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

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

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

$\{$wruan20, 20212010078, whan$^*$$\}$@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: $^{\sharp}{\text{TF-Encrypt}}$ 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 CryptGPUs, 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 square root protocol, we enforce the DPSGD algorithm [1], [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 [18]. We refer to the differentially private versions of TF-Encrypted and Queqiao as TF-Encrypted, and Queqiao, 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 $\epsilon$ as 2, TF-Encrypted, and Queqiao, 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, and Queqiao, 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 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 |
| $|z|$ | One share of a secret $z$ 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 |
| $\theta$ | Trained model |
| $\gamma$ | The gradient clipping bound |
| $B$   | 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 $\tilde{x}$ to an integer $\pi$ such that $\pi = \tilde{x} \times 2^f$. In order to prevent the precision explosion brought by fixed-point number multiplication (i.e. $\pi = \pi \times \tilde{y} = \tilde{x} \times 2^f \tilde{y}$), we apply Trunc protocol [9] to truncate $\pi$ as $\pi'$ such that $\pi' = \tilde{x} \times \tilde{y} \times 2^f$. 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 $\langle x \rangle_1, \langle x \rangle_2, \ldots, \langle x \rangle_{m-1}$, 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 $f(y) = a_0 + a_1 y + a_2 y^2 + \ldots + a_l y^l$, where $a_0 = x, a_1, \ldots, a_l$ are random coefficients, and $l+1$ is the least number of parties to reconstruct the secret; then we construct $m$ points out of it with indexes $1 \leq i \leq m$. These $f(i)$ (i.e. $\langle x_i \rangle$) 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 $\langle x_0 \rangle_i, \langle x_1 \rangle_i, \ldots, \langle x_{k-1} \rangle_i$ such that $x = \sum_{i=0}^{k-1} x_i \times 2^i$.
- 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 $z = x \times y$.
- Comparison outputs $\langle z \rangle_i$ for $P_i$ such that $z = x > y, 0 : 1$.

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

C. Differentially Private Machine Learning

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

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

where $D$ and $D'$ are neighboring datasets, i.e. we can obtain $D'$ by deleting or adding one data sample of $D$. If $\delta$ is set as 0, we say $M$ is $\epsilon$-differentially private. $\epsilon$ 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 $M_1, M_2, \ldots, M_n$ that access the same dataset
DPSGD [1], [2], [64]

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

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

Lemma 1. Parallel Composition Theorem for $(\epsilon, \delta)$-Differential Privacy. For n mechanisms $M_1, M_2, \ldots, M_n$ whose inputs are disjoint datasets $D_1, D_2, \ldots, D_n$, if $M_i$ satisfies $(\epsilon_i, \delta_i)$-DP, the combination of them satisfies $(\epsilon, \delta)$-DP for the dataset $D = D_1 \cup D_2 \cup \cdots \cup D_n$ with $\epsilon = \max(\epsilon_1, \epsilon_2, \ldots, \epsilon_n)$.

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 $D = \{d_1, \ldots, d_n\}$, loss function: $\ell(\theta; d_i)$, privacy parameters: $(\epsilon, \delta)$, number of iterations: $T$, batch size: $B$, gradient clipping bound $C$, learning rate function: $\eta$.

Output: Trained model $\theta_T$:
1: Randomly initialize model $\theta_0$;
2: for $t = 1$ to $T$ do
3: $s_1, \ldots, s_B \leftarrow$ Sample $B$ samples uniformly from $D$;
4: for each $s_i$, compute $\nabla \ell(\theta; s_i)$;
5: Clip gradient
6: $\nabla \ell(\theta; s_i) \leftarrow \nabla \ell(\theta; s_i) \cdot \min(1, C/\|\nabla \ell(\theta; s_i)\|_2)$
7: Add noise and Denoise
8: $b_t \sim N(0, \sigma^2 I_{k \times k})$, where $\sigma = C/\sqrt{2T \log \frac{2}{\delta}}$;
9: $\theta_{t+1} = \theta_t - \eta \frac{B}{T} \sum_{i=1}^{B} \nabla \ell(\theta; s_i) + b_t$;
end for
11: Output $\theta_T$.

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 $g$, parties need to compute the inverse of $\|g\|_2$, i.e. $\sqrt{\frac{1}{g_1^2 + g_2^2 + \cdots + g_d^2}}$. 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 $(x)_i$ to a form $(x' + 2^{\exp_i})_i$, where $0.25 \leq x' \leq 0.5$. Afterwards, they separately compute the $(\frac{1}{\sqrt{x'}})_i$ and $(\frac{1}{\sqrt{1-x'}})_i$. Finally, they obtain $(\frac{1}{\sqrt{y}})_i$ by multiplying $(\frac{1}{\sqrt{x'}})_i$ and $(\frac{1}{\sqrt{1-x'}})_i$.

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 $\|\nabla \ell(\theta; s_i)\|_2 > C$ and approximated $\|\nabla \ell(\theta; s_i)\|_2^{\text{approx}} < C$, 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 $(x)_i$;
Output: $P_i$ obtains the share of the output $(y)_i$ such that $y = \frac{1}{\sqrt{x}}$;
1: Data preparation and transformation
2: $(\{x_i\})_{i=1}^{k-1} \leftarrow \text{Bit-Dec}((x)_i)$;
3: $(\{c_{i1}\})_{i=1}^{k-2} \leftarrow \text{SufOr}(((x_{i1}))_{i=1}^{k-2})$;
4: $(\{b_{i}\}) = 1 + \sum_{i=1}^{k-2} 2^{k-2-(1-(c_{i1}))}$;
5: $(\{x_i\}) = \text{Trunc}(b_{i}, x_{i}, k, k-f-1)$;
6: $(\{exp\}) \leftarrow \sum_{i=0}^{k-2} (c_{i1})_{i} = f$;
7: Compute the square root of $2^{-exp}$;
8: $(\{lsb\}) = \text{Mod2}((\{exp\})_{i})$
9: $(\frac{2^{lsb}}{2}) = \text{Trunc}((\{exp\})_{i}, k, 1)$;
10: $(\{c_{i1}\})_{i=1}^{k-1} = \text{Bit-Dec}(\{f - 2^{lsb}\} _{i})$;
11: for $t = 0$ to $k - 1$ do
12: $(\{c_{i1}\})_{i=1}^{k-1} = (1 + (c_{i1})_{i} \cdot (2^{2t} - 1))$;
end for
14: $(\{c_{i1}\})_{i=1}^{k-1} = \text{PreMulC}((\{c_{i1}\})_{i=1}^{k-1})$;
15: $(\frac{2^{lsb}}{2}) = (c_{k-1})_{i};$
16: $(\frac{2^{lsb}}{2}) = \text{Trunc}((\frac{2^{lsb}}{2}) _{i} + 2^{\frac{t}{2}} + 2^{\frac{l_{sb}}{2}}) _{i}, f, k)$;
17: $(\frac{2^{lsb}}{2}) = \left(2^{\frac{lsb}{2}} - \frac{1}{2} \right) + 2^{\frac{lsb}{2}}$;
18: Compute approximated polynomial
19: $(x_{approx})_{i} = \text{Trunc}(0.8277 \cdot 2^{l_{sb}} \cdot x_{i})$
20: $(x_{approx})_{i} = \text{Trunc}(((x_{approx})_{i} - 2.046 \cdot 2^{l_{sb}}) \cdot x_{i}), k, f)$;
21: $(x_{approx})_{i} = (x_{approx})_{i} + 2.223 \cdot 2^{l_{sb}} - 0.0048 \cdot 2^{l_{sb}}$
22: Compute and output final result
23: $(y_i) = \text{Trunc}((x_{approx})_{i} + 2^{\frac{lsb}{2}} _{i}, k, f)$;
24: $P_i$ obtains $(y_i)$.

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 shown in Protocol 1, we describe each step of the inverse of square root protocol as follows:

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
We further analyze the security and communication complexity when part of parties fails during the noise generating process, Algorithm 1, we can guarantee that gradient vectors whose $\|\nabla \theta \|$ with polynomial $2^{\exp}$, 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 $2^{\exp}$, i.e., $2^{\exp}$, respectively, where $2^{\exp} = \frac{exp}{2}$ if $exp$ is odd. Finally, $P_i$ obtain $2^{\exp}$ by choosing the correct value based on the value of $(\text{lsb})_i$.

**Step 3.** (Line 19-21) $P_i$ approximates $\sqrt{x'}$ 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 $\sqrt{x'}$ of $x'$ to obtain the output $(\sqrt{x})_i$. 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 $\frac{1}{\sqrt{x'}}$ is smaller than the true $\frac{1}{\sqrt{x'}}$. Consequently, according to Algorithm 1, we can guarantee that gradient vectors whose $L_2$ norms are larger than $C$ (i.e., $\|\nabla \theta \|_2 > C$) 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 $\sigma$. 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 $\frac{1}{2\pi} \sigma^2 I_{pxp}$ 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 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 $\sigma$, the dimension of noise $n$.

**Output:** The shares of a Gaussian noise $\langle b \rangle_i$ such that $b \sim N(0, \sigma^2 I_{pxp})$.

1: $P_i$ samples a Gaussian noise $b_i \sim N(0, \frac{1}{2\pi} \sigma^2 I_{pxp})$;
2: $P_i$ sends the shares of $(b_i)_{i \neq i}$ to other parties $P_j$, $j \neq i$;
3: $P_i$ summarizes the noise shares from other parties to obtain the result $(b_i)_{i \neq i}$.

**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 $\sigma$ through calling Protocol 2 and apply them to perturb the gradient vectors. Finally, parties update model parameters $\theta$ 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 $(\epsilon, \delta)$-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 resampling-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: $(\epsilon, \delta)$, number of iterations: $T$, batch size: $B$, gradient clipping bound $C$, learning rate function: $\eta$.

Output: $P_i$ obtains the shares of the trained model $\langle \theta_T \rangle$.

1: $P_i$ generates and sends the shares of its own dataset $\langle D_i \rangle_{j \neq i}$ to $P_j$s. Thus, $P_i$ holds the share of the global dataset $\langle D_i \rangle = \{(D_i)_1, (D_i)_2, \ldots, (D_i)_m \}$.

2: Initializing $(\theta_0)$, with Protocol 4.

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

4: $\langle s_1 \rangle, \ldots, \langle s_B \rangle \leftarrow$ parties collaboratively sample $B$ samples from $D$.

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

6: Clip gradient

7: for each $\langle \nabla \ell(\theta; s_j) \rangle_i$ do

8: $\langle x_j \rangle_i = \text{Trunc}(\langle \nabla \ell(\theta; s_j) \rangle_i) * \langle \nabla \ell(\theta; s_j) \rangle_i$, $k, f$;

9: Compute the inverse square root of $\langle x_j \rangle_i$ (i.e., $\langle \frac{1}{\| \nabla \ell(\theta; s_j) \|_2^2} \rangle_i$) with Protocol 1;

10: $\langle is\_clip \rangle_i = \text{Comparison}(\langle \frac{C}{\| \nabla \ell(\theta; s_j) \|_2^2} \rangle_i^2, 1)$;

11: $\langle \nabla \ell(\theta; s_j) \rangle_i^{\text{clipped}} = \text{Trunc}(\langle \nabla \ell(\theta; s_j) \rangle_i) * (1 - \langle is\_clip \rangle_i) + \langle \nabla \ell(\theta; s_j) \rangle_i^{\text{clipped}} * \langle is\_clip \rangle_i$;

12: end for

13: Add noise and Descent

14: Generate noise $\langle b_i \rangle$ such that $b_i \sim N(0, \sigma^2 I_{p \times p})$ with Protocol 2, where $\sigma \geq \sqrt{\frac{C}{\eta \log(2)}}$

16: $\langle update \rangle_i = \text{Trunc}(\frac{\beta}{B} \sum_{j=1}^{B} \langle \nabla \ell(\theta; s_j) \rangle_i + \langle b_i \rangle_i, k, f)$;

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

18: end for

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

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.
**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 \( D = \{ D_1, D_2, \ldots, D_m \} \) (the proof of Theorem 4 is shown in Appendix A).

**Theorem 4.** Protocol 4 satisfies \((\epsilon_1, \delta_1)\)-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 \( \epsilon_1 \) and \( \delta_1 \), clipping bound \( C_1 \), and aggregation method set \( A = \{ A_1, A_2 \} \);

**Output:** \( P_i \) obtains the share of initial global model \( \langle \theta_i \rangle \);  
1: \( P_i \) trains a local model \( \theta_i^\prime \) with Protocol 1 and \( \epsilon = \epsilon_1, \delta = \delta_1, C = C_1 \);
2: \( P_i \) sends \( \langle \theta_i^\prime \rangle \) to \( P_j, P_k \);
3: Parties randomly initialize a candidate model \( \theta_c \) and securely evaluate the accuracy of \( \theta_c \) as \( \text{Accuracy}_{\epsilon_2, \delta_2} \);
4: for Each \( A_k \in A \) do
5: \( P_i \) obtains the share of \( k \)th candidate model \( \langle \theta_{i,k} \rangle = A_k(\langle \theta_i^\prime \rangle, \langle \theta_1 \rangle, \ldots, \langle \theta_m \rangle) \);
6: All parties collaboratively evaluate the accuracy of \( \theta_{i,k} \) as \( \text{Accuracy}_{\epsilon_2, \delta_2} \);
7: if \( \text{Accuracy}_{\epsilon_2, \delta_2} \geq \text{Accuracy}_{\epsilon_1, \delta_1} \) then
8: \( \langle \theta_i \rangle = \langle \theta_{i,k} \rangle \);
9: \( \text{Accuracy} = \text{Accuracy}_{\epsilon_2, \delta_2} \);
10: end if
11: end for
12: \( P_i \) obtains \( \langle \theta_i \rangle \) 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 \( \epsilon_1, \epsilon_2 \) and \( \delta_1, \delta_2 \), clipping bound \( C_1, C_2 \);

**Output:** \( P_i \) obtains the share of trained global model \( \langle \theta_i \rangle \);

1: \( P_i \) extracts features of the local dataset \( D_i \) as \( F \) with data-independent feature extractors;
2: All parties apply Protocol 4 to initialize the global model with privacy parameter \( (\epsilon_1, \delta_1) \) 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 \( (\epsilon_2, \delta_2) \) and clipping bound \( C_3 \);

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 learning.
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] 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, 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.

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. Finally, we set the field size as $10^{17} + 3$ and $f$ as 20 in Queqiao.

**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 $\times$ 28 gray pixels.
- **CIFAR-10** [40] contains 60,000 32 $\times$ 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 towards movies.

**Dataset Partition.** We partition each dataset into local datasets by 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, and Queqiao, 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. 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 $\delta$ as $\frac{1}{n}$, where $n$ is the size of the training dataset. For the privacy budget $\epsilon$, we set it as 2 and run all global model training processes for two epochs. We use the privacy budget accountant tools 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 $\delta$ and the noise multiplier of Gaussian distribution as the same value as ours during the baseline training. Then we tune $\epsilon$ 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 hyperparameters in Table VII of Appendix B. For the global model training, we set $\delta_2$ as $\frac{9}{100n}$ and tune the total privacy budget $\epsilon$ 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, and Queqiao, 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, and Queqiao, while CryptGPU

---

1https://github.com/tf-encrypted/tf-encrypted
2https://github.com/FudanMPL/SecMML
3https://github.com/tensorflow/privacy/blob/7eea74a6a1cf15e2d2bd890722400edd0c470db8/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    | 3    | 64   | 128  |
|---------|----------------------|------------|------|------|------|------|
| LAN     | TF-Encrypted,        | 28.22      | 28.75| 29.28| 29.93|      |
|         | Queqiao,             | 22.34      | 27.84| 28.31| 28.69|      |
|         | CrypTen              | 17.42      | 19.06| 21.14| 23.52|      |
| WAN     | TF-Encrypted,        | 841.17     | 843.5 | 848.0 | 848.3 |      |
|         | Queqiao,             | 843.8      | 856.19| 859.79| 861.79|      |
|         | CrypTen              | 5098.42    | 5632.96| 6041.90| 6067.64|      |

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, and Queqiao, 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, and Queqiao, 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, 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, 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, by combining multiple protocols. Thus it has more communication rounds than Queqiao.

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, and Queqiao, 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, and Queqiao, 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 tool7 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 $\epsilon$ as the sum of $\epsilon_1$ used in the local training phase and $\epsilon_2$ used in the global training phase, and $\delta$ as $\frac{1}{10}n$.

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

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

| Dataset | Data distribution | Averaging Strategy | Accuracy Strategy |
|---------|-------------------|--------------------|------------------|
| MNIST   | IID               | LR-TF-Encrypted    | 89.53 (±0.26)    |
|         | Non-IID           | LR-TF-Encrypted₂   | 88.45 (±0.40)    |
| CIFAR-10| IID               | LR-TF-Encrypted₂   | 82.55 (±0.39)    |
|         | Non-IID           | LR-TF-Encrypted₂   | 82.16 (±0.42)    |
| IMDb    | IID               | LR-TF-Encrypted₂   | 80.50 (±0.06)    |
|         | Non-IID           | LR-TF-Encrypted₂   | 80.53 (±0.01)    |

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 on extracted features of MNIST and CIFAR-10 datasets as baselines. We set privacy parameters of TF-Encrypted, Queqiao, and CAPC as $\epsilon = 2, \delta = \frac{1}{10^n}$. For DDGauss, we set $\epsilon$ as 5 and $\delta$ as $\frac{1}{10^n}$ as the training process of DDGauss cannot converge when we set $\epsilon$ 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.
Figure 4: Accuracy comparison of the models trained on extracted features and raw data. LR-TF-Encrypted_ε and LR-Queqiao_ε 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_ε and Queqiao_ε and models trained by CAPC and DDGauss. We report the average results of five runs and show the standard deviations in brackets.

| Dataset  | TF-Encrypted_ε | Queqiao_ε | CAPC [15] | DDGauss [36] |
|----------|----------------|-----------|-----------|---------------|
| MNIST    | 95.14 (±0.09)  | 94.90 (±0.14) | 92.32 (±0.20) | 92.43 (±0.38) |
| CIFAR-10 | 88.99 (±0.81)  | 88.18 (±0.33) | 86.70 (±0.17) | 53.31 (±22.92) |

Result. As is shown in Table IV, the models trained by TF-Encrypted_ε and Queqiao_ε 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_ε and Queqiao_ε. 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_ε and Queqiao_ε 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
TABLE V: Accuracy (in %) comparison of models securely 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, because its back-end SMPC protocol (i.e. ABY3) is tailored for the 3-party scenario.

| Party | Local party | TF-Encrypted | Queqiao | PlainText |
|-------|-------------|--------------|---------|-----------|
| 3     | 0.95        | 82.60±(0.30) | 84.03±(0.28) | 84.25±(0.31) |
| 1     | 86.43±(0.21) | 85.57±(0.11) | 87.79±(0.27) |         |
| 2     | 87.10±(0.14) | 86.99±(0.08) | 88.18±(0.14) |         |
| 5     | 0.25        | 79.84±(0.50) | -       | 82.09±(0.35) |
| 1     | 85.70±(0.37) | -       | 86.05±(0.25) |         |
| 2     | 88.07±(0.09) | -       | 88.34±(0.23) |         |
| 10    | 0.25        | 69.49±(1.14) | -       | 77.71±(0.29) |
| 1     | 81.52±(1.01) | -       | 88.05±(0.39) |         |
| 2     | 83.39±(0.62) | -       | 88.09±(0.14) |         |

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 $\epsilon$ as 2 when training the models.

| #Party | MNIST | CIFAR-10 | IMDb |
|-------|-------|----------|------|
| 3     | 0.95  | 85.14±(0.09) | 94.90±(0.14) | 95.35±(0.05) |
| 1     | 88.99±(0.08) | 88.18±(0.33) | 89.00±(0.13) |      |
| 2     | 84.70±(0.44) | 84.35±(1.17) | 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-Encrypted, and Queqiao, can train accurate differentially private classification models within much fewer iterations than differentially private deep neural network models. In particular, compared with CryptoGPU, TF-Encrypted, 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.

ACKNOWLEDGMENT

This paper is supported by NSFC (No. U1836207, 62172100) and STCSM (No. 21511101600). We thank Ninghui Li, Haoqi Wu, Guopeng Lin, Xinyu Tu, Shuyu Chen, and all reviewers for their insightful comments. Weili Han is the corresponding author.
[38] Brian Knott, Shobha Venkataraman, Awni Y. Hannun, Shubho Sengupta, Mark Ibrahim, and Laurents van der Maaten. CrypTen: Secure multi-party computation meets deep learning. In NeurIPS, 2019.

[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 feature detectors. In 🌁 Conference on Neural Information Processing Systems, pages 1097–1104, 2009.

[41] Sven Laur, Jan Willenson, and Bingzheng Zhang. Round-efficient oblivious database manipulation. In 🌁 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. 🌁 Conference on the Foundations of Cryptography, pages 277–346, 2017.

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

[46] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agueray A. Communication-efficient learning of deep networks from decentralized data. In 🌏 Conference on Artificial Intelligence and Statistics, pages 1273–1282. PMLR, 2017.

[47] 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.

[48] Ilya Mironov. On significance of the least significant bits for differential privacy. In 🌏 Conference of the 30th Annual Conference on the Theory of Data Processing (IEEE INFOCOM 2011), pages 263–275, 2011.

[49] Ilya Mironov. 🌏 Conference on the 25th Annual Conference of the IEEE Computer Society on Security and Privacy, pages 392–410. IEEE, 2020.

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

[51] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In 🌏 Conference of the 2017 IEEE Symposium on Security and Privacy, pages 392–410. IEEE, 2020.

[52] Valeria Nikolaeova, Udi Weinsberg, Stratis Ioannidis, Marc Jouy, Dan Boneh, and Nina Taft. Privacy-preserving ridge regression on hundreds of millions of records. In 🌏 Conference on Security and Privacy, pages 334–348, 2013.

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

[54] Nicolas Papernot, Martin Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semisupervised knowledge transfer for deep learning from private training data. In 🌏 Conference on Learning Representations, 2017.

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

[56] Martin Pettai and Peter Laufl. Combining differential privacy and secure multiparty computation. In 🌏 Conference of the 31st Annual Conference on Security Applications Conference, ACSAC 2015, page 421–430, New York, NY, USA, 2015. Association for Computing Machinery.

[57] Daniel Rothchild, Alex Tamkin, Julie Yu, Ujjal Misra, and Joseph Gonzalez. Cs5: Controllable generation of organic molecules with transformers. 🌏 Conference preprint arXiv:2108.10307, 2021.

[58] Amirya Roy Chowdhury, Chenghong Wang, Xi He, Ashwin Machanavajjhala, and Somesh Jha. Crypto: Crypto-assisted differential privacy on untrusted servers. In 🌏 Conference 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.

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

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

[61] Karen Simonian and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In 🌏 Conference of the 3rd International Conference on Learning Representations, 2015.

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

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

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

[65] Sameer Wagh, Shahriar Tole, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. Falcon: Honest-majority maliciously secure framework for private deep learning. 🌏 Conference of Proc. Priv. Enhancing Technol., 2021(1):188–208, 2021.

[66] 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 🌏 Conference of the 2017 ACM International Conference on Management of Data, pages 1307–1322. ACM, 2017.

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

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

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

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

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

[72] Xun Yi, Russell Paulet, and Elisa Bertino. Homomorphic encryption. 🌏 Conference of Homomorphic Encryption and Applications, pages 27–46. Springer, 2014.

[73] Yang You, Zhao Zhang, Cho-Jui Hsieh, James Demmel, and Kurt Keutzer. ImageNet training in minutes. In 🌏 Conference of the Proceedings of the 47th International Conference on Machine Learning, pages 1940–1949. PMLR, 2020.
we assume with \( \frac{\varepsilon}{\delta} \) larger than 0 and is equal to \( D \). The privacy is based on Definition 2 [48].

**APPENDIX A**

**MISSING PROOFS**

**Lemma 1.** For \( n \) mechanisms \( M_1, M_2, \ldots, M_n \) whose inputs are disjoint datasets \( D_1, D_2, \ldots, D_n \), if \( M_i \) satisfies \((\varepsilon_i, \delta_i)\)-differential privacy, the combination of them satisfies \((\varepsilon, \delta)\)-differential privacy for the dataset \( D = D_1 \cup D_2 \cup \cdots \cup D_n \) with \( \varepsilon = \max(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n), \delta = \max(\delta_1, \delta_2, \ldots, \delta_n) \).

**Proof.** We first redefine \((\varepsilon, \delta)\)-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 \( r \in \mathbb{R}^p \), \( Pr(M(D) \in S) \leq e^{\varepsilon \cdot |D \ominus D'|} \cdot Pr(M(D') \in S) + \delta \cdot |D \oplus D'| \)

where \( D \ominus D' \) is the symmetric difference between \( D \) and \( D' \). Then for \( M_1, \ldots, M_n \) and \( D_1', \ldots, D_n' \), let \( M \) be the combination of \( M_1, \ldots, M_n \), 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 \( |D_k \ominus D_k'| > 0 \). Therefore,

\[
\prod_{i=1}^{n} Pr(M_i(D_i) = r_i) \\
\leq \prod_{i=1}^{n} (e^{\varepsilon \cdot |D_i \ominus D_i'|} \cdot Pr(M_i(D_i') = r_i) + \delta_i \cdot |D_i \oplus D_i'|) \\
= \left( \prod_{i=1}^{n} Pr(M_i(D_i') = r_i) \right) \cdot (e^{\varepsilon \cdot |D \ominus D'|} \cdot Pr(M_k(D_k') = r_k) + \delta_k \cdot |D_k \oplus D_k'|) \\
\leq e^{\max(\varepsilon_1, \ldots, \varepsilon_n) \cdot |D_1 \ominus D_1'|} \prod_{i=1}^{n} Pr(M_i(D_i') = r_i) \\
+ \left( \prod_{i=1}^{n} Pr(M_i(D_i') = r_i) \right) \cdot \max(\delta_1, \ldots, \delta_n) \cdot |D_k \ominus D_k'| \\
\leq e^{\max(\varepsilon_1, \ldots, \varepsilon_n) \cdot |D \ominus D'|} \cdot \prod_{i=1}^{n} Pr(M_i(D_i') = r_i) \\
+ \max(\delta_1, \ldots, \delta_n) \cdot |D \ominus D'| \\
\]

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 \((\varepsilon, \delta)\)-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.** \((\lambda, \gamma)\)-Rényi Differential Privacy [50]. For a random mechanism \( M \) whose input is \( D \) and output \( r \in \mathbb{R}^p \), we say \( M \) satisfies \((\lambda, \gamma)\)-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 \( \lambda \)-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 \( \sqrt{\frac{\lambda}{C}} \) and the true \( \sqrt{\frac{\lambda}{C}} \) 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 \((\lambda, \sqrt{\frac{\lambda}{C}})\)-RDP according to Corollary 3 of [50]. Next, according to Proposition 1 of [50], Protocol 3 satisfies \((\lambda, T \sqrt{\frac{\lambda}{C^2}})\)-RDP. Finally, according to Proposition 3 of [50], in order to ensure \((\varepsilon, \delta)\)-differential privacy, we should ensure that

\[ \frac{T C^2 \lambda}{2 \sigma} + \log(\frac{1}{\delta}) \leq \epsilon \]

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

\[ \sigma^2 \geq \frac{T C^2 (1 + 2 \log(\frac{1}{\delta}))}{\epsilon^2} \]

Meanwhile, as we set in Protocol 3

\[ \sigma^2 \geq \frac{T C^2 (1 + 2 \log(\frac{1}{\delta}))}{\epsilon^2} \]

Therefore, Protocol 3 satisfies \((\varepsilon, \delta)\)-differential privacy.

**Theorem 4.** Protocol 4 satisfies \((\varepsilon_1, \delta_1)\)-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.
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 \((\epsilon_i, \delta_i)\), we can obtain the privacy guarantee of Protocol 4 by directly applying Lemma 1.

### 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 \(2^{f-\frac{2n}{k}}\) 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, \(O(mk^2 + mk)\) 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 \(\langle \theta_1 \rangle, \langle \theta_2 \rangle, \ldots, \langle \theta_m \rangle\).

**Output:** \(P_i\) obtains the share of the average of local models \(\langle \theta_n \rangle\);  
1: \(P_i\) compute \(\langle \theta_n \rangle = \sum_{j=1}^{m} \langle \theta_j \rangle / m\);  
2: \(P_i\) obtains \(\langle \theta_n \rangle\) as the share of the average of local models;

| Datasets    | Clipping bound | Batch size | LR   |
|-------------|----------------|------------|------|
| MNIST       | 0.25           | 2 × 10^-3  | 1    |
| CIFAR10     | 0.25           | 1.7 × 10^-3| 1    |
| IMDb        | 0.25           | 4 × 10^-3  | 1    |

| Datasets    | Clipping bound | Batch size | Epoch | LR  |
|-------------|----------------|------------|-------|-----|
| MNIST       | 0.25           | 2 × 10^-3  | 1     | 1   |
| CIFAR10     | 0.25           | 1.7 × 10^-3| 1     | 1   |
| IMDb        | 0.25           | 4 × 10^-3  | 1     | 1   |

| Hyperparameter | Potential values |
|----------------|------------------|
| Clipping bound | 64, 128, 256, 512|
| Batch size     | 64, 128, 256, 512|
| Learning rate schedule | Constant (lr=0.1), OneCycleLR (max_lr=0.1) |
### Protocol 7 Accuracy Strategy

**Input:** $P_i$ holds shares of local models $\langle \theta_i^1 \rangle, \langle \theta_i^2 \rangle, \ldots, \langle \theta_i^n \rangle$.

**Output:** $\theta_{i}$ and $\theta_{j}$ are the share of the most accurate local model $\langle \theta_m \rangle$.

1. **for** Each $\theta_i^j \in \{\theta_i^1, \theta_i^2, \ldots, \theta_i^n\}$ **do**
2.  All parties collaboratively evaluate the accuracy of $\theta_i^j$ as $\text{Accuracy}_i$.
3.  **if** $\text{Accuracy}_i \geq \text{Accuracy}_j$ **then**
4.    $\langle \theta_m \rangle = \langle \theta_i^j \rangle$;
5.    $\text{Accuracy}_m = \text{Accuracy}_i$;
6.  **end if**
7.  **end for**
8.  $P_i$ obtains $\langle \theta_m \rangle$ as the share of the most accurate local model.

### 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\(^1\) 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, 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 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 | $\epsilon$ | Local party | TF-Encrypted | Queqiao |
|--------|------------|-------------|--------------|---------|
| 3-party| 0.25       | 84.93 (±0.44) | 84.93 (±0.42) | 84.93 (±0.41) |
| 5-party| 0.25       | 84.93 (±0.44) | 84.93 (±0.42) | 84.93 (±0.41) |
| 10-party| 0.25      | 84.93 (±0.44) | 84.93 (±0.42) | 84.93 (±0.41) |

### 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 | $\epsilon$ | Local party | TF-Encrypted | Queqiao |
|--------|------------|-------------|--------------|---------|
| 3-party| 0.25       | 84.93 (±0.44) | 84.93 (±0.42) | 84.93 (±0.41) |
| 5-party| 0.25       | 84.93 (±0.44) | 84.93 (±0.42) | 84.93 (±0.41) |
| 10-party| 0.25      | 84.93 (±0.44) | 84.93 (±0.42) | 84.93 (±0.41) |

\(^1\)https://github.com/framer/Handcrafted-DP