, 39(7):2168–2181, 2021.

[44] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. *CoRR*, abs/2007.08432, 2020.

[45] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 1–11, 2019.

[46] Stacey Truex, Ling Liu, Ka Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. Ldp-fed: Federated learning with local differential privacy. *CoRR*, abs/2006.03637, 2020.

[47] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Global-scale secure multiparty computation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 39–56, 2017.

[48] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[49] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167. IEEE, 1986.

[50] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282. IEEE, 2018.

[51] Gathen Joachim von zur and Gerhard Jürgen. *Modern Computer algebra*. Cambridge University Press, 2013.

A Proof of security

Suppose the ideal functionality of noisy vector addition as F, an adversary A. Let v_i and x_i be input and view of client i respectively. Let x_s be the view of the server. n is the LWE security parameter. Suppose a maliciously secure aggregation protocol $\texttt{Sagg}(X,t)$. Let V be the output of π.

Let U be the set of clients, and $C \subset U \cup \{S\}$ be the set of corrupt parties.

In the malicious model, we consider dropping out an adversarial behavior without loss of generality.

Suppose the simulator has access to an oracle $\texttt{IDEAL}(t,v_u)_{u\in U\setminus C}$ where:

$$\texttt{IDEAL}(t,v_u)_{u\in U\setminus C} = \begin{cases} \Sigma_{u\in U\setminus C} v_u & |U\setminus C| > t \\ \perp & otherwise \end{cases}$$

Let $\texttt{REAL}^U_{\pi,C} = \{x_i | i \in C\}, V$.

Theorem 2 *There exists a PPT simulator* $\texttt{SIM}$ *such that for all t, U, C*

$$REAL^U_{\pi,C}(n,t;v_{U\setminus C}) \equiv SIM_C^{U,\texttt{IDEAL}(t,v_u)}(n,t;x_C)$$

Proven through the hybrid argument.

1. This hybrid is a random variable distributed exactly like $\texttt{REAL}^U_{\text{FLDP},C}(n,t;v_{U\setminus C})$
2. In this hybrid $\texttt{SIM}$ has access to $\{x_i|i\in U\}$. $\texttt{SIM}$ runs the full protocol and outputs a view of the adversary from the previous hybrid.
3. In this hybrid, $\texttt{SIM}$ has corrupt parties receive an $\texttt{ABORT}$ if the server sends a U_1 such that $t > |U_1|$.
4. In this hybrid, $\texttt{SIM}$ replaces V with the output of F from any x_C.
5. In this hybrid, $\texttt{SIM}$ generates the ideal inputs of the corrupt parties using the $\texttt{IDEAL}$ oracle, $\texttt{SIM}$ generates a set of random inputs V_C such that $\Sigma_{i\in C} v_i = F(v_U) - \texttt{IDEAL}(t,v_u)_{u\in U\setminus C}$. The output domain of FLDP is any vector $V \in \mathbb{F}_q^m$ and $ABORT$. $\texttt{SIM}$ can replicate any vector output using this process. Therefore, this hybrid is indistinguishable from the previous hybrid.
6. In this hybrid, $\texttt{SIM}$ replaces s, the sum of secret vectors with a vector of random field elements distributed by $\chi * k$. Because s is not used to reconstruct G, and is normally distributed by $\chi * k$, this hybrid is indistinguishable from the previous hybrid.
7. In this hybrid, $\texttt{SIM}$ replaces H with $V + As$.
8. In this hybrid, $\texttt{SIM}$ replaces the run of protocol $\texttt{Sagg}$ with the ideal simulation of $\texttt{Sagg}$. If $\texttt{Sagg}$ returns $\texttt{ABORT}$, $\texttt{SIM}$ returns $\texttt{ABORT}$. Because $\texttt{Sagg}$ is secure, this hybrid is indistinguishable from the previous hybrid using each parties s_i as input.
9. In this hybrid, $\texttt{SIM}$ replaces the s_i of each client with a vector of elements distributed by χ. Because s_i is typically distributed by χ and each s_i is not used to compute s anymore, this hybrid is indistinguishable from the previous hybrid.
10. In this hybrid, $\texttt{SIM}$ replaces the b_i of each client with a vector of uniformly distributed field elements in $\mathbb{F}_q^m$. Given the LWE assumption, b_i should be indistinguishable from random field elements, so this hybrid is indistinguishable from the previous hybrid from the perspective of the adversary.
11. In this hybrid, $\texttt{SIM}$ replaces h_i of each client with a vector of uniformly distributed field elements in $\mathbb{F}_q$. By the definition of one time pad, this hybrid should be indistinguishable from the previous hybrid. Additionally this hybrid does not use any input from the honest parties and thus concludes the proof.

After these steps, the simulator no longer needs any input from the honest clients to simulate Protocol 3, implying that it is secure in the malicious threat model.

Notably, our malicious threat model subsumes the semi-honest threat model. Therefore this proof proves security in that threat model as well. In the case of a semi-honest threat model, the security of $\texttt{Sagg}$ can also eased to semi-honest.