Efficient and Privacy-Enhanced Federated Learning for Industrial Artificial Intelligence

Meng Hao, *Student Member, IEEE*, Hongwei Li, *Senior Member, IEEE*, Xizhao Luo, Guowen Xu, *Student Member, IEEE*, Haomiao Yang, *Member, IEEE*, and Sen Liu

***Abstract*—By leveraging deep learning-based technologies, industrial artificial intelligence (IAI) has been applied to solve various industrial challenging problems in Industry 4.0. However, for privacy reasons, traditional centralized training may be unsuitable for sensitive data-driven industrial scenarios, such as healthcare and autopilot. Recently, federated learning has received widespread attention, since it enables participants to collaboratively learn a shared model without revealing their local data. However, studies have shown that, by exploiting the shared parameters adversaries can still compromise industrial applications such as auto-driving navigation systems, medical data in wearable devices, and industrial robots' decision making. In this article, to solve this problem, we propose an efficient and privacy-enhanced federated learning (PEFL) scheme for IAI. Compared with existing solutions, PEFL is noninteractive, and can prevent private data from being leaked even if multiple entities collude with each other. Moreover, extensive experiments with real-world data demonstrate the superiority of PEFL in terms of accuracy and efficiency.**

***Index Terms*—Federated learning (FL), industrial artificial intelligence, privacy protection.**

Manuscript received June 15, 2019; revised August 10, 2019; accepted September 7, 2019. Date of publication October 4, 2019; date of current version June 22, 2020. This work was supported in part by the National Key R&D Program of China under Grant 2017YFB0802300 and Grant 2017YFB0802000, in part by the National Natural Science Foundation of China under Grant 61972454, Grant 61802051, Grant 61772121, Grant 61728102, and Grant 61472065, in part by the Peng Cheng Laboratory Project of Guangdong Province under Grant PCL2018KP004, and in part by the Guangxi Key Laboratory of Cryptography and Information Security under Grant GCIS201804. Paper no. TII-19-2524. *(Corresponding author: Hongwei Li.)*

M. Hao is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China and also with Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518000, China (e-mail: menghao0303@foxmail.com).

H. Li is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China and also with Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518000, China (e-mail: hongweili@uestc.edu.cn).

X. Luo is with the School of Computer Science and Technology, Soochow University, Suzhou 215006, China (e-mail: xzluo@suda.edu.cn).

G. Xu and S. Liu are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: guowen.xu@foxmail.com; 893551724@qq.com).

H. Yang is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: haomyang@uestc.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

I. Introduction

INDUSTRY 4.0 enhances the operational efficiency of the entire manufacturing process by incorporating multiple emerging technologies, including blockchain, Internet of Things, cloud computing, and artificial intelligence [1], [2]. Recently, industrial artificial intelligence (IAI) has attracted widespread attention and has driven the arrival of Industry 4.0 [3], [4]. As a branch of artificial intelligence, deep learning (DL) is widely used to solve data-driven industrial problems in real-world applications, such as smart grid, autonomous driving, and facial recognition [5], [6]. The development of DL-based products often requires large amount of data collected from different IoT devices, which is essential for learning high-quality DL models. Nevertheless, the collection of massive data for centralized training causes serious privacy threats [7], [8]. For example, malicious adversaries can compromise the privacy of IoT devices by eavesdropping industry data uploaded to cloud service (CS) providers (such as Microsoft Azure Machine Learning,[1] Google Cloud ML Engine[2]) [9], [10]. On the other hand, the large scale of current industrial systems, represented by the number of industrial nodes and the amount of data generated, makes such traditional deployment (e.g., centralized training) nontractable [11], [12]. Therefore, it puts new demands on how to effectively and collaboratively implement artificial intelligence in large-scale industrial applications.

Privacy and scalability issues can be mitigated through distributed training across multiple industrial nodes, which leads to what is currently called federated learning (FL) [13]. FL enables participants to learn the model collaboratively by only sharing local parameters without revealing private data. Intuitively, it is safer than centralized training. However, recent studies have shown that it also brings various privacy concerns [14], [15]. For example, with the shared parameters, a malicious adversary can launch attacks to recover images from a face recognition system [16]. In addition, adversaries could easily violate the health data of patients in wearable devices, and even change the emergency response of autonomous driving [17]. This not only damages the personal privacy, but more seriously, can bring unpredictable economic and life losses. In a sense, the privacy threat in FL is also an important factor hindering the promotion and further development of IAI-based applications.

Digital Object Identifier 10.1109/TII.2019.2945367

[1][Online]. Available: https://azure.microsoft.com

[2][Online]. Available: https://cloud.google.com/ml-engine/

In order to solve above privacy threats, several remarkable solutions have been proposed and applied in FL-based scenarios. For instance, Shokri and Shmatikov [18] proposed the first privacy-preserving distributed learning system, where participants selectively share small part of the gradients to ensure the privacy of training data. However, their proposed approach has been exposed to be insecure even for passive attackers [19]. By exploiting secure multiparty computation (SMC), Bonawitz *et al.* [20] presented a privacy-preserving protocol to support the secure aggregation in FL. Zhang *et al.* [21] also solved the same problem by combining the threshold secret sharing and homomorphic encryption (HE) scheme. Unfortunately, both of the solutions required multiple communication rounds in each aggregation. This inevitably increases communication burden during entire training process. Some recent work [19], [22] can alleviate this problem by delegating the decryption operation to each participant. However, to decrypt successfully, all participants share the same decryption key, which is vulnerable when an adversary compromises some honest entities. Moreover, none of the above solutions consider the privacy issues of the parameters returned by the CS. Therefore, it is urgent to design a noninteractive and privacy-preserving FL scheme that can prevent the privacy leakage from the local gradients as well as the shared parameters even when an adversary colludes with multiple entities.

However, it is challenging to design a solution that meets above requirements. The state-of-the-art schemes for privacy-preserving FL are mainly based on three cryptographic methods: SMC, differential privacy (DP), and HE. First, SMC is obviously not suitable for our solution since we require a noninteractive protocol to perform secure aggregation. Second, DP is an efficient way to prevent the privacy leakage in training process. However, traditional DP [23] may not be suitable for FL, since a trusted third party is required to add noises to the statistical result. This is contrary to the reality that the CS is generally considered dishonest. Third, HE-based scheme is a potential solution, which can be used to support noninteractive FL if all participants are assigned the same secret key. However, this is vulnerable if there are multiple entities colluding during the training. Another variant, e.g., threshold HE, forces that the CS can decrypt the aggregated ciphertext correctly only with the assistance of multiple participants. However, the interactive decryption seriously increases the communication overhead, especially for high-dimensional data in neural networks.

In order to solve above challenges, we propose privacy-enhanced federated learning (PEFL) to achieve efficient and PEFL for IAI. Our contributions are summarized as follows:

1) PEFL is noninteractive in each secure aggregation. To achieve our secure aggregation protocol, the homomorphic ciphertext of private gradients is embedded into augmented learning with error (A-LWE) term. In particular, we provide a concrete instantiation, where we utilize improved BGV HE scheme that removes the key-switching operation and increases the plaintext space.
2) PEFL prevents the privacy leakage from the local gradients as well as the shared parameters even when an adversary colludes with multiple entities. Specifically, to further protect the privacy of training data, we achieve example-level DP by using distributed Gaussian mechanism.
3) A detailed security analysis indicates that PEFL provides postquantum security and guarantees aggregator oblivious security. Performance evaluation demonstrates practical training accuracy, as well as efficient computation and communication overhead.

The remainder of this article is organized as follows. In Section II, we introduce the problem and outline the system model and threat model. In Section III, we discuss the neural network and cryptographic primitives. Then, we present our scheme in detail in Section IV and carry out the security analysis in Section V. And performance evaluation is depicted in Section VI. Finally, we discuss related work in Section VII and Section VIII concludes this article.

II. Problem Statement, System Architecture, and Threat Model

In this section, we first briefly introduce the problem, and then describe the system architecture and the threat model.

A. Problem Statement

We consider the standard settings for FL, where several industrial nodes (called participants) collaboratively learn parametric neural networks. The objective is to find a parameter vector $\mathbf{w}$ of neural network that minimizes the expected empirical loss

$$L(\mathbf{w}) = \sum_{D_i \in D} \sum_{(\mathbf{x},\mathbf{y}) \in D_i} l\left(f(\mathbf{x}, \mathbf{w}), \mathbf{y}\right) \tag{1}$$

where $l(x, y)$ is loss function and each participant i holds private dataset D_i. Specifically, each participant learns local neural network model on its private dataset and uploads the gradients calculated by minimizing the loss function to a centralized server. To further guarantee privacy, we apply the cryptography method on the uploaded gradients while ensuring training accuracy. Finally, the privacy-preserving training is completed until a certain convergence criterion is met.

B. System Architecture

As shown in Fig. 1, our system consists of *key generation center (KGC)*, *CS* and multiple *Participants*.

1) *KGC:* KGC is a trusted entity who initializes the entire system, generates the public parameters, and distributes private key to each participant and the CS.
2) *CS:* The CS is a common place to recursively aggregate encrypted local gradients from the participants in a secure manner, and then broadcasts the updated global parameters to each participant.
3) *Participants:* Each participant has the private dataset and the replica of shared global model. During the learning process, each participant trains its local model over private dataset and exchanges the local gradients through the CS.

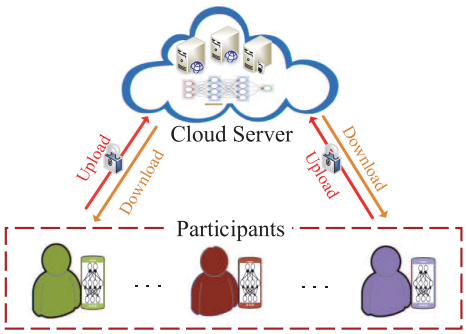


Fig. 1. System Architecture.

C. Threat Model

In our proposed scheme, the KGC is considered as fully trusted and hence would never collude with any entity. We assume that participants and the CS are *honest-but-curious*. Such an adversarial setting has been widely used in cloud computing schemes, such as [24], [25]. In other words, all entities (except the KGC) honestly execute the protocol as designed, but may attempt to infer sensitive information about the training data. Additionally, we also assume the CS may collude with multiple corrupted entities. It means that the corrupted participants can reveal auxiliary information to the CS. Based on the above assumptions, our goal is to ensure the privacy of individual participant's information during the entire training process.

III. Preliminaries

In this section, we review the background of DL and some cryptographic primitives that serve as the foundation of our scheme.

A. Background of DL

Neural networks are trained through iteratively performing *feedforward* and *backpropagation* process. *Feedforward* can be represented as $f(\mathbf{x}, \mathbf{w}) = \bar{\mathbf{y}}$, where $\mathbf{x}$ is input vector and $\mathbf{w}$ is parameter vector. A training dataset is $D = \{(\mathbf{x}_i, \mathbf{y}_i); i \in I\}$ for each example $(\mathbf{x}_i, \mathbf{y}_i)$. Given loss function l, the loss on a training dataset D can be defined as $L(D, \mathbf{w}) = \frac{1}{|D|}\sum_{(\mathbf{x}_i,\mathbf{y}_i)\in D} l(\mathbf{y}_i, f(\mathbf{x}_i, \mathbf{w}))$. In the *backpropagation* phase, the stochastic gradient descent (SGD) is utilized to update parameters

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta \nabla_{\mathbf{w}} L(D^t, \mathbf{w}^t) \tag{2}$$

where hyperparameter η is learning rate, $\mathbf{w}^t$ is the parameter vector after tth iteration and $D^t \subseteq D$ is a mini-batch of training dataset.

The above standard training method requires the training data to be concentrated in one data center or one service provider. In order to alleviate the privacy problem of collecting data, FL was introduced, which simultaneously improves the quality of training model and reduces the power consumption of a single machine. In other words, FL enables participants to collaboratively learn a shared prediction model while keeping all the training data on local device. In detail, each participant $v \in V$ holds a private dataset $D_v \subseteq D$. For mini-batch $D^t = \cup_{v\in V} D_v^t$, SGD rule can be rewritten

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta \frac{\sum_{v\in V} \nabla_{\mathbf{w}} L\left(D_v^t, \mathbf{w}^t\right)}{|V|}. \tag{3}$$

B. Differential Privacy

DP is a privacy mechanism that ensures that a single tuple will not change the overall statistics of the dataset [26]. A formal definition is defined as a randomized algorithm f is (ε, δ) private, for all adjacent datasets $d, d' \in \mathbb{D}$ and all subset of outputs $\mathcal{T} \subseteq \mathbb{R}$, if $Pr[f(d) \in \mathcal{T}] \leq e^{\varepsilon} Pr[f(d') \in \mathcal{T}] + \delta$. Here, δ is introduced to account the probability that plain ε-DP is broken [26].

Definition 1 (Gaussian Mechanism [27]): Let $f : \mathbb{D} \to \mathbb{R}$ be an arbitrary deterministic algorithm. For $\varepsilon \in (0, 1)$ and $\delta \geq \frac{4}{5} exp(-(\sigma\varepsilon)^2/2)$, the Gaussian mechanism $\overline{f}(d) = f(d) + \mathcal{N}(0, (\Delta f)^2\sigma^2)$ satisfies (ε, δ)-DP, where $\Delta f = max_{adj(d,d')} \|f(d) - f(d')\|_2$ represents global sensitivity and $\mathcal{N}(0, (\Delta f)^2\sigma^2)$ is Gaussian distribution with mean 0 and standard deviation $\Delta f\sigma$.

C. Learning With Error

A LWE [28] sample is $(\mathbf{A}, \mathbf{b}^{\mathrm{T}} = \mathbf{s}^{\mathrm{T}}\mathbf{A} + \mathbf{e}^{\mathrm{T}})$, where the matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{\kappa\times\tau}$ and the secret $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^{\kappa}$, and the error term $\mathbf{e}$ is sampled from discrete Gaussian distribution over $\mathbb{Z}_p^{\tau}$. Only with $(\mathbf{A}, \mathbf{b})$ it is very difficult to recover the secret $\mathbf{s}$ or error term $\mathbf{e}$. Furthermore, Bansarkhani *et al.* proposed A-LWE that allows one to hide auxiliary information into the error term $\mathbf{e}$ [29], [30].

Definition 2 (Augmented LWE (A-LWE)): Let κ, τ, p, l be integers, and let f be a function whose the outputs are indistinguishable from random ones. The A-LWE samples can be obtained as follows:

1) *Set* $\mathbf{v} = f(m) \in \mathbb{Z}_p^{\tau/l}$ about the information m.
2) *Sample* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{\kappa\times\tau}$, $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_p^{\kappa}$ and $\mathbf{e} \leftarrow D_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{G}),\sigma} \in \mathbb{Z}_p^{\tau}$.
3) *Return* $(\mathbf{A}, \mathbf{b}^{\mathrm{T}} = \mathbf{s}^{\mathrm{T}}\mathbf{A} + \mathbf{e}^{\mathrm{T}})$.

where $\mathbf{G} = \mathbf{I}_{\tau/l} \otimes \mathbf{g}^{\mathrm{T}} \in \mathbb{Z}_p^{\kappa\times\tau}$, $\mathbf{g}^{\mathrm{T}} = (1, 2, \ldots, 2^{l-1}) \in \mathbb{Z}_p^{l}$ and $D_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{G}),\sigma}$[3] denotes the special error distribution indistinguishable from Gaussian distribution [29].

Lemma 1: Assume full rank matrix $\mathbf{G} \in \mathbb{Z}_p^{\kappa\times\tau}$ and $\rho = negl(\kappa)$, $\mathbf{v} \in \mathbb{Z}_p^{\tau/l}$ with $l = \lceil \log p \rceil$. If $\mathbf{v}$ is computationally indistinguishable from random uniform samples over $\mathbb{Z}_p^{\tau/l}$, then $D_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{G}),\sigma}$ is computationally indistinguishable from standard discrete Gaussian distribution for $\sigma \geq \eta_{\rho}(\Lambda^{\perp}(\mathbf{G}))$. Hence, A-LWE is for certain instantiations at least as hard as the LWE problem.

D. Additively HE

HE allows some computation to be performed on the ciphertext to generate an encrypted result without decryption. The new ciphertext matches the result of the operation performed on the

[3]Note that the n-dimension lattice is denoted as $\Lambda^{\perp}(\mathbf{G}) = \{\mathbf{x} \in \mathbb{Z}_p^n | \mathbf{G}\mathbf{x} = 0 \mod p\}$. And $\Lambda_{\mathbf{v}}^{\perp}(\mathbf{G}) = \{\mathbf{x} \in \mathbb{Z}_p^n | \mathbf{G}\mathbf{x} = \mathbf{v} \mod p\}$ presents the shift from the lattice $\Lambda^{\perp}(\mathbf{G})$, where $\mathbf{G}\mathbf{y} = \mathbf{v} \mod p$.

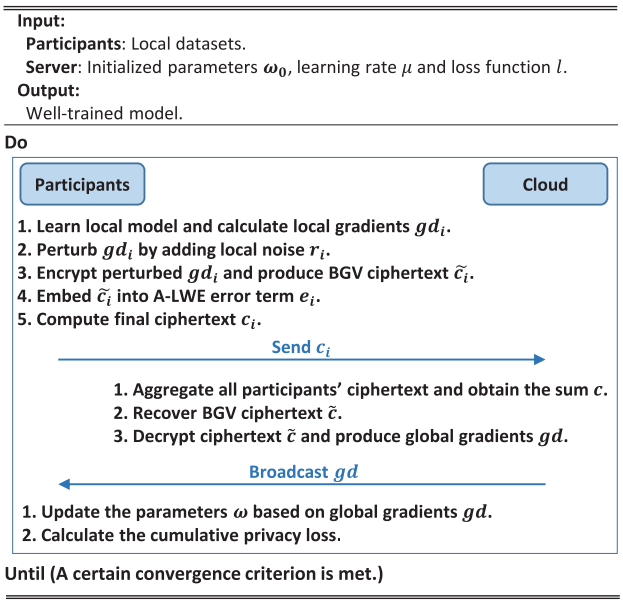


Fig. 2. High-level view of our proposed scheme.

plaintext after decryption. We optimized BGV [31] encryption scheme by removing the key-switching and increasing plaintext space. We refer the readers to [31] for detailed encryption scheme.

IV. Our Proposed Scheme

In this section, we first outline our PEFL at a high level before giving a description of the technical details.

A. Our PEFL at a High Level

Our PEFL is noninteractive in each aggregation. Moreover, PEFL ensures the privacy of training data during and after the training process, even when an adversary colludes with multiple entities.

We summary our proposed scheme in Fig. 2. The protocol that consists of multiple participants and the CS achieves secure FL. Specifically, on the participants side, each participant first perturbs the local gradients vector by using distributed Gaussian mechanism to achieve DP. Then, the perturbed gradients vector is encrypted into the BGV ciphertext (called *internal ciphertext*). Finally, the ciphertext of BGV encrption is embedded into A-LWE ciphertext (called *external ciphertext*) to achieve secure aggregation protocol. On the CS side, only after aggregating all ciphertexts from the participants, the server can correctly decrypt the external ciphertext. Meanwhile, the internal ciphertexts are summed automatically and then the CS can easily decrypt aggregated values without revealing the privacy of individual participant.

B. Basic Construction for PEFL

In this section, we describe PEFL scheme in detail. To implement our protocol efficiently, we utilize the Ring-LWE introduced by Lyubashevsky *et al.* [32]. Let $\Phi_n(X)$ be irreducible polynomial with the degree $\phi(n)$, polynomial ring $R = \mathbb{Z}[X]/(\Phi_n(X))$ and $R_p = R/pR$. Ring-LWE samples are rewritten as $(a, b = s \cdot a + e)$, where s, e are sampled from the Gaussian distribution over the Ring R_p, and a sampled from R_p uniformly at random. Like LWE, Ring-LWE enjoys worst-case hardness guarantees [32].

We first define plaintext and ciphertext space. Ring $R_q = (\mathbb{Z}/q\mathbb{Z})[X]/(\Phi_n(X))$ denotes plaintext space with modulus q. Similarly, $R_{p_1} = (\mathbb{Z}/p_1\mathbb{Z})[X]/(\Phi_n(X))$ denotes *internal ciphertext* space of RBGV and $R'_{p_1} = (\mathbb{Z}/p_1\mathbb{Z})[X]/(\Phi_{n'}(X))$ denotes *external ciphertext* space of A-LWE, where $\phi(n') = 2\,l\phi(n)$ with $l = \lceil \log p_1 \rceil$ and $p_1 = p \cdot p_0$ for primes p, p_0.

Then, for better readability, we define several commonly used sampling subroutines as follows:

1) $\mathcal{SV}(n)$: Generate a "small" vector of n elements from $\{-1, 0, 1\}$ such that the probabilities for each value are $Pr_{-1} = \frac{1}{4}$, $Pr_0 = \frac{1}{2}$, $Pr_1 = \frac{1}{4}$.
2) $\mathcal{GS}(n, \sigma)$: Generate a vector of n elements from Gaussian distribution with mean 0 and standard deviation σ.
3) $\mathcal{UN}(n, p)$: Generate a vector of n elements from randomly uniform distribution modulo p.

PEFL consists of five phases as follows:

1) Setup

Let κ be security parameter and $N \in \mathbb{N}$ be the number of participants. For a clear description, we represent internal encryption with tilde line.

For internal encryption (e.g., RBGV):

1) Draw $\widetilde{a} \leftarrow \mathcal{UN}(\phi(n), p_1)$ and $\widetilde{s}, \widetilde{\gamma} \leftarrow \mathcal{GS}(\phi(n), \widetilde{\sigma})$.
2) Compute $\widetilde{b} = \widetilde{a} \cdot \widetilde{s} + q \cdot \widetilde{\gamma}$.
3) Output $pk = (\widetilde{a}, \widetilde{b}) \in R_{p_1} \times R_{p_1}$ as public key and $SK_{C_2} = \widetilde{s} \in R_{p_1}$ as part of secret key for the CS.

For external encryption (e.g., A-LWE)

1) Draw $a \leftarrow \mathcal{UN}(\phi(n'), p_1)$ and $s_i \leftarrow \mathcal{GS}(\phi(n'), \sigma)$.
2) Set $\mathbf{g}^{\mathrm{T}} = (1, 2, \ldots, 2^{l-1})$ and compute $\mathbf{G} = \mathbf{I} \otimes \mathbf{g}^{\mathrm{T}}$.
3) Output $a, \mathbf{g}^{\mathrm{T}}, \mathbf{G}$ as public parameters.
4) Output $sk_i = s_i \in R'_{p_1}$ as secret key for participant i and $SK_{C_1} = -\sum_i s_i \in R'_{p_1}$ as another part of secret key for the CS.

2) Training networks

We assume the CS has initialized a neural network, including the learning rate η, the original global parameters $\mathbf{w}^0$ and the loss function l. The initialized model has been shipped to all participants.

In tth iteration, each participant $i \in [N]$ learns the neural network on the private dataset $D_i^t \subseteq D^t$ and calculates the loss on D_i^t as follows:

$$L(D_i^t, \mathbf{w}^t) = \frac{1}{|D_i^t|} \sum_{(\mathbf{x}_j, \mathbf{y}_j) \in D_i^t} l\left(\mathbf{y}_j, f(\mathbf{x}_j, \mathbf{w}^t)\right). \quad (4)$$

Each participant i executes back propagation and computes the gradient vector $\mathbf{gd}_i^t = \nabla_{\mathbf{w}} L(D_i^t, \mathbf{w}^t)$.

3) Gradients encryption

In order to achieve the conversion between ring R and vector $\mathbb{Z}^n$ during encryption phase, we define the mappings as follows:

Algorithm 1: : Algorithm **Sample** to Sample from $\Lambda^{\perp}(\mathbf{G})$.

Input: $v_{i_j} \in \mathbb{Z}_{p_1}, \sigma$
Output: $\mathbf{t} = (t_0, t_1, \ldots, t_l)^{\mathrm{T}} \in \Lambda^{\perp}_{v_{i_j}}(\mathbf{G})$
$x_0 = v_{i_j}$;
for each $k \in [0, l-1]$ **do**
  $t_k \leftarrow D_{2\mathbb{Z}+x_k,\sigma}$;
  $x_{k+1} = (x_k - t_k)/2$;
end for

1) $Map_{R\to\mathbb{Z}^n}$: outputs a vector of size n consisting of the entries of the coefficient representation of the input ring element.
2) $Map_{\mathbb{Z}^n\to R}$: takes the vector and outputs the ring element whose coefficient representations are the entries of the vector.

Generation of internal ciphertext (e.g., RBGV)

1) Set $gd_i^t = Map_{\mathbb{Z}^{\phi(n)}\to R_q}(\mathbf{gd}_i^t) \in R_q$.
2) Draw $\widetilde{e}_0, \widetilde{e}_1 \leftarrow \mathcal{GS}(\phi(n), \widetilde{\sigma})$ and $\widetilde{v} \leftarrow \mathcal{SV}(\phi(n))$.
3) Compute $\widetilde{c}_{i,0} = \widetilde{b}\cdot\widetilde{v} + q\cdot\widetilde{e}_0 + gd_i^t$ and $\widetilde{c}_{i,1} = \widetilde{a}\cdot\widetilde{v} + q\cdot\widetilde{e}_1$ for modulus p_1.
4) Output internal ciphertext $\widetilde{c}_i = (\widetilde{c}_{i,0}, \widetilde{c}_{i,1}) \in R_{p_1}\times R_{p_1}$.

Generation of external ciphertext (e.g., A-LWE)

1) Set $\mathbf{v}_i = Map_{R_{p_1}\to\mathbb{Z}^{\phi(n)}}(\widetilde{c}_{i,0}\|\widetilde{c}_{i,1}) \in \mathbb{Z}_{p_1}^{2\phi(n)}$.
2) Invoke algorithm 1 to sample $\mathbf{e}_i \in \mathbb{Z}_{p_1}^{2\phi(n)l}$ subject to the distribution $\Lambda^{\perp}_{\mathbf{v}_i}(\mathbf{G})$, where $\mathbf{e}_i = (Sample(v_{i_1}, \sigma), Sample(v_{i_2}, \sigma), \ldots, \mathbf{Sample}(v_{i_{2\phi(n)}}, \sigma))$.
3) Set $e_i = (Map_{\mathbb{Z}^{\phi(n')}\to R'_{p_1}}(\mathbf{e}_i))$.
4) Compute $c_i = a\cdot s_i + e_i \in R'_{p_1}$.
5) Send final ciphertext c_i to the CS.

Note that the embedding method is efficiently achieved by making use of the gadget matric $\mathbf{G} = \mathbf{I}\otimes\mathbf{g}^{\mathrm{T}}$. According to Definition. 2, $\mathbf{G}\mathbf{e}_i = \mathbf{v}_i \mod p$ holds when the elements of $\mathbf{e}_i$ are sampled from $\Lambda^{\perp}_{\mathbf{v}_i}(\mathbf{G})$. And hence the BGV ciphertext can be recovered successfully from $\mathbf{e}_i$ as well as from $\sum_i \mathbf{e}_i$. In addition, to efficiently facilitate the cryptographic scheme, each participant i first scales its local gradients $\mathbf{gd}_i^t$ from $\mathbb{R}^n$ to $\mathbb{Z}^n$ by multiplying with a scale factor 10^5 and truncating the remaining fractional part. It simultaneously incurs fairly limited accuracy loss.

4) Aggregation decryption

After receiving c_i from all participants, the CS performs aggregation and decryption.

Decryption of external ciphertext (e.g., A-LWE)

1) Aggregate all the ciphertexts $c = \sum_i c_i = a\cdot\sum_i s_i + \sum_i e_i \in R'_{p_1}$.
2) Compute the sum of error terms $e = c + a\cdot SK_{C_1} = \sum_i e_i \in R'_{p_1}$, where $SK_{C_1} = -\sum_i s_i$.
3) Set $\mathbf{e} = Map_{R'_{p_1}\to\mathbb{Z}^{\phi(n')}}(e)$.

Decryption of internal ciphertext (e.g., RBGV)

1) Recover the sum of RBGV ciphertexts by computing $\mathbf{v} = \sum_i \mathbf{v}_i = \mathbf{G}\cdot\mathbf{e} \mod p_1 \in \mathbb{Z}_{p_1}^{2\phi(n)}$.
2) Split the vector $\mathbf{v} = (\mathbf{c}_0, \mathbf{c}_1) \in \mathbb{Z}_{p_1}^{\phi(n)}\times\mathbb{Z}_{p_1}^{\phi(n)}$.
3) Set $\widetilde{c}_0 = Map_{\mathbb{Z}^{\phi(n)}\to R_{p_1}}(\mathbf{c}_0) \in R_{p_1}$ and $\widetilde{c}_1 = Map_{\mathbb{Z}^{\phi(n)}\to R_{p_1}}(\mathbf{c}_1) \in R_{p_1}$.
4) Invoke algorithm Scale$((\widetilde{c}_o, \widetilde{c}_1), p_1, p_0)$ to switch modulus and produce the scaled ciphertext $\widetilde{c}_o', \widetilde{c}_1'$ modulo p_0.
5) Decrypt the ciphertext and produce the sum of plaintext by $gd^t = \sum_{i\in[N]} gd_i^t = \widetilde{c}_o' - SK_{C_2}\cdot\widetilde{c}_1' \mod q \in R_q$.
6) Set $\mathbf{gd}^t = Map_{R_q\to\mathbb{Z}^{\phi(n)}}(gd^t)$.
7) Broadcast the global gradients $\mathbf{gd}^t$.

Observe that *modulus switching* operation is introduced to reduce the error of noise. Rather than the original method that requires to estimate an error level before reducing, we adopt Scale method defined in [33]. Specifically, Scale(c, p_0, p_1) inputs the ciphertext $c \in \mathcal{R}_{p_1}$ and outputs reduced ciphertext $c' \in \mathcal{R}_{p_0}$ such that it holds $c \equiv c' \mod q$ in coefficient representation, where q is the plaintext modulus. In above scheme, the *modulus switching* operation ensures that the reduced ciphertext is still the encryption of same plaintext (e.g., $(\widetilde{c}_o' - SK_{C_2}\cdot\widetilde{c}_1' \mod p_0) \equiv (\widetilde{c}_o - SK_{C_2}\cdot\widetilde{c}_1 \mod p_1) \mod q$).

5) Update networks

After receiving global gradients $\mathbf{gd}^t$, each participant i updates parameters $\mathbf{w}^{t+1}$ according to $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta\frac{\mathbf{gd}^t}{N}$ and updates the local network model. The training will be completed until the loss function reaches the minimum.

Extension: The decryption of internal ciphertext in step 4 can be completed by the participants if secret key SK_{C_2} is delegated to them. This extension is designed to ensure that no parameters of the neural network are exposed to the public during and after training procedure. However, downloading encrypted global parameters and decrypting them entail additional burden on the participant side in communication and computational overhead. This tradeoff between efficiency and security provides a flexible choice for institutions with different demands.

C. Improved Construction for PEFL

To further prevent the privacy leakage from the shared parameters, we adopt distributed Gaussian mechanism to achieve example-level DP. It benefits from the following Cramér's decomposition theorem [34]:

Theorem 1 (Cramér's decomposition theorem): Let a random variable ζ from normal distribution and permit a decomposition as a sum $\zeta = \zeta_1 + \zeta_2$ of two independent random variables. Then, the two independent random variables ζ_1 and ζ_2 are normally distributed as well.

Algorithm 2 outlines FL with distributed DP. Specifically, participant i first "clips" each gradient $gd_i^t(\mathbf{x})$ according to clipping bound θ, to bound the influence of each training data on training updates. The clipping operation bounds the L_2 norm of gradient vector, and hence θ is also the global sensitivity. Then, participant i adds distributed Gaussian noise $\mathcal{N}(0, \frac{\theta^2\rho^2}{N})$ to the aggregated gradients of mini-batch $\sum_{\mathbf{x}\in D_i^t} gd_i^t(\mathbf{x})$. In the end, the CS aggregates all participants' perturbed gradients and computes

$$\sum_{i\in[N]} gd_i^t = \sum_{i\in[N]}\sum_{\mathbf{x}\in D_i^t} gd_i^t(\mathbf{x}) + \sum_{i\in[N]} \mathcal{N}\left(0, \frac{\theta^2\rho^2}{N}\right) \quad (5)$$

Algorithm 2: : Differentially Private Federated Training.

Input: Participants N, dataset $\{D_n\}^{n\in[N]}$, loss $L(\mathbf{w},\mathbf{x})$, global parameter $\mathbf{w}^0$, clip bound θ and iteration T.
for $t \in [T]$ **do**
  for $i \in [N]$ **do**
    Sample mini-batch D_i^t from D_i with probability $\frac{|D_i^t|}{|D_i|}$
    for $\mathbf{x} \in D_i^t$ **do**
      $gd_i^t(\mathbf{x}) \leftarrow \nabla_{\mathbf{w}^t} L(\mathbf{w}^t, \mathbf{x})$
      $gd_i^t(\mathbf{x}) \leftarrow gd_i^t(\mathbf{x}) / \max\left(1, \frac{\|gd_i^t(\mathbf{x})\|}{\theta}\right)$
    end for
    $gd_i^t \leftarrow \sum_{\mathbf{x}\in D_i^t} gd_i^t(\mathbf{x}) + \mathcal{N}(0, \frac{\theta^2\rho^2}{N})$
  end for
  $gd^t \leftarrow \frac{1}{N}(\sum_{n\in[N]} gd_i^t)$
  $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta \cdot gd^t$
  Track the cumulative privacy loss utilizing moment accountant
end for
Output: well-trained parameters $\mathbf{w}^T$ and privacy cost (ε, δ)

where according to Theorem 1, $\sum_{n\in N}\mathcal{N}(0, \frac{\theta^2\rho^2}{N})$ is still Gaussian distribution $\mathcal{N}(0, \theta^2\rho^2)$. According to the argument of Gaussian distribution [27], if $\theta\rho$ is set to be $\sqrt{2\log\frac{5}{4\delta}}/\varepsilon$, each tth iteration in algorithm 2 is (ε, δ)-DP. From the amplification theorem [35], since sampling a batch of training data with probability $\frac{|D_i^t|}{|D_i|}$, each tth iteration in Algorithm 2 is changed to be $(\frac{|D_i^t|}{|D_i|}\varepsilon, \frac{|D_i^t|}{|D_i|}\delta)$-DP for entire dataset.

In order to track the cumulative privacy loss of entire iterations in algorithm 2, rather than strong composition theorem [27], we adopt moment accountant method [23]

Theorem 2 (Moments Accountant): Let $d, d^{'}$ be neighboring datasets and ψ be auxiliary information. For a differentially private mechanism $\overline{f}$, we define the moments accountant at λ^{th} as $\alpha_{\overline{f}}(\lambda) \triangleq \max_{\psi,d,d'} \alpha_{\overline{f}}(\lambda;\psi,d,d^{'})$, where $\alpha_{\overline{f}}(\lambda;\psi,d,d^{'}) \triangleq \log \mathbb{E}[exp(\lambda c(\overline{f},\psi,d,d^{'}))]$ is a moment generating function and $c(\overline{f},\psi,d,d^{'}) \triangleq \log\frac{Pr[\overline{f}(\psi,d)=y]}{Pr[\overline{f}(\psi,d^{'})=y]}$ is privacy loss.

Note that we can calculate the log moments of privacy loss through the privacy accountant, which prematurely terminates learning when the total privacy loss of access training data exceeds the predetermined privacy budget. By doing this, the algorithm 2 achieves $(\mathcal{O}(\frac{|D_i^t|}{|D_i|}\varepsilon\sqrt{T}), \delta)$-DP for entail dataset. Since DP algorithm ensures that the probability whether the trained model contains a particular data record is indistinguishable, the improved construction can prevent the privacy leakage from the shared parameters.

V. Security Analysis

A. Indistinguishability of Ciphertext

Theorem 3: Ciphertext c_i in our PEFL protocol is computationally indistinguishable from random values assuming solving the lattice problem is difficult in worst case.

Proof: According to the security of BGV encryption scheme [31], the internal ciphertext $\widetilde{c}_i$ generated based on specified security parameter κ is indistinguishable from random values. Then, the vector $\mathbf{v}_i$ is transformed from polynomial $\widetilde{c}_i$ and indistinguishable from random vector. From Lemma. 1, $D_{\Lambda^{\perp}_{\mathbf{v}_i}(\mathbf{G}),\sigma}$ is computationally indistinguishable from standard discrete Gaussian distribution for $\sigma \geq \eta_\rho(\Lambda^{\perp}(\mathbf{G}))$. Then, since e_i is from $D_{\Lambda^{\perp}_{\mathbf{v}_i}(\mathbf{G}),\sigma}$, $c_i = a\cdot s_i + e_i$ is indeed a augmented LWE (A-LWE) sample over the ring $R^{'}_{p_1}$. A-LWE problem in ring setting is for certain instantiations at least as hard as the ring LWE problem, and hence the c_i's security comes directly from decisional ring LWE problem. For $\sigma \geq \omega(\sqrt{\log\kappa})\cdot(\kappa N/\log(\kappa N))^{\frac{1}{4}}$, decisional ring LWE problem is difficult if solving the lattice problem is difficult in worst case. That is, a polynomial time adversary can distinguish A-LWE ciphertext from random values only if he can solve the lattice problem in worst case. Therefore, c_i is computationally indistinguishable from random values and hence the Theorem 3 follows. ■

B. Aggregator Obliviousness Security

The obliviousness security focuses on that the CS learns nothing but what can be learned from aggregated sum, even when the CS has arbitrary auxiliary information and compromises some participants. We define aggregator obliviousness as follows:

A protocol is aggregation oblivious if no adversary $\mathcal{A}$ has more than negligible probability in probabilistic polynomial-time $poly(\kappa)$ in winning the security game as follows:

1) *Setup:* Challenger executes Setup algorithm and generates the public parameters *params* to adversary $\mathcal{A}$.
2) *Queries:* Adversary $\mathcal{A}$ makes the following three specific queries adaptively.
   a) *Encrypt:* Adversary may specify (i, m) and request the ciphertext, where i is the index of one entity (including participants and the CS), and m represents the plaintext. Then, the challenger returns the Enc(i, m) to $\mathcal{A}$.
   b) *Compromise:* The adversary comprises an entity i, and then the challenger returns participant i's secret key s_i or the CS's SK_C. Note, this step can be repeated multiple times.
   c) *Challenge:* This query can only be allowed once throughout the entire game. The adversary chooses a participants' subset K, each of which has not been compromised before.
   For each $i \in K$, the adversary generates and sends two plaintext m_i^0 and m_i^1. Then, the challenger flips a random bit b. If $b = 0$, the challenger computes the c_i =Enc (m_i^0). Otherwise, m_i^1 is encrypted in the same way and the challenger sends the c_i to the adversary.
3) *Guess:* The adversary outputs a guess of whether b is 0 or 1.

Theorem 4: Our scheme PEFL is aggregator-oblivious secure assuming solving the lattice problem is difficult in worst case.

Proof: In each aggregation process, an adversary $\mathcal{A}$ can compromise all participants, except μ and ν, and requests their corresponding secret keys $\{s_i\}_{i\neq\mu,i\neq\nu}$. Then, $\mathcal{A}$ arbitrarily specifies the plaintext m_i of participant i for $i\neq\mu, i\neq\nu$ and encrypts it using secret key s_i. Even though $\mathcal{A}$ can compromise the CS to obtain the secret key SK_C, given the ciphertext of all participants, he can only get the sum aggregation of m_μ and m_ν. That is, $\mathcal{A}$ still cannot distinguish the ciphertext of μ and ν from the random values. If $\mathcal{A}$ can distinguish the ciphertext of μ and ν from random values, he has more than negligible advantage to win the aggregator obliviousness security game. In other words, $\mathcal{A}$ can solve decision augmented LWE problem, and hence distinguish an A-LWE sample from a uniform sample in random. However, it is well-known that solving the lattice problem in worst case is difficult [30]. Therefore, our scheme PEFL is aggregator-oblivious secure. ■

C. DP of Participants' Dataset

When the adversary does not collude with the participants, each participant independently adds local noise according to Gaussian distribution $\mathcal{N}(0, \frac{\theta^2\rho^2}{N})$. Since the final aggregated noise still satisfies Gaussian distribution $\mathcal{N}(0, \theta^2\rho^2)$, the DP of PEFL is guaranteed.

When the adversary colludes with a subset of participants, we first define collusion ratio $\beta = \frac{t}{N}$, where t is the number of compromised participants. Since t compromised participants may leak noise to adversaries, to provide DP guarantees, rather than $\mathcal{N}(0, \frac{\theta^2\rho^2}{N})$ each participant adds local noise according to $\mathcal{N}(0, \frac{\theta^2\rho^2}{N-t})$. In the worst case, all participant adds noise according to $\mathcal{N}(0, \frac{\theta^2\rho^2}{N-t})$, and hence the noise scale of the final aggregation is $\frac{N\theta^2\rho^2}{N-t}$ (e.g., $\frac{\theta^2\rho^2}{1-\beta}$). Even though this increases the amount of noise added to gradients, privacy loss of each iteration would be reduced and the training will run more epoch within a given privacy budget. We will show the impact of the collusion on prediction accuracy in the experimental section.

Applying DP to neural networks helps ensure defenses against membership inference and model inversion attacks [36]. That is if the learning process satisfies DP, the probability of generating prediction model from a training dataset containing a particular record is similar to the probability of generating the same model when the record is not included. As these attack methods are very dependent on the training method, our construction can mitigate the risk of membership inference and model inversion attacks.

VI. Performance Evaluation

We conduct experiments on the server with an Intel(R) Xeon(R) CPU E5-2620 v4(2.10 GHz) and 16 GB of RAM. The cryptographic scheme of PEFL utilizes some C++ library HElib,[4] and the FL is simulated by TensorFlow in Python.

A. Functionality

Table I lists the functional comparison of the latest privacy-preserving FL schemes, including PDL [19], PSA [20], and PPM [21]. In particular, although PSA and PPM can resist collusion between an adversary and multiple entities, both of the solutions do not guarantee postquantum security and involve multiple communication rounds. Compared with PSA and PPM, our PEFL is postquantum secure and noninteractive, while can prevent privacy leakage even when an adversary colludes with multiple entities. In addition, PDL achieves noninteractive protocol and post-quantum security. However, all participants share the same decryption key, which is extremely undesirable in real situations. Our PEFL takes into account realistic application, where the adversary may compromise honest parties. Besides, none of the three existing solutions concern privacy leakage from shared parameters. That is vulnerable when adversaries launch model inversion attack or membership inference attack. Our PEFL utilizes DP technology to mitigate the above attacks.

TABLE I
Functional Comparison of Several Privacy-Preserving FL Schemes

	PDL	PSA	PPM	Our PEFL
Gradients confidentiality	✓	✓	✓	✓
Resistance to collusion	×	✓	✓	✓
Noninteractive protocol	✓	×	×	✓
Postquantum Security	✓	×	×	✓
Resilience to attacks *	×	×	×	✓

*Model inversion attack or Membership inference attack.

B. Accuracy

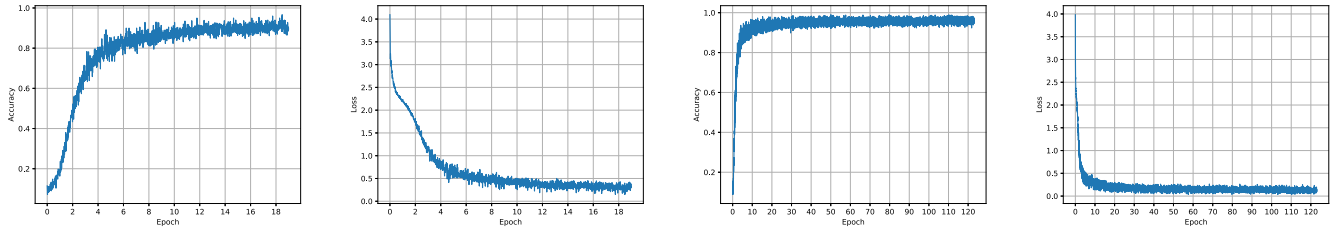
We implemented a convolutional neural network (CNN) and evaluated the performance of PEFL on MNIST dataset. The hidden layers of the CNN consist of two convolutional layers followed by max pooling layers and two fully connected layers. As the convolutional layer is transferable [23], we use an already trained convolutional layers and only retrain the fully connected layers. MNIST is a handwritten digits dataset that consists of 60 000 training examples and 10 000 testing examples. Each example consists of a 28 ×28 size gray-level image and a corresponded class. Our nonprivate baseline model is able to achieve accuracy of 99.3% after 150 epochs. For the sake of accounting privacy loss, we use the code of privacy _ accountant.[5]

Using the moment accountant method for the fixed $\delta = 10^{-5}$, we achieve $(0.5, 10^{-5})$ and $(2, 10^{-5})-$DP for noisy levels $\rho = 8$ and $\rho = 4$, respectively. As shown in Fig. 3, the network reaches 90.8% and 97.5% average accuracy for noisy levels $\rho = 8$ and $\rho = 4$, respectively. Additionally, we find that the accuracy of the neural network with small noise perturbations is more stable. This also means that the more noises added to the gradients, the slower loss function converges.

As shown in Fig. 4(a), for a fixed ε (0.5 or 2), changing the value of δ will have little effect on prediction accuracy. However, for a fixed δ, different privacy budget ε has a huge impact on prediction accuracy. For example, for $\varepsilon = 0.5$, the neural network achieved 90.8% and 91.5% accuracy for $\delta = 10^{-5}$ and $\delta = 10^{-3}$, respectively. But the neural network for fixed

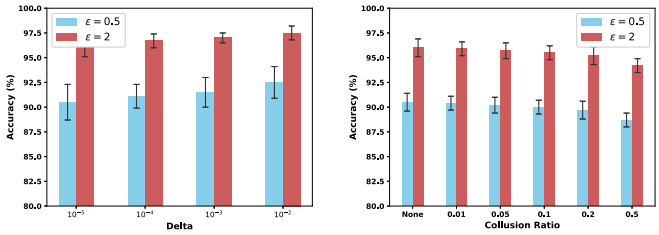
[4][Online]. Available: https://github.com/homenc/HElib

[5][Online]. Available: https://github.com/tensorflow/privacy



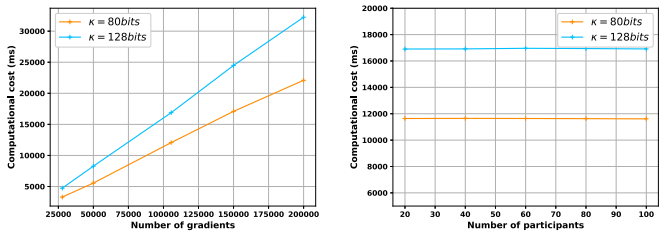
(a) (b) (c) (d)

Fig. 3. Accuracy and loss for different privacy budget on the MNIST dataset. (a) Accuracy ($\varepsilon = 0.5$). (b) Loss ($\varepsilon = 0.5$). (c) Accuracy ($\varepsilon = 2$). (d) Loss ($\varepsilon = 2$).



(a) (b)

Fig. 4. Accuracy for different delta values or collusion ratios on the MNIST dataset. (a) Accuracy for different δ. (b) Accuracy for different β.

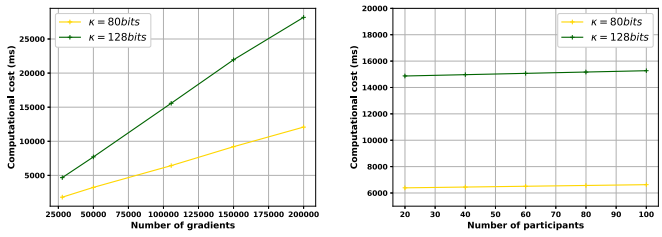


(a) (b)

Fig. 5. Computational cost for each participant. (a) Computational cost with the different number of gradients. (b) Computational cost with the different number of participants.

$\delta = 10^{-5}$ achieved 90.8% and 96.0% accuracy for $\varepsilon = 0.5$ and $\varepsilon = 2$, respectively.

As shown in Fig. 4(b), we compare the accuracy of neural network with different collusion ratios β. When a small part of participants are compromised by the adversary, there is little impact on accuracy. For example, for $\delta = 10^{-5}$ and $\varepsilon = 2$, the baseline neural network reaches accuracy of 96.0%. When collusion ratios are 0.05 and 0.1, the network achieves accuracy of 95.7% and 95.5%, respectively. However, our experiments demonstrate the accuracy of the neural network will be seriously degraded when the collusion ratio is greater than 0.5.



(a) (b)

Fig. 6. Computational cost for the CS. (a) Computational cost with the different number of gradients. (b) Computational cost with the different number of participants.

C. Computational Complexity

We only consider the running time of the ciphertext operation because this is our main contribution. Considering security and correctness requirements, we set $\phi(n) \geq \frac{1}{7.2}(\kappa + 110) \cdot \ln(\frac{p_1}{\sigma})$. We select plaintext space $q = 2^{16}$ and security parameter $\kappa = 80$ b and $\kappa = 128$ b. Computational costs for participants and the CS are plotted in Figs. 5 and Fig. 6, with different lines representing different size of security parameter.

As shown in Fig. 5(a) and (b), the computational cost of each participant increases linearly with the number of gradients, but does keep constant when the number of participants increases. Because our PEFL independently encrypts $\phi(n)$ sized block at each aggregation, the number of encryption increases linearly with the magnitude of the gradients, rather the number of participants. Therefore, the computational overhead of encryption is only related to the number of gradients, regardless of the number of participants. In addition, increased security (e.g., high security parameter κ) reduces efficiency. The computational overhead with $\kappa = 128$ is about 1.45 times slower than the one with $\kappa = 80$.

In the case of the server, Fig. 6(a) shows that the computational cost of the CS increases with the number of gradients. That is because, our PEFL independently decrypts $\phi(n)$ sized block at each aggregation, the number of decryption process increases linearly with the magnitude of the gradients. But, as shown in Fig. 6(b), the computational cost does not change significantly with the number of participants. That is because the number of participants will only affect the evaluation during the decryption process. In the evaluation process, it is only required to perform a simple addition operation. Compared with decryption, the computational cost of evaluation is essentially negligible. Therefore, our solution is suitable for large-scale user scenarios. In addition, for the CS, the computational overhead with $\kappa = 128$ is about 2.33 times slower than the one with $\kappa = 80$.

TABLE II
COMMUNICATION COST FOR DIFFERENT PROCESS

Key distribution	KGC to Pa	$\phi(n^{'})\lceil\log_2 p_1\rceil$
	KGC to CS	$\phi(n)\lceil\log_2 p_1\rceil + \phi(n^{'})\lceil\log_2 p_1\rceil$
Secure aggregation	Pa to CS	$\lvert E\rvert\lvert I\rvert\phi(n^{'})\lceil\log_2 p_1\rceil$
	CS to Pa	$\lvert E\rvert\lvert I\rvert\phi(n)\lceil\log_2 q\rceil$

D. Communication Complexity

We show the communication overhead among KGC, the CS, and participants (Pa) in Table II. In the key distribution phase, KGC communicates with each participant i and the CS to distribute secret key $s_i \in R^{'}_{p_1}$ and $(\mathrm{SK}_{C_1}, \mathrm{SK}_{C_2}) \in R^{'}_{p_1} \times R_{p_1}$, respectively. Key distribution is only performed once during the entire training process. In the secure aggregation phase, we assume our training consists of $|E|$ epochs, each of which has $|I|$ iterations. In each aggregation participant i uploads local gradients' ciphertext $c_i \in R^{'}_{p_1}$ to the CS and downloads shared parameters gd^t from the CS at each aggregation. Therefore, the increased communication factor is $\frac{\phi(n^{'})\lceil\log_2 p_1\rceil}{\phi(n)\lceil\log_2 q\rceil}$ during upload phase, and none of increased factor during download phase.

VII. RELATED WORK

A. Privacy Threats in DL

Despite extensive applications and superior advantages of DL, many researchers have shown that adversaries can steal users' private information from neural network. Fredrikson *et al.* [16] designed the model inversion attacks against neural networks. By accessing the application programming interface (API) of a facial recognition system, attackers can recover the facial image of a user participating in the training process. Additionally, Shokri *et al.* proposed the membership inference attacks [37] against neural networks. Given access to the black-box model, a malicious adversary can launch attacks to infer whether a data record is in the training dataset of the victim model. That seriously violates privacy in special circumstances, such as medical systems or financial systems. However, these attacks will not work for our scheme. Because example-level DP guarantees that any data record in the training set or not in the training set has little effect on the prediction model. In addition, Hitaj *et al.* [14] designed an attack against collaborative learning by using generative adversarial network. In their scheme, a dishonest participant (internal adversary) launches the attack during the training process, where the adversary participates in the collaborative training, forces other participants to disclose more information and utilizes GAN to recover private data. However, different from its adversary setting, in PEFL all participants are honest-but-curious.

B. Privacy-preserving DL

To mitigate the above threats, several privacy-preserving DL schemes are proposed. Shokri and Shmatikov [18] proposed the first privacy-preserving collaborative learning by selectively sharing partial gradients to guarantee the privacy of training data. Unfortunately, even if a small part of the gradients is uploaded to the CS, participant's private information may be leaked [19]. Phong *et al.* [19] proposed a collaborative DL model via additively HE. However, all participants share the same decryption key, which is vulnerable considering collusion between participants and the CS. Moreover, by using SMC, Bonawitz *et al.* designed a secure aggregation scheme in federated DL model. However, these schemes will still suffer from serious privacy threats from the shared parameters.

On the other hand, unlike our training method, several privacy-preserving centralized training schemes are proposed utilizing DP. Abadi *et al.* [23] adopt the Gaussian mechanism to perturb gradients and provide a tighter method (e.g., moment accountant) to keep track of entire privacy loss. To alleviate the privacy leakage, Phan *et al.* [38] achieve learning autoencoder with DP by perturbing loss function. In addition, Mohassel and Zhang [39] proposed a two-server training model utilizing secure two-party computation, where sensitive data will be separated into two parts and sent to the two noncollusive servers. Besides, Chan *et al.* [40] and Bonawitz *et al.* [20] address the problem of dropouts, e.g., completing the protocol even if a user drops out. However, this is beyond the scope of our research and we expect the method developed by Chan *et al.* [40] would also be applied directly to our solution.

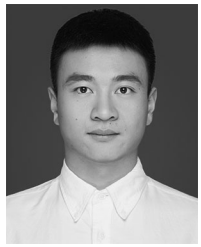
VIII. CONCLUSION

In this article, we had proposed an efficient and PEFL scheme for IAI. Compared with existing solutions, our PEFL can prevent privacy leakage from the local gradients as well as the shared parameters. PEFL is noninteractive in each aggregation and provides high privacy-protection level even when an adversary colludes with multiple honest entities. We prove that PEFL provides postquantum security and guarantees aggregation oblivious security considering collusion attacks. Performance evaluation indicates that PEFL has practical accuracy on the MNIST dataset and we expect to study on the high-dimensional dataset and complex neural network in immediate future work.

REFERENCES

[1] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, and E. Hossain, "Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains," *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 3154–3164, Dec. 2017.
[2] Y. Dai, D. Xu, S. Maharjan, Z. Chen, Q. He, and Y. Zhang, "Blockchain and deep reinforcement learning empowered intelligent 5 g beyond," *IEEE Netw.*, vol. 33, no. 3, pp. 10–17, May/Jun. 2019.
[3] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Deep learning empowered task offloading for mobile edge computing in urban informatics," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7635–7647, Oct. 2019.
[4] K. Zhang, Y. Zhu, S. Maharjan, and Y. Zhang, "Edge intelligence and blockchain empowered 5G beyond for industrial Internet of Things," *IEEE Netw.*, vol. 33, no. 5, pp. 12–19, Sep./Oct. 2019.
[5] H. Li, X. Lin, H. Yang, X. Liang, R. Lu, and X. Shen, "EPPDR: An efficient privacy-preserving demand response scheme with adaptive key evolution in smart grid," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 8, pp. 2053–2064, Aug. 2014.
[6] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, 2017, pp. 2174–2182.

[7] H. Ren, H. Li, Y. Dai, Y. Kan, and X. Lin, "Querying in Internet of Things with privacy preserving: Challenges, solutions and opportunities," *IEEE Netw.*, vol. 32, no. 6, pp. 144–151, Nov./Dec. 2018.
[8] G. Xu, H. Li, Y. Dai, K. Yang, and X. Lin, "Enabling efficient and geometric range query with access control over encrypted spatial data," *IEEE Trans. Inf. Forensics Sec.*, vol. 14, no. 4, pp. 870–885, Apr. 2019.
[9] H. Li, D. Liu, Y. Dai, T. H. Luan, and X. S. Shen, "Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage," *IEEE Trans. Emerg. Topics Comput.*, vol. 3, no. 1, pp. 127–138, Mar. 2015.
[10] G. Xu, H. Li, S. Liu, M. Wen, and R. Lu, "Efficient and privacy-preserving truth discovery in mobile crowd sensing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3854–3865, Apr. 2019.
[11] H. Li, D. Liu, Y. Dai, and T. H. Luan, "Engineering searchable encryption of mobile cloud networks: when QoE meets QoP," *IEEE Wireless Commun.*, vol. 22, no. 4, pp. 74–80, Aug. 2015.
[12] H. Li, Y. Yang, Y. Dai, J. Bai, S. Yu, and Y. Xiang, "Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2017.2769645.
[13] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," *Google Res. Blog*, vol. 3, 2017.
[14] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM Comput. Commun. Secur.*, 2017, pp. 603–618.
[15] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: Secure and verifiable federated learning," *IEEE Trans. Inf. Forensics Sec.*, vol. 6, pp. 911–926, Jul. 2019.
[16] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. ACM Comput. Commun. Secur.*, 2015, pp. 1322–1333.
[17] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, "HealthDep: An efficient and secure deduplication scheme for cloud-assisted ehealth systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 9, pp. 4101–4112, Sep. 2018.
[18] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. ACM Comput. Commun. Secur.*, 2015, pp. 1310–1321.
[19] Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Sec.*, vol. 13, no. 5, pp. 1333–1345, May 2018.
[20] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM Comput. Commun. Secur.*, 2017, pp. 1175–1191.
[21] X. Zhang, S. Ji, H. Wang, and T. Wang, "Private, yet practical, multiparty deep learning," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 1442–1452.
[22] M. Hao, H. Li, G. Xu, S. Liu, and H. Yang, "Towards efficient and privacy-preserving federated deep learning," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–6.
[23] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM Comput. Commun. Secur.*, 2016, pp. 308–318.
[24] W. Jiang, H. Li, G. Xu, M. Wen, G. Dong, and X. Lin, "Ptas: Privacy-preserving thin-client authentication scheme in blockchain-based PKI," *Future Gener. Comput. Syst.*, vol. 96, pp. 185–195, 2019.
[25] H. Li, D. Liu, Y. Dai, T. H. Luan, and S. Yu, "Personalized search over encrypted data with efficient and secure updates in mobile clouds," *IEEE Trans. Emer. Topics Comput.*, vol. 6, no. 1, pp. 97–109, Jan./Mar. 2018.
[26] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. 3rd Conf. Theory Cryptogr.*, 2006, pp. 265–284.
[27] C. Dwork *et al.*, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3–4, pp. 211–407, 2014.
[28] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, 2009, Art. no. 34.
[29] R. El Bansarkhani, Ö. Dagdelen, and J. Buchmann, "Augmented learning with errors: The untapped potential of the error term," in *Proc. 19th Int. Conf. FC*, 2015, pp. 333–352.
[30] D. Becker, J. Guajardo, and K.-H. Zimmermann, "Revisiting private stream aggregation: Lattice-based PSA," in *Proc. Nat. Down Syndrome Soc.*, 2018.
[31] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *ACM Trans. Comput. Theory*, vol. 6, no. 3, 2014, Art. no. 13.
[32] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Proc. EUROCRYPT*, 2010, pp. 1–23.
[33] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the AES circuit," in *Proc. CRYPTO*, 2012, pp. 850–867.
[34] H. Cramér, "Über eine eigenschaft der normalen verteilungsfunktion," *Mathematische Zeitschrift*, vol. 41, no. 1, pp. 405–414, 1936.
[35] A. Beimel, S. P. Kasiviswanathan, and K. Nissim, "Bounds on the sample complexity for private learning and private data release," in *Proc. Trans. Cloud Comput*, 2010, pp. 437–454.
[36] G. Xu, H. Li, H. Ren, K. Yang, and R. H. Deng, "Data security issues in deep learning: Attacks, countermeasures and opportunities," *IEEE Commun. Mag.*, vol. 57, no. 11, pp. 116–122, Nov. 2019.
[37] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Sec. Privacy*, 2017, pp. 3–18.
[38] N. Phan, Y. Wang, X. Wu, and D. Dou, "Differential privacy preservation for deep auto-encoders: An application of human behavior prediction," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 1309–1316.
[39] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *Proc. IEEE Symp. Sec. Privacy*, 2017, pp. 19–38.
[40] T.-H. H. Chan, E. Shi, and D. Song, "Privacy-preserving stream aggregation with fault tolerance," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2012, pp. 200–214.



Meng Hao (S'19) received the B.S. degree in information security in 2018, from the University of Electronic Science and Technology of China, Chengdu, China, where he is currently working toward the master's degree in computer technology with the School of Computer Science and Engineering.

His research interests include applied cryptography and privacy-preserving deep learning.

Hongwei Li (M'12–SM'18) received the Ph.D. degree in computer software and theory from the University of Electronic Science and Technology of China, Chengdu, China, in June 2008.

He is currently the Head and a Professor with the Department of Information Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. He worked as a Postdoctoral Fellow with the University of Waterloo, Waterloo, ON, Canada, from October 2011 to October 2012 under the supervision of Prof. S. Shen. He has authored or coauthored more than 80 technical papers. His research is supported by National Science Foundation of China, and Ministry of Science and Technology of China, and Ministry of Industry and Information Technology, and China Unicom. He is the sole author of a book *Enabling Secure and Privacy Preserving Communications in Smart Grids* (Springer, 2014). His research interests include network security and applied cryptography.

Dr. Li was the Associate Editor of the IEEE INTERNET OF THINGS JOURNAL, and *Peer-to-Peer Networking and Applications*, the Guest Editor of the IEEE NETWORK and IEEE INTERNET OF THINGS JOURNAL. He also serves/served the Technical Symposium Co-Chair of Association for Computer Machinery Turing Celebration Conference—China 2019, IEEE International Conference on Computer and Communications 2016, IEEE Global Communications Conference 2015 and IEEE BigDataService 2015, and many technical program committees for international conferences, such as IEEE Conference on Computer Communications, IEEE International Conference on Communications and IEEE Global Communications Conference. He was a recipient of the Best Paper Award from IEEE International Conference on Mobile Ad-hoc and Sensor Systems 2018 and IEEE International Conference on E-health Networking, Application Services 2015. He is the Distinguished Lecturer of IEEE Vehicular Technology Society.

Xizhao Luo received the B.S. degree in information and computational science and the M.S. degree in applied mathematics from the Xian University of Technology, Xian, China, in 2000 and 2003, respectively, and the Ph.D. degree in computer application from Soochow University, Suzhou, China, in 2010.

He held a Postdoctoral position with the Center of Cryptography and Code, School of Mathematical Science, Soochow University, for three years. His research interests include cryptography and computational complexity.

Guowen Xu (S'15) received the B.S. degree in information and computing science from the Anhui University of Architecture, Hefei, China, in 2014. He is currently working toward the Ph.D. degree in cyber security with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China.

His research interests include cryptography, searchable encryption, and the privacy-preserving deep learning.

Haomiao Yang (M'13) received the M.S. and Ph.D. degrees in computer applied technology from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2004 and 2008, respectively.

He was a Postdoctoral Fellow with the Research Center of Information Cross over Security, Kyungil University, Gyeongsan, South Korea, for one year until June 2013. He is currently an Associate Professor of Cyber Security with the School of Computer Science and Engineering and Center for Cyber Security with UESTC. His research interests include cryptography, cloud security, and the cyber security for aviation communication.

Sen Liu received the B.S. degree in information security from Guizhou University, Guiyang, China, in 2017. He is currently working toward the master's degree in computer technology with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China.

His research interests include cryptography and searchable encryption.