Secure and Privacy-Preserving Vertical Federated Learning

Shan Jin
Visa Research
shajin@visa.com

Sai Rahul Rachuri
Visa Research
srachuri@visa.com

Yizhen Wang
Visa Research
yizhewan@visa.com

Anderson C.A. Nascimento
Visa Research
annascim@visa.com

Yiwei Cai
Visa Research
yicai@visa.com

Abstract

We propose a novel end-to-end privacy-preserving framework, instantiated by three efficient protocols for different deployment scenarios, covering both input and output privacy, for the vertically split scenario in federated learning (FL), where features are split across clients and labels are not shared by all parties. We do so by distributing the role of the aggregator in FL into multiple servers and having them run secure multiparty computation (MPC) protocols to perform model and feature aggregation and apply differential privacy (DP) to the final released model. While a naive solution would have the clients delegating the entirety of training to run in MPC between the servers, our optimized solution, which supports purely global and also global-local models updates with privacy-preserving, drastically reduces the amount of computation and communication performed using multiparty computation. The experimental results also show the effectiveness of our protocols.

Keywords

Vertical federated learning, differential privacy, secure multiparty computation

1 Introduction

Federated Learning [63] is a powerful machine learning paradigm over distributed data sources. It allows multiple data sources, often dubbed as the clients, to collaboratively learn an ML model that outperforms each client's training on its own data. Privacy of clients' sensitive and proprietary data has been a core motivation of FL since its genesis. Recent advances in privacy-preserving federated learning (PPFL) have shown mechanisms that provably enhance data privacy during both communication and computation [91]. In particular, secure aggregation [11] and multiparty computation [21] guarantee that no party can infer any information about the clients' model updates from intermediate values during model aggregation, whereas differential privacy-preserving mechanisms [27, 29] ensure the amount of information that can be inferred from the aggregated model is limited by a provable guarantee. These techniques provably enhance the privacy of FL in the classical FL regime, in which data are split by rows across the clients and the clients' model updates are homogeneous in format.

Proceedings on Privacy Enhancing Technologies 2026(3), 625–645

https://doi.org/10.56553/popets-2026-0099

However, vertical federated learning (VFL)–a challenging FL setting that attracts an increasing amount of attention lately [57]–compels us to rethink the design of PPFL. In VFL, the data, including the label, can be vertically split among clients. Each client may have a unique view of the dataset that consists of only a subset of the columns. For example, banks and credit card companies may want to collaborate against fraud, but typically bank account information and transaction history of the credit card are not aligned in the feature space. Taking advertisement as another example: Normally, an advertising platform may know the browsing history and interests of the user, but only the shopping platform knows whether the click has converted to an actual purchase. The ubiquity of VFL in real-world applications calls for a VFL-friendly PPFL mechanism.

Unfortunately, existing PPFL mechanisms [11, 34] do not apply to VFL. In particular, with each client only holding a partial view of the data, no client can compute the model output or the model update on its own. The flow of backpropagation is disrupted. Moreover, the client and the server may no longer have the same model architecture, i.e., the global model may not be a simple aggregation, e.g., averaging, of the clients' models. Instead, the entire model is often split into separate models between clients and the server. In a general VFL framework [57], a client model extracts an embedding from the client's private data as the intermediate output; the global model at the server then proceeds with the intermediate outputs from all clients to complete the learning task. Although delegating all computation to a secure MPC environment plus applying DP noise during training can ensure both input and output privacy, such a naive combination will be computationally prohibitive, especially when training deep neural networks. Previous work in [70] applied the same idea but restricted it to training a generalized linear model only. Besides, the work in [74, 75] designed a framework for training VFL securely based on secure aggregation [11]. However, this framework did not provide any DP protection on each client's private data, which could easily leak information under membership inference attacks (MIAs) [68, 80, 90]. Most recently, a federated transformer framework for fuzzy VFL was proposed in [87]. This work focuses on a special variant of conventional VFL, fuzzy VFL, which enables *one-to-many* record linkage across clients. While this framework preserves both input and output privacy, it represents only a restricted case in VFL. In contrast, our work aims to develop a privacy-preserving protocol for the more general VFL setting.

In response to these challenges, we propose a novel practical privacy-preserving vertical federated learning mechanism with privacy guarantees on both input and output privacy. Our main contributions are as follows:

(1) We show an efficient differentially private VFL mechanism that couples secure aggregation with the Gaussian mechanism and the Matrix Factorization (MF)-based mechanism to train a general *global* model given pre-trained, frozen local models at the clients. (Section 6.1.)
(2) We extend the VFL mechanism to allow practical privacy-preserving updating of local layers within *local* models as needed without adding DP noises to multiple local layers. To achieve this, we leverage linear estimation techniques to extract insights from the privacy-preserving information published by the server, enabling backward propagation on local models, under the MF-based mechanism. (Section 6.2.)
(3) We empirically evaluate both of our mechanisms on CIFAR-10/EMNIST. Our mechanisms show competitive utility on both datasets across a wide range of privacy parameter εs. (Section 8.)

Our mechanism supports the general VFL pipeline in [57], which allows a flexible split of the global and local models. On a high level, our mechanism securely computes the global model updates and releases the global model with DP protection. This design makes the computation of the global model a choke-point for privacy, and as a result, computation involving only local models at the clients can be done in plaintext, which significantly reduces the computation overhead compared to running MPC end-to-end. This improvement enables training of complex architectures like ResNet-18, which was previously infeasible with end-to-end MPC. Furthermore, we leverage linear estimation techniques to recover the desired per-sample gradients from the received DP-protected release of batch gradient. By the post-processing property of DP, the estimated per-sample gradients are also DP protected, which can then be used by the clients in the clear to perform backpropagation on local models.

2 Related Work

Various previous works have acknowledged and attempted to address the challenge of preserving privacy in VFL. On one hand, multiple encryption-based methods [38, 43, 67, 74, 75] have been proposed to preserve input privacy in computation. However, this line of work does not prevent output privacy leakage, such as membership inference attacks against the released model. On the other hand, several existing mechanisms [83, 84] attempt to enhance output privacy by perturbing the intermediate results. However, these mechanisms either target relatively simple model architectures or lack rigorous privacy guarantees. In [88], the authors present a differentially private VFL framework with multiple server heads, allowing clients to conduct multiple local updates per communication round through an ADMM (Alternating Direction Method of Multipliers)-based optimization scheme. This approach achieves remarkable communication efficiency and also a better privacy–utility tradeoff compared to baseline methods such as split learning [82]. However, ADMM's sophisticated coordination mechanism requires extensive hyperparameter tuning, making it highly sensitive to practical deployment settings. For example, even a slight change in the penalty parameter ρ can lead to significant performance variations. More importantly, the subsampling mechanism applied in this approach still requires synchronized randomness across all clients to maintain vertical data alignment, which breaks the anonymity of the sampling process and may lead to a privacy leakage vector. Recently, [39] proposes a differentially private VFL framework that introduces adaptive constraints and dynamic noise injection to better balance the privacy–utility tradeoff. However, the framework still incurs performance degradation on a complex dataset (CIFAR-10) under stringent privacy budgets, and the added adaptive mechanisms including per-iteration Laplacian score calculation also increase tuning overhead. In contrast, our work systematically studies the possibility of combining MPC and provable DP to enjoy the benefit of both worlds for achieving a better utility-privacy trade-off.

In terms of combining MPC and DP for VFL, the most notable and the closest to ours in spirit is [87], which enhances both input and output privacy in fuzzy-linked VFL by applying local differential privacy (LDP) to the local intermediate representations and aggregating them securely via MPC. This one-to-many fuzzy linkage enables subsampling for privacy amplification, but is not feasible in conventional one-to-one linkage VFL. Compared to [87] which is particularly suited to VFL with fuzzy-link tasks, we propose a framework for the general VFL setting with careful consideration of all surfaces of privacy leakage including model release, intermediate output exchange, and data alignment.

Additional references not directly related to our work are provided in Appendix A, for readers seeking additional background.

3 Preliminaries

Differential Privacy. The notion of DP was first introduced in [28]. Intuitively, differential privacy ensures that the participation of any individual has limited influence on the released output, thereby preventing adversaries, regardless of their auxiliary information, from reliably inferring whether a particular individual's data was included in the dataset. In addition, DP satisfies a desirable post-processing property: The privacy guarantees are preserved, no matter how the output is later used or manipulated [29]. DP is particularly pertinent to ML because ML models, which are the outputs of learning algorithms, will be released or heavily queried at usage. Existing attacks already show extraction of input information, i.e., the training data, from unguarded models [68, 80, 90]. In this paper, we leverage multiple differentially private mechanisms, including the Gaussian mechanism via the moments accountant proposed in [1], as well as the Matrix Factorization mechanism [17, 69]. The basic concepts and formal definitions of these mechanisms are summarized in Appendix D.

Multiparty Computation. MPC allows mutually distrusting parties to jointly compute a function on their private data without revealing the data. It has found wide application, including PPML, and provides two core guarantees: *correctness* (the output is correct) and *privacy* (no information beyond the output is learned). Our work assumes a set of servers that can run MPC with flexible configurations (e.g., number of servers, corruption threshold). We focus on passive security, but by replacing the underlying components with actively secure protocols, one could achieve active security. We leverage MPC protocols to 1) securely compute the model updates without revealing to a centralized server, and 2) re-enable classical DP mechanisms in a decentralized setting in the absence of a trusted curator. In our protocols, data is secret-shared

among servers using a linear scheme (e.g., arithmetic or replicated secret-sharing), enabling local additions and secure multiplications with minimal communication. We use the notation $[\![\cdot]\!]$ to indicate a linear secret-sharing scheme. We also use recent advances allowing mixed-mode computation, which include specialized protocols for non-linear activation functions, secure shuffles, and Gaussian noise sampling. For further details on MPC and the secret-sharing scheme, see Appendix C.

4 Problem Setup

4.1 Notation

Typically, in a FL system, there is a central server (sometimes multiple servers) that seeks to train a model based on private data from N clients. Each client E_i has its own private dataset $\mathcal{D}_i$ and we use $\mathcal{D} = \mathcal{D}_1 \bigcup \mathcal{D}_2 \cdots \bigcup \mathcal{D}_N$ to denote the global dataset space. $\mathcal{D}$ contains the feature space $\mathcal{X}$, the label space $\mathcal{Y}$, which contains S classes, and the sample ID space $\mathcal{I}$. Hence the dataset owned by the i-th client is denoted by $\mathcal{D}_i \triangleq: (\mathcal{X}_i, \mathcal{Y}_i, \mathcal{I}_i)$.

In this paper, we consider the particular scenario of vertical federated learning, where the clients hold a disparate set of features that come from the same sample ID space (henceforth, we omit $\mathcal{I}$). More specifically, the feature space is divided into a finite number of sub-spaces with $\mathcal{X} = \mathcal{X}_1 \bigcup \mathcal{X}_2 \cdots \bigcup \mathcal{X}_N$ where each client, E_i, holds the sub-space $\mathcal{X}_i$. We assume there are M samples in the global dataset, $\mathcal{D} \triangleq: \{\boldsymbol{x}_m, y_m\}_{m=1}^M$ where $\boldsymbol{x}_m \in \mathbb{R}^{d_{\boldsymbol{x}}}$ is the feature vector of each sample, with feature dimension $d_{\boldsymbol{x}}$. Note the feature dimension $d_{\boldsymbol{x}}$ typically refers to the size after flattening and does not affect the input shape fed into the model. Now each client E_i only holds a subset of the features in $\boldsymbol{x}_m$, with feature dimension $d_{\boldsymbol{x}^{(i)}}$, such that $d_{\boldsymbol{x}} = \sum_{i=1}^N d_{\boldsymbol{x}^{(i)}}$. Therefore, we denote $\{\boldsymbol{x}_m^{(i)} \in \mathbb{R}^{d_{\boldsymbol{x}^{(i)}}}\}_{m=1}^M$ as the samples held by the i-th client where $\boldsymbol{x}_m^{(i)} \subseteq \mathcal{X}_i$ with $m \in [1, M]$ and hence $\boldsymbol{x}_m = \boldsymbol{x}_m^{(1)} \bigcup \boldsymbol{x}_m^{(2)} \cdots \bigcup \boldsymbol{x}_m^{(N)}$.

In VFL we distinguish between two kinds of clients. One is called the *label client* (or *active client*), which holds the samples with a subset of features as well as the labels $\{y_m\}_{m=1}^M$. The other clients, which we call *feature clients* (or *passive clients*), only hold a subset of the features for their corresponding samples. For simplicity, we assume there is only one label party, which is the N-th client E_N.

4.2 System Model

We denote $\boldsymbol{\Theta}$ as the full model in VFL. We can decompose $\boldsymbol{\Theta}$ into a global model $\boldsymbol{\mathcal{W}}$ parameterized by $\boldsymbol{\theta}$, and each client E_i holds a local model $\boldsymbol{\mathcal{F}}_i$ parameterized by $\boldsymbol{\phi}_i$. The last client E_N holds the labels $\boldsymbol{y}$, in addition. The loss function for j-th sample is defined as

$$\min_{\boldsymbol{\Theta}} \mathcal{L}_j(\boldsymbol{\Theta}; \boldsymbol{x}_j, \boldsymbol{y}_j) = l\left(\boldsymbol{\mathcal{W}}(\boldsymbol{\theta}; \{\boldsymbol{\mathcal{F}}_i(\boldsymbol{\phi}_i; \boldsymbol{x}_j^{(i)})\}_{i=1}^N), y_j\right),$$

where $l(\cdot)$ denotes the task loss. Since typically the training dataset is divided at the batch level, we set B as the number of samples in a mini-batch and we assume $B_{num} := M/B$ batches are used in each epoch (for simplicity, here we assume M is divisible by B). We also denote the total number of epochs as E_{num}, and therefore the total number of steps T_{num} is obtained through $T_{num} = B_{num} \cdot E_{num}$.

Typically, during model training, at the training step t, the index of the batch used in this step, is denoted by b where $b := t \mod B_{num}$. Hence we define the samples of b-th batch at i-th client as $\mathbf{x}_b^{(i)} = \{\boldsymbol{x}_{b,j}^{(i)}\}_{j=1}^B$ with $b \in [1, M/B]$, where $\boldsymbol{x}_{b,j}^{(i)}$ is the j-th sample of b-th batch at i-th client. Similarly, $\mathbf{y}_b = \{y_{b,j}\}_{j=1}^B$ are the labels for the b-th batch.

Then at the beginning of each step, each client E_i feeds its own batch into the local model $\boldsymbol{\phi}_i$ and obtains the intermediate output: $\mathbf{H}_b^{(i)} = \{\boldsymbol{H}_{b,1}^{(i)}; \ldots; \boldsymbol{H}_{b,B}^{(i)}\} \in \mathbb{R}^{B \times d_{\boldsymbol{H}^{(i)}}}$ where $\boldsymbol{H}_{b,j}^{(i)} = \boldsymbol{\mathcal{F}}_i(\boldsymbol{\phi}_i; \boldsymbol{x}_{b,j}^{(i)}) \in \mathbb{R}^{d_{\boldsymbol{H}^{(i)}}}$ with $j \in [1, B]$ and $d_{\boldsymbol{H}^{(i)}}$ is the feature dimension of the intermediate output at i-th client. Then each E_i uploads its intermediate output $\mathbf{H}_b^{(i)}$ to the server, where the server concatenates the intermediate outputs from all clients to construct $\mathbf{H}_b = \{\mathbf{H}_b^{(1)}, \ldots, \mathbf{H}_b^{(N)}\} \in \mathbb{R}^{B \times d_{\mathbf{H}}}$ where $d_{\mathbf{H}} = \sum_{i=1}^N d_{\boldsymbol{H}^{(i)}}$. Eventually, the server feeds $\mathbf{H}_b$ into the global model $\boldsymbol{\theta}$ and obtains the logits, $\boldsymbol{Z} \in \mathbb{R}^{B \times S}$, as the final outputs. For ease of reference, the notations introduced above are summarized in Table 6 in Appendix B.

4.3 Security and Privacy Definitions

4.3.1 Data Alignment. We assume that the data instances (features and labels) among the clients are aligned: data linked from the same sample ID space, also called *one-to-one linkage*. Since we are in the cross-silo setting, for privacy protection purpose, the alignment can be performed through a privacy-preserving data alignment protocol such as Private Set Intersection (PSI) [51, 71]. This is inherent to any vertically split FL protocol, and we see the data alignment phase as orthogonal to the problem addressed in this paper.

Note that we assume all clients are semi-honest, as defined in Section 4.3.2. Under this threat model, both entity alignment and data participation are strictly determined by the protocol specification.

4.3.2 Threat Model.

Trust Assumptions. We consider K servers $\mathcal{P}_S = \{P_k\}_{k=1}^K$ (instantiated with $K = 3$ in our implementation) connected via pairwise authenticated channels. We follow the three-party setting as it strikes a practical balance: it provides separation of trust among independently managed servers while keeping communication costs low, and it has been adopted in real deployments [4]. A *static*, *semi-honest* adversary $\mathcal{A}$ corrupts at most $t < K/2$ servers before protocol execution begins (i.e., $t = 1$ when $K = 3$); the corrupted server(s) follows the protocol honestly but $\mathcal{A}$ observes its complete view (received messages from other parties and clients, randomness, and state). We adopt the semi-honest model (passive) because our cross-silo deployment targets servers operated by distinct organizations, where the reputational and legal cost of active deviation outweighs potential gains [44]. However, our framework is not tied to a specific MPC protocol. Since all our protocols only make *black-box* use of standard MPC building blocks such as multiplication, secure comparison, etc., replacing them with maliciously (actively) secure counterparts in the UC-secure model, yields active security.

We assume all N clients are semi-honest: They correctly follow the defined protocols but may attempt to infer other clients' and servers' private information, including data features and labels, from the information exchanged between clients and servers during the protocol execution. We further consider external adversaries that can observe the prediction outputs and gradients released during training, and query the deployed model after training, with the goal of inferring clients' private information through membership inference attacks [80].

Input Privacy. Our framework employs MPC to provide input privacy. Each client secret-shares its data among multiple servers, and our protocols guarantee that *no* intermediate values are revealed to any individual server under the semi-honest, honest-majority assumption (Theorem 1).

Output Privacy. In our framework, we say the system provides (ϵ, δ)-output privacy if *all* information released by the servers is (ϵ, δ) differentially private (D.1), with respect to each client E_i's dataset $\mathcal{D}_i$, for all $i \in [1, N]$ (Section 7.2).

5 Single-Server Plaintext Model Training

Before detailing our proposed protocols, we first introduce the basic VFL framework under a single server [57] without any privacy protection, i.e. the plaintext model. This introduction will familiarize the readers with the training flow of a typical VFL framework while also illustrating the privacy challenges.

Global Model Update. In the forward pass, each client computes its local representation $\mathbf{H}_b^{(i)}$ and passes to the server. The server concatenates the representations to $\mathbf{H}_b$ and computes the logits $Z = \mathcal{W}(\boldsymbol{\theta}; \mathbf{H}_b)$ and subsequently the loss $\mathcal{L}$. During back-propagation, the server computes the per-sample gradients for the global model, here the gradient of the j-th sample is represented by

$$\boldsymbol{g}_{\boldsymbol{\theta},j} = \frac{\partial \mathcal{L}_j}{\partial \boldsymbol{\theta}} = \frac{\partial l(\sigma(\boldsymbol{Z}_j), y_{b,j})}{\partial \boldsymbol{\theta}}, \tag{1}$$

where σ is the activation function and $\boldsymbol{Z}_j$ are the logits produced by the j-th sample. The aggregated per-batch gradient is $\boldsymbol{g}_{\boldsymbol{\theta}} = \frac{1}{B}\sum_{j=1}^{B} \boldsymbol{g}_{\boldsymbol{\theta},j}$. Once the gradient is obtained, the server can update the global model using first-order optimization algorithms. For example, in the classic stochastic gradient descent (SGD), a new global model $\boldsymbol{\theta}^{t+1}$ can be obtained as $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \eta_s \boldsymbol{g}_{\boldsymbol{\theta}}$, where η_s is the learning rate and t is the training step.

Local Models Update. Updating the local model is slightly more involved than updating the global model but still achievable via collaboration between the server and the clients. By chain-rule, the per-sample gradient of the local model can be written as

$$\boldsymbol{g}_{\boldsymbol{\phi}_i,j} = \frac{\partial \mathcal{L}_j}{\partial \boldsymbol{\phi}_i} = \frac{\partial l(\sigma(\boldsymbol{Z}_j), y_{b,j})}{\partial \boldsymbol{H}_{b,j}^{(i)}} \cdot \frac{\partial \boldsymbol{H}_{b,j}^{(i)}}{\partial \boldsymbol{\phi}_i} \tag{2}$$

The first term, $\partial l(\sigma(\boldsymbol{Z}_j), y_{b,j})/\partial \boldsymbol{H}_{b,j}^{(i)}$, is the gradient of the loss w.r.t. (with respect to) the client's output at j-th sample. Note that the server can compute the per-sample gradients for each client's intermediate output and then broadcast the results back to the respective clients. The second term, $\partial \boldsymbol{H}_{b,j}^{(i)}/\partial \boldsymbol{\phi}_i$, can be computed locally by the client. Then, each client can combine the two terms to recover the per-sample gradient of its local model and update the model using SGD through $\boldsymbol{\phi}_i^{t+1} = \boldsymbol{\phi}_i^t - \eta_i \boldsymbol{g}_{\boldsymbol{\phi}_i}$, where $\boldsymbol{g}_{\boldsymbol{\phi}_i} = \frac{1}{B}\sum_{j=1}^{B} \boldsymbol{g}_{\boldsymbol{\phi}_i,j}$ and η_i is the learning rate at the i-th client.

The plaintext training protocol protects neither the input nor the output privacy. On the one hand, the intermediate outputs $\mathbf{H}_b$ uploaded from the clients are completely disclosed to the server. On the other hand, the trained global model and the per-sample gradient with respect to the intermediate output ($\partial \mathcal{L}_j/\partial \boldsymbol{H}_{b,j}^{(i)}$) are disclosed without any protection: Such disclosure will leak information about each client's private data to others as studied in previous research [37, 45, 93]. These privacy challenges motivate us to design practical secure and privacy-preserving protocols for VFL.

6 Our : MPC-aided Vertical FL Training with DP

In this section, we introduce our VFL training protocols that guarantee both input and output privacy. We consider a two-stage training scheme: a client-held section (local model) and a server-held section (global model) maintained in secret-shared form. We use the global model as a choke-point for privacy. By applying MPC on the global model with DP protection instead of applying MPC end-to-end, our design significantly reduces MPC usage, which substantially improves the training time compared to naïve MPC *end-to-end* approaches. Also, this improvement enables training of complex architectures, which was previously infeasible with end-to-end MPC. In Section 6.1, we introduce our protocol with frozen local models. In Section 6.2, we show an extension that allows the clients to update their local models with privacy guarantees too. Before proceeding to our protocols, we give a primer on the MPC sub-protocols and blocks used to build our protocols.

Sub-protocols for MPC. Specifically, we use the protocol from [4] as the basis to instantiate the 3PC building blocks of secret-sharing, reconstruction, and multiplication Π_{Mult}. To instantiate the shuffling Π_{Shuffle}, we use the protocol from [5]. We use the protocols from [2] for mathematical functions over fixed-point numbers such as approximated square root, exponentiation, and division, using which we build the ℓ_2-normalization protocol Π_{Norm} and the division protocol Π_{Div}. For more details about how these building blocks are implemented in MP-SPDZ, we refer the reader to [48]. Moreover, we build the forward-passing module Π_{FWD} and the backward-propagation module Π_{BWD} in MP-SPDZ, which are the wrap-ups of the sub-protocols including Π_{Mult} and activation function Π_{Act} introduced in [65], for conducting the forward-passing and backward-propagation.

The Gaussian noise protocol, Π_{GS}, could be implemented from a number of existing protocols such as [15, 23, 30, 47]. Given input (T_{num}, d, σ), where d is the length of the noise vector, and σ is the standard deviation of the noise, a secret-shared version of noise matrix $\mathcal{N}_{\mathsf{Tab}} \in \mathbb{R}^{T_{num} \times d}$, where each noise vector follows the normal distribution of $\mathcal{N}(0, \sigma^2 \boldsymbol{I}_d)$, is obtained among the servers. Here I is an identity matrix. As the noise is not input data-dependent, we assume the servers compute the noise matrix in the pre-processing phase, before the start of the training phase.

6.1 Training a Global model with DP

We first consider the scenario in which the clients have pre-trained local models that will be frozen during the training phase; only the global model, $\mathcal{W}$, will be updated. Typically, to achieve output privacy guarantees, the servers can apply the DP-SGD algorithm [1] during global model training. In practice, privacy amplification techniques, such as subsampling [1], are often employed to improve the privacy–utility tradeoff. These techniques require each mini-batch to be sampled uniformly at random from the training dataset, with the sampling permutation kept hidden from all parties; that is, no party has knowledge of the permutation.

However, this requirement poses a challenge in vertical federated learning: As data holders, once data alignment is completed,

all clients share knowledge of the dataset ordering. Consequently, under the threat model defined in Section 4.3.2, mini-batch permutations cannot be privately generated and distributed to clients, neither directly by the servers nor through a joint agreement among clients that is subsequently revealed, without compromising the anonymity of sample participation.

Therefore, we present two variants of our protocol in order to optimize the privacy-utility trade-off under various privacy and computation constraints. Our first protocol utilizes the Gaussian mechanism from [1] together with privacy amplification via subsampling [85]. We use a secure shuffling procedure to enable subsampling without replacement and therefore name the protocol $\Pi_{\text{Global-Shuffle}}$. Our second protocol targets the case where private shuffling of the data is computationally prohibitive. By utilizing the Banded Matrix Factorization approach from [69], our second protocol, $\Pi_{\text{Global-BMF}}$, can achieve comparable privacy-utility trade-off without relying on amplification by subsampling.

6.1.1 Protocol with Amplification by Subsampling. As mentioned above, to ensure anonymity of sample participation, if one intends to apply subsampling, the permutations cannot be directly provided to clients, neither given by the servers nor through an agreement among the clients themselves. Towards this, we propose an efficient protocol to achieve privacy amplification through subsampling that is also compatible with the VFL framework. Our full protocol consists of three main components: 1) forward-passing and secret-sharing on intermediate outputs, 2) secure computation of gradients for the global model, 3) privacy-preserving noise addition for the securely computed global model gradients and model update.

Forward-passing and Secret-sharing. During local forward passing, each client E_i feeds all M/B batches of its local data into the local model $\mathcal{F}_i$, and obtains the intermediate outputs $\mathbf{H}_b^{(i)} = \{\boldsymbol{H}_{b,1}^{(i)}; \ldots; \boldsymbol{H}_{b,B}^{(i)}\}$ for $b \in [1, M/B]$. Notice that each local model is a *deterministic function* since it is frozen during training. Furthermore, the order of the samples in each client is fixed after data alignment. Therefore, for each client, the intermediate output, $\mathbf{H}_b^{(i)}$ with $b \in [1, M/B]$, is a *fixed mapping* to the local dataset. Hence it suffices for the clients to compute the forward pass on all the batches *once* during the *first* epoch and secret-share the intermediate output (and the labels) to the servers. No further *upstream communications* are needed afterwards from the clients to the servers. The servers can store the intermediate outputs and reuse them for future training epochs, thereby reducing the communication overhead.

Thereafter, each client E_i sends the secret-shares of its intermediate output $\mathbf{H}_b^{(i)}$, which is denoted by $[\![\mathbf{H}_b^{(i)}]\!]$ for $b \in [1, M/B]$, to the servers. The label client also secret-shares the labels $\mathbf{y}_b$ as $[\![\mathbf{y}_b]\!]$. Upon receiving all batches of secret-shares from each client, the servers *vertically* concatenate the shares of the intermediate outputs from all clients for each batch respectively: $[\![\mathbf{H}_b]\!] = \{[\![\mathbf{H}_b^{(1)}]\!], \ldots, [\![\mathbf{H}_b^{(N)}]\!]\}$ for $b \in [1, M/B]$ and then *horizontally* concatenate all $[\![\mathbf{H}_b]\!]$ into $[\![\overline{\mathbf{H}}]\!] = \{[\![\mathbf{H}_1]\!]; \ldots; [\![\mathbf{H}_{M/B}]\!]\}$. All $[\![\mathbf{y}_b]\!]$ are also *horizontally* concatenated into $[\![\overline{\mathbf{y}}]\!] = \{[\![\mathbf{y}_1]\!]; \ldots; [\![\mathbf{y}_{M/B}]\!]\}$.

Secure Gradients Computation for Global Model. During global model training, at the beginning of *every* epoch, the servers make a blackbox call to a *secure shuffling* protocol, Π_{Shuffle}, on $[\![\overline{\mathbf{H}}]\!]$ and $[\![\overline{\mathbf{y}}]\!]$, to obliviously shuffle the rows and obtain $[\![\overline{\mathbf{H}}^*]\!]$ and $[\![\overline{\mathbf{y}}^*]\!]$. Both $[\![\overline{\mathbf{H}}^*]\!]$ and $[\![\overline{\mathbf{y}}^*]\!]$ are further divided into M/B batches again, as $[\![\overline{\mathbf{H}}_b^*]\!]$ and $[\![\overline{\mathbf{y}}_b^*]\!]$ with $b \in [1, M/B]$ which are then *sequentially* fed into the global model, one batch at a time. This shuffling procedure enables subsampling without replacement.

As a result, for *each* step t (note the corresponding batch index in the current epoch is given by $b := t \mod B_{num}$), the servers compute the logits $[\![Z]\!]$ and hence the per-sample loss $[\![\mathcal{L}_j]\!]$ with $j \in [1, B]$, by calling the forward-passing module Π_{FWD}. Thereafter, the per-sample gradients $[\![\boldsymbol{g}_{\boldsymbol{\theta},j}]\!] = [\![\partial \mathcal{L}_j / \partial \boldsymbol{\theta}]\!]$ for the global model are computed through the back-propagation module Π_{BWD}.

Noise Addition for Securely Computed Information. Once the per-sample gradients are computed, they are then clipped into

$$[\![\overline{\boldsymbol{g}}_{\boldsymbol{\theta},j}]\!] = \left[\!\left[\boldsymbol{g}_{\boldsymbol{\theta},j} / \max(1, \|\boldsymbol{g}_{\boldsymbol{\theta},j}\|_2 / \gamma)\right]\!\right] , \tag{3}$$

where the clipping threshold γ is set to the maximum ℓ_2 norm bound on each gradient. We use $\Pi_{\text{Norm}}, \Pi_{\text{Div}}$ to compute the norm and the clipping factors. The clipped gradients in a mini-batch are aggregated locally by the servers into the aggregated gradient $[\![\overline{\boldsymbol{g}}_{\boldsymbol{\theta}}]\!] = \sum_{j=1}^{B} [\![\overline{\boldsymbol{g}}_{\boldsymbol{\theta},j}]\!]$. Note in the pre-processing phase, the servers run Π_{GS} with the input of $(T_{num}, n_{\boldsymbol{\theta}}, \sigma \cdot \gamma)$, where $n_{\boldsymbol{\theta}}$ is the number of parameters in $\boldsymbol{\theta}$, to generate a secret-shared noise matrix $[\![\boldsymbol{\mathcal{N}}_{\text{Tab}}]\!]$. Then, at the step t, the noise vector $[\![\boldsymbol{\mathcal{N}}_{\boldsymbol{\theta}}]\!] = [\![\boldsymbol{\mathcal{N}}_{\text{Tab}}]\!]_{[t,:]}$ is added to the aggregated gradient, which results in the privatized gradient

$$[\![\tilde{\boldsymbol{g}}_{\boldsymbol{\theta}}]\!] = \frac{1}{B}([\![\overline{\boldsymbol{g}}_{\boldsymbol{\theta}}]\!] + [\![\boldsymbol{\mathcal{N}}_{\boldsymbol{\theta}}]\!]) .$$

The servers update $[\![\boldsymbol{\mathcal{W}}]\!]$ through gradient descent with gradient $[\![\tilde{\boldsymbol{g}}_{\theta}]\!]$. The formal description of the protocol is shown in Fig. 1.

6.1.2 Protocol With Banded Matrix Factorization. In 6.1.1, we design a protocol that achieves privacy amplification by subsampling via secure shuffling over the data secret-shared from the clients. However, for large-scale datasets, performing a secure shuffle for each epoch could take up a considerable amount of time. Thus, we propose an alternative protocol that does not rely on the subsampling but still achieves a competitive privacy-utility trade-off. This protocol reduces the communication and computation in the online phase by using a public correlation matrix. In particular, our protocol utilizes the Banded Matrix Factorization (BandMF) mechanism [69], which leverages *banded correlation matrices* to generate *correlated noise* to be applied during training.

We first follow the same concept of b-*min-separated participation* schema proposed in [16]. Under this schema, if a single item (such as a sample) contributed to a model gradient vector at step t, then the earliest it can contribute again is at step $t + \mathrm{b}$. Our VFL training protocol naturally satisfies the schema: The order of the samples at each client is fixed, and therefore the interval between two adjacent participations of any sample point is constant. In fact, our training protocol also satisfies *fixed-epoch-order participation* schema (the (κ, b)-*participation*) [17], a stricter participation schema where each sample participates κ times with the constraint that any two adjacent participations are exactly b steps apart. Note we fix $\kappa = E_{num}$ and $\mathrm{b} = B_{num}$ throughout this paper.

In *Matrix Factorization* mechanisms [17, 26], let $\boldsymbol{A} \in \mathbb{R}^{T_{num} \times T_{num}}$ be an appropriate *workload matrix*, and $\boldsymbol{G} = \{\boldsymbol{g}_{\boldsymbol{\theta}}^1; \ldots; \boldsymbol{g}_{\boldsymbol{\theta}}^{T_{num}}\}$ is a stream of model batch gradients. Here each batch gradient $\boldsymbol{g}_{\boldsymbol{\theta}}^t$ has a bounded ℓ_2 norm, as $\|\boldsymbol{g}_{\boldsymbol{\theta}}^t\| \leq \gamma$. The matrix factorization on $\boldsymbol{A}$ is

Protocol $\Pi_{\text{Global-Shuffle}}$

Parameters: Clients $\mathcal{P}_C = E_1, \ldots, E_N$, aggregation servers $\mathcal{P}_S = \{P_k\}_{k=1}^K$.
Preprocessing: The servers run Π_{GS} with input $(T_{num}, n_\theta, \sigma\gamma)$ to receive $[\![\boldsymbol{\mathcal{N}}_{\text{Tab}}]\!]$.

Client-Server Phase: Each client E_i does the following: Before the global model training starts,

(1) Locally computes $\mathbf{H}_b^{(i)} = \{\boldsymbol{H}_{b,1}^{(i)}; \ldots; \boldsymbol{H}_{b,B}^{(i)}\}$ where $\boldsymbol{H}_{b,j}^{(i)} = \mathcal{F}_i(\boldsymbol{\phi}_i; \boldsymbol{x}_{b,j}^{(i)})$, and secret-shares $[\![\mathbf{H}_b^{(i)}]\!]$, for $b \in [1, M/B]$, then sends all the batches to $\mathcal{P}_S$.
(2) The label client E_N also secret-shares $[\![\mathbf{y}_b]\!]$ for $b \in [1, M/B]$ to $\mathcal{P}_S$.

Server MPC: At the beginning of each epoch, $\mathcal{P}_S$ do the following:

(1) Only if at the beginning of the *first* epoch, vertically concatenate $[\![\mathbf{H}_b^{(i)}]\!]$ received from $\mathcal{P}_C$ for $i \in [1, N]$, which results in $[\![\mathbf{H}_b]\!]$ with $b \in [1, M/B]$, and then horizontally concatenate $[\![\mathbf{H}_b]\!]$ and $[\![\mathbf{y}_b]\!]$, for $b \in [1, M/B]$, to result in $[\![\overline{\mathbf{H}}]\!]$ and $[\![\overline{\mathbf{y}}]\!]$, respectively.
(2) Run Π_{Shuffle} on $[\![\overline{\mathbf{H}}]\!]$ and $[\![\overline{\mathbf{y}}]\!]$, to receive the sorted values $[\![\overline{\mathbf{H}}^*]\!]$ and $[\![\overline{\mathbf{y}}^*]\!]$ respectively.
(3) Split $[\![\overline{\mathbf{H}}^*]\!]$ and $[\![\overline{\mathbf{y}}^*]\!]$ into M/B mini-batches as $[\![\overline{\mathbf{H}}_b^*]\!]$ and $[\![\overline{\mathbf{y}}_b^*]\!]$ with $b \in [1, M/B]$.

For each step in the current epoch, $\mathcal{P}_S$ do the following:

(1) Compute per-sample loss $[\![\mathcal{L}_j]\!]$ for $j \in [1, B]$, by inputting $[\![\overline{\mathbf{H}}_b^*]\!]$, $[\![\boldsymbol{\mathcal{W}}]\!]$ and $[\![\overline{\mathbf{y}}_b^*]\!]$ into Π_{FWD}.
(2) Compute $[\![\boldsymbol{g}_{\theta,j}]\!] = [\![\frac{\partial \mathcal{L}_j}{\partial \theta}]\!]$, for $j \in [1, B]$, by running Π_{BWD}.
(3) Compute $[\![\gamma_j]\!] = \max(1, \mathrm{n}/\gamma)$, where $[\![n]\!] = \Pi_{\text{Norm}}([\![\boldsymbol{g}_{\theta,j}]\!])$, for $j \in [1, B]$.
(4) Finally, obtain $[\![\overline{\boldsymbol{g}}_{\theta,j}]\!] = [\![\frac{\boldsymbol{g}_{\theta,j}}{\gamma_j}]\!]$, by running Π_{Div}.
(5) Locally aggregate the gradients by $[\![\overline{\boldsymbol{g}}_\theta]\!] = \sum_{j=1}^B [\![\overline{\boldsymbol{g}}_{\theta,j}]\!]$.
(6) Locally add noise by $[\![\tilde{\boldsymbol{g}}_\theta]\!] = \frac{1}{B}([\![\overline{\boldsymbol{g}}_\theta]\!] + [\![\boldsymbol{\mathcal{N}}_\theta]\!])$, where $[\![\boldsymbol{\mathcal{N}}_\theta]\!] = [\![\boldsymbol{\mathcal{N}}_{\text{Tab}}]\!]_{[t,:]}$.
(7) Update $[\![\boldsymbol{\mathcal{W}}]\!]$ via gradient descent on $[\![\tilde{\boldsymbol{g}}_\theta]\!]$.

Figure 1: Training a Global model with Frozen Local models under Gaussian Mechanism with Shuffling

represented by $\boldsymbol{A} = \boldsymbol{B}_{\text{MF}}\boldsymbol{C}_{\text{MF}}$, which is used to privately estimate the quantity of $\boldsymbol{AG}$, as

$$\widehat{\boldsymbol{AG}} = \boldsymbol{B}_{\text{MF}}(\boldsymbol{C}_{\text{MF}}\boldsymbol{G} + \boldsymbol{Z}) = \boldsymbol{A}(\boldsymbol{G} + \boldsymbol{C}_{\text{MF}}^{-1}\boldsymbol{Z}) \,, \tag{4}$$

where $\boldsymbol{Z}$ is an appropriately scaled Gaussian noise vector whose scale is determined by $\text{sens}(\boldsymbol{C}_{\text{MF}})$, which is the sensitivity of $\boldsymbol{C}_{\text{MF}}$ defined in Definition 1 in [17] and also summarized in D.4. Typically, $\boldsymbol{C}_{\text{MF}}$ is also called a *query matrix* or *encoder* since it encodes $\boldsymbol{G}$ as $\boldsymbol{C}_{\text{MF}}\boldsymbol{G}$, which is expected to be made private.

Recently, by utilizing the banded root square factorization [69], particularly for the SGD algorithm, the SGD workload matrix $\boldsymbol{A}$ can be further constructed through $\boldsymbol{A} = \eta\boldsymbol{A}_{\alpha,\beta}$ where η is the learning rate, and $\boldsymbol{A}_{\alpha,\beta}$ is a lower triangular Toeplitz-matrix. Based on Theorem 1 in [69], given any value $p \in [1, T_{num}]$, the *p-banded square-root* (BSR) of $\boldsymbol{A}_{\alpha,\beta}$: $C_{\alpha,\beta}^{|p|}$, is computed through

$$C_{\alpha,\beta}^{|p|} = \text{LDToep}(c_0, \ldots, c_{T_{num}-1}) \,,$$

where $c_j = \sum_{i=0}^j \alpha^{j-i} r_{j-i} r_i \beta^i$ with $r_i = |\binom{-1/2}{i}|$ for $0 < j < p$, α is the weight decay parameter and β is the momentum strength. Note here $c_0 = 1$ and $c_j = 0$ when $j \geq p$. This p-BSR matrix serves as the query matrix for computing the correlated noise and is treated as a public parameter, computed during the pre-processing phase and sent in plaintext to the servers. We use Ω to denote the p-BSR matrix for simplicity. Moreover, we set $p = \mathrm{b}$ for repeated data participation. Under b-min-separated-participation, the necessary amount of noise depends on the *sensitivity* of Ω, $\text{sens}_{\kappa,\mathrm{b}}(\Omega)$, which is defined in Theorem 2 in [69] and also summarized in D.5.

After pre-processing, our protocol works as follows: Similar to the protocol in Section 6.1.1, we still describe the protocol using the same three main components.

Forward-passing and Secret-sharing. Each client still passes all batches of local data to obtain the intermediate outputs $\mathbf{H}_b^{(i)}$ for $b \in [1, M/B]$ at the *beginning* and then secret-shares the intermediate outputs to the servers. Similarly, $\mathbf{y}_b$ with $b \in [1, M/B]$ are also secret-shared to the servers. Since all local models are frozen, this upstream communications are still only done once before the *first* epoch. After receiving all uploads, the servers vertically concatenate the shares of the intermediate outputs from all clients for every batch b to obtain $[\![\mathbf{H}_b]\!]$ with $b \in [1, M/B]$. Unlike Section 6.1.1, note here the servers *will not* shuffle the horizontally concatenated version of $[\![\mathbf{H}_b]\!]$ and $[\![\mathbf{y}_b]\!]$ by $b \in [1, M/B]$, which are $[\![\overline{\mathbf{H}}]\!]$ and $[\![\overline{\mathbf{y}}]\!]$, but store $[\![\mathbf{H}_b]\!]$ and $[\![\mathbf{y}_b]\!]$ in their original order from $b = 1$ to M/B to align with the participation pattern defined before.

Secure Gradients Computation for Global Model. In the training phase, for *each* step t, the servers feed $[\![\mathbf{H}_b]\!]$ into the global model for conducting forward-passing through Π_{FWD}. Once the losses for all samples are computed, the back-propagation is conducted by Π_{BWD}, and the per-sample gradients for the global model $[\![\boldsymbol{g}_{\theta,j}]\!]$ with $j \in [1, B]$ are computed.

Noise Addition for Securely Computed Information. After clipping the per-sample gradients, which is conducted through Eq. (3), the aggregated gradient $[\![\overline{\boldsymbol{g}}_\theta]\!] = \sum_{j=1}^B [\![\overline{\boldsymbol{g}}_{\theta,j}]\!]$ is obtained. Unlike Section 6.1.1, the servers now add correlated noise to the aggregated gradient under the BandMF mechanism. During the pre-processing phase, the servers run the noise sampling protocol Π_{DP} with input $(T_{num}, n_\theta, \sigma \cdot \gamma \cdot \text{sens}_{\kappa,\mathrm{b}}(\Omega))$ to generate $[\![\boldsymbol{\mathcal{N}}_{\text{Tab}}]\!]$. Then, the servers compute $[\![\boldsymbol{\mathcal{N}}_{\text{Corr}}]\!] = \Omega^{-1}[\![\boldsymbol{\mathcal{N}}_{\text{Tab}}]\!]$. Hence at the step t, the t-th row of $[\![\boldsymbol{\mathcal{N}}_{\text{Corr}}]\!]$ is added to the aggregated gradient, i.e., $[\![\boldsymbol{\mathcal{N}}_\theta]\!] = [\![\boldsymbol{\mathcal{N}}_{\text{Corr}}]\!]_{[t,:]}$. Eventually, the servers compute the privatized gradient through $[\![\tilde{\boldsymbol{g}}_\theta]\!] = \frac{1}{B}([\![\overline{\boldsymbol{g}}_\theta]\!] + [\![\boldsymbol{\mathcal{N}}_\theta]\!])$, which is then used to update the global model $[\![\boldsymbol{\mathcal{W}}]\!]$. The formal description of this BandMF mechanism based protocol is shown in Fig. 2.

Protocol $\Pi_{\text{Global-BMF}}$

Parameters: Clients $\mathcal{P}_C = E_1, \ldots, E_N$, aggregation servers $\mathcal{P}_S = \{P_k\}_{k=1}^K$.
Preprocessing: The servers run Π_{GS} with $(T_{num}, n_{\boldsymbol{\theta}}, \sigma\gamma\text{sens}_{\kappa,\text{b}}(\Omega))$ to receive $[\![\boldsymbol{\mathcal{N}}_{\text{Tab}}]\!]$. The servers compute $[\![\boldsymbol{\mathcal{N}}_{\text{Corr}}]\!] = \Omega^{-1}[\![\boldsymbol{\mathcal{N}}_{\text{Tab}}]\!]$.

Client-Server Phase: Proceeds the same steps as in Fig. 1, with the clients sending $[\![\mathbf{H}_b^{(i)}]\!]$ and $[\![\mathbf{y}_b]\!]$ for $b \in [1, M/B]$ to $\mathcal{P}_S$.

Server MPC: At the beginning of each epoch, $\mathcal{P}_S$ do the following:

(1) Only if at the beginning of the *first* epoch, vertically concatenate $[\![\mathbf{H}_b^{(i)}]\!]$ received from $\mathcal{P}_C$ for $i \in [1, N]$, denoted by $[\![\mathbf{H}_b]\!]$ with $b \in [1, M/B]$.

For each step in the current epoch, $\mathcal{P}_S$ do the following:

(1) Given $[\![\mathbf{H}_b]\!]$ and $[\![\mathbf{y}_b]\!]$, follow the same steps from 1 to 5, as described in Fig. 1, to obtain $[\![\overline{\boldsymbol{g}}_{\boldsymbol{\theta}}]\!]$.
(2) Locally add noise to the aggregated gradients as $[\![\tilde{\boldsymbol{g}}_{\theta}]\!] = \frac{1}{B}(\overline{\boldsymbol{g}}_{\boldsymbol{\theta}} + [\![\boldsymbol{\mathcal{N}}_{\boldsymbol{\theta}}]\!])$, where $[\![\boldsymbol{\mathcal{N}}_{\boldsymbol{\theta}}]\!] = [\![\boldsymbol{\mathcal{N}}_{\text{Corr}}]\!]_{[t,:]}$.
(3) Update $[\![\boldsymbol{\mathcal{W}}]\!]$ via gradient descent on $[\![\tilde{\boldsymbol{g}}_{\theta}]\!]$.

Figure 2: Training a Global model with Frozen Local models under Banded Matrix Factorization Mechanism

6.2 Training a Global and the Local models with DP

The two protocols introduced in Section 6.1 enable privacy-preserving training of the global model while freezing the local models. In practice, however, clients may also have the incentive to update their local models: A client may not always have a good pre-trained model for all tasks to start with; meanwhile, clients may also want to gain some insights from participating in the VFL for future tasks. Also, as mentioned earlier, directly revealing per-sample gradients from servers would leak too much private information to the clients [93]. These practical needs motivate us to design an alternative protocol that supports the update of local models while preserving privacy.

However, naïvely applying privacy-enhancing techniques to compute the chain-rule for local model updates in Section 5 may be infeasible. Notice that a client needs per-sample gradients of the loss w.r.t. its local model's output from the servers. If the servers publish the per-sample gradients and add DP-preserving noise to each gradient, the amount of noise added will be so tremendously large that the utility is diminished. On the other hand, if the client delegates the local per-sample gradient computation to the servers via MPC, the computation and communication overhead will scale with the size of the local model, which is impractical.

In this section, we introduce a novel training protocol that bypasses the limitation of directly releasing per-sample gradients. The servers release privacy-budget friendly information and the clients can approximate the per-sample gradients from the released information. Note that we will still employ the BandMF mechanism [69], as both upstream and downstream communications between the servers and clients are required at each step in this protocol. This enforces consistency between the information uploaded to the servers and the information received by the clients, ensuring that they correspond to the same mini-batch, which ultimately *precludes* the anonymity of sample participation. We follow the same pre-processing procedure described in Section 6.1.2 to predefine the participation pattern and compute the p-BSR matrix $\boldsymbol{\Omega}$.

Unlike the previous protocols, our full protocol consists of four main components: 1) forward-passing and secret-sharing for intermediate outputs and auxiliary information, 2) secure computation of gradients for the global model and selective layer(s) of local models, 3) privacy-preserving noise addition for the securely computed information including the global model update and auxiliary information for local gradients reconstruction, and 4) reconstruction of per-sample gradient and update of local models.

Forward-passing and Secret-sharing. At each training step t, each client E_i, in addition to secret-sharing $\mathbf{H}_b^{(i)}$ to the servers, also *locally* computes per-sample gradients, $\partial \boldsymbol{H}_{b,j}^{(i)} / \partial \boldsymbol{\phi}_i^L$, for $j \in [1, B]$, with respect to the model parameters $\boldsymbol{\phi}_i^L$ of a client chosen layer(s) L of the local model, and secret-shares those gradients to the servers. These gradients, which are viewed as auxiliary information, will be further processed to construct an equation system for local model gradients estimation. Note that $\boldsymbol{\phi}_i^L$ is allowed to be different at each client. For example, one client could choose the last fully connected (fc) layer, while another client could choose the last LoRA layer [42]. The servers then compute the loss $[\![\mathcal{L}_j]\!]$ for $j \in [1, B]$, through the same steps in previous protocols.

Secure Gradients Computation for Global/Local Models. First, the servers still securely compute the per-sample gradient of the global model $[\![\boldsymbol{g}_{\boldsymbol{\theta},j}]\!]$ for $j \in [1, B]$ via Π_{BWD} as in previous protocols. In addition, the servers securely compute the per-sample gradients for each client's local layer(s) $[\![\boldsymbol{g}_{\boldsymbol{\phi}_i^L,j}]\!]$ for $j \in [1, B]$ using Π_{Mult}: We have shown in Section 6.1 that the servers and the clients can collaborate to securely compute the gradient w.r.t. the intermediate output of the clients. A similar procedure can be used to compute the gradient of loss w.r.t. model parameters in a local model. Note as the client has locally computed $\partial \boldsymbol{H}_{b,j}^{(i)} / \partial \boldsymbol{\phi}_i^L$ and secret-shares the results to the servers during forward-passing, by chain rule, the per-sample gradient w.r.t. $\boldsymbol{\phi}_i^L$ can be securely computed through

$$[\![\boldsymbol{g}_{\boldsymbol{\phi}_i^L,j}]\!] = [\![\frac{\partial \mathcal{L}_j}{\partial \boldsymbol{\phi}_i^L}]\!] = [\![\frac{\partial \mathcal{L}_j}{\partial \boldsymbol{H}_{b,j}^{(i)}}]\!] \cdot [\![\frac{\partial \boldsymbol{H}_{b,j}^{(i)}}{\partial \boldsymbol{\phi}_i^L}]\!] . \tag{5}$$

Noise Addition for Securely Computed Information. After gradient computation, our protocol adds privacy-preserving noise to the gradients before releasing them. For per-sample gradient clipping, the servers *concatenate* all the gradients under the j-th sample (after flattening), which results in:

$$[\![\boldsymbol{g}_j]\!] = [\![\{\boldsymbol{g}_{\boldsymbol{\theta},j}, \boldsymbol{g}_{\boldsymbol{\phi}_1^L,j}, \ldots, \boldsymbol{g}_{\boldsymbol{\phi}_N^L,j}\}]\!] ,$$

where $\boldsymbol{g}_j \in \mathbb{R}^{n_{\text{conca}}}$, with $n_{\text{con}} = (n_\theta + \sum_{i=1}^{N} n_i^L)$, where n_i^L is the number of parameters in $\boldsymbol{\phi}_i^L$. Then, servers apply clipping on $\boldsymbol{g}_j$:

$$[\![\overline{\boldsymbol{g}}_j]\!] = [\![\boldsymbol{g}_j/\gamma_j]\!]\,, \tag{6}$$

where $\gamma_j = \max\left(1, \frac{\|\boldsymbol{g}_j\|_2}{\gamma}\right)$ for $j \in [1, B]$. After clipping, the servers aggregate all clipped gradients to obtain $[\![\overline{\boldsymbol{g}}]\!] = \sum_{j=1}^{B}[\![\overline{\boldsymbol{g}}_j]\!]$ and add the correlated noise under the BMF mechanism to the aggregated gradient by $[\![\tilde{\boldsymbol{g}}]\!] = \frac{1}{B}([\![\overline{\boldsymbol{g}}]\!] + [\![\boldsymbol{\mathcal{N}}_{\tilde{\boldsymbol{g}}}]\!])$ where $\boldsymbol{\mathcal{N}}_{\tilde{\boldsymbol{g}}}$ is obtained through $[\![\boldsymbol{\mathcal{N}}_{\tilde{\boldsymbol{g}}}]\!] = \left[\Omega^{-1}[\![\boldsymbol{\mathcal{N}}_{\text{Tab}}]\!]\right]_{[t,:]}$ and that $[\![\boldsymbol{\mathcal{N}}_{\text{Tab}}]\!]$ is computed by running Π_{GS} with the input of $(T_{num}, n_{\text{conca}}, \sigma\cdot\gamma\cdot\text{sens}_{\kappa,\text{b}}(\Omega))$. After obtaining $[\![\tilde{\boldsymbol{g}}]\!]$, servers use $[\![\tilde{\boldsymbol{g}}_\theta]\!]$, which is the gradient of global model part, to update the global model $[\![\boldsymbol{\mathcal{W}}]\!]$. Meanwhile, the servers take the gradient of the chosen local layer(s) for each client $\tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L}$ out from $[\![\tilde{\boldsymbol{g}}]\!]$, and distribute the gradient back to E_i, for $i \in [1, N]$.

Local Per-sample Gradient Reconstruction and Model Updates. On the client's end, since $\tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L}$ is the privatized gradient (aggregated at batch-level) with noise added, it cannot be directly used to conduct backpropagation on the local model: Recall that in order to recover the back-propagation flow in the plain-text model in Section 5, the client needs $\{\partial\mathcal{L}_j/\partial\boldsymbol{H}_{b,j}^{(i)}\}_{j=1}^{B}$, i.e., the per-sample gradient w.r.t. the intermediate output from the client. Although recovering the exact per-sample gradient is infeasible as the received $\tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L}$ is privacy-preserved, we show that a client can estimate the clipped per-sample gradient w.r.t. its intermediate output. In particular, We can establish the following linear equation system, based on the chain-rule applied on $\tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L}$:

$$\begin{aligned}
\tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L} = \begin{bmatrix} \tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L}[1] \\ \vdots \\ \tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L}[n_i^L] \end{bmatrix} &= \frac{1}{B}\begin{bmatrix} \frac{\partial \boldsymbol{H}_{b,1}^{(i)}}{\partial \boldsymbol{\phi}_i^L[1]} & \frac{\partial \boldsymbol{H}_{b,2}^{(i)}}{\partial \boldsymbol{\phi}_i^L[1]} & \cdots & \frac{\partial \boldsymbol{H}_{b,B}^{(i)}}{\partial \boldsymbol{\phi}_i^L[1]} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \boldsymbol{H}_{b,1}^{(i)}}{\partial \boldsymbol{\phi}_i^L[n_i^L]} & \frac{\partial \boldsymbol{H}_{b,2}^{(i)}}{\partial \boldsymbol{\phi}_i^L[n_i^L]} & \cdots & \frac{\partial \boldsymbol{H}_{b,B}^{(i)}}{\partial \boldsymbol{\phi}_i^L[n_i^L]} \end{bmatrix} \cdot \\
&\quad \overline{\boldsymbol{\Gamma}} \cdot \begin{bmatrix} \frac{\partial \mathcal{L}_1}{\partial \boldsymbol{H}_{b,1}^{(i)}} \\ \vdots \\ \frac{\partial \mathcal{L}_B}{\partial \boldsymbol{H}_{b,B}^{(i)}} \end{bmatrix} + \frac{1}{B}\begin{bmatrix} \boldsymbol{\mathcal{N}}_i[1] \\ \vdots \\ \boldsymbol{\mathcal{N}}_i[n_i^L] \end{bmatrix} \\
&= \frac{1}{B}(\boldsymbol{H}_{\boldsymbol{\phi}_i^L} \cdot \overline{\boldsymbol{\Gamma}} \cdot \boldsymbol{g}_{\mathbf{H}_b^{(i)}} + \boldsymbol{\mathcal{N}}_i) \\
&= \frac{1}{B}(\boldsymbol{H}_{\boldsymbol{\phi}_i^L} \cdot \widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}} + \boldsymbol{\mathcal{N}}_i)\,,
\end{aligned} \tag{7}$$

where matrix $\boldsymbol{H}_{\boldsymbol{\phi}_i^L} = \left\{\partial\boldsymbol{H}_{b,j}^{(i)}/\partial\boldsymbol{\phi}_i^L[l]\right\}_{j=1}^{B} \in \mathbb{R}^{n_i^L \times (d_{\boldsymbol{H}^{(i)}}B)}$ with $l \in [1, n_i^L]$, and vector $\boldsymbol{g}_{\mathbf{H}_b^{(i)}} = \left\{\partial\mathcal{L}_j/\partial\boldsymbol{H}_{b,j}^{(i)}\right\}_{j=1}^{B} \in \mathbb{R}^{d_{\boldsymbol{H}^{(i)}}B}$. More specifically,

$$\partial\boldsymbol{H}_{b,j}^{(i)}/\partial\boldsymbol{\phi}_i^L[l] = \left[\partial\boldsymbol{H}_{b,j}^{(i)}[1]/\partial\boldsymbol{\phi}_i^L[l], \ldots, \partial\boldsymbol{H}_{b,j}^{(i)}[d_{\boldsymbol{H}^{(i)}}]/\partial\boldsymbol{\phi}_i^L[l]\right]\,,$$

and

$$\partial\mathcal{L}_j/\partial\boldsymbol{H}_{b,j}^{(i)} = \left[\partial\mathcal{L}_j/\partial\boldsymbol{H}_{b,j}^{(i)}[1], \ldots, \partial\mathcal{L}_j/\partial\boldsymbol{H}_{b,j}^{(i)}[d_{\boldsymbol{H}^{(i)}}]\right]^T\,.$$

The term $\overline{\boldsymbol{\Gamma}} \in \mathbb{R}^{(d_{\boldsymbol{H}^{(i)}}B)\times(d_{\boldsymbol{H}^{(i)}}B)}$ is a diagonal matrix: $\overline{\boldsymbol{\Gamma}} = \text{diag}(\boldsymbol{\Gamma}_j)$ where each diagonal entry, $\boldsymbol{\Gamma}_j \in \mathbb{R}^{d_{\boldsymbol{H}^{(i)}}\times d_{\boldsymbol{H}^{(i)}}}$, is also a diagonal matrix whose diagonal entries are γ_j^{-1}, i.e., $\boldsymbol{\Gamma}_j = \text{diag}(\gamma_j^{-1})$. The term $\boldsymbol{\mathcal{N}}_i \in \mathbb{R}^{n_i^L}$ is a sub-vector in $\boldsymbol{\mathcal{N}}_{\tilde{\boldsymbol{g}}}$ which is the noise that was added to privatize $\tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L}$.

The key intuition is that the equation in Eq. (7) is a linear system with noisy measurement: We can estimate $\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}$, which is

$$\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}} \triangleq \overline{\boldsymbol{\Gamma}} \cdot \boldsymbol{g}_{\mathbf{H}_b^{(i)}} = \{\gamma_j \cdot \partial\mathcal{L}_j/\partial\boldsymbol{H}_{b,j}^{(i)}\}_{j=1}^{B}\,, \tag{8}$$

from the locally computed $\boldsymbol{H}_{\boldsymbol{\phi}_i^L}$ and the observed $\tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L}$. Since the gradients are estimated from privacy-preserving outputs from the servers, the client can apply standard solvers such as LS (least squares), MMSE (minimum mean square error), RR (ridge regression) or any other linear estimation techniques locally to solve Eq. (7) under the *plaintext model*. Note our protocol is estimator-agnostic. We show how estimation techniques can extract per-sample gradients from the aggregated private gradients to enable backpropagation, without relying on any specific estimator. Exploring alternative estimators and their potential performance gains is left for future work.

Once $\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}$ is estimated, the i-th client can update its local layers by back-propagation. The complete protocol is presented in Fig. 3.

6.2.1 Estimation Performance Analysis. Typically, to obtain a stable and reliable estimate in a linear system with noise, it is desirable to have at least as many observations as unknowns. Under this condition, the system becomes either exactly determined or overdetermined. In our case, this *translates* to $n_i^L \geq d_{\boldsymbol{H}^{(i)}}\cdot B$, which ensures that the estimation problem remains well-posed and can be solved robustly. In general, aside from the ratio between n_i^L and $d_{\boldsymbol{H}^{(i)}}\cdot B$, the estimation error is dependent on the intrinsic properties of the matrix $\boldsymbol{H}_{\boldsymbol{\phi}_i^L}$, the characteristics of the added noise, and the choice of estimator as well as any assumptions on $\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}$ if needed. Although various estimators could be used in the linear system, we take RR estimator, which is defined by

$$\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}^{\text{ridge}} = (\boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T\boldsymbol{H}_{\boldsymbol{\phi}_i^L} + \lambda I)^{-1}\boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T\tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L}\,, \tag{9}$$

where λ is regularization parameter, as an illustrative example, as RR is more stable than LS in ill-conditioned systems and also unlike MMSE, does not require a prior on $\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}$'s distribution, while still allowing bounded-error analysis. As a common heuristic in RR, we set λ to the noise variance. Moreover, as the ℓ_2 norm of $\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}$ is bounded by γ, the estimation error under RR estimator for Eq. (7), is bounded by

$$\mathbb{E}_{\text{Ridge}} \leq \frac{\sigma_t^4\gamma^2 + \sigma_t^2 d_{\boldsymbol{H}^{(i)}}B^3\mu_{\max}}{(B^2\mu_{\min} + \sigma_t^2)^2}\,, \tag{10}$$

where $\sigma_t = \sigma\gamma\text{sens}_{\kappa,\text{b}}(\Omega)\|\Omega^{-1}[t,:]\|_{\ell_2}^2$, $\mu_{\min}$ and $\mu_{\max}$ are the smallest and largest eigenvalues of $\boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T\boldsymbol{H}_{\boldsymbol{\phi}_i^L}$, respectively. The detailed derivation for Eq. (10) and further discussion are in Appendix E.

7 Security and Privacy Analysis

7.1 Input Privacy

Our protocols leak no intermediate values and all computation on the servers is performed entirely under MPC on secret-shared data. Input privacy therefore follows directly from the provable

Protocol $\Pi_{\text{Global-Local-BMF}}$

Parameters: Clients $\mathcal{P}_C = E_1, \ldots, E_N$, aggregation servers $\mathcal{P}_S = \{P_k\}_{k=1}^K$.
Preprocessing: The servers run Π_{GS} with input $(T_{num}, n_{\text{con}}, \sigma\gamma\text{sens}_{\kappa,\text{b}}(\Omega))$ to receive $[\![\boldsymbol{\mathcal{N}}_{\text{Tab}}]\!]$. The servers compute $[\![\boldsymbol{\mathcal{N}}_{\text{Corr}}]\!] = \Omega^{-1}[\![\boldsymbol{\mathcal{N}}_{\text{Tab}}]\!]$.

Client-Server Phase: In each epoch, for each step, each client E_i does the following:

(1) Proceeds the same steps as described in Fig. 1, but only computes and sends the b-th batch's $[\![\mathbf{H}_b^{(i)}]\!]$ and $[\![\mathbf{y}_b]\!]$ to $\mathcal{P}_S$.
(2) Locally computes $\{\partial \boldsymbol{H}_{b,j}^{(i)} / \partial \boldsymbol{\phi}_i^L\}$, and secret-shares $[\![\partial \boldsymbol{H}_{b,j}^{(i)} / \partial \boldsymbol{\phi}_i^L]\!]$ for $j \in [1, B]$, then sends the shares to $\mathcal{P}_S$.

Server MPC: For each step, $\mathcal{P}_S$ do the following:

(1) Vertically concatenate $[\![\mathbf{H}_b^{(i)}]\!]$ received from $\mathcal{P}_C$ for $i \in [1, N]$, which results in $[\![\mathbf{H}_b]\!]$.
(2) Given $[\![\mathbf{H}_b]\!]$ and $[\![\mathbf{y}_b]\!]$, follow the same steps from 1 to 2, as described in Fig. 1, and eventually obtain $[\![\mathcal{L}_j]\!]$ and $[\![\boldsymbol{g}_{\theta,j}]\!]$ for $j \in [1, B]$.
(3) Compute $[\![\boldsymbol{g}_{\boldsymbol{\phi}_i^L,j}]\!] = [\![\frac{\partial \mathcal{L}_j}{\partial \boldsymbol{\phi}_i^L}]\!] = [\![\frac{\partial \mathcal{L}_j}{\partial \boldsymbol{H}_{b,j}^{(i)}}]\!] \cdot [\![\frac{\partial \boldsymbol{H}_{b,j}^{(i)}}{\partial \boldsymbol{\phi}_i^L}]\!]$ for $j \in [1, B]$, with $i \in [1, N]$, using Π_{Mult}.
(4) Concatenate all j-th sample's gradient together to obtain $[\![\boldsymbol{g}_j]\!]$, for $j = [1, B]$.
(5) Follow the same steps from 3 to 5, as described in Fig. 1, and apply them to $[\![\boldsymbol{g}_j]\!]$, for $j \in [1, B]$, and eventually obtain $[\![\overline{\boldsymbol{g}}]\!]$.
(6) Add noise on the aggregated gradients to obtain $[\![\tilde{\boldsymbol{g}}]\!] = \frac{1}{B}([\![\overline{\boldsymbol{g}}]\!] + [\![\boldsymbol{\mathcal{N}}_{\tilde{\boldsymbol{g}}}]\!])$, where $[\![\boldsymbol{\mathcal{N}}_{\tilde{\boldsymbol{g}}}]\!] = [\![\boldsymbol{\mathcal{N}}_{\text{Corr}}]\!]_{[t,:]}$.
(7) Update $[\![\boldsymbol{\mathcal{W}}]\!]$ via gradient descent based on $[\![\tilde{\boldsymbol{g}}_\theta]\!]$. Reconstruct $\tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L}$ and send it back to E_i, for $i \in [1, N]$.

Server-Client Phase: For each step, each client E_i does the following:

(1) Solves Eq. (7) to get the estimation of the clipped per-sample gradients: $\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}$, based on $\tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L}$.
(2) Update the desired local layers through gradient descent, by applying back-propagation on $\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}$.

Figure 3: Training a Global model and Local models under Banded Matrix Factorization Mechanism

security of the employed MPC building blocks. Concretely, every sub-protocol used in our constructions operates in the *arithmetic black-box* ($\mathcal{F}_{ABB}$) hybrid model, whose operations of multiplication, comparison, bit decomposition, truncation, and domain conversion are each UC-secure. We only ever make black-box use of these subprotocols implemented in MP-SPDZ [48], and by the UC composition theorem [13], the composed protocol is itself secure. We provide the theorem statement below:

Theorem 1 (Input privacy)**.** Protocols $\Pi_{\text{Global-Shuffle}}$, $\Pi_{\text{Global-BMF}}$, and $\Pi_{\text{Global-Local-BMF}}$ are secure against a static, semi-honest adversary corrupting at most one of the three servers, in the $\mathcal{F}_{ABB}$-hybrid model. That is, the corrupted server's view during protocol execution can be efficiently simulated given only its input shares and output shares.

Proof. The details of the proof are given in C.1 in Appendix C. □

7.2 Output Privacy

To analyze the privacy guarantee of the proposed protocols, we present the following theorems:

Theorem 2. The protocol in Fig. 1 is (ϵ, δ) differentially private (D.1) with respect to each client E_i's dataset $\mathcal{D}_i$, with $i \in [1, N]$, under Gaussian Mechanism (D.2).

Proof. The details of the proof are given in D.1 in Appendix D. □

Theorem 3. The protocol in Fig. 2 is (ϵ, δ) differentially private (D.1) with respect to each client E_i's dataset $\mathcal{D}_i$, with $i \in [1, N]$, under Matrix Factorization Mechanism (D.6).

Proof. The details of the proof are given in D.2 in Appendix D. □

Theorem 4. The protocol in Fig. 3 is (ϵ, δ) differentially private (D.1) with respect to each client E_i's dataset $\mathcal{D}_i$, with $i \in [1, N]$, under Matrix Factorization mechanism (D.6).

Proof. The details of the proof are given in D.3 in Appendix D. □

7.3 Additional Discussion

Label Privacy. In our setup specified in Section 4, the client storing the label information is the N-th client E_N. This label client is treated identically to the other clients in our protocols. Labels remain protected because they are never revealed to any server in plaintext, and only operated upon through secret shares, under the semi-honest, honest majority assumption. The DP guarantees in Section 7.2 apply to the label party. DP's post-processing property prevents label recovery from noisy aggregated gradients.

Sources of Privacy. When only the global model is trained, no messages are passed to the clients, and thus no information is leaked to the clients. The training of the global model, including gradient computation, DP-noise addition, and model update, are all computed via MPC over the secret-shared data. Only the final model will be reconstructed and published. The servers, which are honest-but-curious, will have the same view as a user of the published model, and thus can only infer as much information as the same DP-guarantee allows.

When both the global and local models are updated, each client will additionally see a privatized gradient with respect to its own

batch data reconstructed from the servers at every training step. The privacy guarantee of the underlying banded matrix factorization mechanism applies to the full sequence of privatized gradients, and hence all intermediate models are DP-protected [16].

8 Experimental Evaluation

We empirically evaluate the performance of our protocols. Specifically, we are interested in:

(1) the model utility of our protocols in various settings,
(2) guidelines for choosing between the protocols based on their strengths and limitations, and
(3) the practicality of our protocols in implementation.

The empirical results show that our protocols achieve competitive utility while preserving privacy; the test accuracies significantly surpass those of local-DP methods and are competitive with other baselines across various privacy budgets ε. Both Gaussian and BMF mechanisms can achieve good performance; each has its own advantage depending on the similarity between the public dataset for pre-training and the private dataset for the actual task. Local model finetuning offers consistent performance improvement when the learner can afford more resources for computation and communication. Our benchmarks for MPC running time and communication costs show that our protocols are practically efficient.

8.1 Experimental Setup

Plaintext Setup. The plaintext experiments were conducted on a Dell PowerEdge XE8545 rack server with 256 CPU cores and 1 TB RAM, with an Nvidia A100 GPU.

MPC Setup. MPC protocols were run on a MacBook Pro with an Apple Silicon M4 Pro processor, and 48 GB RAM on a single thread. The network connections were simulated using packet filtering (pfctl and dnctl). More specifically, we benchmark the online phases of our protocols in two setups, LAN, and WAN. On LAN, we simulate a bandwidth of 1 Gbps and a delay of 1ms, whereas on WAN we use a bandwidth of 200 Mbps and a delay of 20ms.

We implement our protocols in the MP-SPDZ [48] framework. Specifically, we use the replicated-ring protocol with three parties and passive security. The required correlated randomness, such as edaBits, was generated in advance.

Framework. We consider the FL setting where the dataset is vertically split between 2 clients ($N = 2$). Each client has a local ResNet-18 [41] model. We consider *two* scenarios, **Pre-ImageNet** and **Pre-Cifar100**, based on how the local models are pretrained:

In **Pre-ImageNet**, each client loads the ResNet-18 that was pre-trained on the ImageNet dataset [25]. This model is publicly available in [73]. The output size of the pre-trained model is 1000, i.e., $d_{\boldsymbol{H}^{(i)}} = 1000$ for $i \in [1, N]$.

In **Pre-Cifar100**, the local model is pre-trained on the CIFAR-100 dataset [54] which is treated as the public dataset in our case, under the **plaintext model**. We set the output size of the pre-trained model to 512, i.e., $d_{\boldsymbol{H}^{(i)}} = 512$ for $i \in [1, N]$. Enhanced by data augmentation methods including data normalization and random flipping during training, the pre-trained model achieves an average testing accuracy of 77.19%. On the servers' end, we deploy one fully-connected layer as the global model.

We also note that using deeper networks or advanced techniques, such as Sharpness-Aware Minimization (SAM), can lead to significant improvements in accuracy, as shown in [35]. However, our goal is not to employ state-of-the-art methods to maximize performance. Instead, we adopt a standard training approach that yields moderate yet reasonable results. This simplicity makes our setting a practical choice for real-world deployment scenarios.

Datasets. We evaluate our framework on CIFAR-10 [54] and EMNIST [19], which are treated as private datasets in our experiments. As mentioned above, CIFAR-100 dataset is the public dataset used for pre-training. Each data sample in all datasets is vertically partitioned based on the number of participating clients, and the corresponding feature subset is subsequently forwarded to the respective client's local model during forward-passing.

For EMNIST, we split the dataset into 26 classes by letters. We assume that the data distribution across all clients is independently and identically distributed (IID). As DP mechanisms provide distribution-independent theoretical privacy guarantees, the IID or Non-IID setting under our framework *does not* affect the formal DP protection. We acknowledge that heterogeneous data distributions may influence model utility or empirical attack performance [50, 61]. However, a detailed study of their impact on practical privacy leakage is beyond the scope of this work.

Protocol Setups. We use **Plain** to denote the plaintext protocol in Section 5, **G-Shuff** to represent the protocol described in Section 6.1.1, **G-BMF** to denote the protocol in Section 6.1.2, and **GL-BMF** to denote the protocol introduced in Section 6.2.

In **GL-BMF**, each client solves the linear equations Eq. (7) through the RR estimator with $\lambda = (\sigma\gamma \mathrm{sens}_{\kappa,\mathrm{b}}(\Omega)\|\Omega^{-1}[t,:]\|_{\ell_2}^2/B)^2$, at each step. Besides, we apply LoRA finetuning [42] to clients' local models. Instead of modifying all model parameters of the local model, we insert LoRA layers into the local model architecture as the *adapter* to the new task. More specifically, we insert one LoRA layer to each block of the local ResNet model and only the LoRA layers are updated during training. We also organize the LoRA layers in *parallel*: the output of each LoRA layer is directly added to the final output of the local model instead of the next ResNet block. This choice is inspired by the finding in [56, 79]: Intermediate features can capture rich hierarchical information, hence incorporating LoRA layers in a parallel structure enables efficient adaptation across multiple feature extraction layers. Moreover, since the estimated per-sample gradients $\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}$ contain noise, our parallel structure, which allows each layer to be updated independently, may mitigate the accumulation of noise as opposed to the conventional *sequential* LoRA structure where noise propagates across layers and is particularly amplified on the bottom layers during back-propagation.

In addition, under our parallel LoRA structure, we have

$$\boldsymbol{H}_{b,j}^{(i)} = \boldsymbol{H}_{b,j}^{(i),o} + \boldsymbol{H}_{b,j}^{(i),L_1} + \cdots + \boldsymbol{H}_{b,j}^{(i),L_M},$$

where $\boldsymbol{H}_{b,j}^{(i),o}$ is the output of the original model and $\boldsymbol{H}_{b,j}^{(i),L_l}$ is the output of the l-th LoRA layer, where M is the total number of LoRA layers. Based on the principles of differentiation, we have

$$\frac{\partial \mathcal{L}_j}{\partial \boldsymbol{H}_{b,j}^{(i)}} = \frac{\partial \mathcal{L}_j}{\partial \boldsymbol{H}_{b,j}^{(i),L_1}} = \cdots = \frac{\partial \mathcal{L}_j}{\partial \boldsymbol{H}_{b,j}^{(i),L_M}}.$$

Thus, instead of uploading all M layers' $\partial \boldsymbol{H}_{b,j}^{(i),L_l}/\partial \boldsymbol{\phi}_i^{L_l}$ with $j \in [1, B]$ where $\boldsymbol{\phi}_i^L = \{\boldsymbol{\phi}_i^{L_l}\}_{l=1}^M$ to servers for computing $\{\tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^{L_l}}\}_{l=1}^M$, each client could just upload any one layer's $\partial \boldsymbol{H}_{b,j}^{(i),L_l}/\partial \boldsymbol{\phi}_i^{L_l}$ to the servers to compute $\tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^{L_l}}$, and hence obtain the estimation $\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i),L_l}}$. We then approximate $\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}} \approx \widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i),L_l}}$, and update all LoRA layers by $\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}$. Here we set $r = 8$ and $\alpha_{\text{LoRA}} = 1$ for all LoRA layers across all clients in our experiments.

Parameters. In our experiments, we use cross-entropy as the criterion and SGD as the optimization algorithm. We fix $B = 128$, and set E_{num} to be 20 and 100, respectively. For DP testing, we set ℓ_2 norm bound $\gamma = 1.2$, $\delta = 10^{-5}$, and various privacy budgets with $\epsilon = 1, 2, 5, 8, 10$. For the BMF mechanism, we consider two sets of learning parameters:

- **Setting 1**: $(\alpha = 1, \beta = 0)$
- **Setting 2**: $(\alpha = 1, \beta = 0.9)$

Besides, we set learning rate $\eta_s = \eta_i = 0.01$ on both server and clients for **Plain**, and $\eta_s = 0.01$ on servers for **G-Shuff** and **G-BMF**. For **GL-BMF**, we set $\eta_s = 0.01$ and $\eta_i = 0.001$ for $i \in [1, N]$. Here we use smaller η_i to mitigate the impact of the estimation error contained in the estimated per-sample gradients. All the reported test accuracies are averaged over **four** independent runs.

DP Accounting. For DP accounting, for the subsampling-based protocol, we follow the DP-accounting and composition summarized in [1, 64] and implemented using DP library Opacus [92]; For the BMF-based protocols, we use Google's FFT based DP accounting library [40], following the exact setup in [69].

Local Differential Privacy (LDP). In the VFL setting with LDP, each individual perturbs their own output by adding noise, without relying on a trusted party. To provide a direct comparison with our method, we adopt LDP as one of the baselines. Same as the work in [87], we have each client independently privatize its data before transmitting to the server. Specifically, each client applies norm clipping to its per-sample intermediate outputs $\boldsymbol{H}_{b,j}^{(i)}$ for $j \in [1, B]$ to bound sensitivity. A calibrated Gaussian noise is then added to each clipped output. As a result, each party releases only perturbed intermediate outputs to the servers. However, unlike [87], which utilizes subsampling enabled by fuzzy linkage (one-to-many linkage), we adopt the conventional VFL setting where each primary sample has exactly one predetermined match in other parties. Hence the sample-level participation anonymity among clients is no longer preserved after data alignment, and privacy amplification that relies on subsampling is also no longer applicable. We use **LDP-G** and **LDP-GL** to denote the mechanisms for updating global model only and global-local models both under LDP, respectively.

Existing Approaches. We further compare our protocols against existing privacy-preserving VFL approaches, which can be viewed as *variants* of LDP under *different* trust assumptions, used as baselines. Specifically, we consider **ADMM** [88], **Split Learning** [82], and **FedBCD** [58] as in [88], along with **Ada-VFed** [39]. To ensure a fair comparison, we adopt the same experimental configuration as **Ada-VFed**: In particular, two clients each deploy a ResNet-18 model locally, and the server employs an MLP layer as the global model. The clipping threshold is set to $\gamma = 2$, the number of epochs $E_{num} = 20$, the batch size $B = 128$, and various privacy budgets with $\epsilon = 1, 4, 8$. All methods are evaluated on the CIFAR-10 dataset. For **ADMM** and **FedBCD**, we evaluate multiple numbers of local update steps, including $\{1, 5, 10, 30\}$, and fix the value to 5, as it achieves the best performance among these settings. Similarly, after extensive hyperparameter tuning, we fix $\rho = 0.5$ for **ADMM**.

Membership Inference Attacks (MIAs). MIAs are privacy attacks which aim to determine whether a target data point (x, y), was included in the given model's training set. As described in Section 4.3, we assess the robustness of our privacy-preserving protocols against these adversaries by conducting black-box membership inference attacks with the Privacy Meter introduced in [72]. Specifically, we use the enhanced Population Attack (**P-Attack**) proposed in [90] to evaluate the privacy leakage. We adopt the same attack setting as in [6, 90], where the validation dataset which is never used by the model during training, is used to construct the population distribution and compute the attack threshold. When evaluating attacks, we keep using a balanced evaluation dataset: 50% member (from training dataset) and 50% non-member (from test dataset).

8.2 Benchmarks

Utility-Privacy Evaluation. Tab. 1 shows the utilities for different protocols under different privacy budgets after 20 training epochs. We observe that all our protocols yield good utilities for both scenarios and both datasets. In the **Pre-Cifar100** scenario, our protocols perform exceptionally well as the difference in final model accuracies between the plaintext upperbound and our privacy-preserving protocols is less than 15% across all ε. The performance of BMF mechanisms can depend on the choice of hyperparameters in learning. We consider the better of Setting 1 and 2 as it shows the potential of the mechanism with the right choice of hyperparameters. The **Pre-ImageNet** scenario is more challenging because the images used for pretraining differ more from the test data than **Pre-Cifar100** in terms of image size and contents. Nevertheless, our protocols still achieve competitive performances of less than 20% accuracy gap in most cases except for small $\varepsilon = 1, 2$.

The second general pattern is that **GL-BMF** consistently outperforms **G-BMF** across all settings. This pattern indicates that local finetuning helps. Although the local models are only updated with *noisy* gradients estimated from the published privacy-preserving information from the servers, the extracted information is still useful. Although the improvement is moderate because of the large amount of noise added in the BMF mechanism, we see potential in the idea of decoding local gradients from privacy-preserving public information. With the potential development of noise-adding mechanisms that consider our decoding step in the design, we may benefit further from local finetuning.

For the choice between **G-Shuff** and **G-BMF**, we notice that **G-BMF**, with the right choice of hyperparameters, generally outperforms **G-Shuff**. However, **G-Shuff** outperforms other baselines in **Pre-Cifar100** – **CIFAR-10**. The relative performance between **G-Shuff** and **G-BMF** is related to the similarity between the dataset for pre-training and the test dataset; the phenomenon is an intriguing direction for further exploration. The BMF mechanism supports

Dataset	Method	Pre-ImageNet					Pre-Cifar100				
	Plain	Non-DP 90.01					Non-DP 91.42				
		DP					DP				
		$\varepsilon = 1$	$\varepsilon = 2$	$\varepsilon = 5$	$\varepsilon = 8$	$\varepsilon = 10$	$\varepsilon = 1$	$\varepsilon = 2$	$\varepsilon = 5$	$\varepsilon = 8$	$\varepsilon = 10$
CIFAR-10	G-Shuff	69.38	71.78	73.27	73.69	73.76	81.19	81.51	81.62	81.67	81.69
	G-BMF (Setting 1)	69.25	70.73	71.19	71.22	71.24	77.22	77.28	77.31	77.34	77.37
	G-BMF (Setting 2)	51.74	58.69	67.29	70.27	71.59	72.03	77.39	80.68	81.12	81.38
	GL-BMF (Setting 1)	69.63	71.09	71.48	71.53	71.56	77.48	77.55	77.77	77.81	77.82
	GL-BMF (Setting 2)	54.41	60.45	68.19	71.01	72.34	72.58	77.86	81.11	81.39	81.42
	Plain	Non-DP 94.12					Non-DP 93.75				
		DP					DP				
		$\varepsilon = 1$	$\varepsilon = 2$	$\varepsilon = 5$	$\varepsilon = 8$	$\varepsilon = 10$	$\varepsilon = 1$	$\varepsilon = 2$	$\varepsilon = 5$	$\varepsilon = 8$	$\varepsilon = 10$
EMNIST	G-Shuff	73.96	76.71	78.37	78.94	79.22	78.58	78.87	79.02	79.12	79.14
	G-BMF (Setting 1)	73.62	74.53	74.89	75.03	75.05	70.88	70.94	70.99	71.04	71.09
	G-BMF (Setting 2)	73.44	78.95	81.48	81.99	82.06	78.69	79.73	80.08	80.09	80.13
	GL-BMF (Setting 1)	74.09	74.84	75.12	75.25	75.28	71.06	71.15	71.19	71.21	71.23
	GL-BMF (Setting 2)	74.04	79.52	81.84	82.18	82.27	78.99	79.94	80.19	80.21	80.24

Table 1: Test Accuracy (%) versus Different Datasets, Pre-training, and Privacy Budgets (20 epochs)

Dataset	ϵ	Method						
		G-Shuff	G-BMF (Setting 1)	G-BMF (Setting 2)	GL-BMF (Setting 1)	GL-BMF (Setting 2)	LDP-G	LDP-GL
CIFAR-10	1	63.71	65.14	49.14	65.97	51.32	10.29	10.56
	2	69.09	70.66	55.16	71.23	56.26	11.14	11.36
	5	71.93	73.63	61.47	74.09	62.29	16.69	17.14
	8	72.89	74.25	64.42	74.59	65.32	23.97	24.51
	10	73.88	74.41	65.85	74.78	66.61	28.16	28.89

Table 2: Test Accuracy (%) versus Different Privacy Budgets, LDP (100 epochs)

Dataset	ϵ	Method						
		G-Shuff	G-BMF	GL-BMF	ADMM	FedBCD	Split Learning	Ada-VFed
CIFAR-10	1	66.71	67.14	68.49	65.79	68.22	29.66	61.31
	4	71.25	70.96	72.52	68.84	70.43	34.41	62.13
	8	72.37	71.91	73.63	70.03	71.68	40.37	63.50

Table 3: Test Accuracy (%) versus Different Privacy Budgets, Baselines (20 epochs, $\gamma = 2$)

Dataset	Method					
	Plain	G-Shuff	G-BMF (Setting 1)	G-BMF (Setting 2)	GL-BMF (Setting 1)	GL-BMF (Setting 2)
CIFAR-10	0.68	0.52	0.51	0.51	0.51	0.51

Table 4: AUC Score versus Different Protocols under P-Attack (100 epochs, $\epsilon = 8$)

local finetuning, which offers more potential room for improvement. On the other hand, the performance of **G-BMF** is sensitive to the choice of hyperparameters in learning. (See the difference between Setting 1 and 2.) One should also consider the privacy budget for model selection in more complex learning settings.

In addition, we trace the performance of all our protocols with a longer training horizon ($E_{num} = 100$). The results are shown in Tab. 7 and the detailed discussion is given in Appendix B.

We also compare our protocols with Local-DP under $E_{num} = 100$. Since there is no prior work with end-to-end input/output privacy guarantees for general VFL, we use local-DP which can be regarded as the variant of [87] without subsampling, as a privacy-preserving benchmark for utility. Tab. 2 shows the utilities for both our protocols and LDP for **CIFAR-10** in the **Pre-ImageNet** scenario. After 100 epochs of training, the performances of both **LDP-G** and **LDP-GL** are underwhelming. The LDP protocols add noise before any

Dataset	Method	Pre-ImageNet		Pre-Cifar100	
		Runtime (s)	**Communication** (MB)	**Runtime** (s)	**Communication** (MB)
CIFAR-10	G-Shuff	14.88	473.63	7.80	244.03
	G-BMF	14.57	453.6	7.65	233.77
	GL-BMF	132.8	6220.87	83.5	3186.62
EMNIST	G-Shuff	32.80	1228.56	18.89	632.52
	G-BMF	32.00	1178.64	18.50	606.96
	GL-BMF	133.87	6945.87	89.33	3559.79

Table 5: Runtime and Communication cost of the MPC over LAN

data and/or statistics leave a client's local storage. The noise, which is added early in the pipeline, causes difficulties for all downstream calculations. In contrast, our combination of MPC and DP is much more privacy-budget friendly as the privacy-preserving noise is added to aggregated statistics later in the pipeline. The significant gap between our protocols and Local-DP shows the advantage of our approach in privacy-utility trade-off.

In Tab. 3, we compare our protocols with other baselines described in Section 8. All the approaches are evaluated on the CIFAR-10 dataset under **Pre-ImageNet** scenario. For approaches from [88], we set $d_{\boldsymbol{H}^{(i)}} = 60$ to remain consistent with their original configuration. Here both **G-BMF** and **GL-BMF** are under Setting 1. In general, our protocols consistently outperform **Ada-VFed** and **Split Learning**: For example, at $\epsilon = 8$, they achieve a minimum of 8% higher performance than **Ada-VFed**, and an even greater improvement over **SplitLn**. This result indicates that, on complex datasets such as CIFAR-10, the Laplacian Score metric used in **Ada-VFed** fails to capture meaningful feature importance, resulting in performance degradation. We also observe that all our protocols outperform **ADMM** across all privacy budget settings. For **FedBCD**, under the strict privacy constraint ($\epsilon = 1$), it outperforms **G-shuff** and **G-BMF** and is competitive with **GL-BMF**. However, when under the relaxed privacy constraints ($\epsilon \geq 4$), all our protocols perform better than **FedBCD**, with **GL-BMF** showing particular improvement.

Furthermore, we find that **FedBCD** performs better than **ADMM** in our experiments, contrary to the reported results in [88]. **ADMM**'s sophisticated consensus mechanism may be sensitive to changes in the experiment configuration, such as the size of clients and local model architecture. This degradation can also be explained by **ADMM**'s sensitivity to hyperparameters: The penalty parameter ρ and dual variable updates must be extensively re-tuned for each different setting, which limits its practical applicability for general usage. In contrast, **FedBCD** is more robust, as it has fewer hyperparameters to tune and its block coordinate descent enables efficient small-scale coordination.

The extended comparisons with additional baseline methods and further discussion are provided in Appendix B.

Tab. 4 reports the effectiveness of the population-based membership inference attack (P-Attack) against our protocols compared to the **Plain** protocol, for **CIFAR-10** in the **Pre-ImageNet** scenario. All evaluations are conducted under a privacy budget of $\epsilon = 8$, representing a relatively loose privacy constraint. Under this setting, the plaintext model reaches an AUC of 0.68, whereas all of our protocols keep the AUC at 0.52 or below, demonstrating strong resistance to the membership inference attack.

MPC Running Evaluation Results. Tab. 5 shows the MPC runtime and communication cost for different protocols over LAN. Note the results presented in both Tab. 5 and Tab. 8 (see in Appendix B) are for a single batch. For simulating the runtime of **G-Shuff** at the batch-level, we first measure the runtime of shuffling on $[\![\overline{\mathbf{H}}]\!]$, and then divide the runtime by B_{num}. The shuffling runtime at the batch-level is then added to the time spent in the rest of the protocol for a single batch. To the best of our knowledge, there have not been any works performing training of ResNet-18 completely in MPC. Previous works have only performed training on neural networks such as AlexNet [49], which is a much smaller network. Adapting these frameworks to support the complexities of ResNets such as skip connections is non-trivial.

Unsurprisingly, both the **G-Shuff** and **G-BMF** protocols are much faster than **GL-BMF**. This is due to the $B \times N$ (B per client) additional matrix multiplications needed in the **GL-BMF** protocol, for computing $[\![\boldsymbol{g}_{\boldsymbol{\phi}_i^L,j}]\!]$ with $j \in [1, B]$. In general, the online times and communications for the **G-Shuff** and **G-BMF** are very close, as the primary difference between the two is due to the extra shuffling step conducted at each epoch under the **G-Shuff** variant. But we should also note after accumulating the per-batch runtime to per-epoch runtime, on average, **G-Shuff** is still more time-consuming than **G-BMF**: Such as when under **Pre-ImageNet** – **CIFAR-10**, the difference of the runtime at epoch level gets to around 97 seconds, which is non-negligible. Moreover, all observations summarized in Tab. 5 also appear in Tab. 8.

9 Conclusion

In conclusion, we propose a novel end-to-end privacy-preserving framework, instantiated by three efficient protocols for different deployment scenarios, addressing both input and output privacy, for VFL. Our method avoids the high computation and communication costs of naive MPC-based solutions while maintaining model utility, making it practical for real-world deployments. We also observe that the improvement of our global-local model update protocol over the pure global model update protocol is limited. This is because the strong noise introduced by BandMF at each step significantly impairs the linear estimation, which in turn degrades the final model performance. This observation motivates another direction for future work: By developing more efficient DP mechanisms that enhance utility, our PP-VFL protocol could achieve substantial performance gains.

Acknowledgments

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors. The authors used generative AI-based tools to revise the text, improve flow and correct any typos, grammatical errors, and awkward phrasing.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS'16)*. 308–318.

[2] Abdelrahaman Aly and Nigel P. Smart. 2019. Benchmarking Privacy Preserving Scientific Operations. In *ACNS 19International Conference on Applied Cryptography and Network Security (LNCS)*.

[3] Toshinori Araki, Assi Barak, Jun Furukawa, Marcel Keller, Yehuda Lindell, Kazuma Ohara, and Hikaru Tsuchida. 2018. Generalizing the SPDZ Compiler For Other Protocols. In *ACM CCS 2018*. ACM Press.

[4] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. 2016. High-Throughput Semi-Honest Secure Three-Party Computation with an Honest Majority. In *ACM CCS 2016*. ACM Press.

[5] Gilad Asharov, Koki Hamada, Dai Ikarashi, Ryo Kikuchi, Ariel Nof, Benny Pinkas, Katsumi Takahashi, and Junichi Tomida. 2022. Efficient Secure Three-Party Sorting with Applications to Data Analysis and Heavy Hitters. In *ACM CCS 2022*. ACM Press.

[6] Wenxuan Bao, Shan Jin, Hadi Abdullah, Anderson C. A. Nascimento, Vincent Bindschaedler, and Yiwei Cai. 2025. Deep Learning with Plausible Deniability. In *The 39th Annual Conference on Neural Information Processing Systems (NeurIPS'25)*.

[7] Donald Beaver. 1992. Efficient Multiparty Protocols Using Circuit Randomization. In *CRYPTO'91 (LNCS)*.

[8] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. 2020. Secure Single-Server Aggregation with (Poly)Logarithmic Overhead. In *ACM CCS 2020*. ACM Press.

[9] Yaniv Ben-Itzhak, Helen Möllering, Benny Pinkas, Thomas Schneider, Ajith Suresh, Oleksandr Tkachenko, Shay Vargaftik, Christian Weinert, Hossein Yalame, and Avishay Yanai. 2024. ScionFL: Efficient and Robust Secure Quantized Aggregation. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML'24)*. 490–511.

[10] Jonas Böhler and Florian Kerschbaum. 2021. Secure Multi-party Computation of Differentially Private Heavy Hitters. In *ACM CCS 2021*. ACM Press.

[11] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS'17)*. ACM, 1175–1191.

[12] Ran Canetti. 2001. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *42nd FOCS*. IEEE Computer Society Press.

[13] Ran Canetti. 2020. Universally Composable Security. *J. ACM* 67, 5, Article 28 (Sept. 2020), 94 pages.

[14] Octavian Catrina and Amitabh Saxena. 2010. Secure Computation with Fixed-Point Numbers. In *FC 2010 (LNCS)*.

[15] Jeffrey Champion, abhi shelat, and Jonathan Ullman. 2019. Securely Sampling Biased Coins with Applications to Differential Privacy. In *ACM CCS 2019*. ACM Press.

[16] Christopher A. Choquette-Choo, Arun Ganesh, Ryan McKenna, H. Brendan McMahan, Keith Rush, Abhradeep Thakurta, and Zheng Xu. 2023. (Amplified) banded matrix factorization: a unified approach to private training. In *Proceedings of the 37th International Conference on Neural Information Processing Systems ((NeurIPS'23)*. 74856–74889.

[17] Christopher A. Choquette-Choo, H. Brendan McMahan, Keith Rush, and Abhradeep Thakurta. 2023. Multi-epoch matrix factorization mechanisms for private machine learning. In *Proceedings of the 40th International Conference on Machine Learning (ICML'23)*. 5924–5963.

[18] Amrita Roy Chowdhury, Chuan Guo, Somesh Jha, and Laurens van der Maaten. 2022. EIFFeL: Ensuring Integrity for Federated Learning. In *ACM CCS 2022*. ACM Press.

[19] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. 2017. EMNIST: Extending MNIST to handwritten letters. *2017 International Joint Conference on Neural Networks (IJCNN'17)* (2017). https://doi.org/10.1109/ijcnn.2017.7966217

[20] Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. 2018. SPD $\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for Dishonest Majority. In *CRYPTO 2018, Part II (LNCS)*.

[21] Ronald Cramer, Ivan Bjerre Damgård, et al. 2015. *Secure multiparty computation*. Cambridge University Press.

[22] Anders P. K. Dalskov, Daniel Escudero, and Marcel Keller. 2021. Fantastic Four: Honest-Majority Four-Party Secure Computation With Malicious Security. In *USENIX Security 2021*. USENIX Association.

[23] Sankha Das, Sayak Ray Chowdhury, Nishanth Chandran, Divya Gupta, Satya Lokam, and Rahul Sharma. 2025. Communication Efficient Secure and Private Multi-Party Deep Learning. *Proceedings on Privacy Enhancing Technologies (PETS'25)* (2025).

[24] Daniel Demmler, Thomas Schneider, and Michael Zohner. 2015. ABY – A Framework for Efficient Mixed-Protocol Secure Two-Party Computation. In *The Network and Distributed System Security (NDSS'15)*.

[25] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09)*. 248–255.

[26] Sergey Denisov, H. Brendan McMahan, Keith Rush, Adam Smith, and Abhradeep Thakurta. 2022. Improved differential privacy for SGD via optimal private linear operators on adaptive streams. In *Proceedings of the 36th International Conference on Neural Information Processing Systems ((NeurIPS'22)*. 5910–5924.

[27] Cynthia Dwork. 2006. Differential Privacy. In *Automata, Languages and Programming*. 1–12.

[28] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography Conference (TCC'06)*. Springer, 265–284.

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

[30] Reo Eriguchi, Atsunori Ichikawa, Noboru Kunihiro, and Koji Nuida. 2021. Efficient Noise Generation to Achieve Differential Privacy with Applications to Secure Multiparty Computation. In *FC 2021, Part I (LNCS)*.

[31] Daniel Escudero, Satrajit Ghosh, Marcel Keller, Rahul Rachuri, and Peter Scholl. 2020. Improved Primitives for MPC over Mixed Arithmetic-Binary Circuits. In *CRYPTO 2020, Part II (LNCS)*.

[32] Daniel Escudero, Vipul Goyal, Antigoni Polychroniadou, and Yifan Song. 2022. TurboPack: Honest Majority MPC with Constant Online Communication. In *ACM CCS 2022*. ACM Press.

[33] FederatedAI. 2024. FATE. https://github.com/FederatedAI/FATE?tab=readme-ov-file.

[34] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, and Shaza Zeitouni. 2021. SAFELearn: Secure Aggregation for private FEderated Learning. In *2021 IEEE Security and Privacy Workshops (SPW'21)*. 56–62.

[35] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2021. Sharpness-Aware Minimization for Efficiently Improving Generalization. arXiv:2010.01412

[36] David Froelicher, Hyunghoon Cho, Manaswitha Edupalli, Joao Sa Sousa, Jean-Philippe Bossuat, Apostolos Pyrgelis, Juan R. Troncoso-Pastoriza, Bonnie Berger, and Jean-Pierre Hubaux. 2023. Scalable and Privacy-Preserving Federated Principal Component Analysis . In *2023 IEEE Symposium on Security and Privacy (SP'23)*. 1908–1925.

[37] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X. Liu, and Ting Wang. 2022. Label Inference Attacks Against Vertical Federated Learning. In *31st USENIX Security Symposium (Security'22)*. 1397–1414.

[38] Fangcheng Fu, Huanran Xue, Yong Cheng, Yangyu Tao, and Bin Cui. 2022. BlindFL: Vertical Federated Machine Learning without Peeking into Your Data. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD'22)*. 1316–1330.

[39] Keke Gai, Mohan Wang, Jing Yu, Lei Xu, Peng Jiang, Liehuang Zhu, and Bin Xiao. 2025. Differentially Private Vertical Federated Learning With Adaptive Constraints and Dynamic Noise. *IEEE Transactions on Information Forensics and Security (TIFS)* 20 (2025), 11150–11164.

[40] Google. 2025. Differential Privacy. https://github.com/google/differential-privacy.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*. 770–778.

[42] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *Proceedings of the International Conference on Learning Representations (ICLR'22)*.

[43] Yimin Huang, Wanwan Wang, Xingying Zhao, Yukun Wang, Xinyu Feng, Hao He, and Ming Yao. 2023. EFMVFL: An Efficient and Flexible Multi-party Vertical Federated Learning without a Third Party. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 18, 3 (2023), 20 pages.

[44] Matthew Jagielski, Rahul Rachuri, Daniel Escudero, and Peter Scholl. 2025. Covert Attacks on Machine Learning Training in Passively Secure MPC. https://eprint.

iacr.org/2025/906
[45] Xiao Jin, Pin-Yu Chen, Chia-Yi Hsu, Chia-Mu Yu, and Tianyi Chen. 2021. Cafe: Catastrophic data leakage in vertical federated learning. *Proceedings of the 35th International Conference on Neural Information Processing Systems ((NeurIPS'21)* 34 (2021), 994–1006.
[46] Peter Kairouz, Ziyu Liu, and Thomas Steinke. 2022. The Distributed Discrete Gaussian Mechanism for Federated Learning with Secure Aggregation. arXiv:2102.06387 [cs.LG]
[47] Hannah Keller, Helen Möllering, Thomas Schneider, Oleksandr Tkachenko, and Liang Zhao. 2024. Secure noise sampling for DP in MPC with finite precision. In *Proceedings of the 19th International Conference on Availability, Reliability and Security*. 1–12.
[48] Marcel Keller. 2020. MP-SPDZ: A Versatile Framework for Multi-Party Computation. In *ACM CCS 2020*. ACM Press.
[49] Marcel Keller and Ke Sun. 2022. Secure Quantized Training for Deep Learning. Cryptology ePrint Archive, Report 2022/933. https://eprint.iacr.org/2022/933.
[50] Afsana Khan, Marijn ten Thij, and Anna Wilbik. 2025. Vertical Federated Learning: A Structured Literature Review. *Knowledge and Information Systems* (2025).
[51] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. 2017. Practical Multi-party Private Set Intersection from Symmetric-Key Techniques. In *ACM CCS 2017*. ACM Press.
[52] Antti Koskela, Joonas Jälkö, and Antti Honkela. 2020. Computing Tight Differential Privacy Guarantees Using FFT. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS'20) (Proceedings of Machine Learning Research, Vol. 108)*. PMLR, 2560–2569.
[53] Nishat Koti, Arpita Patra, Rahul Rachuri, and Ajith Suresh. 2022. Tetrad: Actively Secure 4PC for Secure Training and Inference. In *Proceedings of the 29th Annual Network and Distributed System Security Symposium NDSS'22*.
[54] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
[55] Sven Laur, Jan Willemson, and Bingsheng Zhang. 2011. Round-Efficient Oblivious Database Manipulation. Cryptology ePrint Archive, Report 2011/429. https://eprint.iacr.org/2011/429.
[56] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature Pyramid Networks for Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*. 2117–2125.
[57] Yang Liu, Yan Kang, Tianyuan Zou, Yanhong Pu, Yuanqin He, Xiaozhou Ye, Ye Ouyang, Ya-Qin Zhang, and Qiang Yang. 2024. Vertical Federated Learning: Concepts, Advances, and Challenges. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* (2024), 1–20.
[58] Yang Liu, Xinwei Zhang, Yan Kang, Liping Li, Tianjian Chen, Mingyi Hong, and Qiang Yang. 2022. FedBCD: A Communication-Efficient Collaborative Learning Framework for Distributed Features. *IEEE Transactions on Signal Processing (TSP)* 70 (2022), 4277–4290.
[59] Hidde Lycklama, Lukas Burkhalter, Alexander Viand, Nicolas Küchler, and Anwar Hithnawi. 2023. RoFL: Robustness of Secure Federated Learning. In *2023 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press.
[60] Yiping Ma, Jess Woods, Sebastian Angel, Antigoni Polychroniadou, and Tal Rabin. 2023. Flamingo: Multi-Round Single-Server Secure Aggregation with Applications to Private Federated Learning. In *2023 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press.
[61] Sindhuja Madabushi, Ahmad Faraz Khan, Haider Ali, and Jin-Hee Cho. 2025. OPUS-VFL: Incentivizing Optimal Privacy-Utility Tradeoffs in Vertical Federated Learning. arXiv:2504.15995 [cs.LG] https://arxiv.org/abs/2504.15995
[62] Saber Malekmohammadi, Yaoliang Yu, and Yang Cao. 2024. Noise-aware algorithm for heterogeneous differentially private federated learning. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*. 34461 – 34498.
[63] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS'17)*, Vol. 54. 1273–1282.
[64] Ilya Mironov. 2017. Rényi Differential Privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF'17)*. IEEE.
[65] Payman Mohassel and Peter Rindal. 2018. ABY3: A Mixed Protocol Framework for Machine Learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS'18)*. 35–52.
[66] Payman Mohassel and Yupeng Zhang. 2017. SecureML: A System for Scalable Privacy-Preserving Machine Learning . In *2017 IEEE Symposium on Security and Privacy (SP'17)*. 19–38.
[67] Vaikkunth Mugunthan, Pawan Goyal, and Lalana Kagal. 2021. Multi-VFL: A Vertical Federated Learning System for Multiple Data and Label Owners. https://arxiv.org/abs/2106.05468
[68] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *2019 IEEE Symposium on Security and Privacy (SP'19)*. 739–753.
[69] Kalinin Nikita and Christoph H Lampert. 2024. Banded Square Root Matrix Factorization for Differentially Private Model Training. In *Proceedings of the 38th International Conference on Neural Information Processing Systems (NeurIPS'24)*, Vol. 37. 17602–17655.
[70] Sikha Pentyala, Davis Railsback, Ricardo Maia, Rafael Dowsley, David Melanson, Anderson Nascimento, and Martine De Cock. 2022. Training Differentially Private Models with Secure Multiparty Computation. arXiv:2202.02625 [cs.CR]
[71] Benny Pinkas, Thomas Schneider, and Michael Zohner. 2018. Scalable Private Set Intersection Based on OT Extension. *ACM Transactions on Privacy and Security (TOPS)* 21, 2, Article 7 (2018), 35 pages.
[72] Data Privacy and Trustworthy Machine Learning Research Lab. 2025. Privacy Meter. https://github.com/privacytrustlab.
[73] Pytorch. 2025. ResNet18. https://pytorch.org/vision/stable/models.html.
[74] Xinchi Qiu, Heng Pan, Wanru Zhao, Yan Gao, Pedro P. B. Gusmao, William F. Shen, Chenyang Ma, and Nicholas D. Lane. 2024. Secure Vertical Federated Learning Under Unreliable Connectivity. arXiv:2305.16794 [cs.CR]
[75] Xinchi Qiu, Heng Pan, Wanru Zhao, Chenyang Ma, Yan Gao, Pedro Porto Buarque de Gusmao, and Nicholas Donald Lane. 2024. vFedSec: Efficient Secure Aggregation for Vertical Federated Learning via Secure Layer.
[76] Michael O. Rabin. 2005. How To Exchange Secrets with Oblivious Transfer. Cryptology ePrint Archive, Report 2005/187. https://eprint.iacr.org/2005/187.
[77] Rahul Rachuri and Peter Scholl. 2022. Le Mans: Dynamic and Fluid MPC for Dishonest Majority. In *CRYPTO 2022, Part I (LNCS)*.
[78] Mayank Rathee, Conghao Shen, Sameer Wagh, and Raluca Ada Popa. 2023. ELSA: Secure Aggregation for Federated Learning with Malicious Actors. In *2023 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press.
[79] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics* 8 (2020), 842–866.
[80] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP'17)*. IEEE, 3–18.
[81] Timothy Stevens, Christian Skalka, Christelle Vincent, John Ring, Samuel Clark, and Joseph Near. 2022. Efficient Differentially Private Secure Aggregation for Federated Learning via Hardness of Learning with Errors. In *31st USENIX Security Symposium (Security'22)*. 1379–1395.
[82] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. 2018. Split learning for health: Distributed deep learning without sharing raw patient data. arXiv:1812.00564 [cs.LG] https://arxiv.org/abs/1812.00564
[83] Praneeth Vepakomma, Abhishek Singh, Otkrist Gupta, and Ramesh Raskar. 2020. NoPeek: Information leakage reduction to share activations in distributed deep learning . In *2020 International Conference on Data Mining Workshops (ICDMW'20)*. 933–942.
[84] Chang Wang, Jian Liang, Mingkai Huang, Bing Bai, Kun Bai, and Hao Li. 2020. Hybrid Differentially Private Federated Learning on Vertically Partitioned Data. https://arxiv.org/abs/2009.02763
[85] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. 2019. Subsampled rényi differential privacy and analytical moments accountant. In *The 22nd International Conference on Artificial Intelligence and Statistics (AISTATS'19)*. PMLR, 1226–1235.
[86] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. 2020. Federated Learning With Differential Privacy: Algorithms and Performance Analysis. *IEEE Transactions on Information Forensics and Security (TIFS)* 15 (2020), 3454–3469.
[87] Zhaomin Wu, Junyi Hou, Yiqun Diao, and Bingsheng He. 2024. Federated Transformer: Multi-Party Vertical Federated Learning on Practical Fuzzily Linked Data. In *The 38th Annual Conference on Neural Information Processing Systems (NeurIPS'24)*.
[88] Chulin Xie, Pin-Yu Chen, Qinbin Li, Arash Nourian, Ce Zhang, and Bo Li. 2024. Improving Privacy-Preserving Vertical Federated Learning by Efficient Communication with ADMM. In *2024 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML'24)*. 443–471.
[89] Runhua Xu, Nathalie Baracaldo, Yi Zhou, Ali Anwar, James Joshi, and Heiko Ludwig. 2021. FedV: Privacy-Preserving Federated Learning over Vertically Partitioned Data. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security (AISec'21)*. 181–192.
[90] Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, Vincent Bindschaedler, and Reza Shokri. 2022. Enhanced Membership Inference Attacks against Machine Learning Models. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS'22)*. 3093–3106.
[91] Xuefei Yin, Yanming Zhu, and Jiankun Hu. 2021. A Comprehensive Survey of Privacy-preserving Federated Learning: A Taxonomy, Review, and Future Directions. *Comput. Surveys* 54, 6, Article 131 (2021), 36 pages.
[92] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. 2021. Opacus: User-Friendly Differential Privacy Library in PyTorch. *arXiv preprint arXiv:2109.12298* (2021).

[93] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep Leakage from Gradients. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems ((NeurIPS'19)*, Vol. 32. 14774–14784.

A Other Related Work

Secure Aggregation and PPML via MPC. A core strand of research in the privacy-preserving machine learning (PPML) community leverages secure multi-party computation (MPC) to enable collaborative model training while preserving input privacy across different settings. Several works have developed MPC-based frameworks to address practical PPML challenges, including floating-point arithmetic, efficient dot-product computation, and non-linear activation functions [3, 24, 31, 53, 65]. However, these techniques are not specifically tailored for FL or VFL, where data partitions and interaction patterns differ significantly from conventional PPML scenarios.

MPC & DP. A smaller subset of the community has explored combining MPC protocols with differential privacy techniques, aiming to achieve complementary advantages of both: secure protocol execution and strong formal privacy bounds [10, 15, 23, 30]. However, to our knowledge, these combined approaches have not been systematically studied in the setting of vertical federated learning.

FL & MPC. In the setting of federated learning, MPC has been used to enable protocols for secure aggregation [11], which focuses on protecting individual local models' updates within FL. Protocols such as [8, 18, 59, 60] focus on the single-server setting, while [9, 78] consider the multi-server setting. While these works have demonstrated promising performance, they predominantly consider horizontally partitioned data (i.e., each party holds the same set of features but on different subsets of samples), and it is non-trivial to extend them to the vertically split scenario. Recent work in [74, 75] proposes a secure layer-based framework for training vertical FL securely through secure aggregation. Although this framework can protect the private data, the output privacy of the trained model is still not guaranteed, which makes the framework vulnerable to attacks such as MIAs.

FL & HE. Another line of research employs encryption-based techniques, particularly homomorphic encryption (HE), to preserve input privacy during model training. In [36], the authors propose a federated Principal Component Analysis (PCA) framework over distributed datasets using HE, but their method is limited to the horizontal FL setting and also doesn't consider the output privacy-preserving. In contrast, the works in [33] consider a vertical FL-style setting and applies HE to protect both the upstream intermediate representations and also downstream gradients. In [89], the authors exploits the functional encryption fore secure computation under VFL scenario. However, both of these approaches focused only on the input privacy and does not address the output privacy issues.

DP in Horizontal FL. Previous works incorporate DP into horizontal federated learning (HFL) [46, 86]. These methods typically add calibrated noise to gradients during training, ensuring that individual client contributions remain obscured. Recent studies [62, 81] further focus on enhancing both the utility and scalability of output privacy preservation in HFL, going beyond conventional DP mechanisms. Such DP-based FL solutions are practical for deployment and have been extensively analyzed in terms of both privacy and utility. However, they are predominantly tailored to the HFL setting and cannot be directly extended to the VFL scenario.

B Additional Details and Experimental Results

This section provides additional materials that complement the main manuscript, including the notation table (Table 6) used throughout the paper and an extended experimental results on communication and computation costs of the MPC over WAN (Tab. 8). Besides, additional experimental results include:

Results under $E_{num} = 100$. Tab. 7 reports the utilities for different protocols after 100 training epoches. The general patterns are consistent. The BMF mechanism seems to benefit more from increasing training epoches. For example, in **Pre-ImageNet – EMNIST**, the performance of **G-BMF** increases from 73.62 to 75.28 at $\varepsilon = 1$, whereas the performance of **G-Shuff** does not improve. Recall that BMF adds correlated noises with the hope that the noises can "offset" among themselves over multiple epoches. Longer training horizon potentially benefit BMF. However, we must also note that the longer training horizon may favor a different choice of hyperparameters. For example, in **Pre-ImageNet – EMNIST** at $\varepsilon = 2$, **G-BMF** used to perform better with the hyperparameters in Setting 1, but now achieves higher model accuracy with the hyperparameters in Setting 2. This phenomenon reminds us practitioners to carefully consider the characteristics of Gaussian v.s. BMF mechanisms.

Extended Baselines Comparisons. A central design principle of our protocols is to strictly preserve sample-level anonymity throughout training. Relaxing this constraint would enable the global-local training protocol **GL** to exploit privacy amplification by subsampling. Consistent with the setting in [88], the servers may randomly sample a subset of B data indices at each step and broadcast them to the clients. Under this modification, **GL-BMF** can employ the subsampled Gaussian mechanism in place of the BMF mechanism. We denote this variant of **GL-BMF** as **GL-Sub**. We stress that this variant is **not** proposed as a new formal privacy-preserving protocol, as it introduces significant privacy risks by breaking the anonymity of sample participation. However, we introduce this variant solely as a special baseline for comparison with our formal **GL-BMF** protocol. While **GL-Sub** relies on much different trust and privacy assumptions, it employs the same denoising techniques as we used in **GL-BMF**.

We follow the same experimental settings as in Tab. 3, but set the number of training epochs to $E_{\text{num}} = 100$ and $\epsilon = 4$. We compare **GL-BMF** (under Setting 1) with **GL-Sub**, as well as **ADMM** and **FedBCD**. The results in Tab. 9 show that **GL-BMF** continues to outperform both **ADMM** and **FedBCD** over a longer training horizon. However, we can also clearly find that by utilizing privacy-amplification, **GL-Sub** outperform all other protocols. This is because when under weaker noise perturbation (the subsampling-based Gaussian mechanism), the denoising procedure applied to the private gradients is more effective than under the BMF mechanism, which injects stronger noise at each step, resulting in an overall boost in utility. We leave further discussion to the readers to assess which privacy condition is most relevant to them and to determine the privacy–utility trade-off that best fits their practical applications.

Table 6: Notation Table

Symbol	Description
$\mathcal{W}$	Global model
$\mathcal{F}_i$	Local model at client i
$\boldsymbol{\theta}$	Parameters of the global model $\mathcal{W}$
$\boldsymbol{\phi}_i$	Parameters of local model $\mathcal{F}_i$
$\boldsymbol{x}_{b,j}^{(i)}$	Client i's local data of the j-the sample in the b-th batch
$\boldsymbol{H}_{b,j}^{(i)}$	Output of local model $\mathcal{F}_i$ over $\boldsymbol{x}_{b,j}^{(i)}$

Dataset	Method	Pre-ImageNet					Pre-Cifar100				
	Plain	**Non-DP** 93.5					**Non-DP** 93.89				
		DP					**DP**				
		$\varepsilon=1$	$\varepsilon=2$	$\varepsilon=5$	$\varepsilon=8$	$\varepsilon=10$	$\varepsilon=1$	$\varepsilon=2$	$\varepsilon=5$	$\varepsilon=8$	$\varepsilon=10$
CIFAR-10	G-Shuff	63.71	69.09	71.93	72.89	73.88	80.16	82.11	82.85	82.98	83.02
	G-BMF (Setting 1)	65.14	70.66	73.63	74.25	74.41	79.51	80.41	80.57	80.59	80.64
	G-BMF (Setting 2)	49.14	55.16	61.47	64.42	65.85	66.83	71.31	78.18	80.61	81.49
	GL-BMF (Setting 1)	65.97	71.23	74.09	74.59	74.78	79.79	80.75	80.81	80.94	81.05
	GL-BMF (Setting 2)	51.32	56.26	62.29	65.32	66.61	67.44	71.87	78.66	80.99	81.83
	Plain	**Non-DP** 94.26					**Non-DP** 94.51				
		DP					**DP**				
		$\varepsilon=1$	$\varepsilon=2$	$\varepsilon=5$	$\varepsilon=8$	$\varepsilon=10$	$\varepsilon=1$	$\varepsilon=2$	$\varepsilon=5$	$\varepsilon=8$	$\varepsilon=10$
EMNIST	G-Shuff	67.41	73.68	77.71	78.62	80.24	77.86	79.55	80.08	80.17	80.19
	G-BMF (Setting 1)	75.28	79.11	80.57	80.82	80.88	77.35	77.95	78.09	78.14	78.17
	G-BMF (Setting 2)	68.12	74.51	80.94	82.76	83.34	77.39	81.59	83.62	84.05	84.14
	GL-BMF (Setting 1)	75.93	79.65	80.83	80.99	81.02	77.62	78.08	78.22	78.23	78.27
	GL-BMF (Setting 2)	68.95	75.22	81.58	83.21	83.68	77.81	81.86	83.79	84.15	84.26

Table 7: Test Accuracy (%) versus Different Datasets, Pre-training, and Privacy Budgets (100 epochs)

Dataset	Method	Pre-ImageNet		Pre-Cifar100	
		Runtime (s)	**Communication** (MB)	**Runtime** (s)	**Communication** (MB)
CIFAR-10	G-Shuff	83.94	473.63	44.62	244.03
	G-BMF	82.60	453.6	43.92	233.77
	GL-BMF	686.60	6220.87	468.19	3186.62
EMNIST	G-Shuff	194.17	1228.56	108.34	632.52
	G-BMF	190.75	1178.64	106.65	606.96
	GL-BMF	725.44	6945.87	490.25	3559.79

Table 8: Runtime and Communication cost of the MPC over WAN

Dataset	Method			
	GL-BMF	GL-Sub	ADMM	FedBCD
CIFAR-10	75.47	84.46	69.64	71.82

Table 9: Test Accuracy (%) versus Different Baselines (100 epochs, $\epsilon = 4$, $\gamma = 2$)

C Multiparty Computation

Corruption. Correctness and privacy are guaranteed by assuming a certain type of adversary in the system. Depending on how we model the adversary there are different settings possible. The two big levers are the number of parties the adversary can control, and what the adversary can do with the corrupted parties. If the adversary can control only up to half of the parties ($t < K/2$) where K is the number of the total parties, we are in an honest

majority setting, whereas if it is higher than half ($t \geq K/2$), we are in dishonest majority. An adversary that only passively listens to the messages in the computation and tries to learn as much as possible from them, is a passive or semi-honest adversary. The adversary in this case will always send the correct message(s), and will not deviate from the expected behavior. If the adversary can behave in arbitrary ways, such as not sending a message, sending the wrong data, etc., it is referred to as an active or malicious adversary.

Active adversaries are much harder to deal with, and in some situations are considered overkill. For instance, when we are dealing with large companies performing a PPML task together, some have argued [44] that the reputational risk of actively deviating from the protocol is *higher* than what could be gained by cheating. Passive protocols on the other hand are easier to design, and often serve as a first step towards building an actively secure protocol. This could be done by identifying the parts in the passive protocol that are vulnerable to active attacks, and proposing techniques to secure them.

Following the corruption threshold, in any MPC protocol, if more parties than the assumed threshold collude, security collapses and intermediate values can be exposed. Below this threshold, however, our MPC+DP protocol ensures that servers only see DP-protected outputs: the clipped, aggregated, and noised gradients, providing an additional layer of privacy even if partial information is inadvertently revealed. In our MPC implementation, we deploy three servers tolerating one corrupted server, which balances efficiency and security; the protocol can be extended to more servers at higher computational cost. In general, security under a corruption-threshold assumption is all-or-nothing: no leakage occurs below the threshold, while full leakage is possible above it.

Secret-Sharing. We use the notation to denote a value that is secret-shared between parties. In this work, we assume that values are secret-shared using a linear secret-sharing scheme, such as arithmetic secret-sharing, or replicated secret-sharing [4]. If a value x was secret-shared between three parties using replicated secret-sharing over a finite field $\mathbb{F}_p$, where p is a prime, each party P_i holds two secret-shares corresponding to x. For example, when , P_0 will hold $[\![x]\!]_1, [\![x]\!]_2$, P_1 holds $[\![x]\!]_2, [\![x]\!]_0$, and P_2 will hold $[\![x]\!]_0, [\![x]\!]_1$.

The linearity of the secret-sharing scheme, such as the one mentioned earlier, allows for parties to compute additions of two secret-shared values locally. To compute a secret-sharing of $z = x + y$, each party simply adds the shares it holds corresponding to the individual values of x, y, resulting in $[\![z]\!]$. To compute a multiplication gate however, parties must communicate with each other. Protocols for efficient multiplication has received a lot of attention over the recent years, with [4, 22, 32, 53, 65] proposing efficient protocols for multiplication in the honest-majority setting, and works such as [20, 31, 66, 77] to name a few in the dishonest-majority setting.

Pre-processing. MPC protocols sometimes involve using correlated randomness, such as Beaver triples [7], Oblivious Transfer (OT) [76], or edaBits [31] in our case. These are typically function-dependent but input-independent, which means that if the parties involved in the MPC know in advance what function they are going to compute, they can pre-process the correlated randomness required to carry out the MPC. Pre-processing takes a significant amount of time, especially for functions such as deep neural networks, so it makes a big difference in performance if one can take advantage of pre-processing.

Mixed-mode Computation. Typically, computations in MPC are expressed as arithmetic ($\mathbb{F}_p$) or boolean circuits ($\mathbb{F}_2$), where p is either a prime or 2^k. Each of these computational domains has its advantages. For instance, performing multiplications is faster in the arithmetic domain, while an operation such as a secure comparison is more well suited for the boolean domain. The structure of neural networks is such that multiplications and non-linear operations such as comparisons and activation functions alternate. This makes it so that sticking to one of the two computational domains is sub-optimal. One of the interesting techniques that has been proposed recently, in works such as [24, 65, 66] is to have efficient ways to switch the secret-sharing domain from arithmetic to boolean and back, mid computation. These techniques allow for parties to take full advantage of the two domains.

C.1 Proof of Theorem 1

The proof, including a description of all building blocks and a simulator construction, is given below:

Building Blocks. Most of our sub-protocols come from ABY3 [65], which is proven UC-secure against a static, semi-honest adversary corrupting at most one of three servers:

- Π_{Mult} (multiplication over replicated secret shares): ABY3 uses the protocol from Araki et al. [4], which is proven UC-secure in the $\mathcal{F}_{ABB}$-hybrid model.
- Π_{Shuffle} (oblivious shuffle): built from local random permutations and re-sharing, both $\mathcal{F}_{ABB}$ operations. The construction follows Laur et al. [55] and is also used as a building block in the sorting protocol of Asharov et al. [5] (Protocol 3.2); We further refer the reader to Theorem 4.6 in [5], which provides a simulation-based proof of the full sorting functionality).
- Π_{Norm} and Π_{Div} (norm and division over fixed-point numbers): constructed by Aly and Smart [2] in the $\mathcal{F}_{ABB}$-hybrid model using secure comparison, fixed-point division (from Catrina and Saxena [14]), and bit decomposition, each independently proven secure under composition.
- Π_{Act} (activation functions): in our protocols the global model uses a softmax activation, which is computed from the exponentiation and division protocols of Aly and Smart [2], the same $\mathcal{F}_{ABB}$ operations used by Π_{Norm} and Π_{Div} above.
- Π_{GS} (secure Gaussian noise generation): we use the specific instantiation from [47], which is built from arithmetic and boolean operations with secure domain conversions, which we use from ABY3 [65].
- Π_{FWD} and Π_{BWD} (forward and backward pass): sequential compositions of Π_{Mult} and Π_{Act}.

All subprotocols are instantiated in the MP-SPDZ [48] framework.

No Leakage from Intermediate Values. No secret-shared value is ever reconstructed between sub-protocol calls. The output shares of one subprotocol (e.g., Π_{Shuffle}) are passed directly as input shares to the next (e.g., Π_{FWD}). Since each server only ever holds its replicated shares, which are uniformly random and independent of the

underlying secret [4], no information is leaked at the transitions between subprotocols. The only reconstruction in the entire protocol occurs at the final step, when the DP-protected output is revealed.

Building on the above discussion, we first present the proof for protocol $\Pi_{\text{Global-Shuffle}}$, which can be easily extended to the remaining protocols.

Corollary 5 (Input privacy of $\Pi_{\text{Global-Shuffle}}$). *$\Pi_{\text{Global-Shuffle}}$ (Fig. 1) is secure against a static, semi-honest adversary corrupting at most one of the three servers, in the $\mathcal{F}_{ABB}$-hybrid model.*

Proof. Every operation in $\Pi_{\text{Global-Shuffle}}$ such as secret sharing, shuffling, forward pass, backpropagation, gradient clipping, noise addition, and model update, is realized by $\mathcal{F}_{ABB}$ operations as established above. By the UC composition theorem [12], the composed protocol is secure in the $\mathcal{F}_{ABB}$-hybrid model.

Simulator Construction. We construct a simulator $\mathcal{S}$ for the corrupted server P_c ($c \in \{1, 2, 3\}$). Following the semi-honest simulation paradigm, $\mathcal{S}$ is given P_c's *input* (the collection of secret shares received from clients) and P_c's *output* (its share of the trained global model), and must produce a simulated *view* consisting of P_c's input, random tape, and all protocol messages received by P_c, that is computationally indistinguishable from P_c's real view.

$\mathcal{S}$ then proceeds as follows:

(1) $\mathcal{S}$ samples a uniformly random string r of appropriate length as P_c's simulated random tape.
(2) For each $\mathcal{F}_{ABB}$ operation, $\mathcal{S}$ generates the messages P_c would have received by running the operation's guaranteed simulator on P_c's input and output shares for that operation.
(3) Local operations (additions, public-scalar multiplications) on shares are deterministic functions of the shares already present in the view, so $\mathcal{S}$ computes these consistently. By the linearity of replicated secret sharing [4], each party's shares remain uniformly distributed and independent of the underlying secret.

Indistinguishability. The simulated view is computationally indistinguishable from P_c's real view because each $\mathcal{F}_{ABB}$ operation's simulator produces an indistinguishable transcript, and all local share operations are deterministic functions of identically distributed inputs. □

Typically, the same argument extends to the remaining protocols:

Corollary 6 (Input privacy of $\Pi_{\text{Global-BMF}}$). *$\Pi_{\text{Global-BMF}}$ (Fig. 2) is secure against a static, semi-honest adversary corrupting at most one server, in the $\mathcal{F}_{ABB}$-hybrid model.*

Proof. The proof for protocol $\Pi_{\text{Global-BMF}}$ is identical to the proof for $\Pi_{\text{Global-Shuffle}}$, with Π_{Shuffle} absent (since $\Pi_{\text{Global-BMF}}$ does not shuffle). The BandMF noise computation $[\![\boldsymbol{\mathcal{N}}_{\text{Corr}}]\!] = \Omega^{-1}[\![\boldsymbol{\mathcal{N}}_{\text{Tab}}]\!]$ involves only a local linear operation on shares (as Ω^{-1} is a public matrix) and requires no additional sub-protocol call. □

Corollary 7 (Input privacy of $\Pi_{\text{Global-Local-BMF}}$). *$\Pi_{\text{Global-Local-BMF}}$ (Fig. 3) is secure against a static, semi-honest adversary corrupting at most one server, in the $\mathcal{F}_{ABB}$-hybrid model.*

Proof. The proof for $\Pi_{\text{Global-Local-BMF}}$ extends Corollary 6: the additional Π_{Mult} calls used to compute $[\![\boldsymbol{g}_{\boldsymbol{\phi}_i^L,j}]\!]$ via Eq. (5) are $\mathcal{F}_{ABB}$ operations. The reconstruction step, in which servers open $\tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L}$ to client E_i, reveals only the DP-protected gradient and introduces no additional leakage. □

Based on the proofs for all above corollaries, we claim we have completed the proof for Theorem 1.

D Differential Privacy

Definition D.1 (Differential Privacy [29])**.** A randomized algorithm $\mathcal{A}$ is said to be (ϵ, δ) differentially private if for any two neighboring input datasets $\mathcal{D}$ and $\mathcal{D}'$, and for any subset S of the output of $\mathcal{A}$, it always holds that

$$\Pr[\mathcal{A}(\mathcal{D}) \in S] \leq e^{\epsilon}\Pr[\mathcal{A}(\mathcal{D}') \in S] + \delta\,,$$

for non-negative constants ϵ and δ.

Definition D.2 (Gaussian Mechanism [29])**.** Given a d-dimensional function f, with ℓ_2 sensitivity $\Delta_2 f$ which is defined as

$$\Delta_2 f = \max \|f(\mathcal{D}) - f(\mathcal{D}')\|_2\,,$$

for any two neighboring input datasets $\mathcal{D}$ and $\mathcal{D}'$, for $\sigma \geq c\Delta_2 f/\epsilon$ and $c^2 > 2\ln(1.25/\delta)$, the Gaussian Mechanism with parameter σ adds noise scaled to $\mathcal{N}(0, \sigma^2)$ to each of the d components of the output of f, is said to be (ϵ, δ) differentially private.

Definition D.3 (Post-Processing [29])**.** Given randomized algorithm $\mathcal{A}$ which is said to be (ϵ, δ) differentially private. Let $\mathcal{M}_R$ be an arbitrary randomized mapping, then $\mathcal{M}_R(\mathcal{A})$ is said to be (ϵ, δ) differentially private.

Definition D.4 (Sensitivity for the Matrix Factorization based Mechanism)**.** The sensitivity of the matrix factorization mechanism Eq. (4) is defined as

$$\text{sens}(\boldsymbol{C}_{\text{MF}}) = \sup_{\boldsymbol{G} \sim \boldsymbol{G}'} \|\boldsymbol{C}_{\text{MF}}\boldsymbol{G} - \boldsymbol{C}_{\text{MF}}\boldsymbol{G}'\|_F\,, \tag{11}$$

where $\boldsymbol{G} \sim \boldsymbol{G}'$ denotes the *neighborhood relation*, which indicates that the two data streams $\boldsymbol{G}$ and $\boldsymbol{G}'$, are differ in the contributions derived from exactly a single example.

Definition D.5 (Sensitivity for Decreasing Non-negative Toeplitz Matrices)**.** Suppose $\Phi = \text{LDToep}(\phi_0, \ldots, \phi_{n-1}) \in \mathcal{R}^{n\times n}$ is a lower triangular Toeplitz matrix, with decreasing non-negative entries, as $\phi_0 \geq \phi_1 \geq \ldots \phi_{n-1} \geq 0$, then the sensitivity of Φ under b-min-separation is defined as

$$\text{sens}_{\kappa,\text{b}}(\Phi) = \left\| \sum_{j=0}^{\kappa-1} \Phi[:, 1 + j\text{b}] \right\|_F\,,$$

where $\Phi[:, 1 + j\text{b}]$ is the $1 + j$b-th column of Φ

Definition D.6 (Matrix Factorization Mechanism [26])**.** Let $\boldsymbol{A}$ be a lower-triangular full-rank query matrix, and let $\boldsymbol{A} = \boldsymbol{B}_{\text{MF}}\boldsymbol{C}_{\text{MF}}$ be any factorization which has the following property that for any two neighboring streams of vectors $\boldsymbol{G}$ and $\boldsymbol{G}'$, we have

$$\|\boldsymbol{C}_{\text{MF}}\boldsymbol{G} - \boldsymbol{C}_{\text{MF}}\boldsymbol{G}'\|_F \leq \zeta\,.$$

Let Z satisfied the distribution of $\mathcal{N}(0, \sigma^2\zeta^2)$ where σ is large enough, so that the mechanism

$$\mathcal{M}_{\text{MF}}(\boldsymbol{G}) = \boldsymbol{B}_{\text{MF}}(\boldsymbol{C}_{\text{MF}}\boldsymbol{G} + Z)\,,$$

is said to be (ϵ, δ) differentially private in the nonadaptive continual release model. With the same parameters, mechanism $\mathcal{M}_{\text{MF}}(\cdot)$ is also said to satisfy the same DP guarantee even the input rows are chosen adaptively.

D.1 Proof of Theorem 2

Proof. To the protocol in Fig. 1, at each step, as the ℓ_2 sensitivity $\Delta_2 f$ of $[\![\overline{\boldsymbol{g}}_\theta]\!]$ is bounded by γ through the clipping, the (ϵ, δ) differential privacy guarantee of $[\![\tilde{\boldsymbol{g}}_\theta]\!]$ follows from the application of the Gaussian Mechanism (D.2). Here let's assume each step satisfies (ϵ_1, δ_1)-DP.

With local models are frozen, and subsampling with the sampling ratio of B/M is conducted through running MPC secure shuffling protocols at servers, the final privacy budget spent on global model training follows from the application of the the Rényi Differential Privacy (RDP) framework [64] for privacy accounting, which formalizes the moments accountant method introduced by [1]. Furthermore, the RDP at order α for subsampled Gaussian mechanism is computed following [85]. Hence, the RDP guarantee at order α^o composes linearly over T_{num} steps, and we convert to (ϵ, δ)-DP through

$$\epsilon = \min_{\alpha^o > 1} \{\epsilon_{\alpha^o}^{(T_{num})} + \log(1/\delta)/(\alpha^o - 1)\} .$$

where $\epsilon_{\alpha^o}^{(T_{num})} = T_{num} \cdot \epsilon_{\alpha^o}^{(1)}$. Note here (ϵ_1, δ_1)-DP equivalently satisfies $(\alpha^o, \epsilon_{\alpha^o}^{(1)})$-RDP, and $\epsilon_{\alpha^o}^{(1)}$ is a function of the sampling ratio B/M and other related mechanism parameters. For a detailed computation on $\epsilon_{\alpha^o}^{(1)}$, we refer the reader to [64] and skip the derivation here. We also refer the reader to [92] for implementation details.

Therefore, the protocol in Fig. 1 is (ϵ, δ) differentially private with respect to each private dataset $\mathcal{D}_i$, with $i \in [1, N]$. □

D.2 Proof of Theorem 3

Proof. To the protocol in Fig. 2, with local models are frozen and the b-min-separated participation that applied on all clients' datasets is pre-defined, the workload matrix $\boldsymbol{A}$ as well as the p-BSR matrix $\boldsymbol{\Omega}$ are therefore constructed. Furthermore, the sensitivity of $\boldsymbol{\Omega}$, denoted $\text{sens}_{\kappa,\text{b}}(\Omega)$, is computed and defined.

At each step, due to the clipping, the ℓ_2 sensitivity $\Delta_2 f$ of $[\![\overline{\boldsymbol{g}}_\theta]\!]$ is bounded by γ. Moreover, we follow the same setting in [69] to use FFT-based DP accounting [52] to compute the appropriate σ given target (ϵ, δ). As a result, the (ϵ, δ) DP guaranteed Gaussian matrix $[\![\boldsymbol{\mathcal{N}}]\!] \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 I_{T_{num} \times n_{\boldsymbol{\theta}}})$ is generated (under secured MPC) for the entire T_{num} steps, and the correlated matrix

$$[\![\boldsymbol{\mathcal{N}}_{\text{Corr}}]\!] = \Omega^{-1}[\![\boldsymbol{\mathcal{N}}_{\text{Tab}}]\!]$$

where $[\![\boldsymbol{\mathcal{N}}_{\text{Tab}}]\!] = \gamma \text{sens}_{\kappa,\text{b}}(\Omega)[\![\boldsymbol{\mathcal{N}}]\!]$ is obtained, which is still (ϵ, δ) satisfied, guaranteed by post-processing (D.3). Hence, the training mechanism that applied on the *stream* of $[\![\overline{\boldsymbol{g}}_\theta]\!]$ across all steps is exact the Matrix Factorization Mechanism (D.6), which makes the *stream* of $[\![\tilde{\boldsymbol{g}}_{\boldsymbol{\theta}}]\!]$ across all steps be (ϵ, δ) differentially private. We also refer the reader to [40] for implementation details.

Therefore, the protocol in Fig. 2 is (ϵ, δ) differentially private with respect to each private dataset $\mathcal{D}_i$, with $i \in [1, N]$. □

D.3 Proof of Theorem 4

Proof. To the protocol in Fig. 3, at each step, the information that revealed by the servers is $[\![\tilde{\boldsymbol{g}}]\!] = \frac{1}{B}([\![\overline{\boldsymbol{g}}]\!] + [\![\boldsymbol{\mathcal{N}}_{\tilde{\boldsymbol{g}}}]\!])$. As discussed in D.2, with the sensitivity of $\boldsymbol{\Omega}$ is defined, as well as the ℓ_2 sensitivity $\Delta_2 f$ of $[\![\overline{\boldsymbol{g}}]\!]$ is also bounded by γ, the *stream* of $[\![\tilde{\boldsymbol{g}}]\!]$ across all T_{num} steps is (ϵ, δ) differentially private under the Matrix Factorization mechanism (D.6).

As the output of the differentially private training mechanism technically consists of the full sequence of per-iteration privatized gradients, hence, as the intermediate result, the privatized gradient at each step $[\![\tilde{\boldsymbol{g}}]\!]$ is also DP protected [16]. Therefore, at each step, at the global model part, $[\![\tilde{\boldsymbol{g}}_{\boldsymbol{\theta}}]\!]$, is naturally differentially private as $[\![\tilde{\boldsymbol{g}}_{\boldsymbol{\theta}}]\!]$ is a sub-vector of $[\![\tilde{\boldsymbol{g}}]\!]$. At each local model part, as a sub-vector of $\tilde{\boldsymbol{g}}$, $\tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L}$ is also (ϵ, δ) differentially private.

Furthermore, all the information with respect to $\mathcal{D}_j, j \neq i$ that is derived by each client E_i from $\tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L}$ is also (ϵ, δ) differentially private, guaranteed by the post-processing (D.3) of differential privacy: In Fig. 3, at each client, here $\mathcal{M}_R(\cdot)$, as defined in D.3, functions to extract estimates of the per-sample gradients from $\tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L}$.

Therefore, the protocol in Fig. 3 is (ϵ, δ) differentially private with respect to each private dataset $\mathcal{D}_i$, with $i \in [1, N]$. □

E Estimation Error Bound Analysis

Considering to solve the following linear system

$$\tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L} = \frac{1}{B}(\boldsymbol{H}_{\boldsymbol{\phi}_i^L} \cdot \widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}} + \boldsymbol{\mathcal{N}}_i) , \quad \text{with} \quad \|\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}\|_{\ell_2} \leq \gamma$$

where $\boldsymbol{\mathcal{N}}_i \sim \mathcal{N}(0, \sigma_t^2 I)$ and $\sigma_t = \sigma\gamma\text{sens}_{\kappa,\text{b}}(\Omega)\|\Omega^{-1}[t,:]\|_{\ell_2}^2$. First, let's define the effective noise variance after scaling as:

$$\sigma_{t(B)}^2 = \frac{\sigma_t^2}{B^2} .$$

As the RR estimator with ridge parameter $\lambda = \sigma_{t(B)}^2$ is defined by

$$\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}^{\text{ridge}} = (\boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \boldsymbol{H}_{\boldsymbol{\phi}_i^L} + \sigma_{t(B)}^2 I)^{-1} \boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L} ,$$

the estimation error can be decomposed as

$$\begin{aligned}\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}^{\text{ridge}} - \widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}} &= (\boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \boldsymbol{H}_{\boldsymbol{\phi}_i^L} + \sigma_{t(B)}^2 I)^{-1} \boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \tilde{\boldsymbol{g}}_{\boldsymbol{\phi}_i^L} - \widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}} \\ &= \underbrace{-\sigma_{t(B)}^2 (\boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \boldsymbol{H}_{\boldsymbol{\phi}_i^L} + \sigma_B^2 I)^{-1} \widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}}_{\text{bias term}} + \\ &\quad \underbrace{(\boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \boldsymbol{H}_{\boldsymbol{\phi}_i^L} + \sigma_{t(B)}^2 I)^{-1} \boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \frac{\boldsymbol{\mathcal{N}}_i}{B}}_{\text{variance term}} .\end{aligned}$$

Furthermore, taking expectation over the noise, which results in

$$\mathbb{E}_{\text{Ridge}} = \mathbb{E}\|\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}^{\text{ridge}} - \widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}\|_{\ell_2}^2 = \sigma_{t(B)}^4 \|(\boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \boldsymbol{H}_{\boldsymbol{\phi}_i^L} + \sigma_B^2 I)^{-1} \widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}\|_{\ell_2}^2 +$$

$$\sigma_{t(B)}^2 \,\text{tr}\Big((\boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \boldsymbol{H}_{\boldsymbol{\phi}_i^L} + \sigma_{t(B)}^2 I)^{-1} \boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \boldsymbol{H}_{\boldsymbol{\phi}_i^L} (\boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \boldsymbol{H}_{\boldsymbol{\phi}_i^L} + \sigma_{t(B)}^2 I)^{-1}\Big) ,$$

where $\mathrm{tr}(\cdot)$ denotes the trace operation on matrix. Let $\mu_{\min}$ and $\mu_{\max}$ denote the smallest and largest eigenvalues of $\boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \boldsymbol{H}_{\boldsymbol{\phi}_i^L}$, respectively. By using $\|\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}\|_{\ell_2} \le \gamma$ and the spectral norm bound on the variance term, we could bound the two terms as follows:

Bias term. As we have

$$\|(\boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \boldsymbol{H}_{\boldsymbol{\phi}_i^L} + \sigma_{t(B)}^2 I)^{-1}\|_{\ell_2} = 1/(\mu_{\min} + \sigma_{t(B)}^2)\,,$$

by using the operator norm inequality, we could obtain

$$\begin{aligned}
&\sigma_{t(B)}^4 \|(\boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \boldsymbol{H}_{\boldsymbol{\phi}_i^L} + \sigma_B^2 I)^{-1} \widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}\|_{\ell_2}^2 \\
&\le \sigma_{t(B)}^4 \|(\boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \boldsymbol{H}_{\boldsymbol{\phi}_i^L} + \sigma_B^2 I)^{-1}\|_{\ell_2}^2 \|\widehat{\boldsymbol{g}}_{\mathbf{H}_b^{(i)}}\|_{\ell_2}^2 \\
&\le \frac{\sigma_{t(B)}^4 \gamma^2}{(\mu_{\min} + \sigma_{t(B)}^2)^2}\,.
\end{aligned}$$

Variance term. Diagonalize $\boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \boldsymbol{H}_{\boldsymbol{\phi}_i^L}$ with eigenvalues μ_i where $i = 1, \ldots, d_{\boldsymbol{H}^{(i)}} B$. The variance term equals

$$\begin{aligned}
&\sigma_{t(B)}^2 \,\mathrm{tr}\Big((\boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \boldsymbol{H}_{\boldsymbol{\phi}_i^L} + \sigma_{t(B)}^2 I)^{-1} \boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \boldsymbol{H}_{\boldsymbol{\phi}_i^L} (\boldsymbol{H}_{\boldsymbol{\phi}_i^L}^T \boldsymbol{H}_{\boldsymbol{\phi}_i^L} + \sigma_{t(B)}^2 I)^{-1}\Big) \\
&= \sigma_{t(B)}^2 \,\mathrm{tr}\big((\Lambda + \sigma_{t(B)}^2 I)^{-1} \Lambda (\Lambda + \sigma_{t(B)}^2 I)^{-1}\big) \\
&= \sigma_{t(B)}^2 \sum_{i=1}^{d_{\boldsymbol{H}^{(i)}} B} \frac{\mu_i}{(\mu_i + \sigma_{t(B)}^2)^2}\,,
\end{aligned}$$

where $\Lambda = \mathrm{diag}(\mu_i)$ for $i = 1, \ldots, d_{\boldsymbol{H}^{(i)}} B$. A relax upper bound is obtained by upper-bounding each fraction by $\mu_{\max}/(\mu_{\min} + \sigma_{t(B)}^2)^2$ and summing, which results in

$$\sigma_{t(B)}^2 \sum_{i=1}^{d_{\boldsymbol{H}^{(i)}} B} \frac{\mu_i}{(\mu_i + \sigma_{t(B)}^2)^2} \le \sigma_{t(B)}^2 \cdot d_{\boldsymbol{H}^{(i)}} B \cdot \frac{\mu_{\max}}{(\mu_{\min} + \sigma_{t(B)}^2)^2}\,.$$

Eventually, we could combine the two above bounds together, which yields the fully bounded error, as

$$\begin{aligned}
\mathbb{E}_{\text{Ridge}} &\le \frac{\sigma_{t(B)}^4 \gamma^2}{(\mu_{\min} + \sigma_{t(B)}^2)^2} + \frac{\sigma_{t(B)}^2 d_{\boldsymbol{H}^{(i)}} B\, \mu_{\max}}{(\mu_{\min} + \sigma_{t(B)}^2)^2} \\
&= \frac{\sigma_{t(B)}^4 \gamma^2 + \sigma_{t(B)}^2 d_{\boldsymbol{H}^{(i)}} B\, \mu_{\max}}{(\mu_{\min} + \sigma_{t(B)}^2)^2} \\
&= \frac{\sigma_t^4 \gamma^2 + \sigma_t^2 d_{\boldsymbol{H}^{(i)}} B^3\, \mu_{\max}}{(B^2 \mu_{\min} + \sigma_t^2)^2}\,. \qquad (12)
\end{aligned}$$

As $\boldsymbol{H}_{\boldsymbol{\phi}_i^L}$ is determined by the clients' data and local model parameters, the client can pre-compute the estimation error bound, which can serve as a "confidence measure" to monitor updates over iterations and track the reliability of the DP-noised gradients. Although the bound cannot be used to adjust batch size or DP noise (to preserve privacy), observing its trend provides valuable diagnostic information about the solver's stability and the confidence of the updates, which in turn can inform the future choice of fixed parameters or guide offline experiments.

However, the above proposal is currently conceptual, and we leave it as a future direction to explore more systematic formulations, theoretical guarantees, and practical implementations that bridge the gap between DP-noised linear estimation and potential DP-safe adaptive optimization strategies.