Efficient Privacy-Preserving Federated Learning with Unreliable Users

Yiran Li, *Student Member, IEEE*, Hongwei Li (Corresponding author), *Senior Member, IEEE*, Guowen Xu, *Member, IEEE*, Xiaoming Huang, *Member, IEEE*, Rongxing Lu, *Fellow, IEEE*

***Abstract*—Federated learning (FL) has emerged as a powerful technology widely applied in Internet-of-Things (IoT). Recently, researchers have shown an increased interest in privacy-preserving FL with *unreliable users*. The goal of such works is to achieve private training under ciphertext mode while ensuring that the FL model is mainly derived from the contributions of users with high-quality data. However, the existing work is still in its infancy, and the main challenge faced by many researchers is how to achieve their schemes for meeting the demands of high accuracy and efficiency. To combat that, we propose an efficient privacy-preserving federated learning (EPPFL) scheme with *unreliable users*. Specifically, we design a novel scheme to mitigate the negative impact of *unreliable users*, where the targeted model is guaranteed to be updated with high-quality data. Through iteratively executing our "*Excluding Irrelevant Components*" and "*Weighted Aggregation*", the FL model converges rapidly while taking limited communication and computation overhead. As a result, not only the model accuracy can be optimized, but also the training efficiency can be improved. Meanwhile, we conduct a secure framework based on threshold Paillier cryptosystem, which can rigorously protect all user-related private information during the training process. Furthermore, extensive experiments demonstrate our EPPFL with high-level performance in terms of accuracy and efficiency.**

***Index Terms*—Federated Learning, Privacy-Preserving, Unreliable Users.**

I. Introduction

Deep learning (DL) has attracted a lot of interest in both academia and industry while being widely exploited in Internet-of-Things (IoT) [1], [2]. However, when traditional centralized DL meets IoT, the privacy issue gradually emerged as an obstacle hindering the proliferation of deep learning in IoT. Specifically, centralized learning requires edge users to provide the service center with raw data, which usually contains some private information, such as home address, identity information, etc. Once the center is corrupted, users' privacy would not be guaranteed anymore. For addressing the privacy issue, federated learning (FL) seems more powerful, since users' local original data is not shared to the center, instead, the gradient information, generated through training on users' local data, will be uploaded. Such a mechanism can not only protect users privacy to a certain extent but also minimize the convergence rate and optimize the model accuracy. Due to this, FL has promised IoT with enormous vertical applications, such as smart city, smart home, e-healthcare, and so on. Undoubtedly, FL has achieved remarkable success in IoT.

However, applying FL into IoT still faces some practical obstacles. One of the most important issues is still the privacy protection. State-of-the-art research [3] has specified that even if a user only uploads gradient information, the user's privacy may still be leaked. An adversary may corrupt the central server, revealing some properties of the training samples, and even completely recovering the original data through the leaked gradient information. Besides, IoT is a complex ecosystem, usually consisting of enormous edge users, cascade network, and service center. It is really hard to guarantee that all these entities perform smoothly all the time. When FL is implemented in IoT, the instability of equipment and irregular operations may ease users to generate local data with low-quality. Training with these data will negatively impact the model accuracy, lower convergence rate, and even cause the model divergence[1]. This phenomenon has also been found in recent studies [4], [5]. Thereby, it is essential to implement an appropriate method to handle these *unreliable users*. On the other hand, IoT applications are usually required to perform with low resource overhead and high processing efficiency. However, FL takes distributed training method, requiring users to iteratively upload tens of thousands of gradient parameters to the central server. It is an insurmountable burden for terminal equipment with limited resources. More seriously, this phenomenon is more obvious under ciphertext because the size of the ciphertext is usually several times that of the plaintext. Therefore, to improve the practicality and security, it is essential to design an efficient distributed learning framework that provides privacy-preserving FL with robustness to unreliable users.

To our best knowledge, only two recent works [6], [7] have been proposed to alleviate the impact caused by *unreliable users*. *Zhao et al.* [6] proposed the first approach named SecProbe. The main idea of SecProbe is to perturb the loss function of DNNs model through exploiting the technology of differential privacy (DP). Their scheme can achieve a good trade-off between accuracy and privacy, based on the

Yiran Li, and Hongwei Li are 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: yiranli842@foxmail.com; hongweili@uestc.edu.cn)

Guowen Xu is with the School of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore, (e-mail: guowen.xu@ntu.edu.sg)

Xiaoming Huang is with the CETC Cyberspace Security Research Institute Co., Ltd., Chengdu 610041, China, (e-mail: apride@gmail.com)

Rongxing Lu is with the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada, (e-mail: rlu1@unb.ca)

[1]In this paper, users who provide low-quality data will be considered as *unreliable users*

custom data detection mechanism [8]. However, state-of-the-art researches [9], [10] have proven that the challenge faced current DP-based privacy-preserving deep learning still exists, especially for balancing the utility and privacy in complicated learning missions. Therefore, under our setting, utilizing differential privacy to construct the secure framework may still cause privacy leakage, if high-level model accuracy is guaranteed. Recently, *Xu et al.* [7] proposed a PPFDL scheme, where they created a method of $\texttt{Meth}_{\texttt{IU}}$ to alleviate the negative impact of *unreliable users*, while protecting the privacy of all user-related information through secure two-party computation (2-PC). However, for achieving their PPFDL, two servers are required to cooperate with each other, but cannot collude with each other. Once the collusion occurs, the entire security mechanism will be destroyed. Thereby, their scheme is not practical in real-world scenarios.

Beyond the above-mentioned limitations, the existing efforts have not considered a fundamental problem, that is, how to reduce the overhead of edge users under ciphertext. As described above, it is highly required to free the terminal equipments from the trouble of resources, if the FL is guaranteed to run smoothly in IoT environments. Especially under ciphertext mode, it is crucial to reduce the edge users overhead. For addressing the above issues, we propose the EPPFL, an efficient and Privacy-Preserving Federated Learning scheme, which can efficiently handle users' low-quality data in IoT. We summarize the contributions of our EPPFL as follows:

- We conduct an efficient and secure aggregation framework in the FL training process. Based on the masterly utilization of the threshold Paillier cryptosystem, the framework can strictly protect the privacy of all users' related information, including gradients and the reliability value of each gradient component.
- We propose a novel scheme (called $\texttt{Sch}_{\texttt{UU}}$) to alleviate the negative impact caused by unreliable users. Through iteratively executing our "*Excluding Irrelevant Components*" and "*Weighted Aggregation*", the FL model converges rapidly with high-level accuracy while taking low overhead, which can be very friendly for edge-users with limited hardware resources.
- We present rigorous security proof for demonstrating the high-level security of our EPPFL. Besides, we conduct extensive experiments to specify the preferable performance of EPPFL in terms of accuracy and efficiency.

The remaining parts of this paper are organized as follows. We specify our research problem and review some primitives in Section II. In Section III, we introduce our scheme in detail. Then, we give a rigorous security proof in Section IV and a comprehensive performance analysis in Section V. Next, we discuss related works in Section VI. Finally, we draw our conclusions in Section VII.

II. Problem Statement and Primitives

A. System Overview

As shown in Fig. 1, two universal entities are considered in our model, i.e., *the cloud server* and *users*, who cooperate with each other to achieve the collaborative FL training [11].

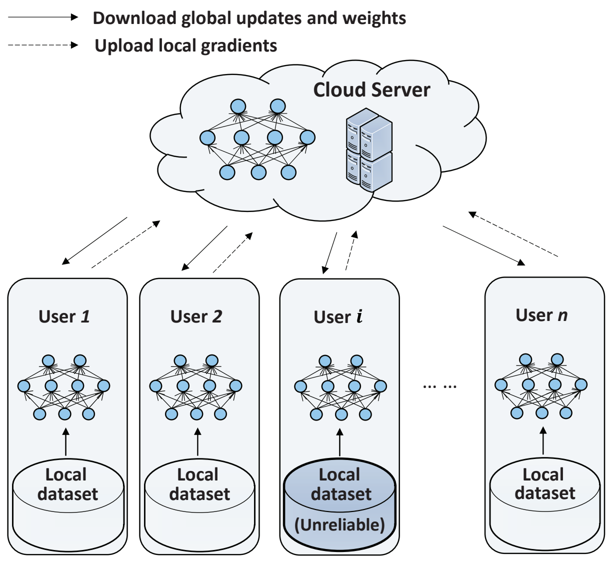


Fig. 1: System Overview

To be specific, all participants initially agree on a unified DNN model with the initialized parameters of weights. Then, in every iteration, each user calculates the local gradient through training with its local dataset, and then uploads its local gradient to the cloud server. After the cloud server achieves the aggregation of the uploaded gradients, a global gradient is generated, based on which the global DNN model is updated. The above operations are executed iteratively until the DNN model satisfies the convergence certification. For protecting each user's privacy during the process, a secure framework is conducted in our EPPFL based on threshold Paillier cryptosystem, which can strongly ensure that all operations on local gradients are performed under ciphertext mode except for some local operations.

Notes: We just consider only one server $\boldsymbol{S}$ in our system, which serves as a platform for mainly achieving the secure aggregation, i.e., one of its significant tasks is to calculate a summation of the encrypted data uploaded from users. Such a point-to-multipoint setting has been widely adopted in previous works [12], [13], [6] on privacy-preserving FL. Through integrating with the "Secure Aggregation Protocol (SAP)" proposed in our EPPFL, this framework provides us three outstanding advantages: (i) all the secure aggregation operations are outsourced to the cloud server $\boldsymbol{S}$, which can greatly alleviate users' pressure on computation, (ii) based on the masterly exploitation of threshold Paillier cryptosystem, our "SAP" only needs two rounds of interactions for obtaining the final aggregation result, which can minimize the communication overhead between users and the cloud server, and (iii) recent solutions [14], [15], [16] based on dual-server or multi-server have also been proposed for addressing the privacy issues in FL. However, all of their schemes require that at least two servers are not allowed to collude with each other, which is not practical in real-world scenarios. Our EPPFL, utilizing a single-server setting, can effectively avoid this issue, being more practical in actual applications. Meanwhile, base on the secure framework in our EPPFL, a certain amount of users are allowed to collude with the cloud server, while users' privacy can still be guaranteed.

Additionally, we consider a significant issue of *unreliable users* during the FL training process. Under the consideration, users learn a unified FL model with their local datasets, and most of them are *reliable*, holding similarly high-quality data, but a small portion of them is *unreliable* (e.g., user m shown in Fig. 1), training with low-quality data. The data held by these *unreliable users* may be not precise compared with others, so that the gradients they conduct may deviate far from the convergence trend. In real-world IoT applications, users may obtain data through their sensors. However, due to the quality discrepancy of sensors, there usually exist *unreliable users* in an FL-based IoT. Considering the scenario where an Internet-of-Vehicles (IoV) company intends to train a unified DNNs model based on federated learning, thereby utilizing this model for intelligently controlling vehicles. In real world, vehicles of different brands may exploit various sensors to collect vehicle information (e.g., speed, position, etc). The most vehicles are usually equipped with high-quality sensors, which are sensitive and highly precise, while some little-known vehicles may be installed with low-quality sensors for some potential reasons, such as saving costs, etc. As a result, the utilization of the model generated with low-quality data may cause serious mistakes (e.g., the vehicle is driven to deviate from the targeted direction, and even the accelerator is misused as a brake, etc).

In this paper, we propose a novel scheme to alleviate the negative impact of unreliable users for improving model accuracy and minimizing convergence rate. Since redundant irrelevant gradient components are excluded from the gradient being uploaded, and the remaining components are appropriately aggregated, our scheme is friendly to edge users in terms of computation and communication overhead. We will introduce the detailed method in Section III.

B. Threat Model and Goals

During the FL training process, the security threats are mainly derived from the inner participants (i.e., the cloud server and users). For acquiring users' private information, the cloud server may try to infer the gradient information of each user. Similarly, the same behavior may also occur among users. Therefore, it is quite important to protect the privacy of users' gradient information. Furthermore, in order to guarantee the non-discriminative training process, the reliability information of each user should also be guaranteed from being exposed to any party (including the user himself). In our EPPFL, we make an assumption that both the cloud server and users are *honest-but-curious*, where each participant strictly abides by the out established protocol to accomplish the training task. However, some of them may try to exploit mastered prior knowledge for compromising other ones' data privacy. Additionally, in our model, a certain number (less than the threshold value) of users are permitted to collude with the cloud server. Based on the threat model, we formally present our privacy demands as follows.

- *Privacy of gradient information*: Each user's gradient information, being potentially utilized to recover the user's private information (e.g., driving route, position, etc.), may be leaked to adversaries, i.e., the cloud server and other users. For protecting users' privacy, it is essential to encrypt each user's local gradient, and operate them under ciphertext mode, except for some local operations.
- *Confidentiality of reliability information*: For guaranteeing fairness and non-discrimination in the training process, each user's reliability information, denoting the quality of user's data, should also be protected against the cloud server and all users.

C. Federated Learning

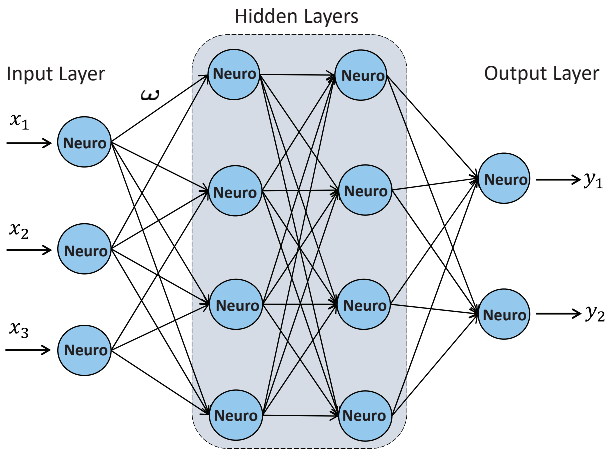


Fig. 2: A typical fully connected neural network

1) Neural Network: Deep neural networks (DNNs) promise various vertical services due to their heterogeneous network structures. Without loss of generality, the fully connected neural network (FCNN), as a representative of DNNs, is introduced in this paper. As shown in Fig. 2, the left-to-right pipeline structure is constituted by one input layer, one output layer, and some hidden layers, where neurons between two adjacent layers are fully connected by the weight $\boldsymbol{\omega}$. Given a data set $(\boldsymbol{x}, \boldsymbol{y})$ labeled by $\boldsymbol{y}$, the DNN can be described as a function: $f(\boldsymbol{x}, \boldsymbol{\omega}) = \hat{\boldsymbol{y}}$, where the input is $\boldsymbol{x} = \{x_1, x_2, x_3\}$, and the output is $\hat{\boldsymbol{y}} = \{\hat{y}_1, \hat{y}_2\}$. The main idea for training a DNN is to adjust the weight $\boldsymbol{\omega}$ for minimizing the distance between the output $\hat{\boldsymbol{y}}$ and the real label $\boldsymbol{y}$. This could be considered as a nonlinear optimization problem, usually addressed by mini-batch stochastic gradient descent (SGD) [17] described as follows.

According to the pseudo-code shown in Alg. 1, given a training data set $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i); i = 1, 2..., \mathcal{N}\}$, the loss function can be defined as $\mathcal{L}_f(\mathcal{D}, \boldsymbol{\omega}) = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \mathcal{L}_f((\boldsymbol{x}_i, \boldsymbol{y}_i), \boldsymbol{\omega})$, where $\mathcal{L}_f((\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{\omega}) = ||\boldsymbol{y} - f(\boldsymbol{x}, \boldsymbol{\omega})||_2 = ||\boldsymbol{y} - \hat{\boldsymbol{y}}||_2$, and $||\cdot||_2$ denotes the 2 $norm$ of a vector.

In j-th iteration, firstly, a mini-batch $\mathcal{D}_j$ is randomly chosen from $\mathcal{D}$, hence the loss function can be defined as

$$\mathcal{L}_f(\mathcal{D}_j, \boldsymbol{\omega}_j) = \frac{1}{|\mathcal{D}_j|} \sum_{(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathcal{D}_j} \mathcal{L}_f((\boldsymbol{x}_i, \boldsymbol{y}_i), \boldsymbol{\omega}_j) \tag{1}$$

Algorithm 1 Stochastic Gradient Descent.

```
Input: Training data set $\mathcal{D}$={$(\boldsymbol{x}_i, \boldsymbol{y}_i); i$=1, 2..., $\mathcal{N}$},
    loss function $\mathcal{L}_f(\mathcal{D}, \boldsymbol{\omega})$=$\frac{1}{\mathcal{N}}\sum_{i=1}^{\mathcal{N}} \mathcal{L}_f((\boldsymbol{x}_i, \boldsymbol{y}_i), \boldsymbol{\omega})$,
    learning rate $\theta$.
Output: $\boldsymbol{\omega}$
1: Randomly choose an initial $\boldsymbol{\omega}$;
2: repeat
3:     In $j$-$th$ iteration, randomly choose $mini$-$\mathcal{D}_j \subseteq \mathcal{D}$;
4:     for each $(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathcal{D}_j$ do
5:         compute $g_j^{(\boldsymbol{x}_i, \boldsymbol{y}_i)} \leftarrow \nabla\mathcal{L}_f((\boldsymbol{x}_i, \boldsymbol{y}_i), \boldsymbol{\omega}_j)$
6:     end for
7:     compute $g_{*,j} \leftarrow \frac{1}{|\mathcal{D}_j|}\sum_{(\boldsymbol{x}_i, \boldsymbol{y}_i)\in\mathcal{D}_j} g_j^{(\boldsymbol{x}_i, \boldsymbol{y}_i)}$;
8:     update $\boldsymbol{\omega}_{j+1} \leftarrow \boldsymbol{\omega}_j - \theta \cdot g_{*,j}$
9: until An approximate minimum is obtained.
10: return $\boldsymbol{\omega}$;
```

where $\mathcal{L}_f((\boldsymbol{x}_i, \boldsymbol{y}_i), \boldsymbol{\omega}_j)$=$||\boldsymbol{y}_i - f(\boldsymbol{x}_i, \boldsymbol{\omega})||^2$.

Then, the gradient $g_j^{(\boldsymbol{x}_i, \boldsymbol{y}_i)}$ for a training pair $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ is obtained through calculating the partial derivative of $\mathcal{L}_f$ with respect to $\boldsymbol{\omega}_j$. After the global gradient $g_{*,j}$ is calculated through averaging the aggregated $\sum_{(\boldsymbol{x}_i, \boldsymbol{y}_i)\in\mathcal{D}_j} g_j^{(\boldsymbol{x}_i, \boldsymbol{y}_i)}$, the weight $\boldsymbol{\omega}$ is updated as:

$$\boldsymbol{\omega}_{j+1} \leftarrow \boldsymbol{\omega}_j - \theta \cdot g_{*,j} \tag{2}$$

For obtaining the approximate minimum, which means satisfying the convergence condition, the above processes will be executed iteratively.

For FL training, considering $\mathcal{M}$ participating users, each user $m\in$[1,$\mathcal{M}$] holds a local data set $\mathcal{D}^m$, and the total data set $\mathcal{D}$ can be defined as $\mathcal{D}$=$\cup_{m\in\mathcal{M}}\mathcal{D}^m$. In j-th iteration, a mini-batch $\mathcal{D}_j^m$ is randomly chosen by user m, and the total data set is $\mathcal{D}_j$=$\cup_{m\in\mathcal{M}}\mathcal{D}_j^m$. Next, each user m computes g_j^m =$\sum_{(\boldsymbol{x}_i, \boldsymbol{y}_i)\in\mathcal{D}_j^m} \nabla\mathcal{L}_f((\boldsymbol{x}_i, \boldsymbol{y}_i), \boldsymbol{\omega}_j)$, and uploads the g_j^m to the cloud server. Then the cloud server can update the weight as

$$\boldsymbol{\omega}_{j+1} \leftarrow \boldsymbol{\omega}_j - \theta \cdot \frac{\sum_{m\in\mathcal{M}} g_j^m}{\sum_{m\in\mathcal{M}} |\mathcal{D}_j^m|} \tag{3}$$

Finally, the cloud server broadcasts $\boldsymbol{\omega}_{j+1}$ to each user to update their local neural network. The above interactive process between the cloud server and users will be executed iteratively until the preset convergence condition is satisfied.

Notes: The Eqn. (3) shows us that, in j-th FL training iteration, the cloud server mainly focuses on aggregating all the gradients of g_j^m uploaded by each user m, calculating an average value $\frac{\sum_{m\in\mathcal{M}} g_j^m}{\sum_{m\in\mathcal{M}} |\mathcal{D}_j^m|}$, and then obtaining the global weight to update the unified DNN. However, the above process has no consideration of the quality of the data sent by users, i.e., actually some unreliable user may upload some data with low quality. In this paper, a novel method will be introduced for handling these unreliable users.

D. Threshold Paillier Cryptosystem

In our scheme, the (t, n)-threshold Paillier cryptosystem [18] is utilized for constructing our secure framework due to its three significant features: (i) asymmetric cryptosystem: public key and private key are different, where public key is used for encryption, while private key is for decryption, (ii) threshold property: only more than or equal to a certain number (i.e., t) of private key owners can decrypt the ciphertext, and (iii) additive homomorphic property: plaintext addition can be achieved by ciphertext multiplication. These above features can provide our scheme with abundant functionalities and adequate privacy protection.

In this asymmetric cryptosystem [18], the public key pk=(z, h) is open to all the participants, while the secret key is divided into n keys based on the method of key generation proposed in the paper [18], denoted as $(sk_1, sk_2, ..., sk_n)$ and privately kept by each unique user, where z=$(1 + h)$, and h =pq, where p,q are two $primes$.

For encrypting a plaintext g to obtain a ciphertext c, the module exponential operation is executed with the public key pk as follows,

$$c = E_{pk}(g) = z^g r^h mod\ h^2 \tag{4}$$

where r is a private random number belonging to the multiplicative group $\boldsymbol{Z}_{h^2}^*$, which consists of invertible elements of group $\boldsymbol{Z}_{h^2}$. According to the above Eqn. (4), the homomorphic properties of this cryptosystem can be described as,

$$\begin{aligned} c = E_{pk}(g_i + g_j) &= z^{g_i+g_j}(r_i r_j)^h mod\ h^2 \\ &= E_{pk}(g_i) \cdot E_{pk}(g_j) \end{aligned} \tag{5}$$

$$\begin{aligned} c = E_{pk}(a \cdot g_i) &= z^{a g_i} {r_i}^{ah} mod\ h^2 \\ &= E_{pk}(g_i)^a \end{aligned} \tag{6}$$

where g_i, g_j are two plaintexts to be encrypted, and r_i, r_j are private random number belonging to $\boldsymbol{Z}_{h^2}^*$.

The decryption process falls into two main steps. Firstly, the unique private secret key sk_i will be respectively utilized to conduct a secret share,

$$s_i = c^{2\Delta sk_i} mod\ h^2 \tag{7}$$

where $\Delta = n!$. Then t shares of s_i are combined, and the plaintext is calculated through Lagrange interpolation algorithm and the “extraction algorithm” in Ref. [18].

Please note that in this (t, n)-threshold cryptosystem, only more than or equal to t participants’ cooperation can decrypt a ciphertext c. Any single participant or even a group of at most $t - 1$ participants doesn’t have the complete private key, being unable to correctly decrypt the ciphertext.

III. Our Proposed Scheme

In this section, we introduce the detailed technology of our EPPFL, which enables the cloud server and users to conduct the encrypted Sch$_{\text{UU}}$ under ciphertext mode while achieving the expected privacy demands, i.e., privacy protection for the user’s private information of local gradient and reliability. We first introduce our proposed Sch$_{\text{UU}}$, then we describe our “Secure Aggregation Protocol”, a significant protocol, which is frequently invoked in our scheme. Finally, we present the detailed method of how to utilize the protocol for accomplishing each step of the EPPFL.

A. Scheme for Handling Unreliable Users

As mentioned in Section. II-C, the global FL model is obtained through collaborative training between the cloud server and users. Specifically, during each iteration of the training process, the global gradient of the model is updated by averaging the aggregated local gradients shared by users. The gradient is essentially a vector constituted by components in multiple dimensions, and together these components determine the direction of the gradient. For minimizing convergence rate and improving model accuracy, it is essential to find a metric to generate the optimal global gradient in each iteration, in this paper, it is named as *ideal gradient*, which will be obtained by optimal operations (not just averaging) on the local gradients.

Indeed, for obtaining the *ideal gradient*, the issue of *unreliable users* has to be considered. As far as we know, *unreliable users* may achieve the training with the data of low-quality, and the existing works [19], [20] show that the direction of the gradient generated by *unreliable users* may deviate far from the collaborative convergence trend, where two significant properties of the gradient may exist: (i) in some dimensions of the gradient, the signs of components may be inconsistent with the signs of these components in the *ideal gradient*, which should be considered as *irrelevant*, and (ii) huge discrepancy in values may exist between local components and components in the *ideal gradient*. Both of these two factors will impair the model training, resulting in slower convergence and even divergence. Therefore, for mitigating the impact of *unreliable users* to generate the global *ideal gradient*, our first intuition is to exclude the *irrelevant components*. Then for addressing the discrepancy, we tend to first find an metric to estimate the reliability of remaining components, and then obtain a weighted aggregation result. In this paper, we propose a scheme named $\mathtt{Sch}_{\mathtt{UU}}$, which mainly consists of two parts: **`Excluding Irrelevant Components`** and **`Weighted Aggregation`**, which will be executed iteratively for generating the *ideal gradient*, introduced as follows.

In our settings, the data held by each user is independent and identically distributed (IID). Specifically, we consider that there are $\mathcal{M}$ users participating in the FL training, and after performing SGD on the local data set, each user obtains a gradient containing $\mathcal{L}$ components denoted as g_m=$(g_m^1, g_m^2, ..., g_m^l, ..., g_m^{\mathcal{L}})$, accordingly, the global *ideal gradient* is defined as g_*=$(g_*^1, g_*^2, ..., g_*^l, ..., g_*^{\mathcal{L}})$. During the FL training, each local gradient g_m will be uploaded to the cloud server, being aggregated to obtain the *ideal gradient* g_* for updating the global DNN. Based on our settings, these two significant parts of $\mathtt{Sch}_{\mathtt{UU}}$ are described in detail.

1) Excluding Irrelevant Components: As discussed above, the signs and values of the components in a gradient determines the direction of the gradient, especially, the sign of a component determines whether the model should be increased or decreased along the dimension of the component. Based on this, our first intuition for excluding *irrelevant components* is to compare the signs of components between local gradient and *ideal gradient*. On the other hand, during FL training process, the global *ideal gradient* is updated in each iteration, so that another issue is how to choose the *ideal gradient*. Intuitively, we can compare local gradient with the global *ideal gradient* in the current iteration. However, the global *ideal gradient* is generated on the aggregated result, which means the comparison will be achieved after all gradients being uploaded by users, so that the process would be full of redundancy and work inefficiently. According to the research [21], we know that the global training model actually converges steadily and smoothly, which means the discrepancy of gradients between two sequential iterations is small. Thereby, it is possible to measure the relevance by comparing the gradients between two sequential iterations, that is, each user can compare the signs of components between their local gradients and the previous global *ideal gradient*.

Algorithm 2 Excluding Irrelevant Components.

```
Input: local gradient $g_m$=$(g_m^1, g_m^2, ..., g_m^l, ..., g_m^{\mathcal{L}})$,
       global gradient $g_*$=$(g_*^1, g_*^2, ..., g_*^l, ..., g_*^{\mathcal{L}})$.
1: Initialize global gradient $g_*$;
2: In $j$-$th$ iteration, given the global gradient $g_{*,j-1}$;
3: for each $g_{m,j}^l \in g_{m,j}$ do
4:     if $I(g_{m,j}^l, g_{*,j-1}^l)$=0 then
5:         $g_{m,j}^l \leftarrow null$;    ▷ exclude irrelevant components
6:     end if
7: end for
```

Specifically, assuming each user m holds a local gradient of g_m. Given a global *ideal gradient* g_*, we introduce the algorithm of "Excluding Irrelevant Components (EIC)" through the pseudo-code in Alg. 2. We first define a function of $I(g_m^a, g_m^b)$ to compare the signs of two components of g_m^a and g_m^b as follows,

$$I(g_m^a, g_m^b) = \begin{cases} 1, & sgn(g_m^a) = sgn(g_m^b) \\ 0, & else \end{cases} \tag{8}$$

where $sgn(\cdot)$ is a sign extraction function to obtain the sign of the component. Then we utilize $I(g_m^a, g_m^b)$ to compare the signs of each g_m^l and the global g_*^l. If the result is **0**, the component g_m^l will be identified as irrelevant, being assigned an invalid value.

Notes: The "EIC" process will be performed locally, i.e., each user will compare their local gradients with the global *ideal gradient* to determine whether to exclude the local components. Hence, there is no need to consider users' privacy issues during the process. In addition, during the aggregation process, the cloud server cannot complete the summation until all users' gradients have been uploaded. Due to this, for ensuring the robustness of the subsequent aggregation process, the irrelevant components will not be just imprudently eliminated, but still be uploaded and labelled invalid. Moreover, during the practical training process, most of the data can be considered with high quality, so even if we exclude some *irrelevant components*, the model can still keep convergency. This can also be specified through our experiment results in Section. V.

2) Weighted Aggregation: "EIC" has shown us how to exclude irrelevant components, in this section, we introduce how to achieve weighted aggregation for further generating the *ideal gradient* in each iteration. The *ideal gradient* is intuitively an aggregated result obtained through the gradients uploaded from users, and each component in the *ideal gradient* is also an aggregation result, i.e., if we obtain all the components in the *ideal gradient*, we can construct the *ideal gradient*. To be specific, the process of weighted aggregation can be divided into two parts as follows:

(i) Reliability Estimation: For obtaining the *ideal component*, we first estimate the reliability of each component uploaded by users. Given the estimated aggregated *ideal component* of g^l_*, each user's reliability $\mathcal{R}_m$ can be updated as follows:

$$\mathcal{R}^l_m = log(\frac{\sum_{m=1}^{m=\mathcal{M}} dis(g^l_m, g^l_*)}{dis(g^l_m, g^l_*)}) \tag{9}$$

where $dis(a,b)$ is for measuring the distance between two values of a and b, denoted as a squared distance function:

$$dis(g^l_m, g^l_*) = (g^l_m - g^l_*)^2 \tag{10}$$

According to existing works [21], [22], the results show that even if IID data is utilized for training DNN by users, disparate data will generate different gradient components, which may cause the discrepancy in the scalars and signs of components in the same dimension. According to Alg. 2, these components whose signs are different from the components in the global *ideal gradient* will be excluded from the uploaded gradients. So that, we can always keep the sign of the remaining local component g^l_m same as the global component g^l_*, so as to guarantee the model convergence. Therefore, it is reasonable to utilize $dis(g^l_m, g^l_*)$ to measure the distance for further reliability estimation, while guaranteeing convergence.

Notes: Based on Eqn. (9), the reliability of a local component is determined by the distance from the *ideal component*, i.e., the shorter distance, the higher reliability the component will be assigned. This mechanism is reasonable under IID situation, but not for non-IID setting, since it cannot fundamentally solve the problem of gradient discrepancies caused by different data distributions through directly measuring the distance, if there exist large discrepancies among different gradients. Additionally, if a small portion of users whose data distributions are significantly different from others, they may be considered as unreliable and excluded from our system, since our scheme focuses on the IID setting. Meanwhile, we have experimentally demonstrated that under the assumption of the IID data setting, erasing the data of users who are obviously contrary to the distribution of the overall data during the training process is beneficial to improve the accuracy of the DNN model. More details are presented in Section V-B. Furthermore, some state-of-the-art works [23], [24], [25], [26] have also been proposed for federated learning with non-IID data. However, there is still a long way to implement these schemes under ciphertext mode, especially for considering the issue of handling unreliable users. We leave this exciting research in the future work.

(ii) Gradient Update: Given each user's reliability $\mathcal{R}^l_m$, the *ideal component* of the *ideal gradient* is updated in each dimension as follows:

$$g^l_* = \frac{\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}^l_m g^l_m}{\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}^l_m} \tag{11}$$

Notes: According to Eqn. (11), in the aggregation process, the higher reliability is assigned for the component g^l_m, the more values of the component will be counted. The result implies that the final aggregated result is primarily derived from the data shared by reliable users. For practical applications, the g^l_m can be considered unreliable if $\frac{\mathcal{R}^l_m - \min_{j\in[1,\mathcal{M}]} \mathcal{R}^l_j}{\max_{s\in[1,\mathcal{M}]} \mathcal{R}^l_s - \min_{j\in[1,\mathcal{M}]} \mathcal{R}^l_j} < \beta$, where β is a threshold value jointly determined by users. The similar setting is also applied in researches [21], [20].

Algorithm 3 Scheme for Handling Unreliable Users.

```
Input:  Local gradient $g_m$=$(g^1_m, g^2_m, ..., g^l_m, ..., g^{\mathcal{L}}_m)$,
        gradient set $G_m$=$\{g_m; m = 1, 2, ..., \mathcal{M}\}$,
        global ideal gradient $g_*$=$(g^1_*, g^2_*, ..., g^l_*, ..., g^{\mathcal{L}}_*)$,
        global weight $\omega_*$=$(\omega^1_*, \omega^2_*, ..., \omega^l_*, ..., \omega^{\mathcal{L}}_*)$,
        learning rate $\theta$.
Output: $\omega_*$ (global weight )
 1: Initialize global gradient $g_*$ and global weight $\omega_*$;
 2: repeat
 3:   for (each $g_m \in G_m$) do
 4:     Exclude irrelevant components through comparing
        with previous global gradient (e.g., Alg. 2);
 5:   end for
 6:   repeat
 7:     for (each $g_m \in G_m$) do
 8:       Update the current reliability of relevant components (e.g., Eqn. (9));
 9:     end for
10:     for (each $l$-th component) do
11:       Update the ideal component based on current reliability (e.g., Eqn. (11));
12:     end for
13:   until Convergence criterion is satisfied;
14:   Update current global weight (e.g., Eqn. (3));
15: until An approximate minimum is obtained;
16: return $\omega_*$.
```

Nextly we describe how to execute our $\texttt{Sch}_{\texttt{UU}}$ through the pseudo-code shown in Alg. 3. Considering $\mathcal{M}$ users participate the FL training, each user holds a local gradient g_m=$(g^1_m, g^2_m, ..., g^l_m, ..., g^{\mathcal{L}}_m)$, and all users' gradients constitute a gradient set G_m=$\{g_m; m = 1, 2, ..., \mathcal{M}\}$. We first initialize the global ideal gradient g_*=$(g^1_*, g^2_*, ..., g^l_*, ..., g^{\mathcal{L}}_*)$ and global weight ω_*=$(\omega^1_*, \omega^2_*, ..., \omega^l_*, ..., \omega^{\mathcal{L}}_*)$. Then in every iteration, firstly, we utilize "EIC" to exclude irrelevant components of each user's local gradient g_m by comparing it with previous global gradient. Secondly, we calculate the reliability of each remaining relevant component based on Eqn. (9). Thirdly, in each dimension of the global ideal gradient, according to the reliability of component, we compute the global component through Eqn. (11), and obtain the global ideal gradient. Finally, we update current global weight based on the generated ideal gradient. The above processes will be iteratively executed until an approximate minimum is obtained.

Notes: Two significant parameters of the gradient component are considered in our scheme, i.e, signs, and values. Specifically, the component, whose sign is inconsistent with the ideal component, will be considered irrelevant, being excluded during the training process. In addition, the remaining relevant component will be assigned an appropriate reliability value, i.e., the component, whose value is farther from the value of the ideal component, will be endowed with less reliability value, and vice versa. Intuitively, this is reasonable for that the negative impact conducted by unreliable users could be minimized through excluding components and reducing reliability for these gradients generated from unreliable users. In Section.V, the experiment results demonstrate the superiority of our approach in terms of model accuracy and convergence rate. Additionally, excluding components will be executed locally, i.e., each user will exclude their local irrelevant components by itself, that there is no need to consider the privacy issues. Nevertheless, weighted aggregation is achieved through the cooperation between the cloud server and users. During the process, not only the confidentiality of each user's gradient data should be protected, but also the reliability of each component needs to be guaranteed for ensuring the non-discrimination of the training.

Next, we will introduce how our EPPFL performs $\mathtt{Sch_{UU}}$ under ciphertext mode, thereby obtaining the expected privacy protection while guaranteeing our scheme's high practicability.

B. Secure Aggregation Protocol

Actually, our "Secure Aggregation Protocol (SAP)" is derived from the threshold Paillier cryptosystem. Considering each user $m\in[1,\mathcal{M}]$ holds a data value $v_m\in\boldsymbol{Z}_h$, after executing our "SAP", the final summation of these values will be obtained, while each user's value being guaranteed private against the cloud server and other users. To be specific, as shown in Fig. 3, in the first step, each user m encrypts value v_m and sends the ciphertext $E_{pk}(v_m)$ to the cloud server $\boldsymbol{S}$. Then, to acquire a summation value under ciphertext, the cloud server $\boldsymbol{S}$ calculates c=$E_{pk}(\sum_{m=1}^{m=\mathcal{M}} v_m)$=$\Pi_{m=1}^{m=\mathcal{M}} E_{pk}(v_m)$ based on Eqn. (5). The equation implies that the summation of plaintexts can be obtained through the multiplication of ciphertexts. Next, the server $\boldsymbol{S}$ randomly selects t users and sends the ciphertext c to them. After receiving the the ciphertext c, each selected user m calculates the secret share s_m based on c through the Eqn. (7), and then sends s_m to the cloud server again. Finally, $\boldsymbol{S}$ combines t secret shares from users, and obtain the summation $\sum_{m=1}^{m=\mathcal{M}} v_m$ through the encryption on the combined result.

Notes: This protocol can be briefly modified for supporting users' dropping out, through slightly changing the processing mechanism. Specifically, in step. 3, instead of randomly selecting t users, the cloud server $\boldsymbol{S}$ sends c to each user m. Then each user m is required to upload the secret share s_m to the cloud server. Until the number of received secret shares is greater than t, the cloud server $\boldsymbol{S}$ randomly selects t shares for decryption. Therefore, even if some unstable users are off-line during the process, the summation value can still be decrypted.

C. Construction of EPPFL

The specific workflow of EPPFL is described in Fig. 4, containing two phases, i.e., system setup and encrypted $\mathtt{Sch_{UU}}$.

Input: Each user's value $v_m \in \boldsymbol{Z}_h$, $m \in [1,\mathcal{M}]$.
Output: Aggregation value $\sum_{m=1}^{m=\mathcal{M}} v_m$.
Procedure:
$\underline{user_m}$:

1. Based on Eqn. (4), each user's v_m is encrypted as $E_{pk}(v_m)$, nextly being sent to the cloud server $\boldsymbol{S}$.

$\underline{\boldsymbol{S}}$:

2. $\boldsymbol{S}$ calculates c=$E_{pk}(\sum_{m=1}^{m=\mathcal{M}} v_m)$=$\Pi_{m=1}^{m=\mathcal{M}} E_{pk}(v_m)$ according to Eqn. (5).
3. Server $\boldsymbol{S}$ randomly selects t users, sending c to them.

$\underline{user_m}$:

4. Each selected user m calculates the secret share s_m of c according to Eqn. (7), then sends s_m to the cloud server.

$\underline{\boldsymbol{S}}$:

5. Server $\boldsymbol{S}$ combines t secret shares from users to get the summation $\sum_{m=1}^{m=\mathcal{M}} v_m$.

Fig. 3: SAP: Secure Aggregation Protocol

1) System Setup: According to the threshold Paillier cryptosystem, given the security parameter k, a trusted agent (TA) is required to generate a public key pk for all the participants and a series of private keys $(sk_1, sk_2, ..., sk_m, ..., sk_{\mathcal{M}})$ for each user $m\in[1,\mathcal{M}]$. The public key pk is broadcasted to all the participants (the cloud server $\boldsymbol{S}$ and each user m), while the private key sk_m is respectively and secretly kept by each user m. For initializing the DNN model, the global weight ω_*=$(\omega_*^1, \omega_*^2, ..., \omega_*^l, ..., \omega_*^{\mathcal{L}})$ and global *ideal gradient* g_*=$(g_*^1, g_*^2, ..., g_*^l, ..., g_*^{\mathcal{L}})$ will be obtained through pre-training the DNN model. This is a universal method widely applied in works such as [19], [21].

2) Encrypted $\mathtt{Sch_{UU}}$*:* Applying threshold Paillier cryptosystem requires that the plaintext should belong to an integer ring, however, in our scheme, the value (e.g., the gradient component g_m^l) needed to be encrypted may not be integers. For addressing this challenge, we utilize a rounding factor 10^N (a magnitude of 10) to scale each g_m^l to an integer $10^N{\cdot}g_m^l$ whenever needed, where N is a positive integer. Additionally, for the sake of simplicity, the $\widetilde{a}$ is utilized to denote the ciphertext of a, calculated through $E_{pk}(a)$. As shown in Fig. 4, our encrypted $\mathtt{Sch_{UU}}$ can be divided into four main parts: (i) excluding irrelevant components, (ii) encrypted reliability estimation, (iii) encrypted gradient update, and (iv) weight update. All the detailed technologies of each part are described as follows.

(i) Excluding irrelevant components: In this part, step. 1$\sim$2 is executed for excluding *irrelevant components*. Firstly, each user $m\in[1,\mathcal{M}]$ obtain a local gradient g_m through training the unified DNN with its local dataset based on SGD algorithm. Then each user m compare each local gradient component g_m^l with previous g_*^l to exclude the irrelevant components based on our "EIC" algorithm. Since this work is achieved locally and respectively by each user m itself, and all the operations can be executed under plaintext mode, there

Implementation of `EPPFL`

Setup: Given the security parameter k, the trusted Third Party (TA) generates a public key pk and a series of private keys $(sk_1, sk_2, ..., sk_m, ..., sk_{\mathcal{M}})$ based on the Paillier $(\mathcal{T}, \mathcal{M})$-threshold cryptosystem. The public key pk is broadcasted to all the participants (the cloud server $\boldsymbol{S}$ and each user m), and each private key sk_m is respectively and secretly kept by each user m. Initialize the DNN model with global weight of ω_*=$(\omega_*^1, \omega_*^2, ..., \omega_*^l, ..., \omega_*^{\mathcal{L}})$ and global *ideal gradient* $g_* = (g_*^1, g_*^2, ..., g_*^l, ..., g_*^{\mathcal{L}})$.

Encrypted $\mathtt{Sch_{UU}}$:

$\underline{user_m}$:

1. Each user m trains the unified DNN to obtain the gradient with the current local data set, through the SGD algorithm.
2. Each user m compares each current local gradient component g_m^l with previous g_*^l to exclude the irrelevant components based on our "EIC" algorithm.
3. Each user m calculates the distance between valid local gradient component and the global *ideal component* g_*^l as $dis(g_m^l, g_*^l)$.
4. Each user m encrypts each $dis(g_m^l, g_*^l)$ as $\widetilde{dis}_m^l$= $E_{pk}(dis(g_m^l, g_*^l))$, and encrypts $log(dis(g_m^l, g_*^l))$ as $\widetilde{log(dis}_m^l)$=$E_{pk}(log(dis(g_m^l, g_*^l)))$ $m \in [1, \mathcal{M}]$ and $l \in [1, \mathcal{L}]$, and then send both of them to the cloud server.

$\underline{\boldsymbol{S}}$:

5. For each l-th component, $\boldsymbol{S}$ multiplies the ciphertexts as $\widetilde{sumdis}^l$=$\Pi_{m=1}^{m=\mathcal{M}} E_{pk}(dis(g_m^l, g_*^l))$= $E_{pk}(\sum_{m=1}^{m=\mathcal{M}} dis(g_m^l, g_*^l))$, based on our "SAP" protocol the cloud server obtain a plaintext of the sum $sumdis^l$.
6.Then $\boldsymbol{S}$ encrypts $log(sumdis^l)$ as $log(\widetilde{sumdis}^l)$=$E_{pk}(log(sumdis^l))$.

For each user m, the cloud server $\boldsymbol{S}$ calculates $\widetilde{\mathcal{R}}_m^l$=$E_{pk}(\mathcal{R}_m^l)$=$log(\widetilde{sumdis}^l)$·$\widetilde{log(dis}_m^l)$, and sends each $\widetilde{\mathcal{R}}_m^l$ to each user m.

$\underline{user_m}$:

7. Each user m calculates $E_{pk}(\mathcal{R}_m^l \cdot g_m^l)$= $E_{pk}(\mathcal{R}_m^l)^{g_m^l}$ and sends them to the cloud server.

$\underline{\boldsymbol{S}}$:

8. $\boldsymbol{S}$ calculates $\Pi_{m=1}^{m=\mathcal{M}} E_{pk}(\mathcal{R}_m^l g_m^l)$ and $\Pi_{m=1}^{m=\mathcal{M}} E_{pk}(\mathcal{R}_m^l)$, and then executes the "SAP" protocol to obtain the plaintexts of $\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l g_m^l$ and $\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l$. Nextly, $\boldsymbol{S}$ calculates $g_*^l = \frac{\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l g_m^l}{\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l}$
9. $\boldsymbol{S}$ obtains all the g_*^l, and construct the current global *ideal gradient* g_*.

$\underline{\boldsymbol{S}, user_m}$:

10. $\boldsymbol{S}$ and each $user_m$ execute step 3∼9 iteratively until the convergence criterion is satisfied and then update the global *ideal gradient* g_*.

$\underline{\boldsymbol{S}}$:

11. Updates the global DNN model (i.e., current global weight ω_*) based on Eqn. (3), and broadcasts g_* and ω_* to each user m.

$\underline{\boldsymbol{S}, user_m}$:

12. $\boldsymbol{S}$ and $user_m$ execute step 1∼11 iteratively until an approximate minimum is obtained.

Fig. 4: Detailed description of `EPPFL`

is no need to consider the privacy issues for users in this part.

(ii) Encrypted reliability estimation: To achieve the privacy-preserving reliability estimation, step. 3∼6 is executed. Based on Eqn. (9) and homomorphic property of threshold Paillier cryptosystem, the encrypted reliability $\widetilde{\mathcal{R}}_m^l$ for each gradient component g_m^l is updated as follows.

$$\begin{aligned}\widetilde{\mathcal{R}}_m^l &= E_{pk}(\mathcal{R}_m^l) = E_{pk}(log(\frac{\sum_{m=1}^{m=\mathcal{M}} dis(g_m^l, g_*^l)}{dis(g_m^l, g_*^l)})) \\ &= E_{pk}(log(\sum_{m=1}^{m=\mathcal{M}} dis(g_m^l, g_*^l)) \cdot E_{pk}(log(dis(g_m^l, g_*^l)))^{-1}\end{aligned} \tag{12}$$

where the $dis(g_m^l, g_*^l)$ denotes the distance between the each local gradient component g_m^l and the global ideal component. and $sumdis^l$=$\sum_{m=1}^{m=\mathcal{M}} dis(g_m^l, g_*^l)$ is the distance summation of all users on l-th component. According to Eqn. (12), the specific steps are introduced as follows.

Firstly, each user m∈$[1, \mathcal{M}]$ calculates the distance $dis(g_m^l, g_*^l)$ between each l-th local gradient component g_m^l and global *ideal component* g_*^l, where l∈$[1, \mathcal{L}]$. Please note that, this process is achieved locally under plaintext mode. Then, each user m encrypts each distance $dis(g_m^l, g_*^l)$ and $log(dis(g_m^l, g_*^l))$ respectively as $\widetilde{dis}_m^l$= $E_{pk}(dis(g_m^l, g_*^l))$ and $\widetilde{log(dis}_m^l)$=$E_{pk}(log(dis(g_m^l, g_*^l)))$, and then sends them to the cloud server $\boldsymbol{S}$.

After receiving these ciphertexts from users, the cloud server $\boldsymbol{S}$ cooperates with users to calculate the plaintext of the sum $sumdis^l$ based on our "SAP" protocol.

Nextly, the cloud server $\boldsymbol{S}$ calculates $log(sumdis^l)$, and encrypts it as $log(\widetilde{sumdis}^l)$=$E_{pk}(log(sumdis^l))$. Finally, the encrypted reliability $\widetilde{\mathcal{R}}_m^l$ is obtained for for each user m as follows.

$$\widetilde{\mathcal{R}}_m^l = E_{pk}(\mathcal{R}_m^l) = log(\widetilde{sumdis}^l) \cdot (\widetilde{log(dis}_m^l))^{-1} \tag{13}$$

(iii) Encrypted gradient update: After the encrypted reliability $\widetilde{\mathcal{R}}_m^l$ being generated for each user m, the step. 7∼9 will be run for privately updating the current gradient. Each user m first calculates the ciphertexts of the component value multiplied by reliability according to Eqn. (6), using the following formula:

$$E_{pk}(\mathcal{R}_m^l \cdot g_m^l) = E_{pk}(\mathcal{R}_m^l)^{g_m^l} \tag{14}$$

Then each user m sends them to the cloud server.

After receiving each user' ciphertexts, the cloud server $\boldsymbol{S}$ executes the "SAP" protocol to respectively obtain the plaintexts of $\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l g_m^l$ and $\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l$, and calculates each current updated gradient component as $g_*^l = \frac{\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l g_m^l}{\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l}$. The part (ii) and (iii) will be executed iteratively until the convergence criterion is satisfied, and then the *ideal gradient* g_* is privately generated.

(iv) Weight update: The global DNN model is updated in step. 11 through Eqn. (3), and the updated $\boldsymbol{\omega}$ is distributed to each user m for local updating. As shown in Fig. 4, $\boldsymbol{S}$ and each

$user_m$ perform step. 1$\sim$11 iteratively until an approximate minimum is obtained.

Notes: For the simplicity of description, it is assumed that no user is off-line during the execution process of our EPPFL. However, it has robustness to users being off-line during the entire implementation. Note that our "SAP" can be divided into two parts: encrypted aggregation and decryption, which are two independent segments, i.e., the users participating in the encrypted aggregation and the users participating in the decryption may be different. As discussed in Sec.III-B, our "SAP" can be easily adjusted to support users dropping out after they have successfully uploaded the encrypted values to be summed. Another issue is the failure of uploading encrypted values, which means not all required users are able to upload the encrypted values for encrypted aggregation. To combat this, the preliminary method is to set a waiting time threshold. However, if some users still can not achieve the uploading tasks timely, they will be labelled as invalid users, so that the subsequent ciphertext operations (e.g., execute "Encrypted Reliability Estimation" and "Encrypted Gradient Update") will be not affected by them. As a result, our EPPFL can still be perform by the cloud server and the remaining valid users. Additionally, if the majority of off-line users are reliable, the model accuracy will be negatively impacted and vice versa. However, in real-world applications, we can hardly confirm whether the off-line users are reliable or unreliable. For convenience, off-line users' effect on training accuracy will not be considered in our scheme.

IV. Privacy Guarantees

According to the analysis in Sec.II-B, the security threats mainly come from the participants (i.e., the cloud server and users). Therefore, the goal of EPPFL is to protect the each user's gradients and the reliability of each gradient component from being exposed to any other participants. Besides, in our encrypted $\mathtt{Sch}_{\mathtt{UU}}$, the 1st part and 4-th part consider no privacy issues, hence, the privacy analysis focuses on the parts of "Encrypted Reliability Estimation" and "Encrypted Gradient Update". Additionally, since the security of our EPPFL is derived from our proposed "Secure Aggregation Protocol", the privacy analysis can be started from this protocol.

In this protocol, all exchanging processes are executed between the cloud server and users, under ciphertext mode. Although user m's ciphertext $E_{pk}(v_m)$ may be observed by other users, it cannot be decrypted by these users due to the security of (t,n)-threshold Paillier cryptosystem adopted in our "SAP" (i.e., only more than t users with secret shares can decrypt the ciphertext). Thus, each user's privacy can be protected from other users. Similarly, the ciphertext $E_{pk}(v_m)$, uploaded by each user m, leaks no useful information to the cloud server, the only value that can be obtained is the summation result $\sum_{m=1}^{m=\mathcal{M}} v_m$. Based on the above analysis, the formal proof is presented as follows.

Proposition 1: (Against honest-but-curious cloud server and each user $m \in M$) There exists a PPT simulator $\mathbf{SIM}$ *such that for the given security parameter k, threshold parameter t, and the participating user set $M = \{1, ..., \mathcal{M}\}$, the output of* $\mathbf{SIM}_{m,\boldsymbol{S}}^{k,t,M}$ *is computationally indistinguishable from the output of* $\mathbf{REAL}_{m,\boldsymbol{S}}^{k,t,M}$*, even if at most $t-1$ users collude with $\boldsymbol{S}$:*

$$\mathbf{SIM}_{m,\boldsymbol{S}}^{k,t,M} \approx \mathbf{REAL}_{m,\boldsymbol{S}}^{k,t,M} \tag{15}$$

Proof: A standard hybrid argument [27], [13] is utilized for proving the theorem. Firstly, a simulator $\mathbf{SIM}$ is defined through a series of (polynomially many) subsequent modifications to the random variable $\mathbf{REAL}$, then the indistinguishability between the output of $\mathbf{SIM}$ and the output of $\mathbf{REAL}$ is proven. The detailed proof is presented as follows.

$\mathbf{hyb_1}$ We initialize a random variable whose distribution is exactly the same as $\mathbf{REAL}$, and then the joint view of each user $m \in M$ and the cloud server $\boldsymbol{S}$ will be obtained in a real execution of the protocol. For the sake of simplicity, the $Rand(a)$ is denoted as a random value with the same length of a.

$\mathbf{hyb_2}$ In this hybrid, for each user $m \in M$, instead of encrypting the original $dis(g_m^l, g_*^l)$ and $log(dis(g_m^l, g_*^l))$, each user m encrypts randomly selected $Rand(dis(g_m^l, g_*^l))$ and $Rand(log(dis(g_m^l, g_*^l)))$ with public key pk under threshold Paillier cryptosystem. Based on the security property of threshold Paillier cryptosystem, the encrypted ciphertext leaks no useful information, and the views of $\mathbf{SIM}$ and $\mathbf{REAL}$ have the same distribution. Therefore, the indistinguishability $\mathbf{SIM}_{m,\boldsymbol{S}}^{k,t,M} \approx \mathbf{REAL}_{m,\boldsymbol{S}}^{k,t,M}$ is guaranteed in this hybrid.

$\mathbf{hyb_3}$ In this hybrid, the input of our "SAP" protocol is replaced, which means the cloud server $\boldsymbol{S}$ computes $E_{pk}(Rand(\sum_{m=1}^{m=\mathcal{M}} g_m^l))$ instead of $E_{pk}(\sum_{m=1}^{m=\mathcal{M}} g_m^l)$. Based on the security property of threshold Paillier cryptosystem, the encrypted ciphertext leaks no useful information, and the views of $\mathbf{SIM}$ and $\mathbf{REAL}$ have the same distribution. Therefore, the indistinguishability $\mathbf{SIM}_{m,\boldsymbol{S}}^{k,t,M} \approx \mathbf{REAL}_{m,\boldsymbol{S}}^{k,t,M}$ is guaranteed in this hybrid.

$\mathbf{hyb_4}$ In this hybrid, the server $\boldsymbol{S}$ encrypts $Rand(\mathcal{R}_m^l)$ instead of $\mathcal{R}_m^l$. Based on the security property of threshold Paillier cryptosystem, the encrypted ciphertext leaks no useful information, and the views of $\mathbf{SIM}$ and $\mathbf{REAL}$ have the same distribution. Therefore, the indistinguishability $\mathbf{SIM}_{m,\boldsymbol{S}}^{k,t,M} \approx \mathbf{REAL}_{m,\boldsymbol{S}}^{k,t,M}$ is guaranteed in this hybrid.

$\mathbf{hyb_5}$ In this hybrid, each user m calculates $E_{pk}(Rand(\mathcal{R}_m^l \cdot g_m^l))$ instead of $\mathcal{R}_m^l \cdot g_m^l$. Based on the security property of threshold Paillier cryptosystem, the encrypted ciphertext leaks no useful information, and the views of $\mathbf{SIM}$ and $\mathbf{REAL}$ have the same distribution. Therefore, the indistinguishability $\mathbf{SIM}_{m,\boldsymbol{S}}^{k,t,M} \approx \mathbf{REAL}_{m,\boldsymbol{S}}^{k,t,M}$ is guaranteed in this hybrid.

$\mathbf{hyb_6}$ In this hybrid, the input of our "SAP" protocol is replaced from $\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l g_m^l$ and $\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l$ to $Rand(\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l g_m^l)$ and $Rand(\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l)$, Based on the property of threshold Paillier cryptosystem, the indistinguishability $\mathbf{SIM}_{m,\boldsymbol{S}}^{k,t,M} \approx \mathbf{REAL}_{m,\boldsymbol{S}}^{k,t,M}$ is guaranteed in this hybrid.

As described above, based on the security properties of our "SAP" and threshold Paillier cryptosystem, we prove that there is a *PPT* simulator $\mathbf{SIM}$ whose output (i.e., $\mathbf{SIM}_{m,\boldsymbol{S}}^{k,t,M}$) is computationally indistinguishable from the out-

TABLE. 1: Functionality Comparison with existing works

Scheme \ Function	Users' privacy protection	Resistance to collusion	Robust to users being off-line	Support for unreliable users	Server setting	Threat model
SecProbe [6]	✔	✔	✗	✔	**Single-Server**	**Honest-but-Curious**
PPFDL[7]	✔	✗	✔	✔	**Dual-Server**	**Honest-but-Curious**
PPML [13]	✔	✔	✔	✗	**Single-Server**	**Honest-but-Curious**
PPDL [12]	✔	✗	✗	✗	**Single-Server**	**Honest-but-Curious**
EPPFL	✔	✔	✔	✔	**Single-Server**	**Honest-but-Curious**

put of $\mathbf{REAL}_{m,\boldsymbol{S}}^{k,t,M}$. Furthermore, through the view of simulation, the plaintext data of each user m will not be exposed to the cloud server or other users under these interactions. ■

V. Performance Evaluation

In this section, we discuss the performance of our EPPFL through extensive experiments, whose settings are as follows: we utilize ($\lfloor \frac{n}{2} \rfloor$,$n$)-threshold Paillier cryptosystem in our EPPFL, which is implemented through the Paillier Threshold Encryption Toolbox[2]. We run our experiments under the environment with Java 1.7.0, where the "Cloud" is simulated by the Lenovo Server with Intel(R) Xeon(R) (6-CORE 2.10GHZ CPU, 32GB RAM), running Ubuntu 18.04, and Huawei honor30 (8-CORE, 8GB RAM) is utilized to simulate the edge users. We achieve the training and testing tasks based on MNIST database[3], which has 60,000 training examples and 10,000 testing examples, while the selected neural network consists of 2 fully connected layers, 2 convolutional layers, and 1 average pooling layer. For comprehensively evaluating our EPPFL, we first conduct the functionality comparison, and then we achieve the comparisons of accuracy and overhead of computation and communication.

A. Functionality

For specifying the superiority of our EPPFL in terms of functionality, we compare our EPPFL with four state-of-the-art approaches of privacy-preserving FL, i.e., SecProbe [6], PPFDL[7], PPML [13], and PPDL [12]. As shown in Table. 1, the SecProbe [6], as the first privacy-preserving method for handling unreliable users during federated training can be resistant to the collusion among the cloud server and users while protecting users' privacy. However, it cannot support users to be off-line during the FL training process. As the following work of SecProbe [6], PPFDL [7] can also handle unreliable users, meanwhile, it supports users being off-line during the whole execution process. However, the dual-server setting applied in their scheme requires that the two servers are not allowed to collude with each other. Once collusion occurs, users' privacy will not be protected any more, being leaked to these two cloud servers. This is not practical in real-world applications. PPDL [12] utilizes additively homomorphic encryption for implementing their federated learning. Due to that, all users exploit the same secret key to encrypt parameters for privacy protection, their scheme requires that there is no collusion between users and the cloud server. Besides, the cases of handling off-line users and unreliable users during the training process are out of their scope of works. PPML [13], exploiting the secret sharing and masking method to guarantee users' privacy of gradients, is the first work with both robustness to users being off-line and resistance to the collusion between users and the cloud server. However, PPML just focuses on how to protect the private gradient information, without any consideration of handling unreliable users.

[2]http://cs.utdallas.edu/dspl/cgi-bin/pailliertoolbox/
[3]http://yann.lecun.com/exdb/mnist/

Compared with the above approaches, we utilize the threshold Paillier cryptosystem to construct the fundamental privacy-preserving framework of our EPPFL. Under our single-server setting, it can rigorously protect users local gradient and reliability information. Meanwhile, due to the threshold property of this framework, even if multiple users collude with the cloud server, users' private information will still be guaranteed from being leaked, besides, this framework can also be robust to users being off-line in the training process, guaranteeing the private training process to be achieved smoothly. Furthermore, for improving the training accuracy and efficiency, we propose a novel scheme $\texttt{Sch}_{\texttt{UU}}$ to alleviate the negative impact of unreliable users.

B. Accuracy

We discuss the accuracy of our EPPFL in this part. For obtaining convincing experiment results, we compare our scheme with two representative methods, i.e., OFL [22] and SecProbe [6]. This is reasonable. OFL [22] is the original model without any additional operations on unreliable data, performing model training under plaintext mode. Thereby, through comparing OFL with our EPPFL, we can present the specific effect of our scheme for handling unreliable users. Additionally, SecProbe is the first scheme for achieving privacy-preserving FL with unreliable users. According to the comparison results, we can clearly specify the discrepancy in effect generated by these two different types of methods.

In our experiments, the accuracy results are obtained on different proportions ($\mathcal{P}$) of unreliable users. For simulating unreliable users, $\mathcal{P}$ of users will be randomly selected, and their data will be added with random noise (ranging from 0 to 1), while the noise-added data will be considered unreliable. For evaluating the robustness to unreliable users in our EPPFL, we change $\mathcal{P}$ to obtain different results. This setting is also applied in OFL [22] and SecProbe [6] for comparison. Additionally, the privacy budget ϵ is set as 10 for SecProbe, which is as same as the setting in the original SecProbe [6].

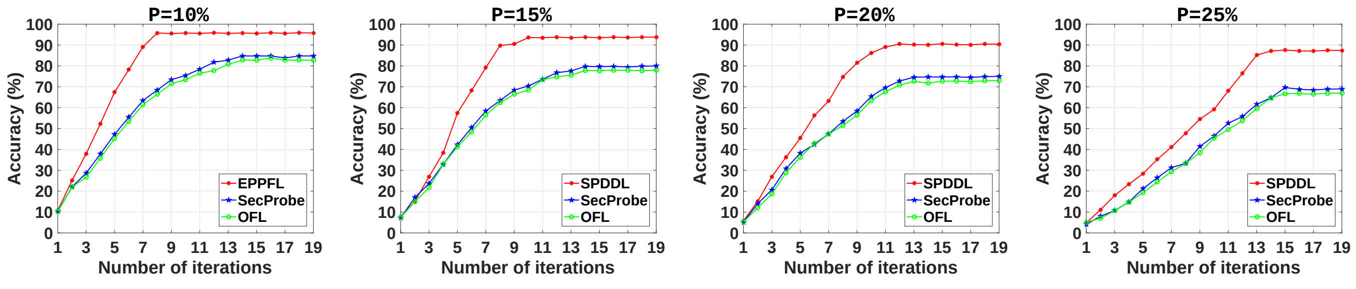


Fig. 5: Accuracy comparison with different proportions of unreliable users.

As shown in Fig. 5, the accuracy values change with the increasing number of iterations, respectively on the proportion of ($\mathcal{P}$=10%, 15%, 20%, 25%). It can be observed that all the accuracy changing curves gradually stabilize, and the final accuracy of our EPPFL is higher than OFL [22] and SecProbe [6], specifically being up to 95.78% in Fig. 5 (a), 93.73% in Fig. 5 (b), 90.38% in Fig. 5 (c), and 87.35% in Fig. 5 (d). Meanwhile, we find that when the same accuracy is obtained, the required number of iterations in our EPPFL is less than OFL and SecProbe, which means that the model in our scheme can converge more rapidly than these two schemes. Moreover, we observe that when the proportion of $\mathcal{P}$ reaches 25%, the accuracy of OFL and SecProbe decreases quite rapidly, while the accuracy of our EPPFL changes more slightly. These above results are mainly caused by two reasons: (i) OFL has not considered how to handle unreliable users. In contrast, our scheme can erasing the low-quality data (also may be a high-quality data but with different distributions) which is obviously contrary to other data during the training process, while remaining data can be processed through a preferable method, and (ii) differential privacy technology, being utilized in SecProbe for constructing privacy-preserving FL, may not supply enough accuracy if high privacy should be guaranteed. However, our $\mathtt{Sch}_{\mathtt{UU}}$ exploits methods of excluding irrelevant gradient components and weighted aggregation to guarantee the aggregation result derived from the contribution of high-quality users, while utilizing Pallier threshold cryptosystem to accomplish our privacy-preserving scheme, all of which establish the superiority of our $\mathtt{Sch}_{\mathtt{UU}}$.

C. Computation Overhead

1) Complexity Analysis: In this part, a comprehensive complexity analysis is presented, focusing on both user-side and server-side. Considering there are $\mathcal{M}$ users participating in the model training, and each user's model takes $\mathcal{L}$ gradient components, the analysis is accomplished as follows.

(i) User side: $O(\mathcal{M}+\mathcal{L})$. The computation cost of each user $user_m$ falls into 4 parts: (1) Encrypting the ciphertexts of $dis(g_m^l, g_*^l)$ and $log(dis(g_m^l, g_*^l))$, which takes $O(\mathcal{L})$. (2) Executing "SAP" for obtaining $sumdis^l$, which takes $O(\mathcal{M}+\mathcal{L})$. (3) Calculating $E_{pk}(\mathcal{R}_m^l \cdot g_m^l)$, which take $O(\mathcal{L})$. (4) Executing "SAP" for obtaining $\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l g_m^l$ and $\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l$, which takes $O(\mathcal{M}+\mathcal{L})$. Overall, the computation complexity of each user is $O(\mathcal{M}+\mathcal{L})$.

(ii) Server side: $O(\mathcal{M}\cdot\mathcal{L}+\mathcal{M}+\mathcal{L})$. The computation cost of the cloud server $\boldsymbol{S}$ can be broken up into 5 parts: (1) Aggregating all users' $E_{pk}(dis(g_m^l, g_*^l))$, which takes $O(\mathcal{M}+\mathcal{L})$. (2) Executing "SAP" for obtaining $sumdis^l$, which takes $O(\mathcal{L})$. (3) Encrypting $log(sumdis^l)$ as $E_{pk}(log(sumdis^l))$, which take $O(\mathcal{L})$. (4) Calculating $\widetilde{\mathcal{R}}_m^l$=$E_{pk}(\mathcal{R}_m^l)$, which takes $O(\mathcal{M}\cdot\mathcal{L})$. (5) Executing "SAP" for obtaining $\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l g_m^l$ and $\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l$, which takes $O(\mathcal{M}+\mathcal{L})$. Overall, the communication complexity of the cloud server is $O(\mathcal{M}\cdot\mathcal{L}+\mathcal{M}+\mathcal{L})$.

2) Experiment Results: As shown in Fig. 6, the computation cost of each user and the cloud server increases as the number of each user's components and the number of users increase. Obviously, the overhead of the cloud server is far more than the overhead of each user, respectively being up to 219s and 32s in Fig. 6 (a), and being up to 219s and 32s in Fig. 6 (b). That means enormous computation tasks are outsourced to the cloud server, which is very friendly to edge user. Additionally, although the overhead of each user will increase as the number of users increases, the increasing rate is so low that abundant users can still join our EPPFL ecosystem.

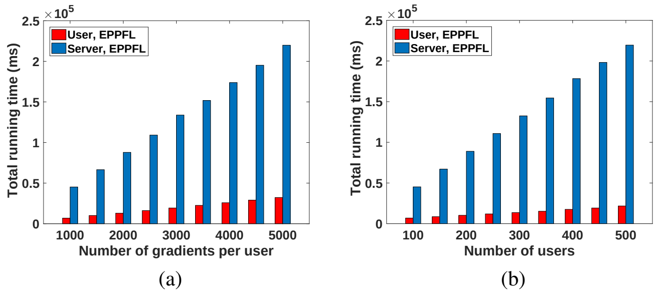
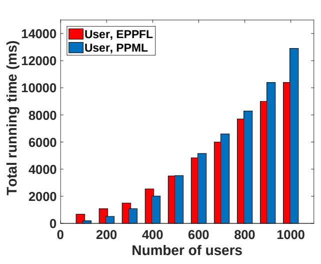
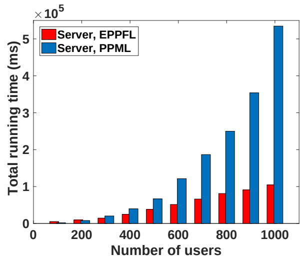


Fig. 6: Computation overhead of EPPFL.

As shown in Fig. 7, the computation overhead is compared between EPPFL and PPML [13]. Specifically, on both user-side and server-side, the overhead of our EPPFL increases linearly as the number of users increases, while the overhead of PPML increases according to a quadratic curve. The reason is that, based on above complexity analysis, our EPPFL takes

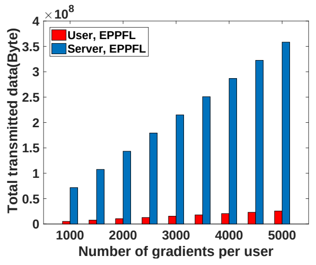


(a)

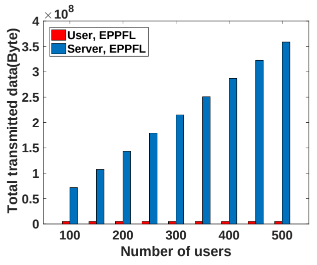


(b)

Fig. 7: Comparison of computation overhead between EPPFL and PPML [13].

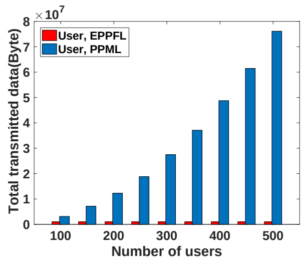


(a)

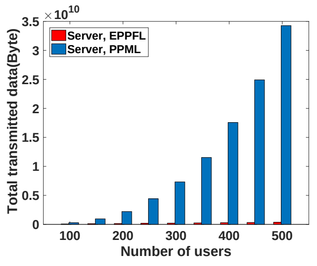


(b)

Fig. 8: Communication overhead of EPPFL.



(a)



(b)

Fig. 9: Comparison of communication overhead between EPPFL and PPML [13].

$O(\mathcal{M}+\mathcal{L})$ for each user and $O(\mathcal{M}\cdot\mathcal{L}+\mathcal{M}+\mathcal{L})$ for the server, nevertheless, the PPML takes $O(\mathcal{M}^2+\mathcal{M}\cdot\mathcal{L})$ and $O(\mathcal{L}\cdot\mathcal{M}^2)$ respectively for each user and the server. As a result, when the number of users reaches over 500, the computation overhead of our EPPFL becomes lower than PPML on the user-side, and the phenomenon is more obvious on the server-side. Obviously, that makes our EPPFL more suitable for Internet of Vehicles, which is constituted with tens of thousands of edge users.

D. Communication Overhead

1) Complexity Analysis: In this part, we give a comprehensive analysis of communication complexity for each user and the cloud server, as follows.

(i) User side: $O(\mathcal{L})$. The communication cost of each user $user_m$ falls into 4 parts: (1) Uploading the ciphertexts of $dis(g_m^l, g_*^l)$ and $log(dis(g_m^l, g_*^l))$ to the cloud server, which takes $O(\mathcal{L})$. (2) Exchanging secret shares for obtaining $sumdis^l$, which takes $O(\mathcal{L})$. (3) Receiving the ciphertexts of $\mathcal{R}_m^l$ and uploading the ciphertexts of $\mathcal{R}_m^l\cdot g_m^l$, which respectively takes $O(\mathcal{L})$. (4) Exchanging secret shares for obtaining the plaintexts of $\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l g_m^l$ and $\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l$, which takes $O(\mathcal{L})$. Overall, the communication complexity of each user is $O(\mathcal{L})$.

(ii) Server side: $O(\mathcal{M}\cdot\mathcal{L})$. The communication cost of the cloud server $\boldsymbol{S}$ can be devided into 4 parts: (1) Receiving the ciphertexts of $dis(g_m^l, g_*^l)$ and $log(dis(g_m^l, g_*^l))$ from all users, which takes $O(\mathcal{M}\cdot\mathcal{L})$. (2) Executing "SAP" for obtaining $sumdis^l$, which takes $O(\mathcal{M}\cdot\mathcal{L})$. (3) Sending the ciphertexts of $\mathcal{R}_m^l$ and receiving the ciphertexts of $\mathcal{R}_m^l\cdot g_m^l$, which take $O(\mathcal{M}\cdot\mathcal{L})$. (4) Executing "SAP" for obtaining $\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l g_m^l$ and $\sum_{m=1}^{m=\mathcal{M}} \mathcal{R}_m^l$, which takes $O(\mathcal{M}\cdot\mathcal{L})$. Overall, the communication complexity of the cloud server is $O(\mathcal{M}\cdot\mathcal{L})$.

2) Experiment Results: As shown in Fig. 8, the communication cost of each user and the cloud server are evaluated along with the increasing number of each user's components and the number of users. Obviously, the overhead of the cloud server is far more than the overhead of each user. Additionally, the overhead of each user just increases as the number of each user's components increases, but keep consistent along with the increasing number of users.

Fig. 9 shows the comparison of communication overhead between EPPFL and PPML [13]. It is observed that the communication overhead of EPPFL is much lower than PPML as the number of users increases, on both user-side and server-side. On the user-side, PPML [13] requires not only encrypted gradient to be sent for aggregation, but also keys to be sent for encrypting the plaintext based on the key sharing protocol. Besides, once some users being off-line during the training process, the secret sharing protocol needs to be invoked for recovering off-line users' secret shares, which will be sent to the server again. Thereby, huge communication should be consumed for each user. However, in our EPPFL, only a few of encrypted data need to be sent to the server, meanwhile, even if some users drop out, no additional communication overhead is required, which can be very friendly for edge users with limited resources. On the server-side, PPML [13] needs the server to receive encrypted gradients and secret shares from users. Besides, additional interaction between the server and on-line users is required to decrypt the aggregated results and guarantee the users' gradient confidentiality. As a result, the communication complexity reaches $O(\mathcal{M}^2+\mathcal{M}\cdot\mathcal{L})$. However, our EPPFL is with a lower complexity of $O(\mathcal{M}\cdot\mathcal{L})$. Thereby, our EPPFL performs with a much smaller communication overhead compared to PPML.

VI. Related Work

For addressing the privacy issues in FL, many state-of-the-art approaches have been proposed, mainly derived from three technologies, i.e., homomorphic encryption (HE) [12], [28], secure multi-party computation (MPC) [13], [29], [30], [31], and differential privacy (DP) [32]. However, the fundamental issue of *unreliable users* has not been considered initially, until *Zhao et al.* [6] proposed the first privacy-preserving FL with unreliable participants. Following *Zhao et al.*'s work, *Xu et al.* [7] proposed another method. In this section, we discuss both of these methods as follows.

For addressing the issue of privacy-preserving FL with *unreliable users*, the first approach is proposed by *Zhao et al.* [6], named SecProbe. In their scheme, for protecting the privacy of participants, differential privacy is exploited to perturb the loss function of the targeted model. Indeed, their SecProbe is outstanding in terms of resource overhead. However, the DP-based method protects data privacy based on the added noise, as a result, the higher privacy is obtained, the more accuracy is lost. Indeed, the work [9] specified that current DP-based methods rarely offered acceptable trade-off between privacy and accuracy for complex learning tasks, through extensive experiments. Specifically, they claimed that even the advanced variants [33], [34] of traditional DP-based methods achieved accuracy loss close to 0.24, and none of the current approaches obtained a zero accuracy loss, even if the privacy was protected with a low level. Besides, research in [10] has claimed that the approach for federated learning can be fundamentally destroyed even if all exchanged parameters are perturbed via DP-based technology. Recently, *Xu et al.* [7] proposed a PPFDL scheme, where a novel method $\mathtt{Meth}_{\mathtt{IU}}$ was conducted to alleviate the negative impact of *unreliable users* on the training accuracy, while the confidentiality of all users' related information was guaranteed through encrypting $\mathtt{Meth}_{\mathtt{IU}}$. Actually, their scheme is based on the secure two-party computation (2-PC) cryptosystems, where two servers cooperate to handle the encrypted data uploaded by users. However, for achieving their PPFDL, these two servers are required not to collude with each other. Once the collusion occurs, the entire security mechanism will be destroyed. Thereby, this limitation makes their scheme unsuitable for real-world application scenarios.

Compared with these approaches, our EPPFL utilizes threshold Paillier cryptosystem to protect the privacy of users' local gradients and reliability information. Based on our proposed secure aggregation framework, only one single server is needed to interact with users, while the privacy can be guaranteed, even if some users collude with the cloud server. Besides, a novel scheme is also presented to alleviate the negative impact of unreliable users, which can minimize model converge rate, improve model accuracy, and work with low overhead of communication and computation.

VII. Conclusion

In this paper, we have proposed an efficient privacy-preserving federated learning (EPPFL), where a novel scheme $\mathtt{Sch}_{\mathtt{UU}}$ is developed for mitigating the negative impact of unreliable users, and a secure framework is conducted for protecting users' private gradient and reliability information. Extensive experiment results demonstrate the our EPPFL with high-level performance in terms of accuracy and efficiency.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grants 62020106013, 61972454, 61802051, 61772121, and 61728102, Sichuan Science and Technology Program under Grants 2020JDTD0007 and 2020YFG0298, the Fundamental Research Funds for Chinese Central Universities under Grant ZYGX2020ZB027.

References

[1] F. Gao, J. Duan, Z. Han, and Y. He, "Automatic virtual test technology for intelligent driving systems considering both coverage and efficiency," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 365–14 376, 2020.

[2] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 413–14 423, 2020.

[3] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *proceedings of IEEE S&P*, 2017, pp. 3–18.

[4] H. Sun, X. Ma, and R. Q. Hu, "Adaptive federated learning with gradient compression in uplink noma," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16 325–16 329, 2020.

[5] P. Luo, F. R. Yu, J. Chen, J. Li, and V. C. M. Leung, "A novel adaptive gradient compression scheme: Reducing the communication overhead for distributed deep learning in the internet of things," *IEEE Internet of Things Journal*, pp. 1–1, 2021, doi:10.1109/JIOT.2021.3051611.

[6] L. Zhao, Q. Wang, Q. Zou, Y. Zhang, and Y. Chen, "Privacy-preserving collaborative deep learning with unreliable participants," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1486–1500, 2020, doi:10.1109/TIFS.2019.2939713.

[7] G. Xu, H. Li, Y. Zhang, S. Xu, J. Ning, and R. Deng, "Privacy-preserving federated deep learning with irregular users," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2020, doi:10.1109/TDSC.2020.3005909.

[8] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, "Functional mechanism: regression analysis under differential privacy," *Proceedings of VLDB Endowment*, vol. 5, no. 11, pp. 1364–1375, 2012.

[9] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *28th USENIX Security Symposium (USENIX Security 19). Santa Clara, CA: USENIX Association*, 2019.

[10] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of ACM CCS*. ACM, 2017, pp. 603–618.

[11] 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 Transactions on Emerging Topics in Computing*, vol. 6, no. 1, pp. 97–109, 2018.

[12] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2018.

[13] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of ACM CCS*, 2017, pp. 1175–1191.

[14] P. Mohassel and P. Rindal, "Aby3: A mixed protocol framework for machine learning," in *Proceedings of ACM CCS*, 2018, pp. 35–52.

[15] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *proceedings of IEEE S&P*, 2017, pp. 19–38.

[16] N. Agrawal, A. Shahin Shamsabadi, M. J. Kusner, and A. Gascón, "Quotient: Two-party secure neural network training and prediction," in *Proceedings of ACM CCS*, 2019, pp. 1231–1247.

[17] G. Xu, H. Li, H. Ren, K. Yang, and R. H. Deng, "Data security issues in deep learning: Attacks, countermeasures, and opportunities," *IEEE Communications Magazine*, vol. 57, no. 11, pp. 116–122, 2019.

[18] I. Damgrd and M. Jurik, "A generalisation, a simpli.cation and some applications of paillier's probabilistic public-key system," in *International Workshop on Practice and Theory in Public Key Cryptography*, 2001, pp. 119–136.

[19] Y. Lin, S. Han, H. Mao, Y. Wang, and B. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=SkhQHMW0W

[20] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching {LAN} speeds," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 629–647.

[21] L. WANG, W. WANG, and B. LI, "Cmfl: Mitigating communication overhead for federated learning," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 954–964.

[22] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proceedings of NeurIPS*, 2017, pp. 4424–4434.

[23] F. Sattler, S. Wiedemann, K.-R. Mller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3400–3413, 2020.

[24] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 1698–1707.

[25] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-iid data," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 15–24.

[26] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, "Self-balancing federated learning with global imbalanced data in mobile systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 59–71, 2021.

[27] C. Gentry, J. Groth, Y. Ishai, C. Peikert, A. Sahai, and A. Smith, "Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs," *Journal of Cryptology*, vol. 28, no. 4, pp. 820–843, 2015.

[28] Y. Li, H. Li, G. Xu, T. Xiang, X. Huang, and R. Lu, "Toward secure and privacy-preserving distributed deep learning in fog-cloud computing," *IEEE Internet of Things Journal*, vol. 7, no. 12, pp. 11 460–11 472, 2020.

[29] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: Secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2020, doi:10.1109/TIFS.2019.2929409.

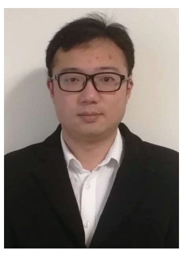
[30] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018, pp. 707–721.

[31] I. Damgård, D. Escudero, T. Frederiksen, M. Keller, P. Scholl, and N. Volgushev, "New primitives for actively-secure mpc over rings with applications to private machine learning," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 1102–1120.

[32] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex, "Differentially private model publishing for deep learning," in *proceedings of IEEE Security & Privacy*, 2019, pp. 309–326.

[33] J. Lee, "Differentially private variance reduced stochastic gradient descent," in *2017 International Conference on New Trends in Computing Sciences (ICTCS)*, 2017, pp. 161–166.

[34] B. Jayaraman and L. Wang, "Distributed learning without distress: Privacy-preserving empirical risk minimization," *Advances in Neural Information Processing Systems*, 2018.



Yiran Li is currently working toward the Ph.D. degree at the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), China. His research interests include Cryptography, Privacy-Preserving Federated Learning, and Data Security.

Hongwei Li is currently the Head and a Professor at Department of Information Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China. He received his Ph.D. degree from University of Electronic Science and Technology of China in 2008. From October 2011 to October 2012, he worked as a Postdoctoral Fellow at the University of Waterloo. His research interests include network security and applied cryptography. He serves as the Associate Editor of IEEE Internet of Things Journal, and Peer-to-Peer Networking and Applications, the Guest Editor of IEEE Network, IEEE Internet of Things Journal and IEEE Transactions on Vehicular Technology. He currently serves as the Secretary of IEEE ComSoc CIS-TC.

Guowen Xu is currently a Research Fellow with Nanyang Technological University, Singapore. He received the PhD degree in cyberspace security from the University of Electronic Science and Technology of China (UESTC) in 2020. As the first author, he has published more than 20 papers in top international conferences and journals, including ACM CCS, ACM ACSAC, ACM ASIACCS, IEEE TDSC and IEEE TIFS. He is the recipient of the Best Paper Award of the 26th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2020) , and the IEEE Student Conference Award. His research interests include Secure Outsourcing Computing and privacy-preserving issues in Deep Learning.

Xiaoming Huang Graduate degree, Senior engineer. He is currently a general manager of Technology Marketing Department of CETC Cyberspace Security Research Institute Co., Ltd., His research interests include cognitive domain security, cryptography and information security theory, trusted computing and trusted network technology, computer and communication security issues. He is a member of Sichuan electronic information expert group.

Rongxing Lu (S'09-M'11-SM'15) is currently an associate professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. Before that, he worked as an assistant professor at the School of Electrical and Electronic Engineering, Nanyang Technological University (N-TU), Singapore from April 2013 to August 2016. He worked as a Postdoctoral Fellow at the University of Waterloo from May 2012 to April 2013. He received his PhD degree from the Department of Electrical & Computer Engineering, University of Waterloo, Canada, in 2012. He is presently a Fellow of IEEE Communications Society. He currently serves as the Vice-Chair (Publication) of IEEE ComSoc CIS-TC.