DMM: Distributed Matrix Mechanism for Differentially-Private Federated Learning using Packed Secret Sharing

Alexander Bienstock
J.P. Morgan AI Research &
J.P. Morgan AlgoCRYPT CoE

Ujjwal Kumar
J.P. Morgan

Antigoni Polychroniadou
J.P. Morgan AI Research &
J.P. Morgan AlgoCRYPT CoE

Abstract

Federated Learning (FL) has gained lots of traction recently, both in industry and academia. In FL, a machine learning model is trained using data from various end-users arranged in committees across several rounds. Since such data can often be sensitive, a primary challenge in FL is providing privacy while still retaining utility of the model. Differential Privacy (DP) has become the main measure of privacy in the FL setting. DP comes in two flavors: central and local. In the former, a centralized server is trusted to receive the users' raw gradients from a training step, and then perturb their aggregation with some noise before releasing the next version of the model. In the latter (more private) setting, noise is applied on users' local devices, and only the aggregation of users' noisy gradients is revealed even to the server. Great strides have been made in increasing the privacy-utility trade-off in the central DP setting, by utilizing the so-called *matrix mechanism*. However, progress has been mostly stalled in the local DP setting. In this work, we introduce the *distributed* matrix mechanism to achieve the best-of-both-worlds; local DP and also better privacy-utility trade-off from the matrix mechanism. We accomplish this by proposing a cryptographic protocol that securely transfers sensitive values across rounds, which makes use of *packed secret sharing*. This protocol accommodates the dynamic participation of users per training round required by FL, including those that may drop out from the computation. We provide experiments which show that our mechanism indeed significantly improves the privacy-utility trade-off of FL models compared to previous local DP mechanisms, with little added overhead.

1 Introduction

In Federated Learning (FL), a machine learning model is trained using data from several end-users. Since such data can often be sensitive, a key challenge in FL is maintaining utility of the trained models, while preserving privacy of the end-users. FL has experienced an explosion of progress in recent years, both in industry and research. In terms of use in practice, there have been numerous deployments of FL recently, such as Google's and Apple's privacy-preserving training of machine learning models for making word suggestions in their mobile keyboards [29, 2] and voice assistants [2]. In FL research, new solutions continue to be proposed with better privacy-utility tradeoffs and usability, e.g., [34, 17, 15, 14].

In more detail, FL typically works in a round-based setting, wherein the current model parameters are sent to a set of clients, which we call a *committee*, who locally execute a step of Stochastic Gradient Descent on their own data to obtain gradients with respect to a loss function. These gradients are then aggregated using different techniques to update the model parameters for the next round

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

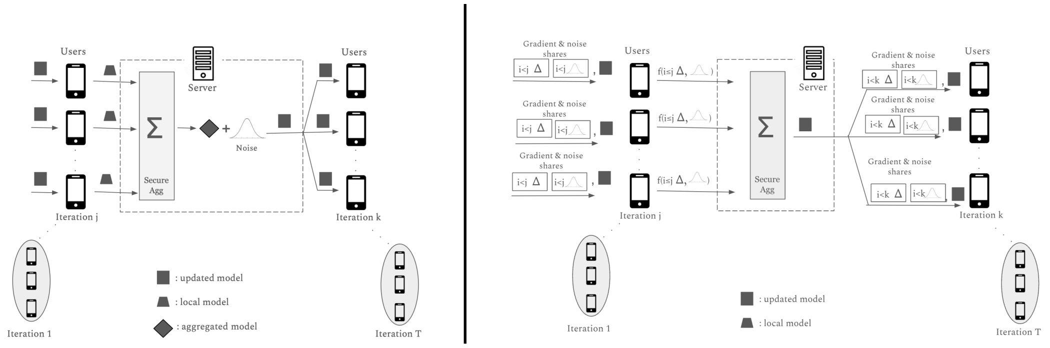

Figure 1: Left: Federated Learning in the central DP model. Right: Federated Learning based on our Distributed Matrix Mechanism in the local DP model.

(e.g., [40, 22, 47]). The main privacy metric for FL is differential privacy (DP) [18]. Roughly speaking, DP guarantees that with high probability, one cannot tell whether or not a user participated in a given FL execution. There are two different notions of DP that can be considered. In *central* DP, there is a centralized server who receives the gradients directly from the clients in each round and then updates the model based on its own noisy aggregation of these gradients. See the left side of Figure 1 for a flowchart illustrating the process involving the committee of users from round j and the committee of users from round $k = j + 1$. A Secure Aggregation [33, 10, 8, 38] protocol is applied in a black-box fashion to conceal local gradients, with noise being added exclusively by the server to preserve privacy. In this case, DP holds with respect to those to whom the server sends the updated models, but not the server itself. In *local* DP, there may still be a centralized server, however, the users utilize a Secure Aggregation protocol to only release to the server a noisy aggregation of their gradients, and thus DP holds with respect to the server as well.

There has been tremendous progress recently in the area of central DP for FL [34, 17, 15, 14]. These works use a sophisticated set of techniques from the DP literature called the *matrix mechanism* [31, 19] to achieve excellent privacy-utility trade-offs. Indeed, in this setting, since the central server receives all of the gradients in the clear and samples all noise on its own, it can *correlate* the noise across rounds in a complex manner. Intuitively, this means that noise can be re-used across rounds without being detected so that the cumulative noise across all rounds is lower compared to sampling new, fresh noise to hide the gradients in each round.

On the other hand, in the setting of local DP, the clients just add noise locally to their gradients, and then these noisy gradients are summed using a Secure Aggregation protocol. Since the noise is not correlated across epochs via the matrix mechanism like in the central DP setting, the privacy-utility trade-off of local DP is not as good as that of central DP thus far.

We note that in both settings, privacy amplification techniques like shuffling [21, 24] or (Poisson) subsampling [7, 54, 51] are sometimes used to increase privacy-utility tradeoffs; however, these require strong assumptions on how data is processed which are often not suitable for practice [34] and thus should be avoided.

Our Contributions In this work, we propose a solution to achieve the "best-of-both-worlds" of the central and local DP settings, without using privacy amplification, called the *Distributed Matrix Mechanism*. We achieve privacy with respect to the central server as in the local DP setting, while using correlated noise to get privacy-utility trade-offs close to that of the central DP setting. To facilitate this, we propose an efficient cryptographic protocol to *(re)share* users' noise and gradients across committees in a way that they remain private.

(Packed) secret sharing is a common technique in the cryptographic literature [25]. In such a protocol, there are n users, t_c of which might be corrupted by an adversary $\mathcal{A}$; by this we mean that $\mathcal{A}$ sees everything that the t_c users see, and can act arbitrarily on behalf of them. In this work, we will focus on the setting where in each round there are n users in the committee for that round and at most $t_c < (1/2 - \mu) \cdot n$ of them are corrupted, for some $0 < \mu < 1/2$. Packed secret sharing allows a user to split a secret vector $\boldsymbol{x} \in \mathbb{F}^k$, where $\mathbb{F}$ is a finite field and $0 < k < n$, amongst the n users, by sending them *shares* of the secrets. These shares are distributed in such a way that the t_c shares that

$\mathcal{A}$ sees reveal nothing about the secret vector $\boldsymbol{x}$. On the other hand, if at least $t_c + k$ users send their shares to another user, then this user can *reconstruct* the original secret vector $\boldsymbol{x}$. Moreover, if some of the corrupted parties send the reconstructing user incorrect shares, then this can be *detected*. Packed secret sharing is additionally *linear*, meaning that if the users have a sharing of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, they can add their shares together to obtain a sharing of $\boldsymbol{x}_3 = \boldsymbol{x}_1 + \boldsymbol{x}_2$ (which can later be constructed as such).

In our FL setting, we have the users in each round secret share vectors containing their noise and gradients. We then propose a *resharing protocol* such that given the secret shares of noise and gradients from users in a given round, the users can efficiently and securely *reshare* them to the users of the next round. This can be repeated for the same (and additional) secrets shares of noise and gradients across many rounds. Given this resharing protocol, we can instantiate the matrix mechanism in a distributed fashion: First, the users take linear combinations of the secret shared gradients and noise and thus introduce noise correlations across epochs. Then, we can reconstruct these aggregrated gradients with (correlated) noise to the server using Secure Aggregation. See the right side of Figure 1 for a flowchart illustrating our approach, where parties from committee j in round j receive noise and gradient shares from previous rounds. These parties combine the received shares with the new gradients and freshly sampled noise, applying a linear combination function f, and then input the result to the Secure Aggregation, whose output will be used to obtain the updated model. Afterward, the gradient and noise shares are reshared with the parties in the subsequent committee k, ensuring continuity in the distributed matrix mechanism.

Our resharing protocol also achieves *dropout tolerance*. In FL, the gradients from end-users often come from mobile devices, and therefore it may not be guaranteed that such users will stay online for the whole round, even if they are honest. Thus, the protocol must not fail if some (honest) users drop out, while still being able to handle other corrupted users. We design our resharing protocol in a particular way to be able to still work even if a certain fraction of users drop out in each committee.

Our main technical contribution is thus instantiating this dropout-resilient, new secure resharing protocol with constant overhead $O(1)$ overhead in the presence of t_c corrupted users per round and t_d dropout (honest) users per round, such that $t_c + t_d < (1/2 - \mu) \cdot n$. We do so by using three main ingredients: (i) packed secret sharing; (ii) parity check matrices, with which we can catch corrupted parties who do not follow the protocol; and (iii) random linear combinations, which allow us to perform such checks with low communication overhead. See Section 3 for a detailed explanation on how the constant overhead is achieved. Importantly, our method maintains differential privacy (DP) even in the presence of corrupted parties who might manipulate the shares, as we show this only leads to an additive attack on the values opened to the server, which can be viewed as a form of post-processing that does not compromise DP.

Another approach without secret sharing includes maintaining aggregate noise and gradients *masked* via a secure aggregation protocol. However, this approach is vulnerable: the server could selectively include or exclude certain masked noise terms as input to the secure aggregation, or manipulate the scaling of the masked gradients inputs, potentially undermining DP and revealing information about the current round's gradients. See Section E for more details on this approach and manipulation attacks.

We implement the Distributed Matrix Mechanism using our resharing protocol and use it to train differentially private FL models. We show that for Federated EMNIST [12] and Stack Overflow Next Word Prediction [4], our approach improves upon the privacy-utility tradeoff of the previous best local DP approach [33] based on a lightweight cryptographic solution.

Related Work DP has been used for various statistical tasks, where privacy is demanded [18, 31, 19]. DP-Follow-The-Regularized-Leader (DP-FTRL) [34] used the DP tree mechanism [31] to achieve high privacy-utility tradeoff for FL, without using any privacy amplification [1, 21, 24, 7, 51, 54]. To improve this, [17] use the matrix mechanism to provide better privacy-utility tradeoff for FL, where each user only participates in the training once. Follow-up work [15] allowed for multiple participations in the training using the matrix mechanism to get even higher privacy-utility tradeoff, while requiring a strict participation pattern amongst users. Then, [14] introduced more relaxed and realistic multi-participation training with the matrix mechanism, while achieving similar privacy-utility tradeoff.

Typically, prior DP mechanisms use secure aggregation in a black box way. The seminal work of [10] introduced secure aggregation for federated learning protocols with dropout resilience. Building on this, subsequent research [8, 32, 37, 49, 48, 53, 52, 38, 36] has focused on optimizing the protocols by either reducing the number of intermediate helper users or minimizing the rounds of communication required between users and the server per secure aggregation protocol

Several works have considered so-called *proactive secret sharing* [43, 6, 39]. This setting is very similar to ours in which secrets are reshared across rounds, however, there the users stay the same in each round (some of the users completely delete their state in each round). Papers that study a similar model to ours exist, but for more general computations and without a central server, and thus are inefficient [27, 9, 16, 44].

2 Packed Secret Sharing

Let $\mathbb{F}$ be some finite field. Let n be the number of parties in each committee; i.e., the number of clients in each round/iteration (assume uniform committee size). Let t_c be the number of maliciously corrupted parties in each committee. A (t_c+1)-out-of-n secret-sharing scheme takes as input a secret z from $\mathbb{F}$ and outputs n shares, one for each party, with the property that it is possible to efficiently recover z from every subset of t_c+1 shares, but every subset of at most t_c shares reveals nothing about the secret z. The value t_c is called the privacy threshold of the scheme.

A secret-sharing scheme consists of two algorithms: the first algorithm, called the *sharing algorithm*, Share, takes as input the secret z and the parameters n and t_c, and outputs n shares: $(z^1, \ldots, z^n) = \mathsf{Share}(z, n, t_c)$. We often denote to the vector of shares as $[z]_{t_c} = (z^1, \ldots, z^n)$. The second algorithm, called the *reconstruction algorithm*, $\mathsf{Reconstruct}$, takes as input party identity i and share z^i and outputs a reconstruction value $\mathsf{Reconstruct}(i, z^i)$. We will utilize secret sharing schemes in which $\lambda_i \cdot z^i = \mathsf{Reconstruct}(i, z^i)$, for some constant λ_i dependent on i. Any set of at least t_c+1 of these reconstruction values can be simply summed to obtain $z = \sum_i \lambda_i \cdot z^i$. It is required that such a reconstruction of shares generated from a value z reconstructs to the same value z. The secret-sharing scheme we use is also *linear*, meaning that if the parties add their shares z_1^i of a secret z_1 with their shares z_2^i of a secret z_2, then invoke $\mathsf{Reconstruct}$ to get reconstruction value $\lambda_i \cdot (z_1^i + z_2^i)$, summing these reconstruction values will yield $z_1 + z_2 = \sum_i \lambda_i (z_1^i + z_2^i)$. Using the notation from above, when all parties sum their shares of $[z_1]_{t_c}$ and $[z_2]_{t_c}$, we will write $[z_1 + z_2]_{t_c} = [z_1]_{t_c} + [z_2]_{t_c}$.

Packed secret sharing is an extension of traditional secret-sharing schemes, where a vector of $k > 1$ secrets $\boldsymbol{z} = (z_1, \ldots, z_k) \in \mathbb{F}^k$ is *packed* into a single set of (individual) shares. This technique is particularly useful for efficiency in cryptographic protocols, as it allows multiple secrets to be shared and reconstructed simultaneously with reduced overhead compared to handling each secret individually. Here, we still have that every subset of at most t_c shares reveals nothing about $\boldsymbol{z}$, but we need at least $t_c + k$ shares to be able to recover $\boldsymbol{z}$. There are also similar Share and $\mathsf{Reconstruct}$ algorithms, and we denote a sharing of some vector $\boldsymbol{z}$ as $[\boldsymbol{z}]_{t_c+k-1} = (\boldsymbol{z}^1, \ldots, \boldsymbol{z}^n)$. In addition, $\mathsf{Reconstruct}$ takes as input an index $j \in [k]$ representing the index of the vector to eventually reconstruct. Here, we utilize packed secret sharing schemes in which $\lambda_i^j \cdot \boldsymbol{z}^i = \mathsf{Reconstruct}(i, \boldsymbol{z}^i, j)$, for some constant λ_i^j dependent on i and j. If at least $t_c + k$ parties run the $\mathsf{Reconstruct}$ algorithm to get reconstruction values $\lambda_i^j \cdot \boldsymbol{z}^i$, then z_j can be computed with these values, which is again a simple sum $z_j = \sum_i \lambda_i^j \cdot \boldsymbol{z}^i$. The packed secret sharing scheme we use is also *linear* with respect to vector addition of the underlying secrets; i.e., $[\boldsymbol{z}_1 + \boldsymbol{z}_2]_{t_c+k-1} = [\boldsymbol{z}_1]_{t_c+k-1} + [\boldsymbol{z}_2]_{t_c+k-1}$.

In the following, t_c and k will be fixed, so we will simply refer to packed secret sharings as $[\boldsymbol{z}]$.

3 Linear Packed Resharing Protocol

In this section, we present our constant overhead Linear Resharing Protocol PSS. Let t_d be the number of (honest party) dropouts in each committee and t_c the number of corrupted parties in each committee. We will aim to handle $t_d + t_c < (1/2 - \mu)n$, for constant $0 < \mu < 1/2$.

Passively-Secure Protocol Our resharing protocol consists of four algorithms: it inherits the first algorithm Share from an underlying linear packed secret sharing scheme. Now, let it be the case that k packed secret sharings $[\boldsymbol{z}_1], \ldots, [\boldsymbol{z}_k]$ are generated for length-k secret vectors $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k \in \mathbb{F}^k$ to

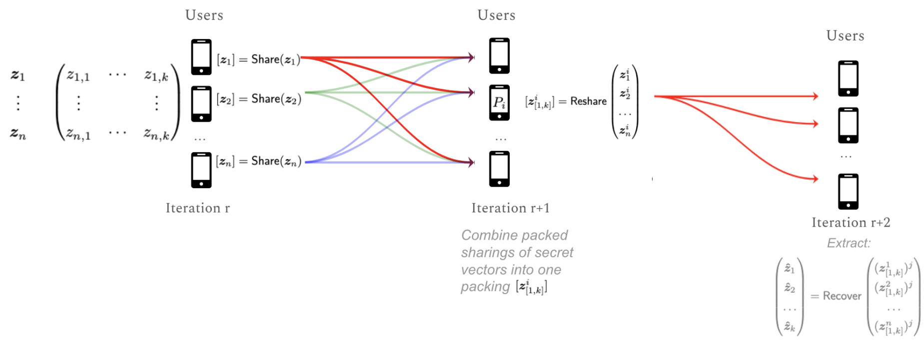

Figure 2: Packed Resharing Protocol

the n parties of iteration r (so there are k^2 total secrets). The next algorithm, called the *resharing algorithm*, Reshare, takes as input the packed shares of party P_i of iteration r, which we denote as the vector $\boldsymbol{z}^i_{[1,k]} = (\boldsymbol{z}^i_1, \ldots, \boldsymbol{z}^i_k)$, and reshares them to the parties of iteration $r+1$: $[\boldsymbol{z}^i_{[1,k]}] = ((\boldsymbol{z}^i_{[1,k]})^1, \ldots, (\boldsymbol{z}^i_{[1,k]})^n) = \mathsf{Reshare}(\boldsymbol{z}^i_{[1,k]})$. Let it be the case that each P_i in iteration r does this. Next, the *recovery algorithm*, Recover takes as input the reshared shares output to party P_j of iteration $r+1$, $(\boldsymbol{z}^1_{[1,k]})^j, \ldots, (\boldsymbol{z}^n_{[1,k]})^j$, and outputs new shares of the original secret vectors $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k$ for party P_j: $(\hat{\boldsymbol{z}}^j_1, \ldots, \hat{\boldsymbol{z}}^j_k) = \mathsf{Recover}((\boldsymbol{z}^1_{[1,k]})^j, \ldots, (\boldsymbol{z}^n_{[1,k]})^j)$.[1] The last algorithm Reconstruct is also inherited from the underlying linear packed secret sharing scheme.

We present the passively-secure version of our protocol below, wherein corrupted parties must follow the protocol and only try to break the privacy of other parties using what they see. The actively-secure protocol where corrupted parties may behave arbitrarily is presented in Section C.

- $\mathsf{Reshare}(\boldsymbol{z}^i_{[1,k]})$ simply executes and outputs $[\boldsymbol{z}^i_{[1,k]}] = \mathsf{Share}(\boldsymbol{z}^i_{[1,k]})$.
- $\mathsf{Recover}((\boldsymbol{z}^1_{[1,k]})^j, \ldots, (\boldsymbol{z}^n_{[1,k]})^j)$ computes and outputs for $m \in [k]$: $\hat{\boldsymbol{z}}^j_m = \sum_i \mathsf{Reconstruct}(i, (\boldsymbol{z}^i_{[1,k]})^j, m)$.

The protocol is also pictorially presented in Figure 2.

Now we observe how $\mathsf{Recover}(\cdot)$ outputs packed shares of the original secrets. Recall that $\mathsf{Reconstruct}(i, (\boldsymbol{z}^i_{[1,k]})^j, m) = \lambda^m_i \cdot (\boldsymbol{z}^i_{[1,k]})^j$, so we can re-write $\hat{\boldsymbol{z}}^j_m = \sum_i \lambda^m_i \cdot (\boldsymbol{z}^i_{[1,k]})^j$. Moreover, each $(\boldsymbol{z}^i_{[1,k]})^j$ is a packed share of sharing of vector $\boldsymbol{z}^i_{[1,k]} = (\boldsymbol{z}^i_1, \ldots, \boldsymbol{z}^i_k)$ for a *linear* packed secret sharing scheme. Thus, we are computing new packed shares of the vectors $\sum_i \lambda^m_i \cdot (\boldsymbol{z}^i_1, \ldots, \boldsymbol{z}^i_k)$. Each $\boldsymbol{z}^i_l$ was itself P_i's share of packed sharing of vector $\boldsymbol{z}_l$. Thus the packed shares we are computing indeed share the vectors $\sum_i \lambda^m_i \cdot (\boldsymbol{z}^i_1, \ldots, \boldsymbol{z}^i_k) = (\sum_i \mathsf{Reconstruct}(i, \boldsymbol{z}^i_1, m), \ldots, \sum_i \mathsf{Reconstruct}(i, \boldsymbol{z}^i_k, m)) = (\boldsymbol{z}_{1,m}, \ldots, \boldsymbol{z}_{k,m})$.

It is clear that the output of Reshare() reveals nothing to the t_c corrupted parties, since it just uses Share() of the underlying packed secret sharing scheme, that is secure against t_c corrupted parties. Since the number of honest parties that do not dropout is at least $n - t_d - t_c > (1/2 + \mu)n$, it is clear that this protocol is resilient to the t_d (honest) dropout parties, if $k \leq 2\mu n$. This is because $t_c + k \leq (1/2 + \mu)n < n - t_d - t_c$, so the shares of the honest parties that do not dropout can still be used to obtain the secrets.

Communication Complexity The total communication complexity of this protocol is n^2 field elements—each party in iteration r sends a share to every party in iteration $r+1$. If we choose $k = 2\mu n$, then this is for $k^2 = 4\mu^2 n^2$ secrets, which is $1/4\mu^2$ communicated field elements per secret.

[1] Note: they are shares of length-k vectors $(z_{1,m}, \ldots, z_{k,m})$ for each $m \in [k]$, instead of $(z_{l,1}, \ldots, z_{l,k})$, for each $l \in [k]$.

4 Differentially Private Federated Learning

In this section, we define some notions important to DPFL before recalling some DP mechanisms for FL from prior work, one with local DP and one with central DP. In the next section, we will describe in more detail our Distributed Matrix Mechanism which achieves the best-of-both-worlds of these two mechanisms. Let T^* be the number of training rounds and d be the dimension of a model to be trained via DPFL.

Adjacency and Participation Schemas DP requires a notion of adjacent datasets. Two data streams $\boldsymbol{X}$ and $\tilde{\boldsymbol{X}}$ are adjacent if the data associated with any single user is altered, in every round where the user participates.[2] Thus, any $\boldsymbol{X}_T$ which a user contributed a gradient $\boldsymbol{g}_{T,i}$ to can be changed subject to the constraint $||\boldsymbol{g}_{T,i}|| \leq c$, where c is the norm clip. The participation pattern does not change in these two adjacent streams. A *participation schema* Φ contains all possible *participation patterns* $\phi \subseteq \Phi$, with each $\phi \in [T^*]$ indicating a set of rounds in which a single user participates. Let $\texttt{Nbrs}$ be the set of all pairs of neighboring streams $\boldsymbol{X}$ and $\mathfrak{D} = \{\boldsymbol{X} - \tilde{\boldsymbol{X}} : (\boldsymbol{X}, \tilde{\boldsymbol{X}}) \in \texttt{Nbrs}\}$ represent the set of all possible differences between neighboring $\boldsymbol{X}, \tilde{\boldsymbol{X}}$. We say a $\mathfrak{D}$ satisfies the participation schema Φ if the indices of all nonzero rows in each $\mathbb{R}^{T^* \times d}$ matrix $\boldsymbol{U} \in \mathfrak{D}$ are a subset of some $\phi \in \Phi$. In this work, we consider the *b-min-sep-participation* schema of [14], where any adjacent participations are at least b steps apart.

Local DP Distributed Discrete Gaussian Mechanism [33] In this setting, the central server only learns noisy versions of the aggregated model gradients, in each round T. This means that each user locally applies some noise to their model gradients. A naive way to do so is for each user to locally compute $\hat{\boldsymbol{g}}_{T,i} \leftarrow \boldsymbol{g}_{t,i} + \boldsymbol{z}_{t,i}$, where $\boldsymbol{z}_{T,i}$ is drawn from some noise distribution. Then, these noisy gradients are combined using Secure Aggregation into $\hat{\boldsymbol{X}}_T \leftarrow \sum_i \hat{\boldsymbol{g}}_{T,i}$ for the sever, which is then used to compute the next model iteration release. Thus, the sensitivity of this setting for a participation schema Φ can be simply computed as $\mathrm{sens}_\Phi = \sup_{(\boldsymbol{X}, \tilde{\boldsymbol{X}}) \in \texttt{Nbrs}} ||\boldsymbol{X} - \tilde{\boldsymbol{X}}||_F = \sup_{\boldsymbol{U} \in \mathfrak{D}} ||\boldsymbol{U}||_F$.

Centralized DP Matrix Mechanism In this setting, the central server learns the (aggregated) gradients in the clear, and adds noise to the new model before releasing it to the users of the next round. Let $\boldsymbol{A} \in \mathbb{R}^{T^* \times T^*}$ be an appropriate linear query work-load (e.g., prefix sums or a matrix encoding of stochastic gradient descent with momentum (SGDM) [17]). Matrix mechanisms in the central DP setting use a factorization $\boldsymbol{A} = \boldsymbol{B}\boldsymbol{C}$ to privately estimate the quantity $\boldsymbol{A}\boldsymbol{X}$ as $\widehat{\boldsymbol{A}\boldsymbol{X}} = \boldsymbol{B}(\boldsymbol{C}\boldsymbol{X} + \boldsymbol{Z})$, where $\boldsymbol{Z}$ is sampled by the central server from some noise distribution.

Each entry of the vector $\widehat{\boldsymbol{A}\boldsymbol{X}}$ corresponds to a model iteration that is released. The matrix $\boldsymbol{A}$ is lower-diagonal, which means that the T-th entry of $\widehat{\boldsymbol{A}\boldsymbol{X}}$ only depends on the first T entries of $\boldsymbol{X}$, for each dimension. Additionally, the T-th entry of $\widehat{\boldsymbol{A}\boldsymbol{X}}$ depends on the first T entries of $\boldsymbol{Z}$, which means that the noise used in each released model iteration is *correlated*. This means that each sampled noise element can have *less variance*, resulting in *better accuracy*.

We now define the *sensitivity* of the central DP matrix mechanism for a particular participation schema Φ with set of neighboring streams $\texttt{Nbrs}$ as $\mathrm{sens}_\Phi(\boldsymbol{C}) = \sup_{(\boldsymbol{X}, \tilde{\boldsymbol{X}}) \in \texttt{Nbrs}} ||\boldsymbol{C}\boldsymbol{X} - \boldsymbol{C}\tilde{\boldsymbol{X}}||_F = \sup_{\boldsymbol{U} \in \mathfrak{D}} ||\boldsymbol{C}\boldsymbol{U}||_F$. As in previous works, it is useful to analyze $\mathrm{sens}_\Phi(\boldsymbol{C})$ when each gradient $\boldsymbol{g}_{t,i}$ has ℓ_2 norm at most $c = 1$, noting that the actual value of $\mathrm{sens}_\Phi(\boldsymbol{C})$ scales with c in general. In our work, however, it is useful to explicitly define the sensitivity with gradients of ℓ_2 norm $c = 1$ as $\mathrm{sens}^1_\Phi(\boldsymbol{C})$.

The expected total squared error on $\boldsymbol{A}$ is typically given as $\mathcal{L}(\boldsymbol{B}, \boldsymbol{C}) = \mathrm{sens}_\Phi(\boldsymbol{C})||\boldsymbol{B}||_F^2$ and the goal is to find a factorization that minimizes this loss.

5 Distributed Matrix Mechanism

In this paper, we achieve the *best-of-both-worlds* of the previous two mechanisms by using the matrix mechanism in a way that still only reveals to the server linear combinations of the noisy aggregated gradients. See Protocol 1 for a detailed description of our FL protocol Π_{PPFL}.

[2] We study the more general user-level DP in this work, as opposed to example-level DP.

Protocol 1 Privacy-Preserving Federated Learning Protocol Π_{PPFL}

Protocol Π_{PPFL} runs with clients $P_1, \ldots, P_N$ and a server S. Let $\mathsf{PSS} = (\mathsf{Share}, \mathsf{Reshare}, \mathsf{Reconstruct}, \mathsf{Recover})$ be a packed resharing protocol (See Section 3) and let $\mathsf{SecAgg} = (\mathsf{SecAgg.Enc}, \mathsf{SecAgg.Dec})$ be a secure aggregation protocol. $\Pi_{\mathsf{PPFL}} = (\mathsf{Setup}, \mathsf{Initialize}, \mathsf{Agg})$ proceeds as follows:

Parameters: Model dimension $d \in \mathbb{N}$; number of rounds T^*; clipping threshold $c > 0$; granularity $\gamma > 0$; noise scale $\sigma > 0$; bias $\beta \in [0, 1)$; finite field $\mathbb{F}$ of bit-width m; public (lower-triangular) matrix encoding of prefix sums or stochastic gradient descent with momentum (SGDM) [17]) $\boldsymbol{A} \in \mathbb{R}^{T^* \times T^*}$; matrices $\boldsymbol{B}, \boldsymbol{C}$ such that $\boldsymbol{A} = \boldsymbol{B}\boldsymbol{C}$.

Inputs: For $i \in [N]$, party P_i holds input dataset D_i. Without loss of generality we assume that committees in each training iteration are of size n.

$\mathsf{Agg}(D_i, \boldsymbol{\theta}_{T-1}, \{[\boldsymbol{g}_{T-1,\eta,j}], [\boldsymbol{z}_{T-1,\eta,j}]\}_{\eta \in [n]}, \{[\boldsymbol{X}^{\eta}_{\tau,[1,k]}], [\boldsymbol{Z}^{\eta}_{\tau,[1,k]}]\}_{\tau \in [T-2], \eta \in [n]})$: Let $\mathcal{C}_T$ be the set of chosen clients for the T-th training iteration. For each T each client P_i in $\mathcal{C}_T$ proceeds as follows:

Round 1:

- Runs training model on $\boldsymbol{\theta}_{T-1}, D_i$ which generates the vector of local gradients $\boldsymbol{g}_{T,i}$ (that are then clipped to norm c, scaled via granularity parameter $\gamma > 0$, flattened, and rounded/discretized with bias $\beta \in [0, 1)$ as in [33]; details of this are provided in the Section A).
- Samples a noise vector $\boldsymbol{z}_{T,i}$ from a Discrete Gaussian distribution $\mathcal{N}_{\mathbb{Z}}(0, \sigma^2/\gamma^2)$.
- For each batch of parameters $j \in [d/k]$ of size k, secret shares the noise vectors and the gradients using the packed secret sharing scheme as $[\boldsymbol{z}_{T,i,j}] = \mathsf{Share}(\boldsymbol{z}_{T,i,j})$ and $[\boldsymbol{g}_{T,i,j}] = \mathsf{Share}(\boldsymbol{g}_{T,i,j})$ to the set $\mathcal{C}_{T+1}$ of clients of the next training iteration. Each η-th share $\boldsymbol{z}^{\eta}_{T,i,j}$ and $\boldsymbol{g}^{\eta}_{T,i,j}$ is encrypted to the η-th client of $\mathcal{C}_{T+1}$ using authenticated and encrypted channels.

If $T = 1$:

- For each model parameter $l \in [d]$, invokes a SecAgg protocol and sends $y_{T,i,l} = \mathsf{SecAgg.Enc}(\boldsymbol{A}_{[1,1]} \cdot g_{T,i,l} + \boldsymbol{B}_{[1,1]} \cdot z_{T,i,l})$ to S.

If $T > 2$:

- Decrypts and recovers each batch using the packed resharing protocol on the sets of k batches from all previous rounds $\tau \in [T-2]$ as $(\hat{\boldsymbol{Z}}^i_{\tau,1}, \ldots, \hat{\boldsymbol{Z}}^i_{\tau,k}) = \mathsf{Recover}((\boldsymbol{Z}^1_{\tau,[1,k]})^i, \ldots, (\boldsymbol{Z}^n_{\tau,[1,k]})^i)$ and $(\hat{\boldsymbol{X}}^i_{\tau,1}, \ldots \hat{\boldsymbol{X}}^i_{\tau,k}) = \mathsf{Recover}((\boldsymbol{X}^1_{\tau,[1,k]})^i, \ldots, (\boldsymbol{X}^n_{\tau,[1,k]})^i)$.
- Then again reshares these shares as $[\hat{\boldsymbol{X}}^i_{\tau,[1,k]}] = \mathsf{Reshare}(\hat{\boldsymbol{X}}^i_{\tau,[1,k]})$ and $[\hat{\boldsymbol{Z}}^i_{\tau,[1,k]}] = \mathsf{Reshare}(\hat{\boldsymbol{X}}^i_{\tau,[1,k]})$ to set $\mathcal{C}_{T+1}$.

If $T > 1$:

- Decrypts and aggregates the shares of each batch of noise vector and gradients $[\boldsymbol{Z}_{T-1,j}] = (\sum_{\eta=1}^{n} [\boldsymbol{z}_{T-1,\eta,j}])$ and $[\boldsymbol{X}_{T-1,j}] = (\sum_{\eta=1}^{n} [\boldsymbol{g}_{T-1,\eta,j}])$ from round $T-1$ and securely reshares each set of k such batches using the packed resharing protocol as $[\boldsymbol{Z}^i_{T-1,[1,k]}] = \mathsf{Reshare}(\boldsymbol{Z}^i_{T-1,[1,k]})$, $[\boldsymbol{X}^i_{T-1,[1,k]}] = \mathsf{Reshare}(\boldsymbol{X}^i_{T-1,[1,k]})$ to the set of clients in $\mathcal{C}_{T+1}$.
- For each model parameter $l \in [k]$ inside batch $j \in [d/k]$, invokes a SecAgg protocol and sends to S:

$$y_{T,i,j \cdot d/k+l} = \mathsf{SecAgg.Enc}\Bigg(\mathsf{Reconstruct}\left(i, \sum_{\tau=1}^{T-1} (\boldsymbol{A}_{[T,\tau]} \cdot \hat{\boldsymbol{X}}^i_{\tau,j} + \boldsymbol{B}_{[T,\tau]} \cdot \boldsymbol{Z}^i_{\tau,j}), l\right)$$
$$+ \boldsymbol{A}_{[T,T]} \cdot \boldsymbol{g}_{T,i,j \cdot d/k+l} + \boldsymbol{B}_{[T,T]} \cdot \boldsymbol{z}_{T,i,j \cdot d/k+l}\Bigg).$$

Round 2:

- S recovers the noisy summed gradients as $Y_{T,l} = \mathsf{SecAgg.Dec}(\sum_i^n y_{T,i,l})$ (then unflattens and rescales as in [33]; details of this are provided in the Section A) and then applies them to the model to obtain $\boldsymbol{\theta}_T$.

In the T-th round, we will assume that the n clients selected have, for $\tau \in [T-2], \eta \in [n]$, encrypted secret shares (i) $[\boldsymbol{Z}^{\eta}_{\tau,[1,k]}]$, which are the (aggregated) noise sampled in the first $T-2$ rounds and reshared by party η in the previous round; and (ii) $[\boldsymbol{X}^{\eta}_{\tau,[1,k]}]$, which are the (aggregated) gradients from the first $T-2$ rounds and reshared by party η in the previous round (both really are shares of batches of the noise and gradients, respectively). The clients will decrypt these, and then recover shares of the same: $(\hat{\boldsymbol{Z}}^{i}_{\tau,1},\ldots,\hat{\boldsymbol{Z}}^{i}_{\tau,k}) = \mathsf{Recover}((\boldsymbol{Z}^{1}_{\tau,[1,k]})^{i},\ldots,(\boldsymbol{Z}^{n}_{\tau,[1,k]})^{i})$ and $(\hat{\boldsymbol{X}}^{i}_{\tau,1},\ldots\hat{\boldsymbol{X}}^{i}_{\tau,k}) = \mathsf{Recover}((\boldsymbol{X}^{1}_{\tau,[1,k]})^{i},\ldots,(\boldsymbol{X}^{n}_{\tau,[1,k]})^{i})$. Additionally, from round $T-1$, the clients will have encrypted shares of (i) $[\boldsymbol{z}_{T-1,i}]$, which is the noise sampled by the i-th client in the last round; and (ii) $[\boldsymbol{g}_{T-1,i}]$, which is the gradient computed by the i-th client in the last round. The clients will decrypt these, and then compute the aggregated versions $[\boldsymbol{Z}_{T-1}] = (\sum_{\eta=1}^{n}[\boldsymbol{z}_{T-1,i}])$ and $[\boldsymbol{X}_{T-1}] = (\sum_{\eta=1}^{n}[\boldsymbol{g}_{T-1,i}])$.

Next, as in the distributed setting, the clients will compute their local gradients $\boldsymbol{g}_i$ (clipped, scaled, flattened, and rounded as in [33]) using current model parameters $\boldsymbol{\theta}_{T-1}$ and data D_i, and sample some noise $\boldsymbol{z}_i$ from a Discrete Gaussian distribution. The parties then take linear combinations, according to $\boldsymbol{A}$ and $\boldsymbol{B}$, of the packed sharings of gradients and noise of all previous rounds as well as their current gradients and noise vectors to obtain packed sharings of the next output of the matrix mechanism, $\widehat{\boldsymbol{AX}}_T$. We employ secure aggregation SecAgg in a black-box way to reconstruct these packed sharings (which are unflattened and rescaled by the server [33]).

Finally, each client will compute some secret shares $[\boldsymbol{z}_i]$, $[\boldsymbol{g}_i]$ of their local gradients and noise. They will also reshare their shares $\hat{\boldsymbol{Z}}^{i}_{\tau,m}$ and $\hat{\boldsymbol{X}}^{i}_{\tau,m}$ of the aggregated noise and gradients from the first $T-1$ rounds.The clients reshare the shares according to the protocol in Section 3.

Privacy We now state the privacy of our protocol. First we explain some parameters: c is the norm to which gradients are clipped, $\gamma > 0$ is used to determine the granularity for the discretization of gradients, β determines the bias of the randomized rounding for discretization, and σ is the noise scale of the Discrete Gaussians. Details on these steps (for which we use the same strategy as [33]) are provided in Section A. The τ value in the theorem bounds the max divergence between the sum of n discrete Gaussians each with scale σ/γ and one discrete Gaussian with scale $\sqrt{n}\sigma/\gamma$. The following theorem is proved in Section B.

Theorem 1. *Consider a query matrix $\boldsymbol{A} \in \mathbb{R}^{T^*\times T^*}$ along with a fixed factorization $\boldsymbol{A} = \boldsymbol{BC}$ with $\Delta = \mathrm{sens}^1_{\Phi}(\boldsymbol{C})$. Let $\tau \coloneqq 10\cdot\sum_{k=1}^{n-1} e^{-2\pi^2\frac{\sigma^2}{\gamma^2}\cdot\frac{k}{k+1}}$ and*

$$\hat{c}^2 \coloneqq \min\left\{c^2 + \frac{1}{4}\gamma^2 d + \sqrt{2\log(1/\beta)}\cdot\gamma\cdot(c + \frac{1}{2}\gamma\sqrt{d}),\quad (c+\gamma\sqrt{d})^2\right\},$$

Assume that the number of corruptions in each committee t_c and number of dropouts (of honest parties) in each committee t_d is such that $t_c + t_d < (1/2-\mu)\cdot n$ for $0 < \mu < 1/2$. Then Π_{PPFL} satisfies $\frac{1}{2}\varepsilon^2$-concentrated differential privacy for $\varepsilon \coloneqq \min\left\{\sqrt{\frac{\Delta^2\hat{c}^2}{n\sigma^2} + 2\tau d},\ \frac{\Delta\hat{c}}{\sqrt{n}\sigma} + \tau\sqrt{d}\right\}$.[3]

Accuracy We now extend the theoretical analysis of the accuracy of the Distributed DP mechanism from [33] to our Distributed Matrix Mechanism (DMM). First, we explain an additional parameter: m is the bit-width of the finite field $\mathbb{F}$ used in Π_{PPFL}. The following theorem is proved in Section B.

Theorem 2. *Let $n, m, d, T^* \in \mathbb{N}$, and $c, \varepsilon > 0$ satisfy:*

$$m \geq \tilde{O}\left(\max_{T\in[T^*]} ||\boldsymbol{A}_{[T:,]}||_2\sqrt{nT} + \max_{T\in[T^*]} ||\boldsymbol{B}_{[T:,]}||_2\frac{\sqrt{d}\Delta}{\varepsilon}\right).$$

Let Π_{PPFL} be instantiated with parameters $\gamma = \tilde{O}\left(\frac{\max_{T\in[T^*]}||\boldsymbol{A}_{[T:,]}||_2 c\sqrt{nT}}{m\sqrt{d}} + \frac{\max_{T\in[T^*]}||\boldsymbol{B}_{[T:,]}||_2 c\Delta}{\varepsilon m}\right)$, $\beta \leq \Theta\left(\frac{1}{n}\right)$ and $\sigma = \tilde{\Theta}\left(\frac{c\Delta}{\varepsilon\sqrt{n}} + \sqrt{\frac{d}{n}}\cdot\frac{\gamma\Delta}{\varepsilon}\right)$. Then Π_{PPFL} satisfies $\frac{1}{2}\varepsilon^2$-concentrated differential privacy and attains the following accuracy. Let each $\boldsymbol{g}_{T,i} \in \mathbb{R}^d$ have $||\boldsymbol{g}_{T,i}||_2 \leq c$ for all $T \in [T^*], i \in [n]$. Then $\sum_{T=1}^{T^*}\mathbb{E}\left[\left|\left|\Pi_{\mathsf{PPFL}}(X) - \boldsymbol{A}_{[T,:]}\sum_{i=1}^{n}\boldsymbol{X}_i\right|\right|_2^2\right] \leq O\left(||\boldsymbol{B}||_F^2\frac{c^2\Delta^2 d}{\varepsilon^2}\right)$.

[3] We note that, just as in [33] and all other works using Secure Aggregation to obtain DP guarantees via aggregated noises, we actually obtain *computational* DP [42].

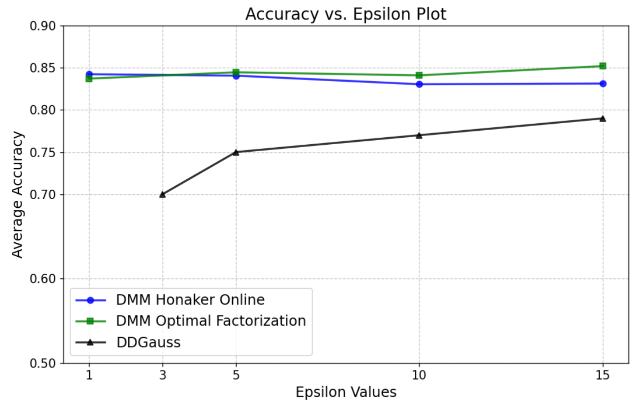

Figure 3: Test accuracies on EMNIST across different ε for the DDG mechanism and our DMM instantiated with the optimal factorization and the Honaker online factorization.

6 Experiments

Here we empirically evaluate our Distributed Matrix Mechanism (DMM) for Federated Learning on the Federated EMNIST public benchmark [12], as in [33]. We provide additional experimental details and results in Section D. Federated EMNIST is an image classification dataset containing 671,585 training handwritten digit/letter images over 64 classes grouped into $N = 3400$ clients by their writer. We use the standard dataset split provided by TensorFlow. We compare to the Distributed Discrete Gaussian Mechanism for FL [33] that also obtains local DP, but with independent noise and reliance upon privacy amplification via sampling [1, 35, 7]. In this setting, users are randomly sampled to participate in each round with replacement (and thus may participate multiple times), without the adversary knowing their identities, which leads to a lower ε for DP.

As in [33], we train a small convolutional net with two 3×3 conv layers with $32/64$ channels followed by two fully connected layers with $128/62$ output units; a 2×2 max pooling layer and two dropout layers with drop rate $0.25/0.5$ are added after the first 3 trainable layers, respectively. The total number of parameters is $d = 1018174$. We use namely momentum 0.9, 1 client training epoch per round, client learning rate $\eta_c = 0.02$, server learning rate $\eta_s = 1$, and client batch size to 16. For Π_{PPFL}, we assume that $\mu = 1/6$; i.e., the number of corrupted parties and dropout parties per round satisfies $t_c + t_d < 1/3n$.

Matrix Factorizations We use two different matrix factorizations $\boldsymbol{A} = \boldsymbol{B}\boldsymbol{C}$ for our experiments. The first is the optimal with respect to the loss function $\mathcal{L}(\boldsymbol{B}, \boldsymbol{C}) = \text{sens}_\Phi(\boldsymbol{C})||\boldsymbol{B}||_F^2$ for the b-min-sep-participation schema Φ, as introduced by [14]. The second is the Honaker Online mechanism [34, 30], where $\boldsymbol{C}$ is essentially the binary tree matrix. This mechanism has the benefit that it allows for implementations with only $\log(T^*)$ overhead; i.e., in the T-th round, the released model can be computed using at most $d \cdot \log(T^*)$ values. Thus, the size of the secret vectors that must be reshared from one committee to the next are at most $d \cdot \log(T^*)$ instead of $d \cdot T^*$, which greatly increases efficiency, as we will see below. For both factorizations, we measure $\text{sens}_\Phi^1(\boldsymbol{C})$ with respect to the b-min-sep-participation schema using [14, Theorems 2 and 3].

Results Figure 3 shows that for several different ε privacy levels, our DMM significantly outperforms the DDGauss Mechanism in terms of classification accuracy, due to the use of correlated noise across rounds. We also see that the Honaker mechanism only sees slight accuracy degradation compared to the mechanism based on the optimal b-min-sep-participation matrix factorization. Therefore, the tree mechanism might be best in practice due to much better efficiency. These experiments all use $n = 40$ clients per round. For the tree mechanism, we use $T^* = 2^9 = 512$ and for the optimal matrix factorization, we use $T^* = 765$. Both use $b = 85$.

Efficiency Table 1 shows the client computation and communication costs of Π_{PPFL} and also the SecAgg protocol Flamingo [38]. We run the experiments on an Ubuntu machine with a 3.0 GHz Intel Xeon GHz processor and 192 GiB of memory, and use 32 bits to represent field values. We take an

Setting	Π_{PPFL} **Comp.**	SecAgg **Comp.**	Π_{PPFL} **Comm.**	SecAgg **Comm.**
Opt.	593 s	0.101 s	1.72 GB	4.07 MB
Honaker	3.95 s	0.101 s	11.5 MB	4.07 MB

Table 1: Client Computation and communication of Π_{PPFL} and SecAgg per round for committee size $n = 64$. SecAgg stats are from Flamingo SecAgg protocol [38].

average over 10 runs for each reported value. For computational experiments, we use $n = 64$, as the Flamingo code requires n to be a power of two. For the optimal matrix factorization results, we report for the worst-case complexity per round, which is the last round, since here, clients need to reshare the noise and gradients from all previous rounds.

In this setting, we see the optimal matrix factorization results in about a 150x increase in both the computation and communication per client compared to the Honaker online factorization. This suggests that the small increase in accuracy from using the optimal matrix factorization may not be worth it in terms of the added efficiency costs.

Compared to Flamingo, we see a large ~ 40x increase in computation from the Honaker online factorization in Π_{PPFL}; however, ~ 4 seconds per round is still very reasonable. In terms of communication, we see a modest 2.8x increase for the Honaker online factorization in Π_{PPFL} compared to that of Flamingo. We believe that this added overhead is worth it given the increased accuracy.

Disclaimer

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS ’16, page 308–318, New York, NY, USA, 2016. Association for Computing Machinery.

[2] Apple. Differential privacy, 2016.

[3] 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)*, page 920–925. IEEE Press, 2020.

[4] T.F.F. Authors. Tensorflow federated stack overflow dataset, 2019.

[5] Borja Balle, Gilles Barthe, Marco Gaboardi, Justin Hsu, and Tetsuya Sato. Hypothesis testing interpretations and renyi differential privacy. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2496–2506. PMLR, 26–28 Aug 2020.

[6] Joshua Baron, Karim El Defrawy, Joshua Lampkins, and Rafail Ostrovsky. How to withstand mobile virus attacks, revisited. In *Proceedings of the 2014 ACM Symposium on Principles of*

Distributed Computing, PODC '14, page 293–302, New York, NY, USA, 2014. Association for Computing Machinery.

[7] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, FOCS '14, page 464–473, USA, 2014. IEEE Computer Society.

[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*, CCS '20, page 1253–1269, New York, NY, USA, 2020. Association for Computing Machinery.

[9] Alexander Bienstock, Daniel Escudero, and Antigoni Polychroniadou. On linear communication complexity for (maximally) fluid mpc. In *Advances in Cryptology – CRYPTO 2023: 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20–24, 2023, Proceedings, Part I*, page 263–294, Berlin, Heidelberg, 2023. Springer-Verlag.

[10] 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.

[11] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Proceedings, Part I, of the 14th International Conference on Theory of Cryptography - Volume 9985*, page 635–658, Berlin, Heidelberg, 2016. Springer-Verlag.

[12] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konecný, H. B. McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *ArXiv*, abs/1812.01097, 2018.

[13] Clément L. Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[14] Christopher A. Choquette-Choo, Arun Ganesh, Ryan McKenna, H. Brendan McMahan, Keith Rush, Abhradeep Thakurta, and Zheng Xu. (amplified) banded matrix factorization: A unified approach to private training. In *37th Conference on Neural Information Processing Systems (NeurIPs 2023)*, 2023.

[15] Christopher A. Choquette-Choo, H. Brendan McMahan, Keith Rush, and Abhradeep Thakurta. Multi-epoch matrix factorization mechanisms for private machine leanring. In *40th International Conference on Machine Learning*, 2023.

[16] Arka Rai Choudhuri, Aarushi Goel, Matthew Green, Abhishek Jain, and Gabriel Kaptchuk. Fluid mpc: Secure multiparty computation with dynamic participants. page 94–123, Berlin, Heidelberg, 2021. Springer-Verlag.

[17] Sergey Denisov, Brendan McMahan, Keith Rush, Adam Smith, and Abhradeep Guha Thakurta. Improved differential privacy for sgd via optimal private linear operators on adaptive streams. In *36th Conference on Neural Information Processing Systems (NeurIPS 2022)*, 2023.

[18] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[19] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, page 715–724, New York, NY, USA, 2010. Association for Computing Machinery.

[20] Cynthia Dwork and Guy N. Rothblum. Concentrated differential privacy. *CoRR*, abs/1603.01887, 2016.

[21] 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 Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2468–2479. SIAM, 2019.

[22] Alireza Fallah, Aryan Mokhtari, and Asuman E. Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In *Neural Information Processing Systems*, 2020.

[23] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to byzantine-robust federated learning. In *Proceedings of the 29th USENIX Conference on Security Symposium*, SEC'20, USA, 2020. USENIX Association.

[24] Vitaly Feldman, Audra McMillan, and Kunal Talwar. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 954–964, 2022.

[25] 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.

[26] Daniel Genkin, Yuval Ishai, Manoj M. Prabhakaran, Amit Sahai, and Eran Tromer. Circuits resilient to additive attacks with applications to secure computation. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, page 495–504, New York, NY, USA, 2014. Association for Computing Machinery.

[27] Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. Yoso: You only speak once. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021*, pages 64–93, Cham, 2021. Springer International Publishing.

[28] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, USA, 2004.

[29] Google. Learn how gboard gets better, 2023.

[30] James Honaker. Efficient use of differentially private binary trees, 2015.

[31] T-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming*, pages 405–417, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[32] Swanand Kadhe, Nived Rajaraman, Onur Ozan Koyluoglu, and Kannan Ramchandran. Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning. *CoRR*, abs/2009.11248, 2020.

[33] Peter Kairouz, Ziyu Liu, and Thomas Steinke. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *38th International Conference on Machine Learning (ICML 2021)*, 2021.

[34] Peter Kairouz, Brendan McMahan, Shuang Song, Om Thakkar, Abhradeep Thakurta, and Zheng Xu. Practical and private (deep) learning without sampling or shuffling. In *38th International Conference on Machine Learning (ICML 2021)*, 2021.

[35] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM J. Comput.*, 40(3):793–826, jun 2011.

[36] Hanjun Li, Huijia Lin, Antigoni Polychroniadou, and Stefano Tessaro. Lerna: Secure single-server aggregation via key-homomorphic masking. In *Advances in Cryptology – ASIACRYPT 2023: 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4–8, 2023, Proceedings, Part I*, page 302–334, Berlin, Heidelberg, 2023. Springer-Verlag.

[37] Zizhen Liu, Si Chen, Jing Ye, Junfeng Fan, Huawei Li, and Xiaowei Li. SASH: efficient secure aggregation based on SHPRG for federated learning. In James Cussens and Kun Zhang, editors, *Uncertainty in Artificial Intelligence, Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence, UAI 2022, 1-5 August 2022, Eindhoven, The Netherlands*, volume 180 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR, 2022.

[38] Y. Ma, J. Woods, S. Angel, A. Polychroniadou, and T. Rabin. Flamingo: Multi-round single-server secure aggregation with applications to private federated learning. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 477–496, Los Alamitos, CA, USA, may 2023. IEEE Computer Society.

[39] Sai Krishna Deepak Maram, Fan Zhang, Lun Wang, Andrew Low, Yupeng Zhang, Ari Juels, and Dawn Song. Churp: Dynamic-committee proactive secret sharing. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 2369–2386, New York, NY, USA, 2019. Association for Computing Machinery.

[40] H. B. McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, 2016.

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

[42] 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.

[43] Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks (extended abstract). In *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '91, page 51–59, New York, NY, USA, 1991. Association for Computing Machinery.

[44] Rahul Rachuri and Peter Scholl. Le mans: Dynamic and fluid mpc for dishonest majority. In *Advances in Cryptology – CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part I*, page 719–749, Berlin, Heidelberg, 2022. Springer-Verlag.

[45] Sashank Reddi, Zachary Burr Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Brendan McMahan, editors. *Adaptive Federated Optimization*, 2021.

[46] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[47] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv: Learning*, 2018.

[48] Jinhyun So, Ramy E. Ali, Basak Güler, and Amir Salman Avestimehr. Secure aggregation for buffered asynchronous federated learning. *CoRR*, abs/2110.02177, 2021.

[49] Jinhyun So, Başak Güler, and A. Salman Avestimehr. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. *IEEE Journal on Selected Areas in Information Theory*, 2(1):479–489, 2021.

[50] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In *Computer Security – ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I*, page 480–501, Berlin, Heidelberg, 2020. Springer-Verlag.

[51] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. Subsampled renyi differential privacy and analytical moments accountant. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1226–1235. PMLR, 16–18 Apr 2019.

[52] Chien-Sheng Yang, Jinhyun So, Chaoyang He, Songze Li, Qian Yu, and Salman Avestimehr. Lightsecagg: Rethinking secure aggregation in federated learning. *CoRR*, abs/2109.14236, 2021.

[53] Yizhou Zhao and Hua Sun. Information theoretic secure aggregation with user dropouts. In *2021 IEEE International Symposium on Information Theory (ISIT)*, pages 1124–1129, 2021.

[54] Yuqing Zhu and Yu-Xiang Wang. Poission subsampled rényi differential privacy. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7634–7642. PMLR, 09–15 Jun 2019.

Protocol 2 Client Gradient Processing

Input: Gradient $\boldsymbol{g}_i \in \mathbb{R}^d$.

Parameters: model dimension d, clipping threshold $c > 0$, granularity γ, modulus m, noise scale $\sigma > 0$ and bias $\beta \in [0, 1)$.

1. Clip and scale gradient: $\boldsymbol{g}_i' = \frac{1}{\gamma} \min\{1, \frac{c}{||\boldsymbol{g}_i||_2}\} \cdot \boldsymbol{g}_i \in \mathbb{R}^d$.
2. Flatten vector: $\boldsymbol{g}_i'' = U \cdot \boldsymbol{g}_i' \in \mathbb{R}^d$.
3. **Repeat**:
   (a) Let $\tilde{\boldsymbol{g}}_i \in \mathbb{Z}^d$ be a randomized rounding of $\boldsymbol{g}_i''$. i.e., $\tilde{\boldsymbol{g}}_i$ is a product distribution with $\mathbb{E}[\tilde{\boldsymbol{g}}_i] = \boldsymbol{g}_i''$ and $||\tilde{\boldsymbol{g}}_i - \boldsymbol{g}_i''||_\infty < 1$.

   until $|||\tilde{\boldsymbol{g}}_|||_2 \leq \min\{c/\gamma + \sqrt{d}, \sqrt{c^2/\gamma^2 + \frac{1}{4}d + \sqrt{2\log(1/\beta)} \cdot (c/\gamma + \frac{1}{2}\sqrt{d})}\}$.
4. **Output**: $\tilde{\boldsymbol{g}}_i$.

Protocol 3 Server Aggregate Noisy Release Value Processing

Input: Vector $\widehat{AX}_T$.

Parameters: model dimension d, clipping threshold $c > 0$, granularity γ, modulus m, noise scale $\sigma > 0$ and bias $\beta \in [0, 1)$.

1. Map $\mathbb{Z}_m$ to $\{1 - m/2, 2 - m/2, \ldots, -1, 0, 1, \ldots, m/2 - 1, m/2\}$ so that $\widehat{AX}_T$ is mapped to $\widehat{AX}_T' \in [-m/2, m/2]^d \cap \mathbb{Z}^d$ (and we have $\widehat{AX}_T' \mod m = \widehat{AX}_T$.

Output: $\gamma \cdot U^\intercal \widehat{AX}_T' \in \mathbb{R}^d$.

Supplementary Material

A Discretization Details of [33]

We use the randomized rounding strategy from [33] for discretization in Π_{PPFL}. At a high-level, each client first clips and scales their input gradient. Then, the clients flatten their gradient vectors using some unitary matrix $\boldsymbol{U}$ (intuitively, this minimizes the risk of modulo overlap in vector elements that are particularly large). Finally, the clients use a randomized process to round their gradient vectors in $\mathbb{R}^d$ to $\mathbb{Z}^d$. On the sever side, after receiving the aggregated, noise outputs $\hat{AX}_T$ in each round, the server unflattens the vector by applying $\boldsymbol{U}^T$ and then descales. Protocols 2 and 3 give more detail, but we refer the readers to [33] for full details on possible flattening matrices $\boldsymbol{U}$ and the randomized rounding procedure used.

To help with the analysis, [33] uses the following definitions to represent the conditional randomized rounding. We present them verabtim.

Definition 1 (Randomized Rounding). *Let $\gamma > 0$ and $d \in \mathbb{N}$. Define $R_\gamma : \mathbb{R}^d \to \gamma\mathbb{Z}^d$ (where $\gamma\mathbb{Z}^d := \{(\gamma z_1, \gamma z_2, \ldots, \gamma z_d) : z_1, \ldots, z_d \in \mathbb{Z}\} \subseteq \mathbb{R}^d$) as follows. For $x \in [0, \gamma]^d$, $R_\gamma(x)$ is a product distribution on $\{0, \gamma\}^d$ with mean x; that is, independently for each $i \in [d]$, we have $\Pr[R_\gamma(x)_i = 0] = 1 - x_i\gamma$ and $\Pr[R_\gamma(x)_i = \gamma] = x_i/\gamma$. In general, for $x \in \mathbb{R}^d$, we have $R_\gamma(x) = \gamma\lfloor x/\gamma \rfloor + R_\gamma(x - \gamma\lfloor x/\gamma \rfloor)$; here $\gamma\lfloor x/\gamma \rfloor \in \gamma\mathbb{Z}^d$ is the point x rounded down coordinate-wise to the grid.*

Definition 2 (Conditional Randomized Rounding). *Let $\gamma > 0$ and $d \in \mathbb{N}$ and $G \subseteq \mathbb{R}^d$. Define $R_\gamma^G : \mathbb{R}^d \to \gamma\mathbb{Z}^d \cap G$ to be R_γ conditioned on the hte output being in G. That is, $\Pr[R_\gamma^G(x) = y] = \Pr[R\gamma(x) = y]/\Pr[R_\gamma(x) \in G]$ for all $y \in \gamma\mathbb{Z}^d \cap G$, where R_γ is as in Definition 1.*

B Proofs for Section 5

Proof of Theorem 1

First we recall the notion of Rényi Divergences and Concentrated Differential Privacy [11, 20], as well as some other standard DP notions. We also define the Discrete Gaussian and provide its DP guarantees. See [33] for more details. Then we prove Thoerem 1

Definition 3 (Rényi Divergences)**.** *Let P and Q be probability distributions on some common domain Ω. Assume that P is absolutely continuous with respect to Q so that the Radon-Nikodym derivative $P(x)/Q(x)$ is well-defined for $x \in \Omega$.*

For $\alpha \in (1, \infty)$, we define the Rényi Divergence of order α of P with respect to Q as:

$$D_\alpha(P||Q) \coloneqq \frac{1}{\alpha - 1} \log \mathbb{E}_{X \leftarrow P}\left[\left(\frac{P(X)}{Q(x)}\right)^{\alpha-1}\right]$$

We also define

$$D_*(P||Q) \coloneqq \sup_{\alpha \in (1,\infty)} \frac{1}{\alpha} D_\alpha(P||Q)$$

Definition 4 (Concentrated Differential Privacy [11, 20])**.** *A randomized algorithm $M : \mathcal{X}^* \to \mathcal{Y}$ satisfies $\frac{1}{2}\varepsilon$-concentrated differential privacy iff, for all $x, x' \in \mathcal{X}$ differing by the addition or removal of a single user's records, we have $D_*(M(x)||M(x')) \leq \frac{1}{2}\varepsilon^2$.*

Definition 5 (Rényi Differential Privacy [41])**.** *A randomized algorithm $M : \mathcal{X}^* \to \mathcal{Y}$ satisfies (α, ε)-Rényi differential privacy iff, for all $x, x' \in \mathcal{X}$ differing by the addition or removal of a single user's records, we have $D_\alpha(M(x)||M(x')) \leq \frac{1}{2}\varepsilon^2$.*

Definition 6 (Differential Privacy [18])**.** *A randomized algorithm $M : \mathcal{X}^* \to \mathcal{Y}$ satisfies (ε, δ)-differential privacy iff, for all $x, x' \in \mathcal{X}$ differing by the addition or removal of a single user's records, we have*

$$\Pr[M(x) \in E] \leq e^\varepsilon \Pr[M(x') \in E] + \delta,$$

for all events $E \subset Y$. We refer to $(\varepsilon, 0)$-DP as pure DP and (ε, δ)-DP for $\delta > 0$ as approximate DP.

We remark that $\frac{1}{2}\varepsilon^2$-concentrated DP is equivalent to satisfying $(\alpha, \frac{1}{2}\varepsilon^2\alpha)$-Rényi DP simultaneously for all $\alpha \in (1, \infty)$. We also have the following conversion lemma from concentrated to approximate DP [5, 13, 3].

Lemma 1. *If M satisfies $(\varepsilon, 0)$-DP, then it satisfies $\frac{1}{2}\varepsilon^2$-concentrated DP. If M satisfies $\frac{1}{2}\varepsilon^2$-DP then, for any $\delta > 0$, M satisfies $(\varepsilon_{aDP}(\delta), \delta)$-DP, where*

$$\varepsilon_{aDP}(\delta) = \inf_{\alpha > 1} \frac{1}{2}\varepsilon^2\alpha + \frac{\log(1/\alpha\delta)}{\alpha - 1} + \log(1 - 1/\alpha) \leq \varepsilon \cdot (\sqrt{2\log(1/\delta)} + \varepsilon/2).$$

Discrete Gaussian Here we define the Discrete Gaussiasn [13] and give its DP guarantees.

Definition 7 (Discrete Gaussian)**.** *The discrete Gaussian with scale parameter $\sigma > 0$ and location parameter $\mu \in \mathbb{Z}$ is a probability distribution supported on the integers $\mathbb{Z}$ denoted by $\mathcal{N}_\mathbb{Z}(\mu, \sigma^2)$ and defined by*

$$\forall x \in \mathbb{Z} \qquad \Pr_{X \leftarrow \mathcal{N}_\mathbb{Z}(\mu,\sigma^2)}(X = x) = \frac{\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)}{\sum_{y \in \mathbb{Z}} \exp\left(\frac{-(y-\mu)^2}{2\sigma^2}\right)}.$$

Proposition 1 ([33], Proposition 14)**.** *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 $\alpha \in (1, \infty)$,*

$$\begin{aligned} D_\alpha(Z_n||Z_n + \Delta) \leq \min\{ & \frac{\alpha||\Delta||_2^2}{2n\sigma^2} + \tau d, \\ & \frac{\alpha}{2} \cdot \left(\frac{||\Delta||_2^2}{n\sigma^2} + 2\frac{||\Delta||_1}{\sqrt{n}\sigma} \cdot \tau + \tau^2 d\right), \\ & \frac{\alpha}{2} \cdot \left(\frac{||\Delta||_2}{\sqrt{n}\sigma} + \tau\sqrt{d}\right)^2 \} \end{aligned}$$

where $\tau := 10 \cdot \sum_{k=1}^{n} e^{-2\pi^2\sigma^2 \frac{k}{k+1}}$. *An algorithm* M *that adds* Z_n *to a query with* ℓ_p *sensitivity* Δ_p *satisfies* $\frac{1}{2}\varepsilon^2$*-concentrated DP for*

$$\varepsilon = \min\{\sqrt{\frac{||\Delta||_2^2}{n\sigma^2} + 2\tau d}, \\ \sqrt{\frac{\Delta_2^2}{n\sigma^2} + 2\frac{\Delta_1}{\sqrt{n}\sigma} \cdot \tau + \tau^2 d}, \\ \frac{\Delta_2}{\sqrt{n}\sigma} + \tau\sqrt{d}\}$$

Proof of Theorem 1

Proof. First, it is sufficient to show that the computation $\boldsymbol{CG} + \boldsymbol{Z}$ satisfies $\frac{1}{2}\varepsilon^2$-concentrated DP, due to the post processing property of DP. Now consider two datasets $\boldsymbol{G}$ and $\boldsymbol{H}$ differing in one data record according to participation schema Φ.[4] By assumption in the theorem statement, we have

$$\text{sens}^1_\Phi(\boldsymbol{C}) = \Delta, \quad \text{and thus} \quad \text{sens}_\Phi(\boldsymbol{C}) = c' \cdot \Delta,$$

where c' is the bound on the ℓ_2 norm of individual gradient vectors that are aggregated. Since we use the randomized rounding techniques from Section A, gradients that are clipped to ℓ_2 norm c can actually end up having ℓ_2 norm $c' = \hat{c}$ after rounding, where $\hat{c}$ is as in the theorem statement. With the bound on the total sensitivity above, we know from [33, Proposition 14] (reproduced above) that the computation is $\frac{1}{2}\varepsilon^2$-concentrated DP, with the ε from the theorem statement. □

Proof of Theorem 2

We first prove the following exact result for the error:

Theorem 3. *Let* $\beta \in [0, 1)$, $\sigma^2 \geq \gamma/2 > 0$, *and* $c > 0$. *Let* $n, d \in \mathbb{N}$ *and* $\rho \geq 1$. *Let* $\boldsymbol{g}_{T,i} \in \mathbb{R}^d$ *with* $||\boldsymbol{g}_{T,i}||_2 \leq c$ *for each* $T \in [T^*], i \in [n]$. *Let* $U \in \mathbb{R}^{d \times d}$ *be a random unitary matrix such that*

$$\forall \boldsymbol{x} \in \mathbb{R}^d \quad \forall i \in [d] \quad \forall t \in \mathbb{R} \quad \mathbb{E}[\exp(t(Ux)_i)] \leq \exp(t^2\rho||x||_2^2/2d).$$

Let

$$\Delta = \text{sens}^1_\Phi(\boldsymbol{C})$$

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

$$\hat{c}^2 = \min\left\{c^2 + \frac{1}{4}\gamma^2 d + \sqrt{2\log(1/\beta)} \cdot \gamma \cdot (c + \frac{1}{2}\gamma d), (c + \gamma\sqrt{d})^2\right\}$$

$$\varepsilon = \min\left\{\sqrt{\frac{\Delta^2\hat{c}^2}{n\sigma^2} + 2\tau d}, \frac{\Delta\hat{c}}{\sqrt{n}\sigma} + \tau\sqrt{d}\right\}.$$

Then Π_{PPFL} *satisfies* $\frac{1}{2}\varepsilon^2$*-concentrated differential privacy.*

Let

$$\hat{\sigma}^2(x) := \frac{\rho \cdot ||\boldsymbol{A}_{[T,:]}||_2^2}{d} \sum_{\tau=1}^{T} \sum_{i=1}^{n} ||\boldsymbol{g}_{\tau,i}||_2^2 + \left(\frac{\gamma^2 \cdot ||\boldsymbol{A}_{[T,:]}||_2^2}{4} + \sigma^2 \cdot ||\boldsymbol{B}_{[T,:]}||_2^2\right) \cdot n$$

$$\leq \frac{\rho||\boldsymbol{A}_{[T,:]}||_2^2}{d} c^2 nT + \left(\frac{\gamma^2 \cdot ||\boldsymbol{A}_{[T,:]}||_2^2}{4} + ||\boldsymbol{B}||_2^2 \cdot \sigma^2\right) \cdot n$$

[4] $\boldsymbol{G}$ and $\boldsymbol{H}$ really consist of entries that are sums of records.

If $\hat{\sigma}^2(x) \leq r^2$ *then*

$$\mathbb{E}\left[\left|\left|\Pi_{\mathsf{PPFL}}(x) - \boldsymbol{A}_{[T,:]}\left(\sum_{i=1}^{n} \boldsymbol{x}_i\right)\right|\right|_2^2\right] \leq \frac{dn}{1-\beta}\left(\frac{2\sqrt{2}\cdot r\cdot e^{-r^2/4\hat{\sigma}^2(x)}}{\sqrt{n(1-\beta)^{nT-1}}} + \left(||\boldsymbol{A}_{[T,:]}||_2^2\cdot\left(\frac{\gamma^2}{4} + \frac{\beta^2\cdot\gamma^2 n}{1-\beta}\right) + ||\boldsymbol{B}_{[T,:]}||_2^2\cdot\sigma^2\right)^{1/2}\right)^2.$$

We start with a modified version of Proposition 26 in [33].

Proposition 2. *Let* R_γ^G *be as in Definition 2 and* $G = \{y \in \mathbb{R}^d : ||y||_2^2 \leq \Delta^2\hat{c}^2\}$*. Let* $\Pi_{\mathsf{PPFL}}{}'(X)$ *be* Π_{PPFL} *up to the point of modular clipping. Consider the parameters from Theorem 3. Then* $\Pi_{\mathsf{PPFL}}{}'(X)$ *satisfies* $\frac{1}{2}\varepsilon^2$*-concentrated differential privacy. Also the following holds.*

$$\mathbb{E}\left[\left|\left|\Pi_{\mathsf{PPFL}}{}'(X) - \boldsymbol{A}_{[T,:]}\sum_{i=1}^{n}\boldsymbol{X}_i\right|\right|_2^2\right] \leq ||\boldsymbol{A}_{[T,:]}||_2^2\cdot\left(\frac{\gamma^2\cdot d\cdot n}{4(1-\beta)} + \left(\frac{\beta}{1-\beta}\gamma\sqrt{d}n\right)^2\right) + ||\boldsymbol{B}_{[T,:]}||_2^2\cdot n\cdot d\cdot\sigma^2.$$

$$\forall \boldsymbol{t}\in\mathbb{R}^d \quad \mathbb{E}\left[\exp\left(\left\langle \boldsymbol{t}, \Pi_{\mathsf{PPFL}}{}'(X) - \boldsymbol{A}_{[T,:]}\sum_{i=1}^{n}\boldsymbol{X}_i\right\rangle\right)\right] \leq \frac{\exp((\frac{\gamma^2\cdot||\boldsymbol{A}_{[T,:]}||_2^2}{8} + \frac{\sigma^2\cdot||\boldsymbol{B}_{[T,:]}||_2^2}{2})\cdot||\boldsymbol{t}||_2^2\cdot n)}{(1-\beta)^{nT}}.$$

Proof. First, the differential privacy claim follows from [33, Proposition 14].

Now, for the utility analysis, we have

$$\begin{aligned}
\mathbb{E}\left[\left|\left|\Pi_{\mathsf{PPFL}}{}'(X) - \boldsymbol{A}_{[T,:]}\sum_{i=1}^{n}\boldsymbol{X}_i\right|\right|_2^2\right] &= \mathbb{E}\left[\left|\left|\sum_{\tau=1}^{T}\boldsymbol{A}_{T,\tau}\cdot\left(\sum_{i=1}^{n}(R_\gamma^G(\boldsymbol{g}_{\tau,i}) - \boldsymbol{g}_{\tau,i})\right) + \boldsymbol{B}_{T,\tau}\cdot\sum_{i=1}^{n}\gamma\cdot\boldsymbol{z}_{\tau,i}\right|\right|_2^2\right] \\
&\leq \sum_{\tau=1}^{T}\boldsymbol{A}_{T,\tau}^2\cdot\mathbb{E}\left[\left|\left|\sum_{i=1}^{n}R_\gamma^G(\boldsymbol{g}_{\tau,i}) - \boldsymbol{g}_{\tau,i}\right|\right|_2^2\right] + \boldsymbol{B}_{T,\tau}^2\cdot n\cdot\sigma^2 \\
&\leq \left\|\boldsymbol{A}_{[T,:]}\right\|_2^2\cdot\left(\frac{\gamma^2\cdot d\cdot n}{4(1-\beta)} + \left(\frac{\beta}{1-\beta}\gamma\sqrt{d}n\right)^2\right) + \left\|\boldsymbol{B}_{[T,:]}\right\|_2^2\cdot n\cdot\sigma^2,
\end{aligned}$$

where the last inequality is due directly to Proposition 26 of [33].

Now, for each $i\in[n], \tau\in[T]$, we have that $R_\gamma(\boldsymbol{g}_{\tau,i}) \in \gamma\lfloor\boldsymbol{g}_{\tau,i}/\gamma\rfloor + \{0,\gamma\}^d$ and is a product distribution with mean $\boldsymbol{g}_{\tau,i}$. Thus, $R_\gamma(\boldsymbol{g}_{\tau,i}) - \boldsymbol{g}_{\tau,i} \in \{0,\gamma\}^d$ and is a product distribution with mean $\mathbf{0}$. Therefore, by Hoeffding's lemma, we have:

$$\forall \boldsymbol{t}\in\mathbb{R}^d \quad \mathbb{E}[\exp(\langle \boldsymbol{t}, \sum_{\tau=1}^{T}\boldsymbol{A}_{T,\tau}\sum_{i=1}^{n}R_\gamma(\boldsymbol{g}_{\tau,i}) - \boldsymbol{g}_{\tau,i}\rangle)] \leq \exp(\frac{\gamma^2}{8}\cdot n\cdot||\boldsymbol{A}_{[T,:]}||_2^2\cdot||\boldsymbol{t}||_2^2).$$

Thus,

$$\begin{aligned}
\forall \boldsymbol{t}\in\mathbb{R}^d \quad \mathbb{E}[\exp(\langle \boldsymbol{t}, \sum_{\tau=1}^{T}\boldsymbol{A}_{T,\tau}\sum_{i=1}^{n}R_\gamma^G(\boldsymbol{g}_{\tau,i}) - \boldsymbol{g}_{\tau,i}\rangle)] &\leq \frac{\mathbb{E}[\exp(\langle \boldsymbol{t}, \sum_{\tau=1}^{T}\boldsymbol{A}_{T,\tau}\sum_{i=1}^{n}R_\gamma(\boldsymbol{g}_{\tau,i}) - \boldsymbol{g}_{\tau,i}\rangle)]}{\Pr[R_\gamma(\boldsymbol{g}_{\tau,i})\in G\ \forall\tau, i]} \\
&\leq \frac{\exp(\frac{\gamma^2}{8}\cdot n\cdot||\boldsymbol{A}_{[T,:]}||_2^2\cdot||\boldsymbol{t}||_2^2)}{(1-\beta)^{nT}}.
\end{aligned}$$

Moreover, we have that [13]:

$$\forall \boldsymbol{t}\in\mathbb{R}^d \quad \mathbb{E}[\exp(\langle \boldsymbol{t}, \sum_{\tau=1}^{T}\boldsymbol{B}_{T,\tau}\sum_{i=1}^{n}\gamma\cdot\boldsymbol{z}_{\tau,i}\rangle)] \leq \exp(\frac{\sigma^2}{2}\cdot n\cdot||\boldsymbol{B}_{[T:,]}||_2^2\cdot||\boldsymbol{t}||_2^2).$$

□

Finally, we are able to prove a modified version of Theorem 36 from [33].

Proof of Theorem 3. First, the differential privacy follows from Proposition 2 and the post-processing property of DP.

Now, for the utility, by assumption, we have that

$$\forall \boldsymbol{x} \in \mathbb{R}^d \; \forall j \in [d] \; \forall t \in \mathbb{R} \quad \mathbb{E}[\exp(t(\boldsymbol{U}x)_j)] \leq \exp(t^2 \rho ||\boldsymbol{x}||_2^2/2d).$$

Therefore,

$$\begin{aligned}\mathbb{E}[\exp(t \cdot (\sum_{\tau=1}^{T} \boldsymbol{A}_{T,\tau} \cdot (\boldsymbol{U} \sum_{i=1}^{n} \boldsymbol{g}_{\tau,i})_j)] &= \prod_{\tau=1}^{T} \cdot \prod_{i=1}^{n} \mathbb{E}[\exp(t \cdot \boldsymbol{A}_{T,\tau} \cdot (\boldsymbol{U}\boldsymbol{g}_{\tau,i})_j)] \\ &\leq \prod_{\tau=1}^{T} \cdot \prod_{i=1}^{n} \exp(t^2 \cdot \boldsymbol{A}_{T,\tau}^2 \cdot \rho \cdot ||\boldsymbol{g}_{\tau,i}||_2^2/2d) \\ &= \exp(t^2 \cdot ||\boldsymbol{A}_{[T,:]}||_2^2 \cdot \rho \cdot \sum_{\tau=1}^{T} \sum_{i=1}^{n} ||\boldsymbol{g}_{\tau,i}||_2^2/2d).\end{aligned}$$

Combining with the result of Proposition 2, we have

$$\begin{aligned}\forall t \in \mathbb{R} \; \forall j \in [d] \quad \mathbb{E}[\exp(t \cdot (\mathcal{A}(\boldsymbol{U}x))_j)] &\leq \exp(\frac{t^2 \cdot ||\boldsymbol{A}_{[T,:]}||_2^2 \cdot \rho}{2d} \cdot \sum_{\tau=1}^{T} \sum_{i=1}^{n} ||\boldsymbol{g}_{\tau,i}||_2^2) \\ &\cdot \frac{\exp((\frac{\gamma^2 \cdot ||\boldsymbol{A}_{[T,:]}||_2^2}{8} + \frac{\sigma^2 \cdot ||\boldsymbol{B}_{[T,:]}||_2^2}{2}) \cdot t^2 \cdot n)}{(1-\beta)^{nT}}\end{aligned}$$

Recall $\hat{\sigma}^2(x) = \frac{\rho \cdot ||\boldsymbol{A}_{[T,:]}||_2^2}{d} \sum_{\tau=1}^{T} \sum_{i=1}^{n} ||\boldsymbol{g}_{\tau,i}||_2^2 + (\frac{\gamma^2 \cdot ||\boldsymbol{A}_{[T,:]}||_2^2}{4} + \sigma^2 \cdot ||\boldsymbol{B}_{[T,:]}||_2^2) \cdot n.$
By Proposition 35 of [33], for all $j \in [d]$,

$$\mathbb{E}[(M_{[a,b]}(\Pi_{\mathsf{PPFL}}'(\boldsymbol{U}x))_j - \Pi_{\mathsf{PPFL}}'(\boldsymbol{U}x)_j)^2] \leq (b-a)^2 \cdot \frac{1}{(1-\beta)^{nT}} \cdot e^{-(b-a)^2/8\hat{\sigma}^2(x)} \cdot (e^{\frac{a^2-b^2}{4\hat{\sigma}^2}} + e^{\frac{b^2-a^2}{4\hat{\sigma}^2}}),$$

where $a = -r$ and $b = r$ here. Summing over $j \in [d]$ gives

$$\mathbb{E}[||M_{[-r,r]}(\Pi_{\mathsf{PPFL}}'(\boldsymbol{U}x)) - \Pi_{\mathsf{PPFL}}'(\boldsymbol{U}x)||_2^2] \leq 4r^2 \cdot \frac{d}{(1-\beta)^{nT}} \cdot e^{-r^2/2\hat{\sigma}^2(x)} \cdot 2$$

Continuing with the proof from [33], we get:

$$\begin{aligned}&\mathbb{E}[||\Pi_{\mathsf{PPFL}}(x) - \boldsymbol{A}_{[T,:]} \sum_{i=1} \boldsymbol{X}_i||_2^2] \\ &\leq \left((8r^2 \cdot \frac{d}{(1-\beta)^{nT}} \cdot e^{-r^2/2\hat{\sigma}^2(x)})^{1/2} + \left(||\boldsymbol{A}_{[T,:]}||_2^2 \cdot \left(\frac{\gamma^2 \cdot d \cdot n}{4(1-\beta)} + \left(\frac{\beta}{1-\beta} \gamma \sqrt{d} n \right)^2 \right) + \right.\right. \\ &\left.\left. ||\boldsymbol{B}_{[T,:]}||_2^2 \cdot n \cdot d \cdot \sigma^2 \right)^{1/2} \right)^2 \\ &= \frac{dn}{1-\beta} \left(\frac{2\sqrt{2} \cdot r \cdot e^{-r^2/4\hat{\sigma}^2(x)}}{\sqrt{n(1-\beta)^{nT-1}}} + \left(||\boldsymbol{A}_{[T,:]}||_2^2 \cdot \left(\frac{\gamma^2}{4} + \frac{\beta^2 \cdot \gamma^2 n}{1-\beta} \right) + ||\boldsymbol{B}_{[T,:]}||_2^2 \cdot \sigma^2 \right)^{1/2} \right)^2.\end{aligned}$$

□

With this error bound, assuming that $\beta \leq 1/\sqrt{n}$ and $\hat{\sigma}^2(x) \leq r^2/4\log(r\sqrt{n}/\gamma^2)$, we get

$$\mathbb{E}[||\tilde{\mathcal{A}}(x) - \boldsymbol{A}_{[T,:]} \sum_{i=1} \boldsymbol{X}_i||_2^2] \leq O(dn((||\boldsymbol{A}_{[T,:]}||_2^2 \cdot \gamma^2 + ||\boldsymbol{B}_{[T,:]}||_2^2 \cdot \sigma^2)).$$

Proof of Theorem 2. Note that $r = \frac{1}{2}\gamma m$. We verify that setting the parameters as specified yields $\frac{1}{2}\varepsilon^2$-concentrated DP and the desired accuracy. First, we have that

$$\varepsilon^2 \leq \frac{\Delta^2 \hat{c}^2}{n\sigma^2} + 2\tau d \leq \frac{\Delta^2(c+\gamma\sqrt{d})^2}{n\sigma^2} + 20nde^{-\pi^2(\sigma/\gamma)^2} \leq \frac{2\Delta^2 c^2}{n\sigma^2} + \frac{2d\Delta^2}{n(\sigma/\gamma)^2} + 20nde^{-\pi^2(\sigma/\gamma)^2}.$$

Thus the privacy requirement is satisfied as long as $\sigma \geq 2c\Delta/\varepsilon\sqrt{n}$ and $(\sigma/\gamma)^2 \geq 8d\Delta^2/\varepsilon^2 n$, and $20nde^{-\pi^2(\sigma/\gamma)^2} \leq \varepsilon^2/4$. So we can set

$$\sigma = \max\{\frac{2c\Delta}{\varepsilon\sqrt{n}}, \frac{\gamma\Delta\sqrt{8d}}{\varepsilon\sqrt{n}}, \frac{\gamma}{\pi^2}\log(\frac{80nd}{\varepsilon^2})\} = \tilde{\Theta}(\frac{c\Delta}{\varepsilon\sqrt{n}} + \sqrt{\frac{d}{n}} \cdot \frac{\gamma\Delta}{\varepsilon} + \gamma\log(\frac{nd}{\varepsilon^2}).$$

We set $\beta = \min\{1/n, 1/2\} = \Theta(\frac{1}{n})$.

Next,

$$\begin{aligned}
\hat{\sigma}^2 &\leq \frac{\rho||\boldsymbol{A}_{[T,:]}||_2^2}{d}c^2 nT + (\frac{\gamma^2||\boldsymbol{A}_{[T,:]}||_2^2}{4} + \sigma^2||\boldsymbol{B}_{[T,:]}||_2^2) \cdot n \\
&\leq \frac{\rho||\boldsymbol{A}_{[T,:]}||_2^2}{d}c^2 nT + \gamma^2||\boldsymbol{A}_{[T,:]}||_2^2 n + \sigma^2||\boldsymbol{B}_{[T,:]}||_2^2 \cdot n \\
&\leq O(\frac{\rho||\boldsymbol{A}_{[T,:]}||_2^2}{d}c^2 nT + \gamma^2||\boldsymbol{A}_{[T,:]}||_2^2 n + ||\boldsymbol{B}_{[T,:]}||_2^2(\frac{c^2\Delta^2}{\varepsilon^2} + \frac{\gamma^2 d\Delta}{\varepsilon^2} + \gamma^2 n\log^2(\frac{nd}{\varepsilon^2})) \\
&\leq O(\frac{\rho||\boldsymbol{A}_{[T,:]}||_2^2}{d}c^2 nT + ||\boldsymbol{B}_{[T,:]}||_2^2\frac{c^2\Delta^2}{\varepsilon^2})) + \gamma^2 \cdot O(||\boldsymbol{A}_{[T,:]}||_2^2 n + ||\boldsymbol{B}_{[T,:]}||_2^2(\frac{d\Delta}{\varepsilon^2} + n\log^2(\frac{nd}{\varepsilon^2})).
\end{aligned}$$

Now we work out the asymptotics of the accuracy guarantee:

$$\begin{aligned}
&\mathbb{E}[||\Pi_{\mathsf{PPFL}}(X) - \boldsymbol{A}_{[T,:]}\sum_{i=1}\boldsymbol{X}_i||_2^2] \\
&\leq \frac{dn}{1-\beta}\left(\frac{2\sqrt{2}\cdot r\cdot e^{-r^2/4\hat{\sigma}^2(x)}}{\sqrt{n(1-\beta)^{nT-1}}} + \left(||\boldsymbol{A}_{[T,:]}||_2^2 \cdot \left(\frac{\gamma^2}{4} + \frac{\beta^2\cdot\gamma^2 n}{1-\beta}\right) + ||\boldsymbol{B}_{[T,:]}||_2^2\cdot\sigma^2\right)^{1/2}\right)^2. \\
&\leq O(nd(\frac{re^{-r^2/4\hat{\sigma}^2}}{\sqrt{n}} + \sqrt{||\boldsymbol{A}_{[T:,]}||_2^2\gamma^2 + ||\boldsymbol{B}_{[T:,]}||_2^2\sigma^2})) \\
&\leq O(nd(\frac{r^2e^{-r^2/2\hat{\sigma}^2}}{n} + ||\boldsymbol{A}_{[T:,]}||_2^2\gamma^2 + ||\boldsymbol{B}_{[T:,]}||_2^2\sigma^2)) \\
&\leq O(nd(\frac{\gamma^2 m^2}{n}\exp(\frac{-\gamma^2 m^2}{8\hat{\sigma}^2}) + ||\boldsymbol{A}_{[T:,]}||_2^2\gamma^2 + ||\boldsymbol{B}_{[T:,]}||_2^2(\frac{c^2\Delta^2}{\varepsilon^2 n} + \frac{d\gamma^2\Delta^2}{\varepsilon^2 n} + \gamma^2\log^2(\frac{nd}{\varepsilon^2})))) \\
&\leq O(||\boldsymbol{B}_{[T:,]}||_2^2\frac{c^2\Delta^2 d}{\varepsilon^2} + \gamma^2 nd(\frac{m^2}{n}\exp(\frac{-\gamma^2 m^2}{8\hat{\sigma}^2}) + ||\boldsymbol{A}_{[T:,]}||_2^2 + ||\boldsymbol{B}_{[T:,]}||_2^2(\frac{d\Delta^2}{\varepsilon^2 n} + \log^2(\frac{nd}{\varepsilon^2}))))
\end{aligned}$$

Similarly to the analysis of Theorem 2 in [33], if

$$\begin{aligned}
m^2 &\geq O((||\boldsymbol{A}_{[\boldsymbol{T}:,]}||_2^2 n + ||\boldsymbol{B}_{[\boldsymbol{T}:,]}||_2^2(\frac{d\Delta}{\varepsilon^2} + n\log^2(\frac{nd}{\varepsilon^2}))) \cdot \log(1+m^2/n) \\
&= \tilde{O}(||\boldsymbol{A}_{[\boldsymbol{T}:,]}||_2^2 n + ||\boldsymbol{B}_{[\boldsymbol{T}:,]}||_2^2(\frac{d\Delta}{\varepsilon^2} + n)),
\end{aligned}$$

then we can set

$$\gamma^2 = O(\frac{\rho||\boldsymbol{A}_{[T:,]}||_2^2 c^2 nT}{d} + \frac{||\boldsymbol{B}_{[T:,]}||_2^2 c^2\Delta^2}{\varepsilon^2}) \cdot \frac{\log(1+m^2/n)}{m^2}$$

so that $\frac{m^2}{n}\exp(\frac{-\gamma^2 m^2}{8\hat{\sigma}}^2) \leq 1$.

This gives us,

$$\begin{aligned}
&\mathbb{E}[||\tilde{\mathcal{A}}(x) - \boldsymbol{A}_{[T,:]} \sum_{i=1} \boldsymbol{X}_i||_2^2] \\
&\leq O(||\boldsymbol{B}_{[T:,]}||_2^2 \frac{c^2\Delta^2 d}{\varepsilon^2} + \gamma^2 nd(1 + ||\boldsymbol{A}_{[T:,]}||_2^2 + ||\boldsymbol{B}_{[T:,]}||_2^2(\frac{d\Delta^2}{\varepsilon^2 n} + \log^2(\frac{nd}{\varepsilon^2})))) \\
&\leq O(||\boldsymbol{B}_{[T:,]}||_2^2 \frac{c^2\Delta^2 d}{\varepsilon^2} + (\frac{\rho||\boldsymbol{A}_{[T:,]}||_2^2 c^2 nT}{d} + \frac{||\boldsymbol{B}_{[T:,]}||_2^2 c^2\Delta^2}{\varepsilon^2}) \cdot \\
&\frac{\log(1+m^2/n)}{m^2} nd(1 + ||\boldsymbol{A}_{[T:,]}||_2^2 + ||\boldsymbol{B}_{[T:,]}||_2^2(\frac{d\Delta^2}{\varepsilon^2 n} + \log^2(\frac{nd}{\varepsilon^2})))) \\
&\leq O(||\boldsymbol{B}_{[T:,]}||_2^2 \frac{c^2\Delta^2 d}{\varepsilon^2} + ||\boldsymbol{B}_{[T:,]}||_2^2 \frac{c^2\Delta^2 d}{\varepsilon^2}(\frac{\log(1+m^2/n)}{m^2} n \cdot (\rho||\boldsymbol{A}_{[T:,]}||_2^2 T + \\
&1 + ||\boldsymbol{A}_{[T:,]}||_2^2 + ||\boldsymbol{B}_{[T:,]}||_2^2(\frac{d\Delta^2}{\varepsilon^2 n} + \log^2(\frac{nd}{\varepsilon^2}))))) \\
&\leq O(||\boldsymbol{B}_{[T:,]}||_2^2 \frac{c^2\Delta^2 d}{\varepsilon^2}(1 + \frac{\log(1+m^2/n)}{m^2} n \\
&\cdot (\rho||\boldsymbol{A}_{[T:,]}||_2^2 T + 1 + ||\boldsymbol{A}_{[T:,]}||_2^2 + ||\boldsymbol{B}_{[T:,]}||_2^2(\frac{d\Delta^2}{\varepsilon^2 n} + \log^2(\frac{nd}{\varepsilon^2}))))).
\end{aligned}$$

So, if

$$\begin{aligned}
m^2 &\geq O(\log(1+m^2/n) n \cdot (\rho||\boldsymbol{A}_{[T:,]}||_2^2 T + 1 + ||\boldsymbol{A}_{[T:,]}||_2^2 + ||\boldsymbol{B}_{[T:,]}||_2^2(\frac{d\Delta^2}{\varepsilon^2 n} + \log^2(\frac{nd}{\varepsilon^2})))) \\
&= \tilde{O}(\rho||\boldsymbol{A}_{[T:,]}||_2^2 nT + ||\boldsymbol{B}_{[T:,]}||_2^2 \frac{d\Delta^2}{\varepsilon^2}),
\end{aligned}$$

then the mean squared error is $O(||\boldsymbol{B}_{[T:,]}||_2^2 \frac{c^2\Delta^2 d}{\varepsilon^2})$, as required. The final bound is obtained by simply summing the above over each round from $T = 1$ to $T = T^*$. $\square$

C Resharing Security Model and Proof

Security proofs

We first provide an intuition on the current analysis for proving the security of cryptographic protocols. In the security proof, we compare between an n-party function $f(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$ and a protocol $P(x_1, \ldots, x_n)$ that allegedly privately computes the function f. Intuitively, a protocol P correctly and privately computes f if the following hold: (a) *Correctness:* For every input $\vec{x} = (x_1, \ldots, x_n)$, the output of the parties at the end of the protocol interaction P is the same as $f(\vec{x})$; (b) *Privacy*: There exists a simulator $\mathcal{S}$ that receives the input and output of the corrupted parties, and can efficiently generate the messages that the corrupted parties received during the protocol execution. The simulator does not know the input/outputs of the honest parties. Intuitively, the fact that the messages sent by the honest parties can be generated from the input/output of the corrupted parties implies that these messages do not contain any additional information about the inputs of the honest parties besides what is revealed from the output of the computation.

Security Model

We now introduce the formal security model. We first note that we consider robustness checks on inputs out of the scope of our security model; i.e., we do not cover *poisoning attacks*,which have been extensively studied in the literature, e.g., [50, 23]. Indeed, it is the case that malicious parties can input to the protocol whatever they want as their gradients and noise $\boldsymbol{x}, \boldsymbol{z}$, which can lead to a meaningless model.

We follow the standard real/ideal world security paradigm of [28]. Consider some multi-party protocol Π that is executed by some parties $P_1, \ldots, P_N$ that are grouped into committees $\mathcal{C}_1, \ldots, \mathcal{C}_{T^*}$ from round 1 to round T^* and a server S. Note: the committees can be arbitrarily chosen, but our protocol only provides security if the assumption that the number of parties $\mathcal{A}$ corrupts is at most t holds;

in other words, we abstract out the committee selection process.[5] Each of these parties has inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, and they want to evaluate some given *functionality* $\mathcal{F}$. In our case, the functionality $\mathcal{F}_{\mathsf{PPFL}}$ is resharing the inputs from all previous committees to the next committee, in each round, and then outputting the $\widehat{AX}_T$ value to the sever in each round T, given some factorization $\boldsymbol{A} = \boldsymbol{BC}$. The security of protocol Π is defined by comparing the real-world execution of the protocol with an *ideal*-world evaluation of $\mathcal{F}$ by a trusted party (ideal functionality), who receives the inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ from the parties in the clear and simply sends the relevant parties their outputs $\mathcal{F}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ periodically. There is an adversary $\mathcal{A}$ that chooses to corrupt at most $t < N$ of the parties $P_1, \ldots, P_N$. This adversary $\mathcal{A}$ sees all of the messages and inputs and outputs of the corrupted parties and is allowed to act arbitrarily on their behalf. We also assume that the server is corrupted and thus $\mathcal{A}$ can see all of the messages sent to the server and all of its outputs. Informally, it is required that for every adversary that corrupts some parties during the protocol execution, there is an adversary $\mathcal{S}$, also referred to as the *simulator*, which can achieve the same effect and learn the same information in the ideal-world. This simulator only sees what the corrupted parties send to the honest parties and the output $\boldsymbol{y}$ vectors, not the inputs $\boldsymbol{x}$ of the honest parties.We now formally describe the security definition.

Real Execution. In the real execution, Π is executed in the presence of the adversary $\mathcal{A}$. The *view* of a party P during an execution of Π, denoted by View_P^Π consists of the messages P receives from the other parties during the execution and P's input. The execution of Π in the presence of $\mathcal{A}$ on inputs $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ denoted $\mathsf{Real}_{\Pi,\mathcal{A}}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ is defined as $\{\mathsf{View}_P^\Pi\}_{P\in\mathfrak{C}}$. The output of Π in the presence of $\mathcal{A}$ on inputs $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ is noted as Output.

Ideal Execution. In the ideal execution, the parties and an ideal world adversary $\mathcal{S}$ interact with a trusted party (ideal functionality). The ideal execution proceeds as follows: As a committee $\mathcal{C}_T$ comes online, the parties $P_{T,1}, \ldots, P_{T,n}$ in that committee send their inputs $\boldsymbol{x}_{T,1}, \ldots, \boldsymbol{x}_{T,n}$ to the trusted party, who computes the output $\mathcal{F}(\boldsymbol{x}_{1,1}, \ldots, \boldsymbol{x}_{T,n})$ to the server for that round. $\mathcal{S}$ is also allowed to release a vector $\boldsymbol{\chi}$, which will be added to the output, to simulate additive attacks.

Definition 8. *Protocol Π securely computes $\mathcal{F}$ if for every adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that*

$$\mathsf{SD}((\{\mathsf{View}_P^\Pi\}_{P\in\mathfrak{C}}, \mathsf{Output}), (\mathcal{S}(\{\boldsymbol{x}_{T^*,j}\}_{T,j\in\mathfrak{C}(T)}, \mathcal{F}(\boldsymbol{x}_{1,1}, \ldots, \boldsymbol{x}_{T^*,n}), \mathcal{F}(\boldsymbol{x}_{1,1}, \ldots, \boldsymbol{x}_{T^*,n}) + \boldsymbol{\chi})) \leq \mathsf{negl}(\lambda),$$ [6]

where SD is the statistical distance between the two distributions and $\mathfrak{C}(T)$ is the set of corrupted parties in round T.

Additional Protocol Details for Active Security

Before proving the security of our protocol, we provide additional details that are needed for an adversary that is allowed to act arbitrarily on behalf of the corrupted parties, or an *active* adversary. For active security, our protocol relies on four main techniques/properties:

1. *More on Packed Secret Sharing*: We first give a property of packed secret sharing relevant to active security that we omitted from Section 2. A packed secret sharing $[\boldsymbol{z}]$ is actually equivalent to a Reed-Solomon Encoding [46] of the underlying secret $\boldsymbol{z}$. This means that packed secret sharings inherit the *error-detection* property of Reed Solomon codes. Indeed, writing $n = w + t_c + k$, if $w \geq t_c$, and at most t_c of the shares are changed before one attempts to use them to reconstruct the underlying secret $\boldsymbol{z}$, then either the reconstruction succeeds, or the reconstructor knows that at least one of the shares was tampered with.
2. *Parity Check Matrices*: Now, we have the following, which is essentially the check that the reconstructor performs to see if any of the shares were tampered with. Let $\boldsymbol{H} \in \mathbb{F}^{(n-t_c-k)\times n}$ be the *parity check matrix* of the Reed Solomon code such that $\boldsymbol{H} \cdot \boldsymbol{z} = 0$ if and only if $\boldsymbol{z} \in \mathbb{F}^n$ is a valid codeword. This matrix intuitively takes the first $t_c + k$ shares in $\boldsymbol{z}$, computes what the other $n - (t_c + k)$ shares should be (which can be done with Reed Solomon codes), and compares them to those that are actually in $\boldsymbol{z}$.

[5] In practice, the committee selection is done by the server.

[6] $\mathsf{negl}(\lambda)$ is any function in $\lambda^{\omega(1)}$

3. *Commitments*: Commitments are a two-stage protocol where first a party P_i commits to some value x by using $c \leftarrow \mathsf{Comm}(x)$ and sending c to the other parties. The important property is that $\mathsf{Comm}(x)$ *hides* x from the other parties. Next, the party P_i can open c by using $o \leftarrow \mathsf{Open}(c, x)$ and sending (o, x) to the other parties. The important property is that P_i cannot convince the other parties that it committed to another value $x' \neq x$ in its original commitment c. There are several well-known constructions of commitments.
4. *Random Linear Combinations*: If $\beta \in \mathbb{F}$ is random and unknown to all, then to check that some secret sharings $\mathsf{Share}(\boldsymbol{\delta}_j)$ for $j \in [n-d-1]$ each share $\mathbf{0}$, we can compute and reconstruct $\mathsf{Share}(\boldsymbol{\delta}_j) \leftarrow \sum_{j=1}^{n-d-1} \beta^j \cdot \mathsf{Share}(\boldsymbol{\delta}_j)$, then check that the reconstructed value is $\mathbf{0}$. Intuitively, we are evaluating the polynomial defined by the $\boldsymbol{\delta}_j$ on random point β. So if some $\boldsymbol{\delta}_j \neq \mathbf{0}$, then by the Schwartz-Zippel Lemma, the reconstructed value will be non-zero with high probability.

With these tools in hand, we can describe the modifications to our passively-secure protocol above, to make it actively secure. After committee $\mathcal{C}_{T+1}$ receives the re-shared $([\boldsymbol{Z}^1_{[1,k]}], \ldots, [\boldsymbol{Z}^n_{[1,k]}])$ from each P_i in committee $\mathcal{C}_T$, each party P_j in committee $\mathcal{C}_{T+1}$ samples random β_j, sends $c \leftarrow \mathsf{Comm}(\beta_j)$ to the other parties of committee $\mathcal{C}_{T+1}$ and finally opens β_j to the other parties. The parties of $\mathcal{C}_{T+1}$ then agree on the m parties from $\mathcal{C}_T$ that actually sent them reshared values[7] and compute

$$([\boldsymbol{y}_1], \ldots, [\boldsymbol{y}_{m-t_c-k}]) \leftarrow \boldsymbol{H} \cdot ([\boldsymbol{Z}^1_{[1,k]}], \ldots, [\boldsymbol{Z}^n_{[1,k]}]).$$

Note that since the secret sharing is linear, by the properties of parity check matrices above, the shared $\boldsymbol{y}_l$ will be equal to $\mathbf{0}$ if and only if the underlying shares of the $\boldsymbol{Z}^i_1, \ldots \boldsymbol{Z}^i_k$ correspond to valid codewords and thus shares that were not tampered with. Finally, the parties compute

$$[\boldsymbol{y}] \leftarrow \sum_{l=1}^{d(m-t_c-k)/(4\mu^2 n^2)} \beta^l \cdot [\boldsymbol{y}_l],$$

then reconstruct it to the server who check if the reconstructed value is $\mathbf{0}$, and aborts if not. Otherwise, they abort.

Security Intuition Let t_{c_1} be the number of corrupted parties in committee $\mathcal{C}_T$ that do not send to enough parties in $\mathcal{C}_{T+1}$ and $m = n - t_d - t_{c_1}$ be the number of parties from committee $\mathcal{C}_T$ that do *not* drop out (including those corrupted parties that do not send to enough parties). Writing $m = w + t_c + k$, we have that $w = m - t_c - k = n - t_d - t_{c_1} - ((1/2 + \mu)n) = (1/2 - \mu)n - t_d - t_{c_1} > t_{c_2}$, where t_{c_2} is the number of corrupted parties that do send to enough parties in $\mathcal{C}_{T+1}$, and thus $t_{c_1} + t_{c_2} = t_c$. The last inequality holds, since we assume that $t_d + t_c < (1/2 - \mu)n$. This means that if the corrupted parties from committee $\mathcal{C}_T$ that do send to enough parties, do not reshare their actual shares to committee $\mathcal{C}_{T+1}$, then the parity check sharing will not share $\boldsymbol{y}_i = \mathbf{0}$. This is because the number of honest parties who do not drop out is at least $t_c + k$ and thus their shares completely define the correct codeword and so if the corrupted parties' shares do not match with this codeword, it will be reflected.Using similar logic, the server in round $\mathcal{C}_{T+1}$ will be able to either successfully reconstruct the parity check sharing, or otherwise detect malicious behavior during the reconstruction.

Added Communication Complexity Note that most of the updates to achieve active security are done *locally*. The only added communication is for committing to and opening the randomness β_i, then reconstructing the $\boldsymbol{y}$. Moreover, if we use the passively-secure protocol many times in parallel, then we can use the same β to take the random linear combination across all such instances. Thus the total communication complexity of the actively secure protocol is marginally changed with respect to the passively secure protocol, as long as if enough instances of the passive protocol are used at the same time.

Security Proof

Theorem 4 (Security)**.** Π_{PPFL} *securely computes* $\mathcal{F}_{\mathsf{PPFL}}$ *with functionalities* $\mathcal{F}_{\mathsf{SecAgg}}$ *and* $\mathcal{F}_{\mathsf{Comm}}$.

[7]This can be done by each party sending to the other parties those identities from which they received reshared values, then including an identity if at least $n - t_c$ parties said they received from that identity.

Proof. We first build the simulator $\mathcal{S}$. We first note that we model the SecAgg protocol as a trusted functionality $\mathcal{F}_{\mathsf{SecAgg}}$ which takes inputs $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m$ from some parties via SecAgg.Enc and outputs their sum $\sum_{i=1}^{m} \boldsymbol{a}_i$ to the server S via SecAgg.Dec. We also model commitments as a trusted Functionality $\mathcal{F}_{\mathsf{Comm}}$ that in the first stage takes in x from P_i and then does not reveal x to the other parties until the next stage. Indeed, the simulator emulates these trusted functionalities and thus can see whatever the corrupted parties input to them.

We describe the simulator for the first rounds $T = 1$ and then inductively for the rest. Throughout, we will (inductively) show that the simulator knows all of the corrupted parties' shares. We start with the case of a corrupted server S.

Corrupted Server In round 1, $\mathcal{S}$ simulates the shares sent by honest parties of round 1 to corrupted parties of round 2 by sampling random values from the field $\mathbb{F}$. In round 2, $\mathcal{S}$ receives on behalf of the honest parties in committee $\mathcal{C}_2$ the shares sent by corrupted parties from round 1. Note that the honest shares completely (and exactly) define these sharings since the number of honest parties is exactly $t_c + k$, and thus $\mathcal{S}$ can compute the corrupted parties' shares.

In subsequent rounds $T > 1$, $\mathcal{S}$ first simulates the resharing of honest parties of round T to corrupted parties of round $T + 1$ by sampling random values from the field $\mathbb{F}$. In round $T + 1$, $\mathcal{S}$ first inputs to $\mathcal{F}_{\mathsf{Comm}}$ random β_i on behalf of the honest parties. It also receives on behalf of the honest parties in committee $\mathcal{C}_{T+1}$ the reshared shares sent by corrupted parties from round T. Note that the honest shares completely (and exactly) define these sharings since the number of honest parties is exactly $t_c + k$, and thus $\mathcal{S}$ can compute the corrupted parties' shares as well as the actual underlying reshared shares $\tilde{\boldsymbol{Z}}_1^i, \ldots, \tilde{\boldsymbol{Z}}_k^i$ of each corrupted party P_i in $\mathcal{C}_T$. Note that these might be different from the actual underlying shares $\hat{\boldsymbol{Z}}_1^i, \ldots, \hat{\boldsymbol{Z}}_k^i$ of the corrupted parties which, inductively, $\mathcal{S}$ knows. Thus, $\mathcal{S}$ can compute $\boldsymbol{e}_m^i \leftarrow \hat{\boldsymbol{Z}}_m^i - \tilde{\boldsymbol{Z}}_m^i$ for each $m \in [k]$. We have for $k \in [m]$:[8]

$$\boldsymbol{H} \cdot (\tilde{\boldsymbol{Z}}_m^1, \ldots, \tilde{\boldsymbol{Z}}_m^n)^\intercal = \boldsymbol{H} \cdot (\hat{\boldsymbol{Z}}_m^1 + \boldsymbol{e}_m^1, \ldots, \hat{\boldsymbol{Z}}_m^1 + \boldsymbol{e}_m^n)^\intercal = \boldsymbol{H}(\boldsymbol{e}_m^1, \ldots, \boldsymbol{e}_m^n)^\intercal.$$

Since these are the underlying values of the shared vectors when the parties compute $\boldsymbol{H} \cdot ([\boldsymbol{Z}_{[1,k]}^1], \ldots, [\boldsymbol{Z}_{[1,k]}^n])^\intercal$, $\mathcal{S}$ can compute the underlying values of the shared vector defined by the shares $[\boldsymbol{y}]$ (also by using β). Thus, along with the corrupted parties' shares $\boldsymbol{y}^j$, which it can compute manually with the corrupted parties' shares $\hat{\boldsymbol{Z}_m}^j$ and β which it knows, it can reconstruct the honest parties' shares $\boldsymbol{y}^j$ and send these to the corrupted server.

Now we show that this is a good simulation. By the properties of Shamir Secret Sharing, we know that the at most t_c shares that the adversary receives in the real world for every sharing will be distributed randomly. Thus the shares that $\mathcal{S}$ sends are distributed the same way. Also the $\boldsymbol{y}^j$ shares that $\mathcal{S}$ sends to the corrupted server are computed exactly as they are in the real world, since $\mathcal{S}$ can compute the $\boldsymbol{e}_m^i$ exactly and also inductively computes the corrupted parties' shares of all sharings exactly. Thus $\mathcal{S}$ perfectly simulates the real world.

Honest Server In the case of an honest server, we can use all of the same simulation above, except we do not need to simulate the messages sent to the server. We do need to show that, even in the presence of honest dropout parties, the random linear combinations of the parity checks do indeed reconstruct to $\mathbf{0}$ if and only if the adversary did not tamper with its shares (which the simulator can trivially check and abort if so, since it keeps track of the corrupted parties' shares). Since the packed secret sharing scheme we use is linear, it is clear that applying the parity check matrix to the shares of shares will result in shares of $\mathbf{0}$ if and only if the adversary reshared the correct underlying shares: Let t_{c_1} be the number of corrupted parties in committee $\mathcal{C}_T$ that do not send to everyone in $\mathcal{C}_{T+1}$ and $m = n - t_d - t_{c_1}$ be the number of parties from committee $\mathcal{C}_T$ that do *not* drop out (including those corrupted parties that do not send to enough parties). Writing $m = t_c + k + w$, we have that $w = m - t_c - k = n - t_d - t_{c_1} - ((1/2 + \mu)n) = (1/2 - \mu)n - t_d - t_{c_1} > t_{c_2}$, where t_{c_2} is the number of corrupted parties that do send to $\mathcal{C}_{T+1}$, and thus $t_{c_1} + t_{c_2} = t_c$. The last inequality holds, since we assume that $t_d + t_c < (1/2 - \mu)n$. This means that if the corrupted parties from committee $\mathcal{C}_T$ that do send, do not reshare their actual shares to committee $\mathcal{C}_{T+1}$, then the parity check sharing will not share $\boldsymbol{y}_i = \mathbf{0}$. This is because the number of honest parties who do not drop out is at least t_+k and thus their shares completely define the correct polynomial and so if the corrupted parties'

[8] For honest parties, $\boldsymbol{e}_m^i = 0$.

shares do not match with this polynomial, it will be reflected. Using similar logic, the server in round $\mathcal{C}_{T+1}$ will be able to either successfully reconstruct the parity check sharing, or otherwise detect malicious behavior during the reconstruction.

In fact, this holds even after the parties take the random linear combination $[\boldsymbol{y}] \leftarrow \sum_{l=1}^{d(n-t_c-k)/4\mu^2 n^2} \beta^l \cdot [\boldsymbol{y}_l]$, where d is the dimension of the model. This is because β was random and unknown to the adversary before it generated its shares of shares. Thus, the underlying values of this linear combination can be seen as the evaluation of a polynomial defined by coefficients being the underlying values of the $\boldsymbol{y}_l$, on a random input β. By the Schwartz-Zippel Lemma, if any of the underlying values of the $\boldsymbol{y}_l \neq \mathbf{0}$, then the result of this polynomial evaluation will not be $\mathbf{0}$ with probability $d(n-t_c-k)/(4\mu^2 n^2 \cdot |\mathbb{F}|)$.[9] Thus, if the adversary does not tamper with its shares $\boldsymbol{y}^j$, then the reconstruction to the server will be $\mathbf{0}$ if and only if the adversary reshared the correct shares. If the adversary does tamper with its shares $\boldsymbol{y}^j$, then we know by the properties of packed secret sharing that the server will detect this and abort.

We also need to show that the output of the server is the same in the real and ideal worlds. Indeed, if an adversary tampers with its shares before inputting them to SecAgg.Enc, the worst this can achieve is an *additive attack* [26]: Let's consider the reconstruction of the shares of some $\widehat{\boldsymbol{AX}}_T$ through SecAgg, assuming w.l.o.g., that the first d parties are honest:

$$\sum_{i=1}^{n} \lambda_i^j \cdot \widehat{\boldsymbol{AX}}_T^{i,tamp} = \sum_{i=1}^{d} \lambda_i^j \cdot \widehat{\boldsymbol{AX}}_T^{i} + \sum_{i=d+1}^{n} \lambda_i^j \cdot (\widehat{\boldsymbol{AX}}_T^{i} + \boldsymbol{\chi}^i) = \widehat{\boldsymbol{AX}}_T + \boldsymbol{\chi}.$$

Indeed, since $\mathcal{S}$ sees the values input to SecAgg.Enc by the corrupted parties and also inductively knows what the corrupted parties' real input values should be, it can compute $\sum_{i=d+1}^{n} \lambda_i^j \cdot \boldsymbol{\chi}^i$ and thus $\boldsymbol{\chi}$. This completes the security proof. □

D Additional Experimental Results

Here we empirically evaluate our Distributed Matrix Mechanism (DMM) for Federated Learning on the Stack Overflow Next Word Prediction public benchmark [4], as in [33, 15]. Stack Overflow is a large-scale text dataset based on the question answering site Stack Overflow. It contains over 108 training sentences extracted from the site grouped by the $N = 342477$ users, and each sentence has associated metadata such as tags. The task of SO-NWP involves predicting the next words given the preceding words in a sentence We use the standard dataset split provided by TensorFlow. We compare to the Distributed Discrete Gaussian Mechanism for FL [33] that also obtains local DP, but with independent noise and reliance upon privacy amplification via sampling [1, 35, 7], as well as the central DP version of our paper for multiple epochs [15], where noise is correlated, but the server applies it.

As in [33, 15], we use the LSTM architecture defined in [45] directly, which has a model size of $d = 4050748$ parameters (slightly under 2^{22}). We use namely momentum 0.9, 1 client training epoch per round, client learning rate $\eta_c = 0.02$, server learning rate $\eta_s = 1$, and client batch size to 16. For Π_{PPFL}, we assume that $\mu = 1/6$; i.e., the number of corrupted parties and dropout parties per round satisfies $t_c + t_d < 1/3n$.

Matrix Factorizations We use the optimal matrix factorization $\boldsymbol{A} = \boldsymbol{BC}$ with respect to the loss function $\mathcal{L}(\boldsymbol{B}, \boldsymbol{C}) = \mathrm{sens}_\Phi(\boldsymbol{C})||\boldsymbol{B}||_F^2$ for the b-min-sep-participation schema Φ, introduced in [14].Again, we compute $\mathrm{sens}_\Phi^1(\boldsymbol{C})$ based on [14, Theorems 2 and 3].

Results Figure 4 shows that for several different ε privacy levels, our DMM significantly outperforms the DDGauss Mechanism in terms of prediction accuracy, while getting close to that of the central-DP matrix mechanism of [14]. We also see that the Honaker mechanism only sees slight accuracy degradation compared to the mechanism based on the optimal b-min-sep-participation matrix factorization. Therefore, the tree mechanism might be best in practice due to much better efficiency, as we see below. These experiments all use $n = 40$ clients per round. For the tree mechanism, we use $T^* = 2^{10} = 1024$ For the optimal matrix factorization, we use $T^* = 1500$. For both, we use $b = 85$.

[9] We assume that $|\mathbb{F}| > \lambda$.

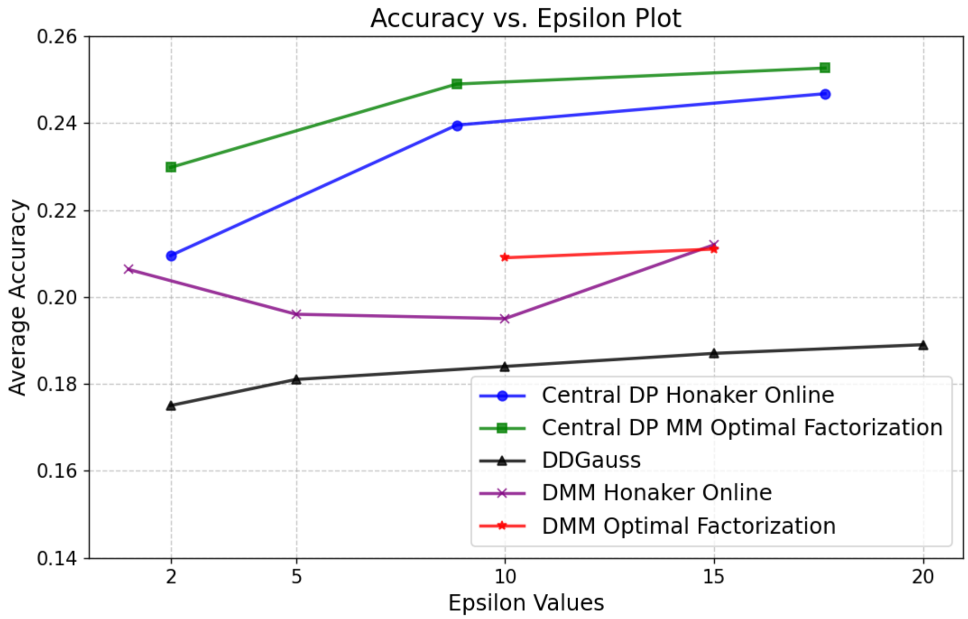

Figure 4: Test accuracies on SO NWP across different ε for the DDG mechanism [33], the central-DP matrix mechanism for multiple epochs [15], and our DMM instantiated with the optimal factorization for multiple epochs and the Honaker online factorization.

Setting	Π_{PPFL} **Comp.**	SecAgg **Comp.**	Π_{PPFL} **Comm.**	SecAgg **Comm.**
Opt.	3230 s	0.164 s	9.35 GB	16.2 MB
Honaker	17.3 s	0.164 s	50.1 MB	16.2 MB

Table 2: Client computation and communication of Π_{PPFL} and SecAgg for committee size $n = 64$. SecAgg stats are from Flamingo SecAgg protocol [38].

Efficiency Table 2 shows the client computation and communication costs of Π_{PPFL} and also the SecAgg protocol Flamingo [38]. We run the experiments on an Ubuntu machine with a 3.0 GHz Intel Xeon GHz processor and 192 GiB of memory, and use 32 bits to represent field values. We take an average over 10 runs for each reported value. For computational experiments, we use $n = 64$, as the Flamingo code requires n to be a power of two. For the optimal matrix factorization results, we report for the worst-case complexity per round, which is the last round, since here, clients need to reshare the noise and gradients from all previous rounds.

In this setting, we see the optimal matrix factorization results in about a 187x increase in both the computation and communication per client compared to the Honaker online factorization. This suggests that the small increase in accuracy from using the optimal matrix factorization may not be worth it in terms of the added efficiency costs.

Compared to Flamingo, we see a large ~ 105x increase in computation from the Honaker online factorization in Π_{PPFL}; however, ~ 4 seconds per round is still very reasonable. In terms of communication, we see a modest 3.1x increase for the Honaker online factorization in Π_{PPFL} compared to that of Flamingo. We believe that this added overhead is worth it given the increased accuracy.

E Attacks on Other Approaches and Future Work

Instead of maintaining secret-shared versions of the aggregated gradients and noise vectors, the server could preserve the aggregated noise vectors and gradients of previous training iterations within the system by masking them with an appropriate mask $\boldsymbol{mk}$ invoking a secure aggregation protocol

SecAgg_1. The masks $\boldsymbol{mk}$ themselves would be secret shared and reshared among the clients. That said, the black-box secure aggregation SecAgg_1 protocol would output aggregated gradients $\boldsymbol{G}$ and noise vectors masked by $\boldsymbol{mk}$, i.e., $\boldsymbol{G}+\boldsymbol{mk}$ to the server. When it is time to aggregate in each training iteration, another black-box SecAgg_2 protocol is called in which the server would input the masked aggregated gradients and noise vectors along and the clients would input the negative shares of the masks $\boldsymbol{mk}$. This ensures that the secure aggregation SecAgg_2 protocol outputs the unmasked (the masks of the gradients and noise vectors from previous iterations would cancel out) noisy aggregate for the current iteration to the server.

However, this approach faces a fundamental issue: the server holds the masked aggregated noise and gradients and could input any dishonest combination into the aggregation protocol to undermine DP. Specifically, the server might:

- Selective Noise Cancellation: In the matrix mechanism, noise is added directly by the clients in the current training iteration, and past aggregated correlated noise is added to enhance utility by canceling out some of the total noise. If the server has access to the masked aggregated noise, it could selectively include or exclude certain masked noises as input to the secure aggregation protocol SecAgg_2, effectively canceling out noise terms across training iterations. This would enable selective noisy cancellation, potentially weakening the overall differential privacy guarantees.
- Manipulation of Scaled Aggregated Gradients: The server might multiply the aggregated masked gradients by a malleable value when inputting them into the secure aggregation protocol SecAgg_2, causing the noise to be incorrectly scaled relative to the proper sensitivity. This manipulation could reveal information about the current iteration's aggregated gradients, thereby compromising the privacy guarantees.

Future work An alternative method for rolling noise forward to the next committee is to encrypt the noise rather than secret-sharing it based on our resharing protocol. However, an efficient solution is not straightforward, as the noise must remain encrypted while being used by the clients. The challenge lies in determining which keys to use for encryption. If the noise is encrypted using the server's key, the server could decrypt it, compromising privacy. Conversely, if it is encrypted under the client' keys, they would be able to decrypt it. Identifying an advanced encryption scheme that can maintain privacy and offer better efficiency remains an open question for future research.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We provide theorems and experiments that show this is true.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We show that our techniques are somewhat more computationally expensive than the prior work in the local DP setting and suffer some accuraccy loss due to the privacy guarantee.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Yes we clearly state assumptions and provide complete proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide pseudocode for our algorithms and describe datasets, models, and hyperparameters used, as well as our instantiations of the matrix mechanism. We also provide the code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Yes we include all of the above.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, as above.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: It would be too computationally expensive to do so. The models took days to train.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes we provide this information

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We conform to the guidelines.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We acknowledge that a tradeoff between privacy and accuracy exists

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We use public datasets

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We point out the datasets that we use.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide documentation and comments in the code.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our paper did not involve this.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our paper did not invovle this.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.