Private, Efficient, and Accurate: Protecting Models Trained by Multi-party Learning with Differential Privacy

Wenqiang Ruan$^\$$, Mingxin Xu$^\$$, Wenjing Fang$^+$, Li Wang$^+$, Lei Wang$^+$, Weili Han$^\$$

$^\$$*Laboratory of Data Analytics and Security, Fudan University*, $^+$*Ant Group*

$^\$${*wqruan20, 20212010078, wlhan**}*@fudan.edu.cn*, $^+${*bean.fwj, raymond.wangl, shensi.wl*}*@antgroup.com*

Abstract—Secure multi-party computation-based machine learning, referred to as multi-party learning (MPL for short), has become an important technology to utilize data from multiple parties with privacy preservation. While MPL provides rigorous security guarantees for the computation process, the models trained by MPL are still vulnerable to attacks that solely depend on access to the models. Differential privacy could help to defend against such attacks. However, the accuracy loss brought by differential privacy and the huge communication overhead of secure multi-party computation protocols make it highly challenging to balance the 3-way trade-off between privacy, efficiency, and accuracy.

In this paper, we are motivated to resolve the above issue by proposing a solution, referred to as PEA (Private, Efficient, Accurate), which consists of a secure differentially private stochastic gradient descent (DPSGD for short) protocol and two optimization methods. First, we propose a secure DPSGD protocol to enforce DPSGD, which is a popular differentially private machine learning algorithm, in secret sharing-based MPL frameworks. Second, to reduce the accuracy loss led by differential privacy noise and the huge communication overhead of MPL, we propose two optimization methods for the training process of MPL: (1) the data-independent feature extraction method, which aims to simplify the trained model structure; (2) the local data-based global model initialization method, which aims to speed up the convergence of the model training. We implement PEA in two open-source MPL frameworks: `TF-Encrypted` and `Queqiao`. The experimental results on various datasets demonstrate the efficiency and effectiveness of PEA. E.g. when $\epsilon = 2$, we can train a differentially private classification model with an accuracy of 88% for CIFAR-10 within 7 minutes under the LAN setting. This result significantly outperforms the one from `CryptGPU`, one state-of-the-art MPL framework: it costs more than 16 hours to train a non-private deep neural network model on CIFAR-10 with the same accuracy.

Index Terms—Secure Multi-party Computation, Multi-Party Learning, Differential Privacy, Privacy Computing

I. INTRODUCTION

Since European Union released General Data Protection Regulation [68], data transfer has been strictly restricted more than before [60]. In order to utilize data stored in different parties with privacy preservation, secure multi-party computation (SMPC for short) based machine learning [51], [52], [65], referred to as MPL [63], is an important technology. Benefiting from the rigorous security guarantee provided by SMPC, through using MPL, data analysts can train machine learning models based on data from multiple sources without leaking any private information except for the trained models.

However, many attacks [43], [61], e.g. membership inference attacks [61], on the machine learning models themselves have been proposed in recent years. These attacks only require access to the trained models (i.e. no raw data, even no intermediate results). For example, by using membership inference attacks, attackers can infer the membership of one data point by querying the target model [61]. As these attacks are missed by the threat models of SMPC, they would seriously break the security of MPL frameworks.

Differential privacy (DP for short) is a widely-used technology to defend against the above attacks with a rigorous privacy guarantee [32], which is different from the protections provided by current heuristic methods [34], [43]. Meanwhile, a recent user study [72] shows that users prefer to share their data when a rigorous privacy guarantee is provided. Hence, DP could be a more significant defense than existing heuristic defenses. However, the DP noise would lead to significant accuracy loss of trained models, especially for deep neural network models [1], [66]. In addition, even though there have been many studies on the efficiency optimization of MPL [51], [65], there is still a huge efficiency gap between plaintext training and secure training. Therefore, it is highly challenging to balance the 3-way trade-off between privacy, efficiency, and accuracy in the secure training of MPL. Concretely, (1) directly combining the plaintext data from all parties to efficiently train an accurate model would bring severe privacy concerns; (2) training an accurate model with existing MPL frameworks can partially protect privacy, but it requires a large amount of time to complete the training process; (3) while integrating DP into MPL frameworks without further optimizations can resolve the privacy concerns, the accuracy loss led by DP would cancel out the accuracy gain of combining multiple parties' data.

In this paper, we are motivated to resolve the above issue by proposing a solution, referred to as PEA (Private, Efficient, Accurate), which consists of a secure differentially private stochastic gradient descent (DPSGD for short) protocol and two optimization methods. First, by designing a secure inverse of square root protocol, we enforce the DPSGD algorithm [1], [2], [64] in secret sharing-based MPL frameworks to enhance their privacy protection. Second, to reduce the accuracy loss led by DP and improve the efficiency of MPL, we propose two optimization methods: (1) extracting the features of input data with data-independent feature extractors; (2) initializing the global model by aggregating local models trained on local data from parties. With the first optimization method, parties only need to securely train a differentially private simple model on extracted features, which can achieve similar, even higher

accuracy than differentially private complex models, such as Resnet with tens of layers [27], trained on raw data. In addition, the second optimization method initializes a relatively accurate global model for the secure training process, thus significantly speeding up the convergence of the model training process.

We summarize our main contributions as follows:

- We propose the secure DPSGD protocol to enforce the DPSGD algorithm in secret sharing-based MPL frameworks, thus defending against the membership inference attacks on machine learning models (Section IV).
- We propose two optimization methods to reduce the accuracy loss brought by DP and improve the efficiency of MPL frameworks: (1) the data-independent feature extraction method (Section V-A); (2) the local data-based global model initialization method (Section V-B).
- We implement PEA in two open-source MPL frameworks, namely `TF-Encrypted` [17] and `Queqiao`[1]. We refer to the differentially private versions of `TF-Encrypted` and `Queqiao` as `TF-Encrypted`$_\epsilon$ and `Queqiao`$_\epsilon$ respectively. The experimental results on three widely-used datasets show that our proposed feature extraction method and model initialization method can significantly improve the efficiency of MPL frameworks. In particular, under the LAN setting, when we set ϵ as 2, `TF-Encrypted`$_\epsilon$ and `Queqiao`$_\epsilon$ can train a differentially private classification model for CIFAR-10 with an accuracy of 88% within 7 and 55 minutes, respectively. While `CryptGPU` [65], one state-of-the-art MPL framework, requires more than 16 hours (about $137\times$ and $17\times$ of ours respectively) to train a non-private deep neural network model for CIFAR-10 with the same accuracy (Section VI). Therefore, with our proposed PEA, `TF-Encrypted`$_\epsilon$ and `Queqiao`$_\epsilon$ can balance privacy, efficiency, and accuracy in the secure model training of MPL.

II. OVERVIEW

As is shown in Figure 1, we present an overview of PEA, which works in the training process with the support of SMPC protocols. In the following parts, we introduce the motivation and design of our proposed protocol and optimization methods.

A. Secure DPSGD Protocol

In the multi-party setting, local differential privacy [23], which has no dependence on a trusted server, is a straightforward method to protect privacy. However, the accuracy loss caused by local differential privacy is one or more orders of magnitude larger than that caused by central DP [37], which requires a trusted server to handle all data. Benefit from the security guarantee provided by SMPC protocols, even though in the multi-party setting, MPL can simulate a trusted server that trains the model on all data from distributed parties.

According to the above analysis, we leverage the DPSGD algorithm [1], [2], [64] in the multi-party setting to protect models trained by MPL frameworks. DPSGD adds noise sampled from Gaussian distribution to gradient vectors with
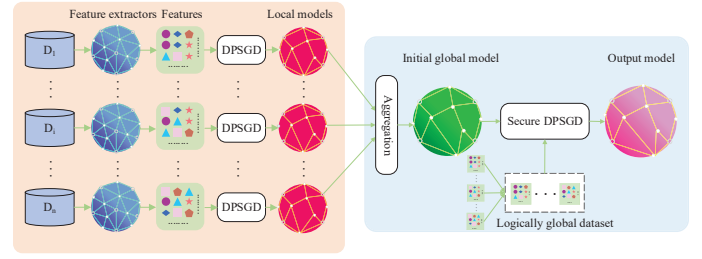
Figure 1: An overview of PEA. The computations in the orange (left) frame are performed locally on plaintext data, and computations in the blue (right) frame are performed globally on secret shared data. During the global computation, all local datasets are merged as the logically global dataset through using SMPC protocols.

bounded L_2 norms to protect training data. We thus design a secret sharing-based inverse of square root protocol to clip L_2 norms of gradient vectors and utilize the distributed noise generation protocol proposed by Dwork et al. [22] to generate random Gaussian noise securely.

B. Efficiency and Accuracy Optimization Methods

Existing MPL frameworks [51], [52], [65], [70] try to securely train end-to-end deep neural network models (i.e. training neural network models that combine feature extraction and classification). However, combining DP with deep neural network models could result in significant accuracy loss [1], [66]. Furthermore, under the computation and communication complexity of current SMPC protocols, it is difficult, if not impractical, to securely train an end-to-end deep neural network model. We elaborate on these two arguments in Section V-A.

Instead of training an end-to-end neural network model, we let each party first apply data-independent feature extractors, including foundation models [5] (e.g. BERT [21]) and feature extraction rules, to extract features of input datasets locally. Consequently, all parties collaboratively train shallow models (e.g. linear models or neural network models with a few layers) on extracted features. In this way, parties only need to perform secure computation and add noise in the shallow model training process. As a result, the accuracy loss and communication overhead can be significantly reduced.

Furthermore, when we train a machine learning model, starting from an accurate initial model would significantly improve the convergence speed. Our main idea is to let each party train a local model on his or her local data with DPSGD. Subsequently, parties securely aggregate their local models with SMPC protocols to initialize a global model. By performing parts of the computation of the training process locally, parties can therefore significantly improve the efficiency of MPL.

III. PRELIMINARIES

We first describe the security model used in this paper. After that, we introduce background knowledge of SMPC and differentially private machine learning. For clarity purposes, we list notations used in the paper in Table I.

TABLE I: Notations used in this paper.

P_i	Party i participating in MPL
m	The number of parties involved in MPL
$\langle x \rangle_i$	One share of a secret x held by P_i
k	The number of bits used to represent a fixed-point number
f	The bit number of fixed-point number's fraction part
p	The dimension of gradient vectors
n	The number of data samples
θ	Trained model
C	The gradient clipping bound
η	Learning rate
B	Batch size

A. Security Model

In this paper, we consider the semi-honest security model, which is widely used in MPL frameworks [51], [52]. In the model, each party tries to infer private information from all received messages but will follow the steps of protocols. The semi-honest model should be a reasonable setting because the primary goal of MPL is to bridge the data sources separated by strict privacy protection regulations, and parties are willing to share their data. Furthermore, following previous studies [51], [65], we assume an honest majority, i.e. less than half of parties would collude. In summary, PEA guarantees the security of data during the model training process in the semi-honest setting with an honest majority, and provides DP guarantees to defend against membership inference attacks on output models.

B. Secure Multi-party Computation

SMPC enables multiple parties to collaboratively execute a function without leaking any information of each party's input except the computation results. There are several technical routes [16], [76], [77] to implement SMPC, including secret sharing [16]. The main idea of secret sharing is to divide a private value x into m shares, where m is the number of parties. Subsequently, parties perform computations of a target function on the shares and finally reconstruct the computation results as the output. The major MPL frameworks are based on linear secret sharing protocols (i.e. additive secret sharing [25] and Shamir's secret sharing [3]) for their high efficiency of arithmetic computations [51], [52], [65], [70].

We construct the secure DPSGD protocol based on linear secret sharing protocols. Because different secret sharing schemes have different sharing semantics, we use the primitives that are supported by all linear secret sharing protocols to construct our proposed secure DPSGD protocol. Thus our proposed protocol can be applied to linear secret sharing-based MPL frameworks, including `ABY3` [51] and `Queqiao`.

We hereby introduce the fixed-point number representation method and cryptographic primitives used in this paper. As these primitives are used as black-boxes, we only introduce details of secret sharing and reconstruction. For other primitives, we present the functionality of these primitives and refer details to the related literature [3], [9], [10], [25]. Note that in our computation process, all parties are usually equal to each other. This feature is different from federated learning [74], where there is usually a central server among parties.

Fixed-point Number Representation. Original linear secret sharing protocols [3], [25] only support secure computations on integers. However, the model training process often involves floating-point numbers. Therefore, we encode a fixed-point number with f-bit precision by mapping a fixed-point number $\widetilde{x}$ to an integer $\overline{x}$ such that $\overline{x} = \widetilde{x} * 2^f$. In order to prevent the precision explosion brought by fixed-point number multiplication (i.e. $\overline{z} = \overline{x} * \overline{y} = \widetilde{x} * \widetilde{y} * 2^{2f}$), we apply Trunc protocol [9] to truncate $\overline{z}$ as $\overline{z}'$ such that $\overline{z}' = \widetilde{x} * \widetilde{y} * 2^f$. Trunc receives three parameters: x, k, q, where x is the data to be truncated, k, q indicate the number of bits to represent the input data and the number of bits to be truncated respectively.

Secret Sharing and Reconstruction. For additive secret sharing [25], given a secret x, we first generate $m-1$ random number $\langle x \rangle_1, \langle x \rangle_2, \cdots, \langle x \rangle_{m-1}$, and let $\langle x \rangle_m = x - \sum_{i=1}^{m-1} \langle x \rangle_i$. P_i holds $\langle x \rangle_i$. To reconstruct x, all parties exchange $\langle x \rangle_i$ and add them together to obtain x. For Shamir's secret sharing [3], given a secret x, we first generate a polynomial $f(y) = a_0 + a_1 y + a_2 y^2 + ... + a_t y^t$, where $a_0 = x, a_1, \cdots, a_t$ are random coefficients, and $t+1$ is the least number of parties to reconstruct the secret; then we construct m points out of it with indexes $1 \le i \le m$. These $f(i)$ (i.e. $\langle x \rangle_i$) are exactly the shares of x. To reconstruct x, all or some of all parties exchange $\langle x \rangle_i$ and reconstruct the polynomial $f(y)$ with the Lagrange interpolation to obtain x.

Unary Operations. Given one share $\langle x \rangle_i$ held by P_i:
- Bit_Dec outputs $\langle x_0 \rangle_i, \langle x_1 \rangle_i, \cdots, \langle x_{k-1} \rangle_i$ such that $x = \sum_{t=0}^{k-1} x_t * 2^t$.
- Mod2 outputs $\langle t \rangle_i$ for P_i such that $t = x \mod 2$.

Binary Operations. Given two shares $\langle x \rangle_i, \langle y \rangle_i$ held by P_i:
- Addition outputs $\langle z \rangle_i$ for P_i such that $z = x + y$.
- Multiplication outputs $\langle z \rangle_i$ for P_i such that $z = x * y$.
- Comparison outputs $\langle z \rangle_i$ for P_i such that $z = x > y?0 : 1$.

Multiple-Input Operations. Given a sequence of shares $\langle x_0 \rangle_i, \langle x_1 \rangle_i, \cdots, \langle x_{k-1} \rangle_i$ held by P_i:
- SufOr outputs $\langle c_0 \rangle_i, \langle c_1 \rangle_i, \cdots, \langle c_{k-1} \rangle_i$ such that $c_0 = x_0 \vee x_1 \vee \cdots \vee x_{k-1}, \cdots, c_{k-1} = x_{k-1}$.
- PreMulC output $\langle c_0 \rangle_i, \langle c_1 \rangle_i, \cdots, \langle c_{k-1} \rangle_i$ such that $c_0 = x_0, c_1 = x_0 * x_1, \cdots, c_{k-1} = x_0 * x_1 * \cdots * x_{k-1}$.

C. Differentially Private Machine Learning

Definition 1. (ϵ, δ)**-Differential Privacy [23].** *For a random mechanism M whose input is D and outputs $r \in R^p$, we say M is (ϵ, δ)-differential private if for any subset $S \subseteq R^p$,*

$$Pr(M(D) \in S) \le e^\epsilon \cdot Pr(M(D') \in S) + \delta$$

where D and D' are neighboring datasets, i.e. we can obtain D' by deleting or adding one data sample of D. If δ is set as 0, we say M is ϵ-differentially private. ϵ is also called privacy budget that represents the privacy loss of a DP mechanism. When multiple DP mechanisms are performed on the same dataset simultaneously, the total privacy budget is estimated by composition theorems, e.g. sequential composition theorem [23] and parallel composition theorem [48].

Theorem 1. *Sequential Composition Theorem [23]. For n mechanisms $M_1, M_2, \cdots, M_n$ that access the same dataset*

D, if each M_i satisfies (ϵ_i, δ_i)-DP, the combination of their outputs satisfies (ϵ, δ)-DP with $\epsilon = \epsilon_1 + \epsilon_2 + \cdots + \epsilon_n$, $\delta = \delta_1 + \delta_2 + \cdots + \delta_n$.

Theorem 2. *Parallel Composition Theorem [48]. For n mechanisms $M_1, M_2, \cdots, M_n$ whose inputs are disjoint datasets $D_1, D_2, \cdots, D_n$, if M_i satisfies ϵ_i-DP, the combination of them satisfies ϵ-DP for the dataset $D = D_1 \cup D_2 \cup \cdots \cup D_n$ with $\epsilon = Max(\epsilon_1, \epsilon_2, \cdots, \epsilon_n)$.*

Based on Theorem 2, we elaborate the parallel composition theorem for (ϵ, δ)-DP in the Lemma 1, whose proof is shown in Appendix A.

Lemma 1. *Parallel Composition Theorem for (ϵ, δ)-Differential Privacy. For n mechanisms $M_1, M_2, \cdots, M_n$ whose inputs are disjoint datasets $D_1, D_2, \cdots, D_n$, if M_i satisfies (ϵ_i, δ_i)-DP, the combination of them satisfies (ϵ, δ)-DP for the dataset $D = D_1 \cup D_2 \cup \cdots \cup D_n$ with $\epsilon = Max(\epsilon_1, \epsilon_2, \cdots, \epsilon_n)$, $\delta = Max(\delta_1, \delta_2, \cdots, \delta_n)$.*

In addition to composition theorems, another important property of DP is post-processing immunity [23]. That is, performing data-independent computations over the differentially private output will not influence its privacy guarantee.

Algorithm 1 DPSGD [1], [2], [64]

Input: Data set $D = \{d_1, \cdots, d_n\}$, loss function: $\ell(\theta; d_i)$, privacy parameters: (ϵ, δ), number of iterations: T, batch size: B, gradient clipping bound C, learning rate function: η;
Output: Trained model θ_T;
1: Randomly initialize model θ_1;
2: **for** $t = 1$ to T **do**
3:     $s_1, \cdots, s_B \leftarrow$ Sample B samples uniformly from D;
4:     For each s_i, compute $\nabla\ell(\theta; s_i)$;
5:     **Clip gradient**
6:     $\nabla\ell(\theta; s_i) \leftarrow \nabla\ell(\theta; s_i) * min(1, \frac{C}{\|\nabla\ell(\theta;s_i)\|_2})$;
7:     **Add noise and Descent**
8:     $b_t \sim \mathcal{N}(0, \sigma^2 I_{p\times p})$, where $\sigma \geq C\frac{2\sqrt{T\log\frac{1}{\delta}}}{\epsilon}$;
9:     $\theta_{t+1} = \theta_t - \frac{\eta(t)}{B}[\sum_{i=1}^{B} \nabla\ell(\theta; s_i) + b_t]$;
10: **end for**
11: Output θ_{T+1};

We then introduce the DPSGD algorithm, whose details are shown in Algorithm 1. In order to securely perform DPSGD, besides gradient computation and descent that have been implemented in existing MPL frameworks, there are two key steps to be completed: (1) securely clipping the gradient vectors such that their L_2 norms are all lower than or equal to a given constant C; (2) securely generating the random noise vectors sampled from Gaussian distribution.

IV. SECURE DPSGD

In this section, we first introduce the secure inverse of square root and secure Gaussian noise generation protocols. Then we combine these two protocols to design the secure DPSGD protocol.

A. Inverse of Square Root

In order to clip the L_2 norm of one gradient vector $\mathbf{g}$, parties need to compute the inverse of $\|\mathbf{g}\|_2$, i.e. $\frac{1}{\sqrt{g_1^2 + g_1^2 + \cdots + g_p^2}}$. As this computation involves two functions of square root and division, which are both non-linear, directly implementing them with secret sharing protocols implies huge communication overhead. To reduce the overhead, a feasible way is to use polynomial to approximate the non-linear functions. Lu et al. [45] proposed to transform the original secret value $\langle x \rangle_i$ to a form $\langle x' * 2^{exp} \rangle_i$, where $0.25 \leq x' \leq 0.5$. Afterwards, they separately compute the $\langle \frac{1}{\sqrt{x'}} \rangle_i$ and $\langle 2^{-\frac{exp}{2}} \rangle_i$. Finally, they obtain $\langle \frac{1}{\sqrt{x}} \rangle_i$ by multiplying $\langle \frac{1}{\sqrt{x'}} \rangle_i$ and $\langle 2^{-\frac{exp}{2}} \rangle_i$. However, their method has an issue, i.e. the error of their polynomial approximation might destroy the privacy guarantee of DP. That is, if true $\frac{C}{\|\nabla\ell(\theta;s_i)\|_2}$ is slightly smaller than 1 (i.e. $\|\nabla\ell(\theta;s_i)\|_2 > C$) and approximated $\frac{C}{\|\nabla\ell(\theta;s_i)\|_2}$ is slightly higher than 1 (i.e. $\|\nabla\ell(\theta;s_i)\|_2^{approx} < C$), the gradient vector will not be clipped according to Line 6 of Algorithm 1, and its L_2 norm will exceed C. Consequently, the privacy guarantee of DP is destroyed.

Protocol 1 Inverse of Square Root

Input: P_i holds the shares of the input $\langle x \rangle_i$;
Output: P_i obtains the shares of the output $\langle y \rangle_i$, such that $y = \frac{1}{\sqrt{x}}$;
1:   **Data preparation and transformation**
2:   $\{\langle x_t \rangle_i\}_{t=0}^{k-1} \leftarrow \mathsf{Bit\_Dec}(\langle x \rangle_i)$;
3:   $\{\langle c_t \rangle_i\}_{t=0}^{k-2} \leftarrow \mathsf{SufOr}(\{\langle x_t \rangle_i\}_{t=0}^{k-2})$;
4:   $\langle b \rangle_i = 1 + \sum_{t=0}^{k-2} 2^{k-2-t}(1 - \langle c_t \rangle_i)$;
5:   $\langle x' \rangle_i = \mathsf{Trunc}(\langle b \rangle_i * \langle x \rangle_i, k, k - f - 1)$;
6:   $\langle exp \rangle_i \leftarrow \sum_{t=0}^{k-2} \langle c_t \rangle_i - f$;
7:   **Compute the square root of 2^{-exp};**
8:   $\langle lsb \rangle_i = \mathsf{Mod2}(\langle exp \rangle_i)$;
9:   $\langle \frac{exp}{2} \rangle_i = \mathsf{Trunc}(\langle exp \rangle_i, k, 1)$;
10: $\{\langle e_t \rangle_i\}_{t=0}^{k-1} = \mathsf{Bit\_Dec}(\langle f - \frac{exp}{2} \rangle_i)$;
11: **for** $t = 0$ to $k - 1$ **do**
12:    $\langle e_t \rangle_i = (1 + \langle e_t \rangle_i * (2^{2^t} - 1))$;
13: **end for**
14: $\{\langle c_t \rangle_i\}_{t=0}^{k-1} = \mathsf{PreMulC}(\{\langle e_t \rangle_i\}_{t=0}^{k-1})$;
15: $\langle 2^{f - \frac{exp}{2}} \rangle_i = \langle c_{k-1} \rangle_i$;
16: $\langle 2^{f - \frac{exp}{2} - \frac{1}{2}} \rangle_i = \mathsf{Trunc}(\langle 2^{f - \frac{exp}{2}} \rangle_i * 2^{f - \frac{1}{2}}, k, f)$;
17: $\langle 2^{f - \frac{exp}{2}} \rangle_i = \langle 2^{f - \frac{exp}{2} - \frac{1}{2}} \rangle_i * \langle lsb \rangle_i + \langle 2^{f - \frac{exp}{2}} \rangle_i * (1 - \langle lsb \rangle_i)$;
18: **Compute approximated polynomial**
19: $\langle x'_{approx} \rangle_i = \mathsf{Trunc}(0.8277 * 2^f * \langle x' \rangle_i, k, f)$;
20: $\langle x'_{approx} \rangle_i = \mathsf{Trunc}((\langle x'_{approx} \rangle_i - 2.046 * 2^f) * \langle x' \rangle_i, k, f)$;
21: $\langle x'_{approx} \rangle_i = \langle x'_{approx} \rangle_i + 2.223 * 2^f - 0.0048 * 2^f$;
22: **Compute and output final result**
23: $\langle y \rangle_i = \mathsf{Trunc}(\langle x'_{approx} \rangle_i * \langle 2^{f - \frac{exp}{2}} \rangle_i, k, f)$;
24: P_i obtains $\langle y \rangle_i$;

In order to resolve the above issue, we propose the inverse of square root protocol based on linear secret sharing protocols and prove an error bound of the approximated polynomial to keep the rigorous privacy guarantee of DP. As is shown in Protocol 1, we describe each step of the inverse of square root protocol as follows:

• **Step 1.** (Line 2-6) P_i first decomposes $\langle x\rangle_i$ as binary encoding $\{\langle x_t\rangle_i\}_{t=0}^{k-1}$ and performs SufOr on $\{\langle x_t\rangle_i\}_{t=0}^{k-2}$ so that the bits after the first 1 in the binary encoding are all set as 1. Finally, P_i transforms $\langle x\rangle_i$ to $\langle x'\rangle_i$ and obtains the $\langle exp\rangle_i$ such that $x' \in [0.5, 1)$ and $x = x' * 2^{exp}$.

• **Step 2.** (Line 8-17) P_i computes the fixed-point representation of $\langle 2^{-\frac{exp}{2}}\rangle_i$, i.e. $\langle 2^{f-\frac{exp}{2}}\rangle_i$. First, P_i obtains the least significant bit (i.e. $\langle lsb\rangle_i$) of exp, which indicates the parity of exp through applying Mod2. Subsequently, P_i computes $\langle 2^{f-\frac{exp}{2}}\rangle_i$ and $\langle 2^{f-\frac{exp}{2}-\frac{1}{2}}\rangle_i$ respectively, where $\frac{exp}{2} = \frac{exp-1}{2}$ if exp is odd. Finally, P_i obtain $\langle 2^{f-\frac{exp}{2}}\rangle_i$ by choosing the correct value based on the value of $\langle lsb\rangle_i$.

• **Step 3.** (Line 19-21) P_i approximates $\frac{1}{\sqrt{x'}}$ for $x' \in [0.5, 1)$ with polynomial $0.8277x'^2 - 2.046x' + 2.223$, which is obtained through using the polyfit function of NumPy [2]. Next, P_i subtracts the approximated value with error bound 0.0048 to ensure that the approximated value is strictly smaller than the true value, thus keeping the rigorous privacy guarantee.

• **Step 4.** (Line 23-24) P_i multiplies the approximated value $\langle x'_{approx}\rangle_i$ with $\langle 2^{f-\frac{exp}{2}}\rangle_i$ to obtain the output $\langle y\rangle_i$.

We bound the difference between our approximated polynomial and true output as follows.

Lemma 2. *For any* $x' \in [0.5, 1), (0.8277x'^2 - 2.046x' + 2.223 - 0.0048) - \frac{1}{\sqrt{x'}} < 0.$

We show the proof of Lemma 2 in Appendix A. When the approximated $\frac{1}{\sqrt{x'}}$ is smaller than the true $\frac{1}{\sqrt{x'}}$, the approximated $\frac{1}{\sqrt{x}}$ is smaller than the true $\frac{1}{\sqrt{x}}$. Therefore, the approximated $\frac{C}{\|\nabla \ell(\theta;s_i)\|_2^{approx}}$ (i.e. $\frac{1}{\sqrt{\sum_{j=0}^{p-1} \nabla \ell(\theta;s_i)_j^2}} * C$) is smaller than the true value. Consequently, according to Algorithm 1, we can guarantee that gradient vectors whose L_2 norms are larger than C (i.e. $\frac{C}{\|\nabla \ell(\theta;s_i)\|_2} < 1$) are all clipped. Finally, we can obtain the DP guarantee of true gradient vectors. We further analyze the security and communication complexity of Protocol 1 in Appendix C-A.

B. Noise Generation

We refer to the noise generation protocol proposed by Dwork et al. [22] to securely generate the Gaussian noise with mean 0 and standard deviation σ. The details are shown in Protocol 2. Benefit from the additivity of Gaussian distribution, the sum of independent Gaussian noises from parties is the target Gaussian noise. According to [22], each party generates a Gaussian noise with variance $\frac{3}{2m}\sigma^2 I_{p\times p}$ to preserve Byzantine robustness of Protocol 2. Byzantine robustness means that if not more than 1/3 of parties fail, parties can generate the noise that is random enough. Without the Byzantine robustness here, when part of parties fails during the noise generating process, the generated noise may not be random enough to provide privacy protection, which is the main goal of our proposed PEA. Note that this is not the property of the end-to-end secure DPSGD protocol. Meanwhile, as the noise generation process only requires secure addition, its security is trivially guaranteed

---

[2]https://numpy.org/doc/stable/reference/generated/numpy.polyfit.html

by the security of secure addition. As to the communication complexity, Protocol 2 can be completed in one communication round with $O(mpk)$ bits message transfer. Besides, as the noise generation process is data-independent, it can be generated offline and has no impact on the efficiency of the online training process.

Protocol 2 Secure Gaussian Noise Generation [22]

Input: The standard deviation of the target Gaussian distribution σ, the dimension of noise p;

Output: The shares of a Gaussian noise $\langle b\rangle_i$ such that $b \sim \mathcal{N}(0, \sigma^2 I_{p\times p})$;

1: P_i samples a Gaussian noise $b_i \sim \mathcal{N}(0, \frac{3}{2m}\sigma^2 I_{p\times p})$;
2: P_i sends the shares of $\langle b_i\rangle_{j\neq i}$ to other parties $P_{j\neq i}$;
3: P_i summarizes the noise shares from other parties to obtain the result $\langle b\rangle_i = \sum_{j=1}^m \langle b_j\rangle_i$;

C. Secure DPSGD Protocol

With Protocol 1 and Protocol 2, we then introduce the secure DPSGD protocol, as is shown in Protocol 3. First, parties securely compute gradient vectors. Second, parties clip the L_2 norms of the gradient vectors through calculating the inverse of L_2 norms of the gradient vectors with Protocol 1 and storing the comparison results between them with 1 in is_clip. After that, parties update the gradient vectors according to the values of is_clip. Subsequently, all parties collaboratively generate Gaussian noises with a pre-defined standard deviation σ through calling Protocol 2 and apply them to perturb the gradient vectors. Finally, parties update model parameters θ by following the regular gradient descent process. Note that the model parameters are kept in the secret shared form through the whole secure DPSGD process. We analyze the security and communication complexity of Protocol 3 in Appendix C-B.

Privacy Guarantee. We show the privacy guarantee of Protocol 3 in Theorem 3 (its proof is shown in Appendix A).

Theorem 3. *For any* $\epsilon \leq 2\log(1/\delta)$ *and* $\delta \in (0, 1)$, *Protocol 3 satisfies* (ϵ, δ)-*DP.*

Random Sampling of Minibatches. Protocol 3 amplifies the privacy guarantee of the training process by randomly sampling minibatches from the logically global dataset. We apply the resharing-based oblivious shuffling protocol [41], which can securely shuffle the logically global dataset while keeping the new permutation of data samples invisible to parties, to implement the random sample operation. With the shuffling protocols, we can securely sample minibatches to meet the requirement of DPSGD. Note that the oblivious shuffling protocol assumes an honest majority.

V. OPTIMIZATION METHODS FOR THE TRAINING PROCESS

In this section, we introduce two optimization methods on the accuracy and efficiency of MPL. The core ideas of the optimization methods are to simplify the structure of trained models and finish parts of computations locally, thus reducing the random noise added in the training process and the secure computations required for the training process.

Protocol 3 Secure DPSGD

Input: P_i holds a local dataset D_i, loss function: $\ell(\theta; d_i)$, privacy parameters: (ϵ, δ), number of iterations: T, batch size: B, gradient clipping bound C, learning rate function: η;

Output: P_i obtains the shares of the trained model $\langle \theta_T \rangle_i$;

1: P_i generates and sends the shares of its own dataset $\langle D_i \rangle_{j \neq i}$ to $P_{j \neq i}$. Thus, P_i holds the share of the global dataset $\langle D \rangle_i = \{\langle D_1 \rangle_i, \langle D_2 \rangle_i, \cdots, \langle D_m \rangle_i\}$;

2: Initializing $\langle \theta_1 \rangle_i$ with Protocol 4;

3: **for** $t = 1 \ to \ T$ **do**

4: $\quad \langle s_1 \rangle_i, \cdots, \langle s_B \rangle_i \leftarrow$ parties collaboratively sample B samples from D;

5: $\quad$ For each $\langle s_j \rangle_i$, compute $\langle \nabla \ell(\theta_t; s_j) \rangle_i$;

6: $\quad$ **Clip gradient**

7: $\quad$ **for** each $\langle \nabla \ell(\theta_t; s_j) \rangle_i$ **do**

8: $\qquad \langle x_j \rangle_i = \mathsf{Trunc}(\langle \nabla \ell(\theta_t; s_j) \rangle_i * \langle \nabla \ell(\theta_t; s_j)^T \rangle_i, k, f)$;

9: $\qquad$ Compute the inverse square root of $\langle x_j \rangle_i$ (i.e. $\left\langle \frac{1}{\|\nabla \ell(\theta_t; s_j)\|_2^{approx}} \right\rangle_i$) with Protocol 1;

10: $\qquad \langle is\_clip \rangle_i = \mathsf{Comparison}(\left\langle \frac{C}{\|\nabla \ell(\theta_t; s_j)\|_2^{approx}} \right\rangle_i, 1)$;

11: $\qquad \langle \nabla \ell(\theta_t; s_j)_{clipped} \rangle_i = \mathsf{Trunc}(\langle \nabla \ell(\theta_t; s_j) \rangle_i * \left\langle \frac{C}{\|\nabla \ell(\theta_t; s_j)\|_2^{approx}} \right\rangle_i, k, f)$;

12: $\qquad \langle \nabla \ell(\theta_t; s_j) \rangle_i \leftarrow \langle \nabla \ell(\theta_t; s_j) \rangle_i * (1 - \langle is\_clip \rangle_i) + \langle \nabla \ell(\theta_t; s_j)_{clipped} \rangle_i * \langle is\_clip \rangle_i$;

13: $\quad$ **end for**

14: $\quad$ **Add noise and Descent**

15: $\quad$ Generate noise $\langle b_t \rangle_i$ such that $b_t \sim \mathcal{N}(0, \sigma^2 I_{p \times p})$ with Protocol 2, where $\sigma \geq C \frac{2\sqrt{T \log \frac{1}{\delta}}}{\epsilon}$;

16: $\quad \langle update \rangle_i = \mathsf{Trunc}(\frac{\eta}{B}[\sum_{j=1}^{B} \langle \nabla \ell(\theta_t; s_j) \rangle_i + \langle b_t \rangle_i], k, f)$;

17: $\quad \langle \theta_{t+1} \rangle_i = \langle \theta_t \rangle_i - \langle update \rangle_i$;

18: **end for**

19: P_i obtains $\langle \theta_T \rangle_i$;

A. Data-independent Feature Extraction Method

Reason to Simplify the Trained Model in MPL. From the perspective of DP, the complex structures of deep neural network models require adding much random noise during the training process, thus resulting in large accuracy loss. First, the gradient vectors of deep neural network models are mainly high-dimension [81]. In DPSGD, as random noise is added to each gradient vector component, gradient vectors with higher dimensions require more noise. Bassily et al. [2] theoretically prove that DPSGD implies extra loss with lower bound linear in the dimension of the gradient vectors. Second, deep neural network models have many redundant parameters. Thus adding noise to these redundant parameters would cause extra accuracy loss to neural network models. The empirical results of Section VI-B further verify the above claim. Yu et al. [81] show that although optimizing original DPSGD with the assistance of a little public data, the accuracy gap between differentially private and non-private deep neural network models still keeps more than 15% for CIFAR-10.

On the other hand, from the perspective of SMPC, the representation power of deep neural network models comes at the cost of efficiency, i.e. deep neural network models have many redundant parameters to fit the data. Learning these parameters requires a large number of computations. For example, training a ResNet-50 model [27], a popular deep neural network model, on the ImageNet-1k dataset requires about 10^{18} single-precision operations [78]. In the centralized setting, domain-specific processors (e.g. GPU) and distributed computation techniques can be used to accelerate the training process. However, when training a deep neural network model with MPL, the huge communication overhead brought by SMPC protocols would become the bottleneck of performance.

In linear secret sharing protocols, to complete one fixed-point number multiplication, it is necessary for parties to exchange messages. Therefore, training a deep neural network model on high-dimension data inevitably implies a huge communication overhead. For example, Falcon [70], one communication-efficient MPL framework, requires about 46 TB data transfer and more than one month to train a VGG-16 model [62] on CIFAR-10 for 25 epochs in the LAN setting. CryptGPU, which utilizes GPU to accelerate the computation, still requires ten days to train the same model with Falcon [65], [70]. In summary, with the current secret sharing protocols and differentially private optimization algorithms, it is still impractical to securely train an accurate and differentially private end-to-end deep neural network model.

Method. In order to securely train a differentially private model with small accuracy loss and high efficiency simultaneously, we let each party first apply data-independent feature extractors, including public foundation models and heuristic feature extraction rules, to extract high-level features of input data. After that, all parties train a shallow model (e.g. linear models or shallow neural network models) on the extracted features. In recent years, training models based on foundation models [5], [13], [35], [58], which are deep neural network models pre-trained on large-scale public data, has rapidly emerged as a new paradigm of the artificial intelligence field. By utilizing the knowledge transferred from large-scale public data, foundation models can effectively extract high-level features of input data from various domains. Thus the models trained on the extracted features can be very accurate. Classical feature extraction rules also have proven their effectiveness on various tasks [20], [39]. We then analyze how the above method reduces the accuracy loss brought by DP and improves the efficiency of MPL. Because foundation models and feature extraction rules are both data-independent, parties only need to add noise in the shallow model training process. Recent studies [31], [66] have shown that we can apply DP to train shallow models with small accuracy loss privately. Therefore, training models on extracted features can significantly reduce the accuracy loss brought by DP. We further empirically verify this claim in Section VI. In addition, as feature extraction is performed locally on plaintext data, parties only need to perform the shallow model training over data in the secret shared form. Thus the number of secure computations can be significantly reduced. In summary, extracting features with data-independent feature extractors enables parties to reduce the large accuracy loss and huge communication overhead of securely training a differentially private end-to-end deep neural network model.

DP Guarantee of Data-independent Feature Extraction Method. Because we perform the global model training process on secret shared data, the extracted features are invisible to potential adversaries (i.e. other parties, the users of trained models). Meanwhile, as we utilize secure DPSGD to protect the trained model, the feature extraction operation is oblivious to the adversaries. In other words, the data-independent feature extraction can be viewed as a data-independent data pre-processing step, such as adding 1 to the features of data samples. Therefore, it should be unnecessary to add random noise to the data-independent feature extraction process. This conclusion is the same as those of previous studies [66], [80].

B. Local Data-based Global Model Initialization Method

Utilizing the local data held by each party to initialize an accurate global model would significantly reduce the number of iterations in the global model training process, thus improving the efficiency of MPL. Hereby, we introduce how to aggregate local models from all parties to initialize the global model.

Aggregation Methods. Referring to the classical aggregation method in the Federated Learning field [74] and considering that training data might be not independent and identically distributed (Non-IID for short) among parties, we propose two methods to aggregate the local models:

- **Averaging Strategy.** Averaging local model parameters as parameters of the initial global model. This method is originated from the literature of federate learning [74].
- **Accuracy Strategy.** When data are Non-IID, initializing the global model by averaging local models might cause that the accuracy of the initial global model is lower than local models. Therefore, an alternative strategy is to choose the most accurate local model as the initial model.

With the above aggregation methods, we show the global model initialization method in Protocol 4. Because the distribution status of data is difficult to measure directly, we choose an initial global model by selecting the most accurate candidate model generated by different aggregation methods. We show the details of the initial global model selection method in Lines 4-11 of Protocol 4. Parties generate candidate models with different aggregation methods and evaluate the accuracy of each candidate model, respectively. Finally, all parties obtain the most accurate initial global model by comparing the accuracy of candidate models. We analyze the security and communication complexity of Protocol 4 in Appendix C-C, where we introduce the details of the two aggregation methods.

Privacy Guarantee. Benefit from the post-processing immunity property of DP [23], we only need to analyze the privacy guarantee of the local model training phase. In the following theorem, we show the privacy guarantee of Protocol 4 respect to the logically global dataset $D = \{D_1, D_2, \cdots, D_m\}$ (the proof of Theorem 4 is shown in Appendix A).

Theorem 4. *Protocol 4 satisfies (ϵ_1, δ_1)-DP.*

C. Put Things Together: PEA

Combining the secure DPSGD protocol and the above two optimization methods, we come to PEA shown in Protocol 5.

Protocol 4 Local Data-based Global Model Initialization

Input: P_i holds a local dataset D_i, the privacy parameters ϵ_1 and δ_1, clipping bound C_1 and aggregation method set $A = \{A_1, A_2\}$;
Output: P_i obtains the share of initial global model $\langle \theta_c \rangle_i$;
1: P_i trains a local model θ_i^l with Protocol 1 and $\epsilon = \epsilon_1, \delta = \delta_1, C = C_1$;
2: P_i sends $\langle \theta_i^l \rangle_{j \neq i}$ to $P_{j \neq i}$;
3: Parties randomly initialize a candidate model θ_c and securely evaluate the accuracy of θ_c as $Accuracy_c$;
4: **for** Each $A_k \in A$ **do**
5:    P_i obtains the share of kth candidate model $\langle \theta_{ck} \rangle_i = A_k(\langle \theta_1^l \rangle_i, \langle \theta_2^l \rangle_i, \cdots, \langle \theta_m^l \rangle_i)$;
6:    All parties collaboratively evaluate the accuracy of θ_{ck} as $Accuracy_{ck}$;
7:    **if** $Accuracy_{ck} \geq Accuracy_c$ **then**
8:       $\langle \theta_c \rangle_i = \langle \theta_{ck} \rangle_i$;
9:       $Accuracy_c = Accuracy_{ck}$;
10:   **end if**
11: **end for**
12: P_i obtains $\langle \theta_c \rangle_i$ as the share of the initial global model;

Each party first extracts features of his or her input dataset with data-independent feature extractors. Note that this step is completed locally. And the following training processes are performed on the extracted features. Next, they collaboratively initialize the global model through using the method introduced in Protocol 4. Finally, all parties improve the global model on the logically global dataset with the secure DPSGD protocol introduced in Section IV. As the privacy guarantees of Protocol 3 and Protocol 4 have been shown in Theorem 3 and Theorem 4, with the sequential composition theorem introduced in Section III-C, we can show that Protocol 5 satisfies $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DP.

Theorem 5. *Protocol 5 satisfies $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DP.*

Protocol 5 Private, Efficient, and Accurate Training of MPL

Input: P_i holds a local dataset D_i, the privacy parameters ϵ_1, ϵ_2 and δ_1, δ_2, clipping bound C_1, C_2.
Output: P_i obtains the share of trained global model $\langle \theta_T \rangle_i$.
1: P_i extracts features of the local dataset D_i as FD_i with data-independent feature extractors;
2: All parties apply Protocol 4 to initialize the global model with privacy parameter (ϵ_1, δ_1) and clipping bound C_1;
3: All parties send the shares of local datasets to other parties;
4: All parties obtain the trained global model through applying secure DPSGD protocol described in Protocol 3 to improve the initial global model with privacy parameters (ϵ_2, δ_2) and clipping bound C_2;

VI. IMPLEMENTATION AND EVALUATION

In this section, we first conduct an end-to-end comparison to illustrate the effectiveness of PEA. Subsequently, we evaluate the running time and approximation error of the inverse of square root protocol, which is the main component of the secure DPSGD protocol. After that, we evaluate the effectiveness of optimization methods and compare the performance of different aggregation methods. Subsequently, we compare PEA with two state-of-the-art methods on federated learning and secure

aggregation with DP. Finally, we evaluate the accuracy of models trained by MPL to show that MPL can significantly improve the accuracy of differentially private models.

A. Implementation and Experiment Setup

Implementation. We implement PEA in two open-source MPL frameworks: `TF-Encrypted` [17][3] and `Queqiao`[4].

`TF-Encrypted` is implemented in Python and contains multiple back-end SMPC protocols. We implement PEA based on its supported `ABY3` [51] back-end protocol. As `ABY3` is designed on the integer ring rather than the prime field, we implement some primitives involved in Protocol 1 in `TF-Encrypted`$_\epsilon$ by combining some primitives of `ABY3` with other protocols. For example, we implement Bit_Dec through combining A2B protocol and the sharing conversion protocol proposed by Knott et al. [38]. We set the ring size as 2^{64} and the bit number of the fractional part of the fixed-point number f as 20 in `TF-Encrypted`$_\epsilon$.

`Queqiao` is implemented in C++ based on BGW protocol [3]. It supports secure model training on the distributed data from three+ parties. As some primitives involved in Protocol 1 are not implemented by `Queqiao`, we supplement the missed primitives in `Queqiao` to implement `Queqiao`$_\epsilon$. Finally, we set the field size as $10^{17} + 3$ and f as 20 in `Queqiao`$_\epsilon$.

Datasets. We introduce three widely-used datasets involved in our experiments as follows:

- **MNIST [42]** is a handwritten digit dataset, which is widely used in the evaluation of multi-party learning studies [51], [65]. MNIST contains 60,000 training samples and 10,000 testing samples. Each data sample represents one digit between 0 to 9 and has 28×28 gray pixels.
- **CIFAR-10 [40]** contains 60,000 32×32 RGB images in ten classes, and each class has 6,000 images. We thus randomly sample 50,000 images as the training dataset and let the rest of 10,000 images as the testing dataset.
- **IMDb [46]** is a popular dataset in the natural language processing field. It contains 50,000 film reviews from IMDb with 25,000 training samples and 25,000 testing samples. The binary label of IMDb indicates the sentiment of film reviews (i.e. positive or negative).

Dataset Partition. We partition each dataset into local datasets by randomly dividing it into m disjointed parts, where m is the number of parties. In experiments of Sections VI-B, VI-C, VI-D, VI-E, and VI-F, we set m as 3, which is the party number `TF-Encrypted`$_\epsilon$ and `Queqiao`$_\epsilon$ both support.

Feature Extractors and Classification Models. We employ HOG [18] and two foundation models in the computer vision and natural language processing (NLP) fields, namely SimCLR [14] and BERT [21], to extract features of MNIST, CIFAR-10, and IMDb, respectively. Moreover, in the following experiments, we use logistic regression as the classification model trained on the extracted features.

Experiment Settings. We run experiments on three Linux servers, each of them with 2.7 GHZ of CPU and 128GB

of RAM. We consider two types of network environments. The first one is the LAN setting with 1Gbps bandwidth and negligible latency. The second one is the WAN setting with 100 Mbps bandwidth and 20ms round-trip-time latency. We simulate these two network environments through using the tc tool[5]. To avoid the bias brought by the randomness of noise sampling, we run all experiments that involve randomness five times and plot the corresponding error bars in figures.

Parameter Settings. We set the batch size B as 128 and the gradient clipping bound C as 3. In addition, we set δ as $\frac{1}{10n}$, where n is the size of the training dataset. For the privacy budget ϵ, we set it as 2 and run all global model training processes for two epochs. We use the privacy budget accountant tools[6] released by Google to set the parameters of Gaussian distributions.

B. End-to-end Comparison

Baselines. We employ several popular differentially private deep neural network models trained on raw datasets as baselines. For MNIST and CIFAR-10, we train the differentially private LeNet model [42] and VGG-16 model [62] respectively. The non-private training of these two models is also supported by `CryptGPU` [65] and `Falcon` [70]. For IMDb, we train the same differentially private neural network model that is used by Bu et al. [6] as the baseline. We refer to it as SampleNet. We set δ and the noise multiplier of Gaussian distribution as the same value as ours during the baseline training. Then we tune ϵ as the iteration number increases. All baseline models are trained by `Opacus` framework released by Yousefpour et al. [79]. We also apply a grid search procedure to find the best hyperparameter configuration for them. The potential hyperparameters are listed in Table VIII of Appendix B.

Privacy Budget Allocation. With local datasets, we train local classification models with DPSGD. In this phase, we set $\epsilon_1 = 0.25$ and $\delta_1 = \frac{1}{100n}$. We show the values of other hyperparamters in Table VII of Appendix B. For the global model training, we set δ_2 as $\frac{9}{100n}$ and tune the total privacy budget ϵ as the iteration number increases.

Results. As is shown in Figure 2, under the same number of iterations, the differentially private logistic regression models trained by `TF-Encrypted`$_\epsilon$ and `Queqiao`$_\epsilon$ have higher accuracy than differentially private deep neural network models in all three datasets. Note that the baseline models trained on different datasets have different privacy budgets as we set different batch sizes for them. The detailed parameter settings are shown in Table IX of Appendix B.

As to the efficiency, since we only need to train a shallow model, the time consumption of one iteration is much smaller than that of the deep neural network model training. For example, the training of differentially private classifiers for CIFAR-10 costs about 0.5 seconds and 4 seconds per iteration in `TF-Encrypted`$_\epsilon$ and `Queqiao`$_\epsilon$, while `CryptGPU`

---

[3]https://github.com/tf-encrypted/tf-encrypted
[4]https://github.com/FudanMPL/SecMML

[5]https://man7.org/linux/man-pages/man8/tc.8.html
[6]https://github.com/tensorflow/privacy/blob/
7eea74a6a1cf15e2d2bd890722400edd0e470db8/research/hyperparameters_
2022/rdp_accountant.py

TABLE II: Online running time (in ms) comparison of the inverse square root protocols. We run all experiments ten times and report the average running time. The shortest running times are marked in bold.

Setting	Framework	Batch size			
		1	32	64	128
LAN	TF-Encrypted$_\epsilon$	28.22	**28.75**	**29.28**	**29.93**
	Queqiao$_\epsilon$	**22.94**	76.43	123.37	233.52
	CrypTen	170.42	236.66	271.64	318.23
WAN	TF-Encrypted$_\epsilon$	841.17	843.6	848.0	848.3
	Queqiao$_\epsilon$	**456.8**	**556.19**	**595.79**	**701.70**
	CrypTen	4696.42	6632.96	6641.89	6647.64

requires about 12 seconds to complete one iteration of a non-private VGG-16 model training, which serves as the baseline. As a result, TF-Encrypted$_\epsilon$ and Queqiao$_\epsilon$ can train a differentially private classifier for CIFAR-10 with an accuracy of more than 88% within 7 minutes and 55 minutes, while CryptGPU requires more than 16 hours to train a non-private model to achieve the same accuracy.

In addition, as we only add noise to the linear model training process, the accuracy gaps between differentially private and non-private linear models are all smaller than those of differentially private and non-private deep neural network models. These results further verify the claim of Section V-A.

C. Efficiency and Approximation Error of Protocol 1

Baseline. We compare the efficiency and accuracy of Protocol 1 with CrypTen [38], an MPL framework released by Facebook. As CrypTen cannot directly compute the inverse of square root, we implement it by sequentially calling the sqrt() and the reciprocal() APIs of CrypTen. Note that we run all three frameworks in the 3-party scenario and compare the online running time of all MPL frameworks here.

Efficiency of Protocol 1. We show the running time comparison results in Table II. Under all batch sizes, TF-Encrypted$_\epsilon$ and Queqiao$_\epsilon$ are more efficient than CrypTen in both LAN and WAN settings. This is because we use a polynomial to approximate the target function directly and only require constant communication rounds. In contrast, CrypTen uses the Newton-Raphson method to approximate the target function iteratively, thus requiring more communication overhead.

In addition, under the LAN setting, the running time of TF-Encrypted$_\epsilon$ and CrypTen changes little as the batch size increases. The main reason is that these two frameworks are built on TensorFlow and Pytorch, which both support efficient vectorized computations. While in the WAN setting, where the communication round number is the dominant factor of efficiency, the running time of Queqiao$_\epsilon$ is the smallest as it has the least communication round number. Note that we implement a part of primitives involved in Protocol 1 in TF-Encrypted$_\epsilon$ by combining multiple protocols. Thus it has more communication rounds than Queqiao$_\epsilon$.

Approximation Errors. We show the approximation error, which is calculated through subtracting exact results by approximated results, of three frameworks in Figure 3. Compared with CrypTen, which is accurate only when input values are higher than 0.1 and smaller than 200, the approximation

errors of TF-Encrypted$_\epsilon$ and Queqiao$_\epsilon$ stably keep small as the input value changes from 0.01 to 300. CrypTen has such a large error for parts of input values because it uses the Newton-Raphson method to approximate the target function. Its approximation errors are highly dependent on the pre-defined initial point and the iteration number. Therefore, CrypTen could be accurate for those input values in a fixed range while having a large error for other input values. Besides, the approximation errors of TF-Encrypted$_\epsilon$ and Queqiao$_\epsilon$, which are all smaller than 0, can prevent the privacy guarantee from being destroyed as we discuss in Section IV-A.

D. Effectiveness of Optimizations

Effectiveness of Feature Extraction. We keep other experimental settings unchanged and compare the accuracy of differentially private models trained on extracted features and raw data, respectively. For MNIST and CIFAR-10, we directly use their pixel matrices as training data. For IMDb, we perform tokenization on its textual data with BertTokenizerFast tool[7] to produce the training data that are represented as a matrix.

We show the experimental results in Figure 4. Compared with differentially private models trained on raw data, models trained on features extracted by our method can achieve higher accuracy. For CIFAR-10, which has the highest dimension (3,072) among the three datasets, the accuracy gap between models trained on extracted features and raw data is higher than 60%. For IMDb, which has rich semantics, the models trained on raw data are almost totally random. The above results show that extracting features of input data with data-independent feature extractors can significantly improve the accuracy of classification models.

Effectiveness of Global Model Initialization with Local Data. To set the baseline, we use random variables sampled from Gaussian distribution to initialize the global model, which is a standard initialization method in the machine learning field [26]. We set the mean and the standard deviation of the Gaussian distribution as 0 and 0.2. As to the privacy parameter settings, we set ϵ as the sum of ϵ_1 used in the local training phase and ϵ_2 used in the global training phase, and δ as $\frac{1}{10n}$.

We show the experimental results in Figure 5. The experimental results demonstrate the effectiveness of our proposed global model initialization with local data method. When we start training from a randomly initialized global model, even though after twice of iterations, the accuracy of trained models is still smaller than the models trained from the models that are initialized with local data. Therefore, by consuming part of the privacy budget locally, we can significantly reduce the number of secure computations in the global model training phase and improve the efficiency of the model training process.

E. Comparison of Aggregation Methods

In this section, we compare the accuracy of the global model initialized by averaging local models and choosing the most accurate one. We consider two scenarios: independent and

[7]https://github.com/huggingface/transformers/blob/master/src/transformers/models/bert/tokenization_bert_fast.py

(a) TF-Encrypted$_\epsilon$ _MNIST (b) TF-Encrypted$_\epsilon$ _CIFAR-10 (c) TF-Encrypted$_\epsilon$ _IMDb

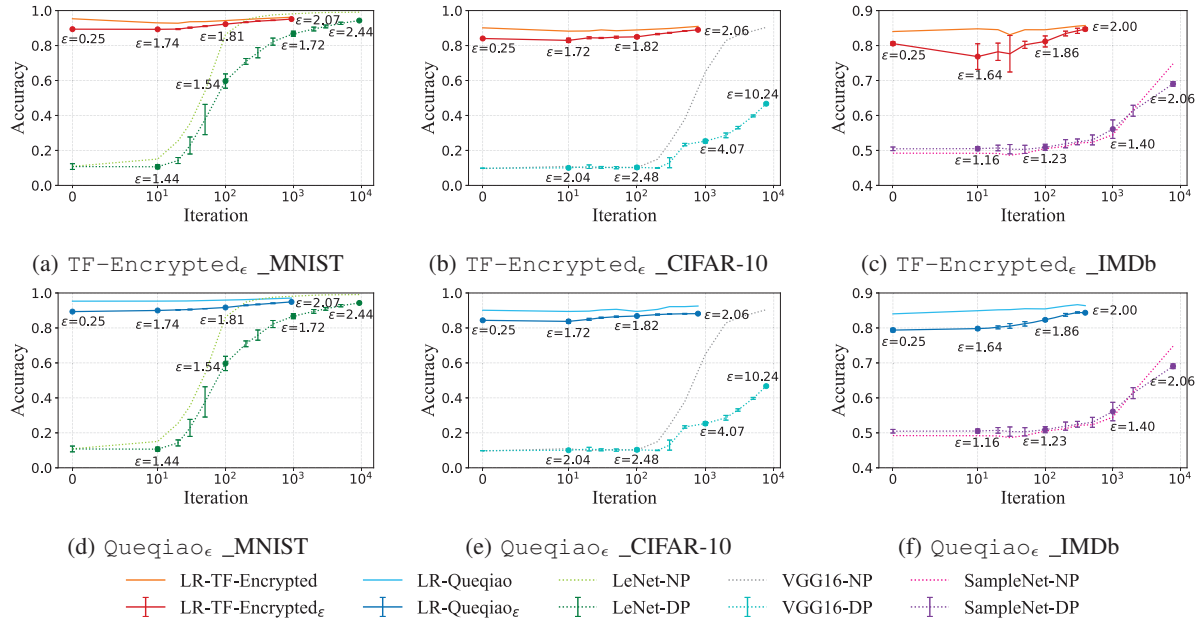(d) Queqiao$_\epsilon$ _MNIST (e) Queqiao$_\epsilon$ _CIFAR-10 (f) Queqiao$_\epsilon$ _IMDb

Figure 2: End-to-end comparison of different model training methods. LR-TF-Encrypted$_\epsilon$ and LR-Queqiao$_\epsilon$ are our proposed methods. LR-TF-Encrypted and LR-Queqiao are non-private logistic regression models trained by `TF-Encrypted` and `Queqiao`.
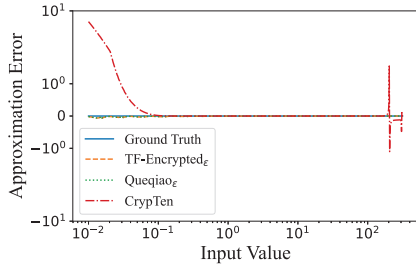

Figure 3: Approximation error comparison of the inverse square root computations.

identically distributed (IID), and Non-IID. We randomly divide the whole dataset into three parts to simulate the IID scenario. For the Non-IID scenario, following the previous study [47], we first sort the whole datasets by label values, then divide them into three parts. After partitioning the datasets, we train logistic regression models with the same parameter settings of Section VI-B and aggregate them with different methods.

TABLE III: Accuracy (in %) comparison of global models initialized by different aggregation methods. The highest accuracy values are marked in bold.

Dataset	Data distribution	Averaging Strategy	Accuracy Strategy
MNIST	IID	**89.53** (±0.26)	88.45 (±0.40)
	Non-IID	**50.56** (±1.67)	39.59 (±0.41)
CIFAR-10	IID	**83.75** (±0.36)	82.16 (±0.42)
	Non-IID	**65.39** (±1.87)	38.80 (±0.06)
IMDb	IID	**81.11** (±0.57)	80.53 (±1.01)
	Non-IID	67.48 (±3.04)	**79.10** (±1.36)

We show the accuracy comparison results in Table III. In

the IID scenario, the performance of averaging local models is slightly higher than choosing the most accurate local model. In contrast, in the Non-IID scenario, these two aggregation methods have different performances on different datasets. For MNIST and CIFAR-10, the global models initialized by averaging local models are much more accurate than those initialized by choosing the most accurate local model. While for IMDb, the global models initialized by choosing the most accurate local model are much more accurate than those initialized by averaging parameters of local models. As we choose the optimal aggregation method by comparing the accuracy of candidate models in Protocol 4, we can initialize an accurate global model for different scenarios.

F. Comparison with Federated Learning and Secure Aggregation with DP

In this section, we compare PEA with two state-of-the-art methods on federated learning and secure aggregation with DP, namely CAPC Learning (CAPC for short) [15] and Distributed Discrete Gaussian mechanism for federated learning with secure aggregation (DDGauss for short) [36]. We directly run their open-source codes[8,9] on extracted features of MNIST and CIFAR-10 datasets as baselines. We set privacy parameters of `TF-Encrypted`$_\epsilon$, `Queqiao`$_\epsilon$, and CAPC as $\epsilon = 2, \delta = \frac{1}{10n}$. For DDGauss, we set ϵ as 5 and δ as $\frac{1}{10n}$ as the training process of DDGauss cannot converge when we set ϵ as 2. For other settings (e.g. types of trained models), we set them following the description of corresponding studies [15], [36]. Note that we only run experiments on MNIST and CIFAR-10 datasets
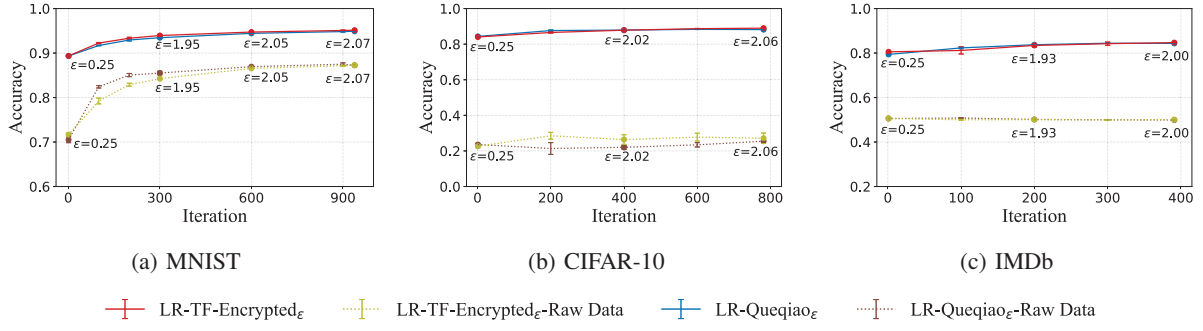
[8]https://github.com/cleverhans-lab/capc-iclr
[9]https://github.com/google-research/federated

Figure 4: Accuracy comparison of the models trained on extracted features and raw data. LR-TF-Encrypted$_\epsilon$ and LR-Queqiao$_\epsilon$ are our proposed methods.
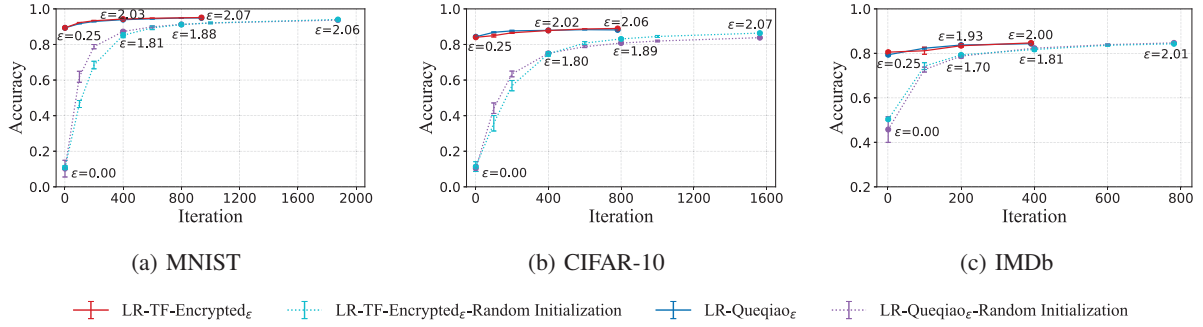

Figure 5: Convergence speed comparison of local data initialized global models and randomly initialized global models.

because the sentiment analysis task is not supported by the open-source codes of CAPC and DDGauss.

TABLE IV: Accuracy (in %) comparison of models trained by TF-Encrypted$_\epsilon$ and Queqiao$_\epsilon$ and models trained by CAPC and DDGauss. We report the average results of five runs and show the standard deviations in brackets.

	TF-Encrypted$_\epsilon$	Queqiao$_\epsilon$	CAPC [15]	DDGauss [36]
MNIST	95.14 ($\pm$0.09)	94.90 ($\pm$0.14)	92.52 ($\pm$0.20)	92.45 ($\pm$0.38)
CIFAR-10	88.99 ($\pm$0.08)	88.18 ($\pm$0.33)	86.70 ($\pm$0.17)	53.31 ($\pm$2.52)

Result. As is shown in Table IV, the models trained by TF-Encrypted$_\epsilon$ and Queqiao$_\epsilon$ are all more accurate than the models trained by CAPC and DDGauss. CAPC applies SMPC to use other parties' models to securely label one party's local data, thus the party can retrain its model on the augmented dataset. DDGauss applies SMPC to securely aggregate noisy gradients from parties. These two methods inevitably miss some information about the original dataset. In contrast, we apply SMPC protocols to simulate a trusted server that could virtually hold all data from parties to train target models [44]. Therefore, even though CAPC and DDGauss can efficiently support larger numbers of parties (50 and 500 in our experiments) than PEA, the models trained by CAPC and DDGauss suffer more accuracy loss than the models trained by TF-Encrypted$_\epsilon$ and Queqiao$_\epsilon$. Meanwhile, it is worth noting that CAPC applies DP to protect parties' private information during the data labeling process and does not consider the membership

inference attacks from end-users. We further discuss this point in Section VII.

G. Accuracy Evaluation of Models Trained by MPL

In this section, we conduct two experiments to evaluate the accuracy of models trained by MPL: (1) comparing the accuracy of global models with the accuracy of local models, and (2) evaluating the impact of SMPC on the accuracy of trained models. For the first experiment, we randomly divide the whole dataset into 3, 5, and 10 disjoint parts to simulate scenarios with different numbers of parties. After extracting features, we train differentially private logistic regression models on one plaintext subset and the logically global dataset with the same privacy parameters respectively. For the second experiment, we fix other experiment settings and compare the accuracy of models trained on the global dataset in the plaintext form and the logically global dataset merged by SMPC.

Results. We show the experimental results in Table V and Table VI respectively. For the first experiment, due to the space limitation, we only show the results of CIFAR-10 and show the results of other datasets in Appendix D. As is shown in Table V, the models globally trained by TF-Encrypted$_\epsilon$ and Queqiao$_\epsilon$ are all more accurate than the models locally trained on one subset of the whole dataset. Moreover, as the number of parties increases, the accuracy gap between locally trained and globally trained models becomes larger. There are mainly two reasons behind the above result. First, more

1936

TABLE V: Accuracy (in %) comparison of models securely trained on the logically global dataset and models locally trained on one subset of the whole dataset for CIFAR-10. We report the average results of five runs and show the standard deviations in brackets. Note that we only show the 3-party result of `TF-Encrypted`$_\epsilon$ because its back-end SMPC protocol (i.e. `ABY3`) is tailored for the 3-party scenario.

#Party	ϵ	Local party	TF-Encrypted$_\epsilon$	Queqiao$_\epsilon$
3	0.25	82.65 (±0.30)	84.03 (±0.28)	84.35 (±0.51)
	1	86.43 (±0.21)	88.57 (±0.11)	87.79 (±0.27)
	2	87.15 (±0.13)	88.99 (±0.08)	88.18 (±0.33)
5	0.25	79.84 (±0.55)	-	82.09 (±0.38)
	1	85.70 (±0.37)	-	88.05 (±0.25)
	2	86.07 (±0.09)	-	88.34 (±0.24)
10	0.25	69.49 (±1.14)	-	77.71 (±0.39)
	1	81.32 (±1.01)	-	88.05 (±0.39)
	2	83.35 (±0.52)	-	88.09 (±0.14)

training data provide more features, thus producing a more accurate model. Second, a larger training dataset implies a smaller sample probability, thus requiring less random noise to achieve the same privacy protection level.

For the second experiment, as is shown in Table VI, SMPC protocols have a negligible impact (i.e. less than 0.9%) on the accuracy of trained models. This is because we apply SMPC protocols to simulate a trusted server that could virtually hold all data to train the target model [44]. The possible impact of SMPC mainly comes from two factors: (1) we perform the computation over fixed-point numbers. However, they can only express the decimal number with limited precision, which brings a negligible accuracy loss. (2) we use linear piecewise functions to approximate non-linear activation functions (e.g. sigmoid function), which would bring another negligible accuracy loss. In summary, MPL can significantly improve the accuracy of differentially private models.

TABLE VI: Accuracy (in %) comparison of differentially private models trained by MPL on distributed data and models centrally trained on the global dataset in plaintext. We report the average results of five runs and show the standard deviations in brackets. We set ϵ as 2 when training the models.

	TF-Encrypted$_\epsilon$	Queqiao$_\epsilon$	Plaintext
MNIST	95.14 (±0.09)	94.90 (±0.14)	95.35 (±0.07)
CIFAR-10	88.99 (±0.08)	88.18 (±0.33)	89.00 (±0.13)
IMDb	84.70 (±0.44)	84.35 (±0.17)	85.19 (±0.14)

VII. RELATED WORK

Multi-party Learning. In recent years, implementing privacy-preserving machine learning based on SMPC protocols has become a hot topic in the security field. Nikolaenko et al. [53] combined homomorphic encryption and garbled circuits to train ridge regression models securely. Mohassel and Zhang [52] first applied additive secret sharing to train neural network models securely. After that, many following studies were proposed to improve the efficiency of the training process [51], [65], [70], enhance security models [19], or support the training of more complex models [65], [70]. For example, `CryptGPU` [65] and `Falcon` [70] improved the efficiency of computation

primitives and added more computation primitives to support the training of deep neural network models (e.g. VGG-16). Different from previous studies, which optimize the efficiency of computation primitives involved in the model training process, we try to optimize the training of MPL from the model training process aspect, i.e. reducing the secure computations required for the training process.

Differentially Private Machine Learning. There are mainly three paradigms in differentially private machine learning algorithms: objective perturbation, output perturbation, and gradient perturbation. Objective perturbation adds random noise to loss functions as a regularization term. Output perturbation [12], [71] adds random noise to trained models. They both assume that loss functions are convex and continuous to preserve the rigorous privacy guarantees, which significantly limits the application scenarios of these two paradigms. Therefore, we choose to enforce DPSGD [1], [2], [64], which follows the gradient perturbation paradigm and has no assumption on the loss functions, in secret sharing-based MPL frameworks to enhance the privacy of these frameworks.

Recently, there have been several studies [56], [66], [81] that utilize priors knowledge to improve the utility of DPSGD. Papernot et al. [56] improved the accuracy of models trained by DPSGD by using public data to initialize model parameters. Yu et al. [81] utilized few public data to project the high-dimension gradient vectors to a low-dimension space, thus reducing the noise added in the training process. Tramèr and Boneh [66] showed that applying heuristic rules or knowledge transferred from public data to extract features of input data could significantly reduce the accuracy loss brought by DPSGD on image classification tasks. However, their feature extraction method might be less effective in the multi-party setting. We empirically show that our feature extraction method is more suitable for the multi-party setting than that proposed by Tramèr and Boneh [66] in Appendix E. Meanwhile, we consider the sentiment analysis, which is an important task of NLP.

Intersection of SMPC and DP. Because the security models of SMPC and DP are complementary, i.e. SMPC guarantees the security of the computation process and DP preserves output privacy, there have been many studies on the intersection of SMPC and DP. Wagh et al. [69] provided an overview of recent studies on the intersection of SMPC and DP. Pettai et al. [57] applied secret sharing protocols to improve the utility of differentially privacy aggregation queries on sensitive data. Next, He et al. [28] combined secret sharing protocols and DP to securely link similar records in different databases. In recent years, as many strict privacy protection regulations have been published, some related studies [4], [7], [59] have been proposed to securely collect and analyze sensitive data in a differentially private manner. While there are several studies on secure and differentially private statistical analyses, such as aggregation queries [57], [59], median [4] or heavy hitters [7], there still lacks a combination of differentially private machine learning algorithms and MPL.

Federated Learning and Secure Aggregation with DP. In addition to our proposed secure DPSGD protocol, there is

another routine of studies on securely training differentially private machine learning models on distributed data, namely federated learning and secure aggregation with DP [11], [15], [36]. Chase et al. [11] applied garbled circuits [76] to securely aggregate noisy gradient vectors from parties. Choquette-Choo et al. proposed CAPC [15] to extend PATE [55] to the distributed setting by combining homomorphic encryption [77] and garbled circuits [76]. Kairouz et al. [36] extended the discrete gaussian mechanism [8] to the distributed setting. These studies apply SMPC protocols to securely aggregate the intermediate information (i.e. noisy labels or noisy gradient vectors) of the training process, which implies relatively higher efficiency and scalability. However, transferring intermediate information inevitably causes considerable accuracy loss to trained models [74], especially when the data are not independent and identically distributed among parties [74]. As a result, when the number of parties is small, our proposed secure DPSGD protocol is more suitable than these methods.

In addition, the security model of PEA concerns membership inference attacks from end-users, which are not concerned by the security model of CAPC [15]. The training process in CAPC consists of three steps: (1) each party trains its local model; (2) a querying party employs CAPC protocol to confidentially and privately query other parties to obtain (query data, label) pairs; (3) the querying party retrains an improved local model with the pairs. The CAPC protocol guarantees both the confidentiality of the query data and the privacy of the local models in Step (2). However, they assume that the retrained model is only used by the querying party and do not consider the membership inference attacks from other users.

VIII. DISCUSSION AND FUTURE WORK

Side-channel Attacks against Differentially Private Mechanisms. There have been several side-channel attacks [30], [49] on the implementations of DP mechanisms. Mironov [49] proposed an attack that exploits the limited precision of floating-point number arithmetic to destroy the privacy guarantee of Laplacian mechanisms. While these side-channel attacks severely break the privacy guarantee of DP mechanisms in practice, they mainly target the statistical queries on databases, where random noises are added once to query results. In DPSGD, the random noises are iteratively added in the training process, and only the trained model is released. Without the knowledge of the initial model and the information of intermediate gradient vectors, attackers cannot exploit side-channel attacks to destroy the privacy guarantee of our proposed secure DPSGD protocol.

Feature Extraction for Other Types of Data. Following previous studies [6], [52], [65], we consider the image data and textual data in our experiments. However, our proposed optimization methods are also applicable to other types of data. There have emerged several foundation models that can effectively extract the features of other types of data (e.g. protein sequences [35]) to finish downstream tasks. Therefore, our proposed optimization methods are general enough to support various types of input data.

Recent Backdoor Attacks on Foundation Models. Recently, Jia et al. [33] and Zhang et al. [82] proposed two backdoor attacks on foundation models. Here, attackers inject backdoors into downstream classification models by adding adversarial samples in the training process of foundation models. However, these two attacks assume that attackers hold a few training samples for the target tasks. With the strict protection of highly sensitive training data (e.g. medical images) in the context of MPL, attackers would be difficult to obtain these data samples, thus leading to little privacy risk to the trained models. Besides, to further defend the backdoor attacks on foundation models, the owners of trained models can employ recently proposed defenses (e.g. MNTD [73]) to detect the backdoors hidden in the trained models.

Future Work. We will integrate recent optimizations [81] on DPSGD into existing MPL frameworks to further reduce the accuracy loss brought by DP. Concretely, we first project high-dimension gradient vectors to a low-dimension space with a few auxiliary public data. Then we perturb the low-dimension gradient vectors and project perturbed gradient vectors to the original high-dimension space. In this way, we can reduce the random noise added in the model training process, then improve the accuracy of the trained differentially private model. Moreover, we will implement the state-of-the-art defenses (e.g. prediction poisoning [54] and prediction purification [75]) against model stealing attacks [67] and model inversion attacks [24], which cannot be defended by DPSGD, in MPL frameworks to further enhance their privacy protection.

IX. CONCLUSION

In this paper, we propose PEA, which can help secret sharing-based MPL frameworks to securely and efficiently train a differentially private machine learning model with little accuracy loss. After implementing PEA in two open-source MPL frameworks: `TF-Encrypted` and `Queqiao`, we conduct experiments on three datasets: MNIST, CIFAR-10, and IMDb. The experimental results demonstrate the efficiency and effectiveness of PEA. For all three datasets, $TF\text{-}Encrypted_\epsilon$ and $Queqiao_\epsilon$ can train accurate differentially private classification models within much fewer iterations than differentially private deep neural network models. In particular, compared with `CryptGPU`, $TF\text{-}Encrypted_\epsilon$ only requires less than 1% (7 minutes vs. 16 hours) of its time to train a classification model for CIFAR-10 with the same accuracy. In conclusion, with PEA, multiple parties can balance the 3-way trade-off between privacy, efficiency, and accuracy in the model training process of MPL.

REFERENCES

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

[2] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473. IEEE, 2014.

[3] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 1–10, 1988.

[4] Jonas Böhler and Florian Kerschbaum. Secure multi-party computation of differentially private median. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2147–2164. USENIX Association, August 2020.

[5] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[6] Zhiqi Bu, Jinshuo Dong, Qi Long, and Su Weijie. Deep learning with gaussian differential privacy. *Harvard Data Science Review*, 9 2020. https://hdsr.mitpress.mit.edu/pub/u24wj42y.

[7] Jonas Böhler and Florian Kerschbaum. Secure multi-party computation of differentially private heavy hitters. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, New York, NY, USA, 2021. ACM.

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

[9] Octavian Catrina. Round-efficient protocols for secure multiparty fixed-point arithmetic. In *2018 International Conference on Communications (COMM)*, pages 431–436, 2018.

[10] Octavian Catrina and Sebastiaan de Hoogh. Improved primitives for secure multiparty integer computation. In Juan A. Garay and Roberto De Prisco, editors, *Security and Cryptography for Networks*, pages 182–199, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[11] Melissa Chase, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, and Peter Rindal. Private collaborative neural network learning. *Cryptology ePrint Archive*, 2017.

[12] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

[13] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34, 2021.

[14] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020.

[15] Christopher A. Choquette-Choo, Natalie Dullerud, Adam Dziedzic, Yunxiang Zhang, Somesh Jha, Nicolas Papernot, and Xiao Wang. Ca{pc} learning: Confidential and private collaborative learning. In *International Conference on Learning Representations*, 2021.

[16] Ronald Cramer, Ivan Bjerre Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[17] Morten Dahl, Jason Mancuso, Yann Dupis, Ben Decoste, Morgan Giraud, Ian Livingstone, Justin Patriquin, and Gavin Uhma. Private machine learning in tensorflow using secure computation. *CoRR*, abs/1810.08130, 2018.

[18] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005.

[19] Anders Dalskov, Daniel Escudero, and Marcel Keller. Fantastic four: Honest-majority four-party secure computation with malicious security. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2183–2200. USENIX Association, August 2021.

[20] Quang Vinh Dang and Claudia-Lavinia Ignat. Quality assessment of wikipedia articles without feature engineering. In *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries*, pages 27–30, 2016.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding, 2018.

[22] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, pages 486–503, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[23] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, August 2014.

[24] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[25] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229. ACM, 1987.

[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[28] Xi He, Ashwin Machanavajjhala, Cheryl Flynn, and Divesh Srivastava. Composing differential privacy and secure computation: A case study on scaling private record linkage. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1389–1406, New York, NY, USA, 2017. Association for Computing Machinery.

[29] Zhicong Huang, Wen jie Lu, Cheng Hong, and Jiansheng Ding. Cheetah: Lean and fast secure Two-Party deep neural network inference. In *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA, August 2022. USENIX Association.

[30] Christina Ilvento. Implementing the exponential mechanism with base-2 differential privacy. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 717–742, New York, NY, USA, 2020. Association for Computing Machinery.

[31] Roger Iyengar, Joseph P Near, Dawn Song, Om Thakkar, Abhradeep Thakurta, and Lun Wang. Towards practical differentially private convex optimization. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 299–316. IEEE, 2019.

[32] Matthew Jagielski, Jonathan R. Ullman, and Alina Oprea. Auditing differentially private machine learning: How private is private sgd? In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[33] Jinyuan Jia, Yupei Liu, and Neil Zhenqiang Gong. BadEncoder: Backdoor attacks to pre-trained encoders in self-supervised learning. In *IEEE Symposium on Security and Privacy*, 2022.

[34] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. Memguard: Defending against black-box membership inference attacks via adversarial examples. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 259–274, New York, NY, USA, 2019. Association for Computing Machinery.

[35] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

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

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

[38] Brian Knott, Shobha Venkataraman, Awni Y. Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. Crypten: Secure multi-party computation meets machine learning. In *NeurIPS*, 2021.

[39] Takumi Kobayashi. Bfo meets hog: feature extraction based on histograms of oriented pdf gradients for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 747–754, 2013.

[40] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

[41] Sven Laur, Jan Willemson, and Bingsheng Zhang. Round-efficient oblivious database manipulation. In *International Conference on Information Security*, pages 262–277. Springer, 2011.

[42] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[43] Jiacheng Li, Ninghui Li, and Bruno Ribeiro. Membership inference attacks and defenses in classification models. In *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, pages 5–16, 2021.

[44] Yehuda Lindell. How to simulate it–a tutorial on the simulation proof technique. *Tutorials on the Foundations of Cryptography*, pages 277–346, 2017.

[45] Wen-jie Lu, Yixuan Fang, Zhicong Huang, Cheng Hong, Chaochao Chen, Hunter Qu, Yajin Zhou, and Kui Ren. Faster secure multiparty computation of adaptive gradient descent. In *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*, PPMLP'20, page 47–49, New York, NY, USA, 2020. Association for Computing Machinery.

[46] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, page 142–150, USA, 2011. Association for Computational Linguistics.

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

[48] Frank D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, page 19–30, New York, NY, USA, 2009. Association for Computing Machinery.

[49] Ilya Mironov. On significance of the least significant bits for differential privacy. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, page 650–661, New York, NY, USA, 2012. Association for Computing Machinery.

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

[51] Payman Mohassel and Peter Rindal. Aby3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 35–52, 2018.

[52] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy, SP 2017*, pages 19–38, San Jose, CA, USA, 2017. IEEE Computer Society.

[53] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE Symposium on Security and Privacy*, pages 334–348, 2013.

[54] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction poisoning: Towards defenses against dnn model stealing attacks. In *International Conference on Learning Representations*, 2020.

[55] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *International Conference on Learning Representations*, 2017.

[56] Nicolas Papernot, Steve Chien, Shuang Song, Abhradeep Thakurta, and Ulfar Erlingsson. Making the shoe fit: Architectures, initializations, and tuning for learning with privacy, 2020.

[57] Martin Pettai and Peeter Laud. Combining differential privacy and secure multiparty computation. In *Proceedings of the 31st Annual Computer Security Applications Conference*, ACSAC 2015, page 421–430, New York, NY, USA, 2015. Association for Computing Machinery.

[58] Daniel Rothchild, Alex Tamkin, Julie Yu, Ujval Misra, and Joseph Gonzalez. C5t5: Controllable generation of organic molecules with transformers. *arXiv preprint arXiv:2108.10307*, 2021.

[59] Amrita Roy Chowdhury, Chenghong Wang, Xi He, Ashwin Machanavajjhala, and Somesh Jha. Cryptϵ: Crypto-assisted differential privacy on untrusted servers. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 603–619, New York, NY, USA, 2020. Association for Computing Machinery.

[60] Wenqiang Ruan, Mingxin Xu, Haoyang Jia, Zhenhuan Wu, LuShan Song, and Weili Han. Privacy compliance: Can technology come to the rescue? *IEEE Security & Privacy*, 19(4):37–43, 2021.

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

[62] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[63] Lushan Song, Haoqi Wu, Wenqiang Ruan, and Weili Han. Sok: Training machine learning models over multiple sources with privacy preservation. *arXiv preprint arXiv:2012.03386*, 2020.

[64] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248. IEEE, 2013.

[65] Sijun Tan, Brian Knott, Yuan Tian, and David J. Wu. CRYPTGPU: Fast privacy-preserving machine learning on the gpu. In *IEEE S&P*, 2021.

[66] Florian Tramer and Dan Boneh. Differentially private learning needs better features (or much more data). In *International Conference on Learning Representations*, 2021.

[67] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 601–618, 2016.

[68] Paul Voigt and Axel Von dem Bussche. *The EU General Data Protection Regulation (GDPR)*. Springer, New York, USA, 2017.

[69] Sameer Wagh, Xi He, Ashwin Machanavajjhala, and Prateek Mittal. Dp-cryptography: Marrying differential privacy and cryptography in emerging applications. *Commun. ACM*, 64(2):84–93, January 2021.

[70] Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. Falcon: Honest-majority maliciously secure framework for private deep learning. *Proc. Priv. Enhancing Technol.*, 2021(1):188–208, 2021.

[71] Xi Wu, Fengan Li, Arun Kumar, Kamalika Chaudhuri, Somesh Jha, and Jeffrey Naughton. Bolt-on differential privacy for scalable stochastic gradient descent-based analytics. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1307–1322. ACM, 2017.

[72] Aiping Xiong, Tianhao Wang, Ninghui Li, and Somesh Jha. Towards effective differential privacy communication for users' data sharing decision and comprehension. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 392–410. IEEE, 2020.

[73] Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A Gunter, and Bo Li. Detecting ai trojans using meta neural analysis. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 103–120. IEEE, 2021.

[74] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[75] Ziqi Yang, Bin Shao, Bohan Xuan, Ee-Chien Chang, and Fan Zhang. Defending model inversion and membership inference attacks via prediction purification. *arXiv preprint arXiv:2005.03915*, 2020.

[76] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 160–164, 1982.

[77] Xun Yi, Russell Paulet, and Elisa Bertino. Homomorphic encryption. In *Homomorphic Encryption and Applications*, pages 27–46. Springer, 2014.

[78] Yang You, Zhao Zhang, Cho-Jui Hsieh, James Demmel, and Kurt Keutzer. Imagenet training in minutes. In *Proceedings of the 47th International*

Conference on Parallel Processing, ICPP 2018, New York, NY, USA, 2018. Association for Computing Machinery.

[79] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Gosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. Opacus: User-friendly differential privacy library in pytorch. *CoRR*, abs/2109.12298, 2021.

[80] Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. Differentially private fine-tuning of language models. In *International Conference on Learning Representations*, 2022.

[81] Da Yu, Huishuai Zhang, Wei Chen, and Tie-Yan Liu. Do not let privacy overbill utility: Gradient embedding perturbation for private learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[82] Xinyang Zhang, Zheng Zhang, Shouling Ji, and Ting Wang. Trojaning language models for fun and profit. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 179–197, Los Alamitos, CA, USA, sep 2021. IEEE Computer Society.

APPENDIX A
MISSING PROOFS

Lemma 1. For n mechanisms $M_1, M_2, \cdots, M_n$ whose inputs are disjoint datasets $D_1, D_2, \cdots, D_n$, if M_i satisfies (ϵ_i, δ_i)-differential privacy, the combination of them satisfies (ϵ, δ)-differential privacy for the dataset $D = D_1 \cup D_2 \cup \cdots \cup D_n$ with $\epsilon = Max(\epsilon_1, \epsilon_2, \cdots, \epsilon_n)$, $\delta = Max(\delta_1, \delta_2, \cdots, \delta_n)$.

Proof. We first redefine (ϵ, δ)-differential privacy in a new form based on Definition 2 [48].

Definition 2. *For two datasets D and D', we say a random mechanism M that outputs $r \in R^p$ satisfies (ϵ, δ)-differential privacy if, for any subset $S \subseteq R^p$, $Pr(M(D) \in S) \leq e^{\epsilon * |D \oplus D'|} \cdot Pr(M(D') \in S) + \delta * |D \oplus D'|$*

where $D \oplus D'$ is the symmetric difference between D and D'. Then for $D_1, \cdots, D_n$ and $D'_1, \cdots, D'_n$, let M be the combination of $M_1 \cdots, M_n$, the probability that M outputs r is $Pr(M(D) = r) = \prod_{i=1}^{n} Pr(M_i(D_i) = r_i)$. Meanwhile, as D and D' are neighboring datasets, there is one $|D_i \oplus D'_i|$ larger than 0 and is equal to $|D \oplus D'|$. Without loss of generality, we assume $|D_k \oplus D'_k| > 0$. Therefore,

$$\prod_{i=1}^{n} Pr(M_i(D_i) = r_i)$$

$$\leq \prod_{i=1}^{n} (e^{\epsilon_i * |D_i \oplus D'_i|} Pr(M_i(D'_i) = r_i) + \delta_i * |D_i \oplus D'_i|)$$

$$= (\prod_{i=1, i \neq k}^{n} Pr(M_i(D'_i) = r_i)) * (e^{\epsilon_k * |D_k \oplus D'_k|} Pr(M_k(D'_k) = r_k)$$
$$+ \delta_k * |D_k \oplus D'_k|)$$

$$\leq e^{Max(\epsilon_1, \cdots, \epsilon_n) * |D_k \oplus D'_k|} \prod_{i=1}^{n} Pr(M_i(D'_i) = r_i)$$

$$+ (\prod_{i=1, i \neq k}^{n} Pr(M_i(D'_i) = r_i)) * Max(\delta_1, \cdots, \delta_n) * |D_k \oplus D'_k|$$

$$\leq e^{Max(\epsilon_1, \cdots, \epsilon_n) * |D \oplus D'|} \prod_{i=1}^{n} Pr(M_i(D'_i) = r_i)$$

$$+ Max(\delta_1, \cdots, \delta_n) * |D \oplus D'|$$

It completes the proof. □

Lemma 2. For any $x' \in [0.5, 1), (0.8277x'^2 - 2.046x' + 2.223 - 0.0048) - \frac{1}{\sqrt{x'}} < 0$.

Proof. Let $f(x') = (0.8277x'^2 - 2.046x' + 2.223 - 0.0048) - \frac{1}{\sqrt{x'}}$. As function f is differentiable on interval $[0.5, 1)$, we can get its first-order derivative function as $f'(x') = 1.6554x' - 2.046 + \frac{1}{2\sqrt{x'^3}}$ and second-order derivative function as $f''(x') = 1.6554 - \frac{3}{4\sqrt{x'^5}}$. Because $f''(x')$ is monotonically increasing on $[0.5, 1)$ and has one zero point $x'_0 = 0.7286$, $f'(x')$ is monotonically decreasing on $[0.5, 0.7286)$ and monotonically increasing on $[0.7286, 1)$. Then as $f'(0.7286) = -0.0359 < 0$ and $f'(x')$ is larger than 0 on two endpoints 0.5 and 1, $f'(x')$ has two zero points on $[0.5, 1)$: $x'_1 = 0.6257$ and $x'_2 = 0.8517$. Therefore, $f(x')$ is monotonically increasing on $[0.5, 0.6257)$, monotonically decreasing on $[0.6257, 0.8517)$, and monotonically increasing on $[0.8517, 1)$. Finally, for any $x' \in [0.5, 1)$, $f(x') < \max\{f(0.6257), f(1)\} = -0.0001 < 0$. □

Theorem 3. For any $\epsilon \leq 2\log(1/\delta)$ and $\delta \in (0, 1)$, Protocol 3 satisfies (ϵ, δ)-differential privacy.

Proof. We first introduce a differential privacy concept, namely Rényi differential privacy (RDP) [50], to tightly analyze the privacy budget of DPSGD in a simple way.

Definition 3. *(λ, γ)-Rényi Differential Privacy [50]. For a random mechanism M whose input is D and output $r \in R^p$, we say M satisfies (λ, γ)-RDP if, for any two neighboring datasets D, D', it holds that,*

$$D_\lambda(M(D) \| M(D')) \leq \gamma$$

where $D_\lambda(M(D) \| M(D'))$ is λ-Rényi divergence between the distributions of $M(D)$ and $M(D')$. Then we show the proof of Theorem 3.

According to Lemma 2, the maximum difference between the approximated $\frac{1}{\sqrt{x'}}$ and the true $\frac{1}{\sqrt{x'}}$ is smaller than 0. Thus Protocol 1 can ensure that for each iteration, the L_2 sensitivity of Protocol 3 is smaller than C. Then each iteration of Protocol 3 satisfies $(\lambda, \frac{\lambda C^2}{2\sigma^2})$-RDP according to Corollary 3 of [50]. Next, according to Proposition 1 of [50], Protocol 3 satisfies $(\lambda, T\frac{\lambda C^2}{2\sigma^2})$-RDP. Finally, according to Proposition 3 of [50], in order to ensure (ϵ, δ)-differential privacy, we should ensure that

$$\frac{TC^2\lambda}{2\sigma^2} + \frac{\log(\frac{1}{\delta})}{\lambda - 1} \leq \epsilon$$

We set $\lambda = 1 + \frac{2\log(\frac{1}{\delta})}{\epsilon}$, thus we should ensure that

$$\sigma^2 \geq \frac{TC^2(\epsilon + 2\log(\frac{1}{\delta}))}{\epsilon^2}$$

Meanwhile, as we set in Protocol 3

$$\sigma^2 \geq \frac{TC^2(4\log(\frac{1}{\delta}))}{\epsilon^2} \geq \frac{TC^2(\epsilon + 2\log(\frac{1}{\delta}))}{\epsilon^2}$$

Therefore, Protocol 3 satisfies (ϵ, δ)-differential privacy. □

Theorem 4. Protocol 4 satisfies (ϵ_1, δ_1)-differential privacy.

Proof. Each local dataset D_i held by P_i can be viewed as a subset of the logically global dataset. Meanwhile, as different parties hold different data, their local datasets are disjoint.

TABLE VII: Hyperparameter settings of local model training.

Datasets	ϵ	δ	Clip bound	#Epoch	Batch size	LR
MNIST	0.25	2×10^{-7}	1	3	128	0.1
CIFAR-10	0.25	1.7×10^{-7}	1	1	128	0.1
IMDb	0.25	4×10^{-7}	1	3	128	0.1

TABLE VIII: Potential hyperparameters involved in the grid search procedure of baselines training.

Hyperparameter	Potential values
Clipping bound	1, 2, 5
Batch size	64, 128, 256, 512
Learning rate schedule	Constant (lr=0.1), OneCycleLR (max_lr=0.1)

In addition, the aggregation methods are data-independent. According to the post-process immunity property of differential privacy, the aggregation has no impact on the privacy guarantee. Therefore, we only need to consider the privacy guarantee of local model training. As all parties privately train their models with privacy parameters (ϵ_1, δ_1), we can obtain the privacy guarantee of Protocol 4 by directly applying Lemma 1. $\square$

APPENDIX B
EXPERIMENT SETTINGS

In this section, we show the hyperparameter settings of experiments. The hyperparamters of local model training are shown in Table VII. The potential hyperparameters involved in the grid search procedure are shown in Table VIII. We also show the parameters found by the grid search procedure in Table IX. Note that we run the baseline training process of MNIST, CIFAR-10, and IMDb for 20, 80, 20 epochs, respectively.

APPENDIX C
SECURITY AND COMMUNICATION COMPLEXITY ANALYSIS OF PROTOCOLS

A. Analysis of Protocol 1

Security. As Protocol 1 only contains the cryptographic primitives whose security has been proven in previous studies [9], [10] and the rest of computation is performed locally. Therefore, as long as the security of these cryptographic primitives holds, Protocol 1 is secure.

Communication Complexity. In Protocol 1, the computation of $2^{f - \frac{exp}{2}}$ and the evaluation of approximated polynomial can be parallelized. Therefore, in the critical path of Protocol 1, there are four Multiplication and six Trunc invocations, two Bit_Dec invocations, one Mod2, SufOr and PreMulC invocation, respectively. According to Table 2 and Table 3 of [9], Table 4 of [10], the total communication round number and the communication complexity of Protocol 1 is 15, $O(mk^2)$ bits respectively. Meanwhile, the offline communication round number and the communication complexity of Protocol 1 is 1,

TABLE IX: Hyperparameter settings of baseline model training.

Datasets	Clipping bound	Batch size	LR
MNIST	1	128	Constant (lr=0.1)
CIFAR-10	1	512	OneCycleLR (max_lr=0.1)
IMDb	2	64	OneCycleLR (max_lr=0.1)

$O(mk^2 + mfk)$ bits respectively. Note that the communication round is 1 because the offline computations have no mutual dependence and can all be executed in parallel.

B. Analysis of Protocol 3

Security. In addition to the invocations of Protocol 1 and Protocol 2, the other secure computation primitives used in Protocol 3 (i.e. comparison, multiplication and subtraction) all have the standard protocols [3], [9], [25], which have been proven secure by previous studies [3], [9], [25]. Meanwhile, because the security of Protocol 1 and Protocol 2 has been stated in Section IV-A and Section IV-B, as long as our security model holds, Protocol 3 is secure.

Communication Complexity. In the online phase, for each batch with batch size B, in addition to invoke two Trunc, four Multiplication, one Protocol 1 and one Comparison B times in parallel, Protocol 3 contains one extra Trunc to update the model parameters. Therefore, according to Table 2 of [9], when we execute Multiplication and Trunc in parallel, the communication round number and communication complexity of Protocol 3 is 22 and $O(mBk^2 + mpBk)$ bits. Note that we consider a linear model here. For complex models with multiple layers, the communication round number keeps unchanged because gradient clipping and perturbation can be performed in parallel, while the communication complexity linearly increases as the number of parameters increases. As to the offline communication complexity, according to Table 2 of [9], the offline communication round number and offline communication complexity of Protocol 3 is 1 and $O(mBk^2 + mpfBk)$ bits.

C. Analysis of Protocol 4

Security. When parties train their local models, they only perform computations on their local data. As to the aggregation phase, the model aggregation and accuracy evaluation are completed with cryptographic primitives introduced in Section III-B. The only information revealed to parties is the accuracy of candidate models. As these candidate models are protected by DP and testing data are not visible to parties, the accuracy does not leak private information. Therefore, as long as the security of these cryptographic primitives holds, Protocol 4 is secure.

Communication Complexity. In order to analyze the communication complexity of Protocol 4, we first introduce two aggregation methods of Section V-B. We show the details of the averaging strategy and the accuracy strategy in Protocol 6 and Protocol 7 respectively. We then analyze the communication complexity of each aggregation method. We first analyze

---

Protocol 6 Averaging Strategy

Input: P_i holds shares of local models $\langle \theta_1^l \rangle_i, \langle \theta_2^l \rangle_i, \cdots, \langle \theta_m^l \rangle_i$.
Output: P_i obtains the share of the average of local models $\langle \theta_a \rangle_i$;
  1: P_i compute $\langle \theta_a \rangle_i = \sum_{j=1}^{m} \langle \theta_j^l \rangle_i / m$;
  2: P_i obtains $\langle \theta_a \rangle_i$ as the share of the average of local models;

1942

Protocol 7 Accuracy Strategy

Input: P_i holds shares of local models $\langle\theta_1^l\rangle_i, \langle\theta_2^l\rangle_i, \cdots, \langle\theta_m^l\rangle_i$:
Output: P_i obtains the share of the most accurate local model $\langle\theta_m\rangle_i$;
1: **for** Each $\theta_i^l \in \{\theta_1^l, \theta_2^l, \cdots, \theta_m^l\}$ **do**
2:     All parties collaboratively evaluate the accuracy of θ_i^l as $Accuracy_i$;
3:     **if** $Accuracy_i \geq Accuracy_m$ **then**
4:         $\langle\theta_m\rangle = \langle\theta_i^l\rangle$;
5:         $Accuracy_m = Accuracy_i$;
6:     **end if**
7: **end for**
8: P_i obtains $\langle\theta_m\rangle_i$ as the share of the most accurate local model;

TABLE X: Accuracy (in %) comparison of models securely trained on the logically global dataset and models locally trained on one subset of the whole dataset for MNIST. We report the average results of five runs and show the standard deviations in brackets.

#Party	ϵ	Local party	TF-Encrypted$_\epsilon$	Queqiao$_\epsilon$
3-party	0.25	87.23 ($\pm$0.34)	89.36 ($\pm$0.25)	89.50 ($\pm$0.22)
	1	93.56 ($\pm$0.22)	95.03 ($\pm$0.11)	94.04 ($\pm$0.21)
	2	94.25 ($\pm$0.15)	95.14 ($\pm$0.09)	94.90 ($\pm$0.14)
5-party	0.25	86.92 ($\pm$0.39)	-	88.18 ($\pm$0.21)
	1	92.33 ($\pm$0.29)	-	93.99 ($\pm$0.09)
	2	93.92 ($\pm$0.24)	-	95.13 ($\pm$0.19)
10-party	0.25	79.23 ($\pm$2.23)	-	85.60 ($\pm$0.58)
	1	89.25 ($\pm$0.37)	-	94.32 ($\pm$0.31)
	2	92.96 ($\pm$0.21)	-	95.35 ($\pm$0.15)

the communication complexity of Protocol 6 and Protocol 7. Protocol 6 only includes a constant division computation and performs one Trunc computation. Hence, Protocol 6 requires 2 communication rounds and $O(mpk)$ bits communication. For Protocol 7, the main overhead is from the accuracy evaluation of each local model. Assuming the size of the testing dataset is n_t, the communication round complexity of Protocol 7 is $O(m)$ and the communication complexity is $O(m^2 n_t p k^2)$, where the constant depends on the complexity of the trained model.

We then analyze the communication complexity of Protocol 4 introduced in Section V-B. Besides the invocations of Protocol 6 and Protocol 7, Protocol 4 evaluates the accuracy of two candidate models. Thus its communication round complexity is $O(m)$ and communication complexity is $O(m^2 n_t p k^2)$. We do not analyze the offline communication complexity of the above three protocols because they do not have offline phase.

TABLE XI: Accuracy (in %) comparison of models securely trained on the logically global dataset and models locally trained on one subset of the whole dataset for IMDb. We report the average results of five runs and show the standard deviations in brackets.

#Party	ϵ	Local party	TF-Encrypted$_\epsilon$	Queqiao$_\epsilon$
3-party	0.25	79.48 ($\pm$0.72)	80.55 ($\pm$0.49)	80.20 ($\pm$0.51)
	1	83.84 ($\pm$0.32)	84.46 ($\pm$0.67)	84.49 ($\pm$0.28)
	2	84.08 ($\pm$0.19)	84.70 ($\pm$0.44)	84.35 ($\pm$0.17)
5-party	0.25	77.2 ($\pm$1.01)	-	77.46 ($\pm$0.82)
	1	80.33 ($\pm$0.26)	-	84.01 ($\pm$0.15)
	2	80.69 ($\pm$0.21)	-	84.59 ($\pm$0.37)
10-party	0.25	72.00 ($\pm$1.81)	-	75.48 ($\pm$1.03)
	1	77.84 ($\pm$1.30)	-	84.64 ($\pm$0.13)
	2	78.22 ($\pm$0.69)	-	84.56 ($\pm$0.31)

APPENDIX D
SUPPLEMENTAL EXPERIMENTAL RESULTS OF SECTION VI-G

In this section, we show the accuracy comparison results of global models and local models for MNIST and IMDb. The results are shown in Table X and Table XI. Like the results of CIFAR-10, the accuracy of the global models is higher than local models in all scenarios. Meanwhile, the accuracy gaps between global models and local models also become larger as the number of parties increases. These results further verify the effectiveness of training models on the logically global datasets that are composed of local datasets through MPL.

APPENDIX E
COMPARISON OF FEATURE EXTRACTION METHODS

In this section, we keep other experimental settings unchanged and compare our feature extraction method with the method proposed by Tramèr and Boneh [66]. We run their source codes[10] to extract the features of MNIST and CIFAR-10 datasets with methods described in their paper.

We show the experimental results in Table XII. Compared with differentially private models trained on the features extracted by Tramèr and Boneh method, the models trained on the features extracted by our method can achieve higher accuracy. Because the features extracted by Tramèr and Boneh method are high-dimension (3969 vs. 1800 (PEA) in MNIST, 4096 vs. 2048 (PEA) in CIFAR-10) and sparse, the information of these features that are represented as fixed-point number with limited precision might be erased when we perform truncation and gradient clipping with SMPC protocols. Specifically, the low accuracy of models trained by TF-Encrypted$_\epsilon$ is from the truncation error of ABY3 protocol [51], i.e. it might truncate a fixed-point number with a very large error. When the dimension of features becomes higher, the truncation error occurs with higher probability, which is also shown in one previous study [29].

TABLE XII: Accuracy (in %) comparison of the models trained on the features extracted by our method (PEA) and the models trained on features extracted by Tramèr and Boneh method. We report the average results of five runs and show the standard deviations in brackets.

	PEA		Tramèr and Boneh [66]	
	MNIST	CIFAR-10	MNIST	CIFAR-10
TF-Encrypted$_\epsilon$	95.14 ($\pm$0.09)	88.99 ($\pm$0.08)	9.80 ($\pm$0.00)	8.35 ($\pm$2.06)
Queqiao$_\epsilon$	94.90 ($\pm$0.14)	88.18 ($\pm$0.33)	92.81 ($\pm$0.06)	86.04 ($\pm$0.15)

[10]https://github.com/ftramer/Handcrafted-DP