Differentially Private Byzantine-Robust Federated Learning

Xu Ma, Xiaoqian Sun, Yuduo Wu, Zheli Liu, Xiaofeng Chen, *Senior Member, IEEE*, and Changyu Dong

Abstract—Federated learning is a collaborative machine learning framework where a global model is trained by different organizations under the privacy restrictions. Promising as it is, privacy and robustness issues emerge when an adversary attempts to infer the private information from the exchanged parameters or compromise the global model. Various protocols have been proposed to counter the security risks, however, it becomes challenging when one wants to make federated learning protocols robust against Byzantine adversaries while preserving the privacy of the individual participant. In this article, we propose a differentially private Byzantine-robust federated learning scheme (DPBFL) with high computation and communication efficiency. The proposed scheme is effective in preventing adversarial attacks launched by the Byzantine participants and achieves differential privacy through a novel aggregation protocol in the shuffle model. The theoretical analysis indicates that the proposed scheme converges to the approximate optimal solution with the learning error dependent on the differential privacy budget and the number of Byzantine participants. Experimental results on MNIST, FashionMNIST and CIFAR10 demonstrate that the proposed scheme is effective and efficient.

Index Terms—Federated learning, differential privacy, byzantine-robust

1 Introduction

Machine learning shows outstanding performance in many data-driven applications like image classification, speech recognition, recommendation systems, and self-driving cars, etc. The success of machine learning is largely due to the availability of vast volumes of data. However, for most organizations, it is often difficult to collect data with enough volume and variety to produce a good predictive model. Therefore, collaborative learning which enables learning from multiple data sources becomes an increasingly popular solution. However, the training dataset may contain highly private information about an individual. Therefore, sharing the training data among all the participants or to a central server is undesirable from the security and privacy view, and even sometimes is not allowed by law. Hence, how to realize privacy-preserving collaborative learning becomes a challenging research problem recently.

Federated learning [1] was first proposed by Google AI, aiming to enable mobile phones to collaboratively learn a global shared model without sharing the training data. Different from the centralized machine learning infrastructure, in which all the training data are collected and processed on one master server, in federated learning the training data is kept on the mobile devices (workers) that run the machine learning algorithm locally. Federated learning reduces the risk of privacy leakage by separating model training from the need for direct access to raw training data.

However, federated learning itself does not completely guarantee the security and privacy of the whole learning process. There are two seemingly independent but entangled challenges: protecting the privacy of individual worker's training data and ensuring the robustness of the global model against Byzantine faults. In federated learning, local parameters (gradients) and the global models are exchanged, knowing those can enable multiple privacy attacks. This is the first challenge in federated learning. For instance, Zhang *et al.* [2] proposed a model inversion attack with a high success rate of inverting deep neural networks by leveraging partial public information. Nasr *et al.* [3] proposed a white-box membership inference attack that allows an attacker to tell whether a given sample is in the training dataset or not. To defend against privacy attacks such as model-inversion attacks and membership inference attacks in federated learning, various privacy protection mechanisms have been proposed based on cryptography [4], [5], [6], [7], [8], [9] or differential privacy [10], [11], [12], [13], [14]. However, they are unable to resist the Byzantine manipulation attacks explained below.

The second challenge is the Byzantine faults. The Byzantine adversaries, who could control a subset of workers,

- *Xu Ma is with the School of Cyber Science and Engineering, Qufu Normal University, Qufu 273165, China, and also with the State Key Laboratory of Cryptology, Beijing 100878, China. E-mail: xma@qfnu.edu.cn.*
- *Xiaoqian Sun is with the School of Cyber Science and Engineering, Qufu Normal University, Qufu 273165, China. E-mail: xqsunqfnu@126.com.*
- *Yuduo Wu and Zheli Liu are with the College of Cyber Science and College of Computer Science, Nankai University, Tianjin 300071, China. E-mail: doria@mail.nankai.edu.cn, liuzheli@nankai.edu.cn.*
- *Xiaofeng Chen is with the State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an 710071, China. E-mail: xfchen@xidian.edu.cn.*
- *Changyu Dong is with the School of Computing, Newcastle University, NE1 7RU Newcastle Upon Tyne, U.K. E-mail: changyu.dong@newcastle.ac.uk.*

Manuscript received 23 Aug. 2021; revised 1 Apr. 2022; accepted 11 Apr. 2022. Date of publication 14 Apr. 2022; date of current version 11 July 2022.
This work was supported by the National Natural Science Foundation of China under Grants 62072132, 61960206014, and 62032012.
(Corresponding author: Changyu Dong.)
Recommended for acceptance by R. Tolosana.
Digital Object Identifier no. 10.1109/TPDS.2022.3167434

may send malicious parameters to the master server to indiscriminately jeopardize the accuracy of the global model or control the global model to output the specified result via poisoning attacks [15]. They can seriously damage the robustness of the federated model. Bagdasaryan *et al.* [16] proposed a new backdoor attack and showed that it is sufficient to reach backdoor accuracy of approximately 50% for word prediction by controlling only 0.01% of the participants. Some approaches have been proposed to realize Byzantine-robust aggregation for federated learning [17], [18], [19], [20], [21]. However, most of the existing Byzantine-robust techniques require access to the workers' model updates, so they cannot adopt cryptography and differential privacy technologies, which cause the risk of privacy leakage. Privacy and robustness are two important issues that have to be addressed in federated learning. In this paper, we present a shuffle protocol for summation (SPS) which satisfies differential privacy. Then, we combine SPS with a Byzantine-robust stochastic aggregation algorithm to construct a differentially private Byzantine-robust federated learning scheme (DPBFL). Especially, our contributions are as follows:

- We introduce DPBFL, a new method to improve the privacy and robustness of federated learning. DPBFL achieves differential privacy for the exchanged parameters during the learning process, and it can converge to the optimum (sub-optimum) solution even if there are Byzantine adversaries (workers).
- We present an efficient shuffle protocol for summation (SPS), based on which we design a differentially private aggregation scheme for DPBFL. Then, we analytically prove that DPBFL satisfies differential privacy. Compared to existing local differential privacy or cryptography-based methods, DPBFL exhibits better privacy utility and efficiency, respectively.
- We numerically validate the performance and efficiency of DPBFL on MNIST, FashionMNIST, CIFAR10 datasets. To do so, we implement DPBFL in a distributed network of $n = 1000$ workers with different proportions of Byzantine workers. The experimental results demonstrate that DPBFL can guarantee convergence against approximately 40% of Byzantine workers with certain privacy protection, and its test accuracy is higher than conventional federated learning benchmarks.

1.1 Related Work

By sharing the model parameters, federated learning completely opens the local models to the honest-but-curious server and Byzantine workers. The model itself also reveals information about its training data. To overcome the problem of shared parameter leakage during the federated training process, implementing a machine learning algorithm that learns from noisy [22] or encrypted data [23] has also been studied in recent years.

From the perspective of differential privacy (DP) [22], the methods involve adding noise to the data to obscure the sensitive information until an accessor cannot distinguish individuals. Centralized differential privacy (CDP), which requires a central trusted party to add noise to the aggregated gradients, is a focus of most works [24], [25]. Papernot *et al.* [25] proposed a semi-supervised knowledge transfer method(PATE) by learning noisy knowledge from the "teacher" models trained directly with private data to train the final "student" models used for publication and application and avoid information leakage of training data. Though CDP is a powerful privacy protection method, computing the statistics and adding noise requires a trusted third party. Multiparty computation (MPC) is one of the powerful tools that can simulate a central model algorithm without a trusted server, but due to the considerable overhead in terms of computing, communication, etc., the existing technology is difficult to be applied in federated learning.

Consequently, some recent works tried to integrate local differential privacy (LDP) into federated learning to achieve privacy protection [10], [11], [13], [26]. Zhao *et al.* [11] proposed integrating federated learning and LDP to facilitate training machine learning models via crowdsourcing. However, each query result in federated learning involves the worker's dataset information, the amount of information leaked will increase with the number of queries and communication rounds increases. Most differential privacy methods also face this problem [12]. Bhowmick *et al.* [13] introduced an approach to training large-scale local models, however, their schemes require at least 200 communication rounds with a very high privacy budget, i.e., ϵ=500 on MNIST and ϵ=5000 on CIFAR10, which means a weak privacy guarantee. Therefore, how to obtain a strong privacy guarantee in complex federated learning scenarios through LDP is still an open problem.

The shuffle model [14], [27], [28], [29] lies somewhere between centralized differential privacy and local differential privacy. Erlingsson *et al.* [14] showed that local differential privacy can be amplified through a shuffler, who randomly permute anonymized data submitted by clients. Cheu *et al.* [27] formally defined an augmented local differentially private model, including an asymmetric shuffler, and studied sample complexity to solve several problems with the shuffle model. In this paper, we also present a shuffle protocol for summation, which is the main building block to achieving a federated learning scheme with good performance and privacy.

Another line of work involves ensuring that the model is robust to Byzantine attacks during the protocol execution and training process. The Byzantine adversaries can gain control of compromised devices and launch poisoning attacks or backdoor attacks by sending arbitrary malicious messages to the server, which seriously threatens the security of the model. The main way to counteract Byzantine attacks is to combine federated aggregation methods with robust rules, such as Krum [20], Geometric Median [17], Buyan [21], HeteroSAg [30]. The core of which is to compare the local models received from different workers and remove outliers. However, new attacks continue to emerge, and almost all these defense methods have been proven to be insecure. Therefore, Li *et al.* proposed the RSA [31] method for Byzantine attacks, which is also suitable for a wider class of applications because it does not rely on the assumption of i.i.d. data. However, these methods do not guarantee the privacy. In the study of federated learning,

few schemes [32], [33], [34], [35] address both privacy and robustness. For example, Ma *et al.* [32] proposed a privacy-preserving Byzantine-robust method that combines Byzantine-robust algorithm, distributed Paillier encryption and zero-knowledge proof.

1.2 Organization

The remainder of this paper is organized as follows: In Section 2, we provide the fundamental knowledge regarding federated learning and differential privacy that underpins our protocol and basic model framework. In Section 3, the system model and security model are described in detail. The internal architecture and security analysis of our scheme are presented in Section 4, and the results of efficiency analysis and experimental evaluation are given in Section 5. Finally, Section 6 concludes the paper.

2 Preliminaries

In this section, we review the concepts of federated learning, Byzantine-robust federated learning, and the shuffle model.

2.1 Federated Learning

Federated learning, proposed by Google [1], has emerged as a promising collaboration paradigm in which the motivation is to share the model parameters instead of the private data for better privacy protection. During each iteration of the training process, each worker, the owner of private training datasets, downloads the current global model from the master server and updates a model using its local dataset, then uploads all gradients or weight parameters to the master server. The server aggregates such parameters from workers and updates a new central model (e.g., averaging all workers' gradients) and then distributes it back to a fraction of workers for another round of model updates. This process is repeated until the global model reaches a certain precision. Note that the model exchanged between the master server and workers involve communication in the network, which can lead to privacy and robustness issues.

2.2 Byzantine-Robust Federated Learning

To cope with Byzantine workers, most of the previous Byzantine-robust federated learning schemes utilize Byzantine-robust aggregation rules to eliminate outliers and aggregate the "correct" gradients. For example, Krum [20] chooses a gradient sent by workers, compared with other gradients, the euclidean distances between the chosen gradient and its adjacent gradients is the smallest. GeoMed [17] tries to find the coordinate-wise geometric median of a set of gradients, and so on. However, the sizable stochastic gradient noise introduced by the heterogeneous datasets and Byzantine workers makes it challenging to distinguish malicious messages sent by Byzantine workers from noisy stochastic gradients sent by honest workers. Consequently, Li *et al.* proposed RSA [31], which significantly reduces the negative impact of Byzantine workers and heterogeneous datasets.

RSA for Robust Distributed Learning. In the server-worker architecture, the goal of RSA [31] is to train a global model using the private data of the workers. The objective function introduces an l_1-norm regularized form to restrict the negative influence of Byzantine workers as follows:

$$\omega^* = \underset{\omega=[\omega_0;\omega_i]}{argmin} \sum_{i=1}^{n} (\mathbb{E}[L(\omega_i, \xi_i)] + \lambda ||\omega_0 - \omega_i||_1) + f_0(\omega_0), \quad (1)$$

where $\omega^* \in \mathbb{R}^{(r+1)d}$ is a vector of the desirable optimality for the global model, $L(\cdot)$ denotes the loss function of the local model for a data point ξ_i sampled from dataset $\mathcal{DB}_i$ of worker p_i, and $||\omega_0 - \omega_i||_1$ is the l_1-norm penalty, whose minimization enables ω_i to approach ω_0, λ is a positive constant and the last term $f_0(\cdot)$ is a regularization term. In RSA [31], each worker possesses a local ω_i, while the master server holds the global ω_0. At time $k+1$, in the ideal case in which the identities of Byzantine workers are revealed, the updates of the master server and worker p_i are:

$$\omega_0^{k+1} = \omega_0^k - \alpha^{k+1} \left(\nabla f_0(\omega_0^k) + \lambda \left(\sum_{i \in \mathcal{H}} \mathsf{sign}(\omega_0^k - \omega_i^k) \right) \right) \quad (2)$$

$$\omega_i^{k+1} = \omega_i^k - \alpha^{k+1} (\nabla L(\omega_i^k, \xi_i^{k+1}) + \lambda \mathsf{sign}(\omega_i^k - \omega_0^k)), \quad (3)$$

where α^{k+1} is the learning rate, $\nabla L(\cdot)$ denotes gradient of the loss function. $\mathsf{sign}(\cdot)$ is the elementwise sign function that returns the sign of the input. When the input to the function is greater than 0, it returns 1, and when it is less than 0, it returns -1; otherwise, it returns 0.

Rather than sending the value computed from (3) to the master server, Byzantine workers will send an arbitrary value, h_i^k. $\mathcal{H}$ is the set of honest workers and $\mathcal{B}$ is the set of Byzantine workers. Therefore, because the identities of Byzantine workers are indistinguishable, the update of the master server in the (k+1)-th iteration no longer follows (2), but:

$$\omega_0^{k+1} = \omega_0^k - \alpha^{k+1} \left(\nabla f_0(\omega_0^k) + \lambda \left(\sum_{i \in \mathcal{H}} \mathsf{sign}(\omega_0^k - \omega_i^k) + \sum_{i \in \mathcal{B}} \mathsf{sign}(\omega_0^k - h_i^k) \right) \right). \quad (4)$$

The above is the specific construction of the Byzantine-robust RSA scheme. RSA [31] lacks the privacy protection of ω_i sent by worker p_i to the master server, which can lead to the disclosure of workers' private information.

2.3 Shuffle Model

As illustrated in Fig. 1, a protocol $P = (\mathcal{R}, \mathcal{S}, \mathcal{A})$ in the shuffle model [36] consists of three randomized algorithms:

- Local Randomizer $\mathcal{R}: \mathcal{X} \rightarrow \mathcal{Y}$, $\mathcal{R}$ randomly outputs worker's messages, e.g., the protocol gives it a probability γ, and it will output its real value with a probability of $1-\gamma$ and uniform random noise otherwise.
- A shuffler $\mathcal{S}$: $\mathcal{Y} \rightarrow \mathcal{Y}$, the shuffler outputs the messages sent by the workers in a manner that applies a uniformly random permutation.
- Analyzer $\mathcal{A}$: $\mathcal{Y} \rightarrow \mathcal{Z}$, the analyzer computes on messages received from the shuffler $\mathcal{S}$ by the specific requirement of the model.

A shuffler (e.g., a mixnet [27]) collects all (locally randomized) messages, randomly permutes them, before

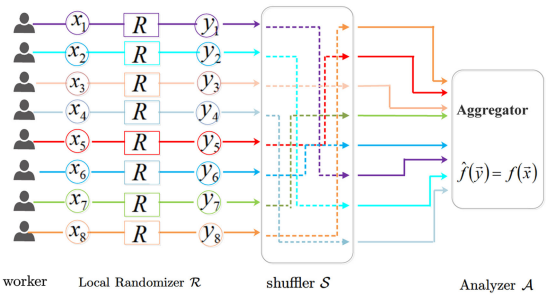


Fig. 1. A simple instance of the shuffle model.

sending them to the analyzer. The messages are anonymized so that the analyzer cannot link a message back to the message owner who generated it. In our scheme, each worker runs a local randomizer, the shuffler is treated as an abstract service that randomly permutes a set of messages, which implementation method is out of the scope of this paper, and the master server is the analyzer.

3 System Model and Security Model

In this section, we present the system model and security definition of privacy and robustness in our scheme.

3.1 System Model

As illustrated in Fig. 2, we divide the parties in our system into two classes: one is a master server that aggregates the local model parameters, the other consists of n workers, among which the b percent of workers are attacked by Byzantine adversaries. In contrast to classical federated learning, we integrate the shuffle protocol into our system to further protect workers' privacy.

Our system consists of four steps: (1) Initialization: All the participants perform an initialization operation to obtain the model parameters and the shuffle protocol during the first iteration; (2) LocUpdate: Each worker trains the local model based on its dataset and the global model and computes a matrix of signs; (3) Shuffle: The sign matrices from workers are shuffled and sent to the master server; (4) Aggregation: The master server aggregates the data from workers and updates the global model. Finally, each participant returns to step (2) to start the next iteration. Iteration continues until the learning process reaches the optimum or a maximum number of iterations.

DPBFL consists of four subalgorithms as follows:

- $(\omega_0^0, \omega_1^0, \ldots, \omega_n^0, \gamma) \leftarrow \mathsf{Init}(1^n)$: Initialize all the participants' learning models $\omega_0^0, \ldots, \omega_n^0$ and choose a parameter $\gamma \in (0, 1)$ that will be used in the shuffle protocol.
- $x_i^{k+1} \leftarrow \mathsf{LocUpdate}(\omega_0^k, \omega_i^k), i = 1, 2, \ldots, n$: In the $(k{+}1)$th iteration, the master server broadcasts the parameters of its local model to all the workers, and then each worker p_i computes $\mathsf{sign}(\cdot)$. Each worker updates the local model based on these signs and its dataset.
- $y_i^{k+1} \leftarrow \mathsf{Shuffle}(\gamma, x_i^{k+1}), i = 1, 2, \ldots, n$: In the $(k{+}1)$th iteration, the signs received from worker p_i are shuffled and then uploaded to the master server.

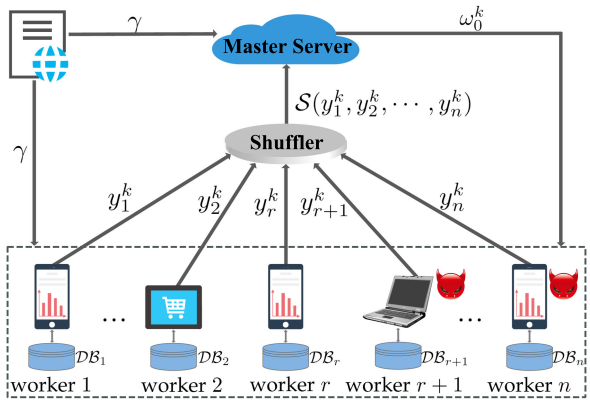


Fig. 2. Framework of differentially private Byzantine-robust federated learning.

- $\omega_0^{k+1} \leftarrow \mathsf{Agg}(\gamma, y_1^{k+1}, \ldots, y_n^{k+1}), i = 1, 2, \ldots, n$: The master server aggregates all the shuffled signs and updates its learning model in the $(k{+}1)$th iteration.

3.2 Security Model

The definition of differential privacy (DP) proposed by Dwork *et al.* [22] in 2006 is as follows:

Definition 1. *(Differential Privacy) An algorithm A: $T \rightarrow S$ satisfies (ε, δ)-differential privacy if, for every pair of neighboring datasets T and T' and every subset $S \subset T$,*

$$\mathbb{P}[A(T) \in S] \leq e^{\varepsilon}\mathbb{P}[A(T') \in S] + \delta,$$

where T and T' are neighboring datasets that differ in at most one piece of data, ε is the privacy budget and denotes the distinguishable bound of all outputs on neighboring datasets; δ is the probability that the difference of the output distributions from A when using T and T' cannot be bounded by ϵ. DP is not impacted by post-processing. Formally, let $A{:}T \rightarrow S$ be (ε, δ)-differential privacy mechanism and $P{:}S \rightarrow S'$ be non-differential privacy mechanism, then $P \circ A$: $T \rightarrow S'$ is (ε, δ)-differential privacy, where the notation $\circ$ denotes a composition, that is, the output of A is the input of P.

However, local differential privacy (LDP) in distributed learning is a model of differential privacy with the added restriction that even if the master server and an adversary has access to the workers' shared parameters of an individual, they will still unable to learn too much about the worker's dataset.

Definition 2. *(Robustness) Robustness is that when the proportion b of Byzantine workers is less than $(1 - \frac{1}{2-\gamma})$ throughout, the global model can still converge to the optimum.*

Where $\gamma \in (0, 1)$ is the probability that a worker will send uniform random noise in shuffle model. In our scheme, the robustness is mainly affected by two aspects: Byzantine workers who may send malicious messages, and the honest workers who follow the SPS protocol may send random noise (to achieve differential privacy). However, our scheme can still converge to the optimum and has good robustness with the $(1 - \frac{1}{2-\gamma})$ percent of Byzantine workers.

4 Our Construction

We build DPBFL upon the baseline Byzantine-robust federated learning algorithm RSA presented by Li *et al.* [31] for

robustness, which is described in Section 2.2. To realize differential privacy, we utilize the shuffle model to construct a differentially private summation protocol(SPS). In the construction of the DPBFL scheme, the worker and master server comply with the SPS protocol in the process of parameter uploading and aggregation respectively to achieve parameter privacy protection and accurate aggregation, as shown in the last two subalgorithms of Section 4.2. Finally, we present the security analysis of the DPBFL scheme.

4.1 Shuffle Protocol for Summation (SPS)

SPS protocol $\mathcal{P}_{n,\gamma} = (\mathcal{R}, \mathcal{S}, \mathcal{A})$ consists of three components, Local Randomizer $\mathcal{R}$, Shuffler $\mathcal{S}$ and Analyzer $\mathcal{A}$. According to the given probability γ, the specific structure is as follows:

- Local Randomizer $\mathcal{R}$: The local randomizer $\mathcal{R}$ randomizes the worker's message $x \in \{-1, 0, 1\}$. It first outputs a bit $g \leftarrow \text{Ber}(\gamma)$:

$$g = \begin{cases} 0 & \text{with the possibility of } 1-\gamma \\ 1 & \text{with the possibility of } \gamma \end{cases}. \tag{5}$$

  As shown in formula (5), the function $\text{Ber}(\gamma)$ in Algorithm 1 outputs 0 or 1 according to probability γ. Then, according to the result in formula (5), it outputs y:

$$y = \begin{cases} x & g = 0 \\ \text{Unif}(\{\text{-1,0,1}\}) & g = 1 \end{cases}. \tag{6}$$

  The bit g indicates whether the worker outputs the truthful value or a uniformly random value from $\{-1, 0, 1\}$.
- Shuffler $\mathcal{S}$: The shuffler receives y from each local randomizer, then randomly permute them before outputting them to the analyzer, we omit this step in Algorithm 1.
- Analyzer $\mathcal{A}$: The analyzer $\mathcal{A}$ estimates the sum on values $y_i (i = 1, \ldots, n)$ as follows:

$$z \leftarrow \frac{1}{1-\gamma} \sum_{i=1}^{n} y_i. \tag{7}$$

Algorithm 1. A Shuffle Protocol $\mathcal{P}_{n,\gamma} = (\mathcal{R}, \mathcal{S}, \mathcal{A})$ for Computing the Sum of $\{-1, 0, 1\}$

```
//Local Randomizer R
Input: x ∈ {−1, 0, 1}, parameters n ∈ ℕ, γ ∈ (0, 1)
Output: y ∈ {−1, 0, 1}
Sample g ← Ber(γ)
if g=0 then
  Let y ← x
else
  Sample y ← Unif({−1, 0, 1})
end if
return y
//Analyzer A
Input: (y_1, ..., y_n) ∈ {−1, 0, 1}^n, parameters n ∈ ℕ, γ ∈ (0, 1)
Output: z ∈ [−n, n]
Let z ← 1/(1−γ) Σ_{i=1}^{n} y_i
return z
```

4.2 The DPBFL Scheme

DPBFL consists of four subalgorithms, each of which is specified below:

- $(\omega_0^0, \omega_1^0, \ldots, \omega_n^0, \gamma) \leftarrow \mathsf{Init}(1^n)$: Initialize all the participants' learning models $\omega_0^0, \ldots, \omega_n^0$ and generate a probability $\gamma \in (0, 1)$, that will be used in the shuffle operation.
- $x_i^{k+1} \leftarrow \mathsf{LocUpdate}(\omega_0^k, \omega_i^k), i = 1, 2, \ldots, n$: In the ($k$+1) th iteration, worker p_i has a local learning model ω_i^k and downloads the master server's learning model ω_0^k, and then computes

$$x_i^{k+1} = \mathsf{sign}(\omega_0^k - \omega_i^k). \tag{8}$$

  Worker p_i updates its local learning model ω_i^{k+1} using private dataset $\mathcal{DB}_i$ and current x_i^{k+1} by a deep learning algorithm $\mathcal{DL}_i$ is given by

$$\omega_i^{k+1} = \omega_i^k - \alpha^{k+1}(\nabla L(\omega_i^k, \xi_i^{k+1}) - \lambda x_i^{k+1}), \tag{9}$$

  where $\mathsf{sign}(\cdot)$ is the elementwise sign function, α^{k+1} is the learning rate, $\nabla L(\cdot)$ denotes the local loss gradient of worker p_i, and λ is a positive constant.
- $y_i^{k+1} \leftarrow \mathsf{Shuffle}(\gamma, x_i^{k+1}), i = 1, 2, \ldots, n$: Worker p_i launches the local randomizer $\mathcal{R}$ of shuffle protocol for summation (SPS) using x_i^{k+1} as the input, obtains the noisy output y_i^{k+1}, then uploads it shuffled to the master server. The formal representation is as follows:

$$y_i^{k+1} = \mathcal{R}(x_i^{k+1}). \tag{10}$$

- $\omega_0^{k+1} \leftarrow \mathsf{Agg}(\gamma, y_i^{k+1}), i = 1, 2, \ldots, n$: In the ($k$+1)th iteration, the master server performs a summation operation on the values y_i^{k+1} obtained from the honest workers in $\mathcal{H}$ and the h_i^{k+1} values obtained from unidentified Byzantine workers in $\mathcal{B}$. Then, it computes an approximate solution z^{k+1} according to the analyzer $\mathcal{A}$ of SPS by:

$$z^{k+1} = \frac{1}{1-\gamma}\left(\sum_{i \in \mathcal{H}} y_i^{k+1} + \sum_{i \in \mathcal{B}} h_i^{k+1}\right). \tag{11}$$

  Subsequently, the master server updates the local model

$$\omega_0^{k+1} = \omega_0^k - \alpha^{k+1}(\nabla f_0(\omega_0^k) + \lambda z^{k+1}), \tag{12}$$

  where the $\nabla f_0(\cdot)$ denotes the gradient of regularization term.

Repeat the last three steps until the stop condition is reached. To better protect the privacy of the worker p_i, the worker's local model ω_i^{k+1} is private and not sent to the master server. Finally, we privately train an available global model ω_0^*.

4.3 Security Analysis

In this section, we present the privacy analysis and convergence analysis of DPBFL. The results demonstrate that DPBFL satisfies differential privacy and converges to the approximate optimal solution. To show the superiority of our scheme in achieving a trade-off between privacy and utility, we present the comparison of SPS and the LDP-based protocol for summation.

Privacy Analysis. In the following analysis, we show the level of DP that can be achieved when there are n honest workers in our scheme. We do not consider Byzantine workers here because as shown in formula (11), the summation can be viewed as to first sum up the honest workers' y_i, then sum the result with the Byzantine workers' h_i. This step can be viewed as post-processing, hence the Byzantine workers' inputs will not weaken the privacy level of the honest workers. Moreover, the shuffle protocol in our scheme has the parallel composition property [37] of differential privacy, i.e., the execution of the shuffle protocol $\mathcal{P}_{n,\gamma}$ on each worker satisfies (ϵ, δ)-differential privacy, DPBFL is (ϵ, δ)-differentially private.

According to the shuffle protocol $\mathcal{P}_{n,\gamma}$, approximate γn workers will randomly select a value from $\{-1, 0, 1\}$ and send it to the master server as random noise, and approximate $(1-\gamma)n$ workers will submit their truthful signs, we then use histograms Y_1 and Y_2 to represent the statistics of random noise and truthful signs, respectively. The information obtained by the master server is the histogram $Y = Y_1 \cup Y_2$ figured by the mixture of Y_1 and Y_2.

Without loss of generality, we assume two neighboring datasets $\mathcal{D} = (x_1, x_2, \ldots, x_n)$ and $\mathcal{D}' = (x_1, x_2, \ldots, x'_n)$ that represent the sets of workers' truthful signs as the input of local randomizer and differ on x_n from the nth worker p_n. We further assume that the master server knows the inputs from all workers except the nth worker p_n.

We then set the value γ of protocol $\mathcal{P}_{n,\gamma}$, so that when computing on these neighboring datasets in which only the nth worker's value changes, Y can change in an appropriate limited amount. Intuitively, the privacy of protocol $\mathcal{P}_{n,\gamma}$ is essentially set γ to allow Y_1 to hide x_n appropriately. The proofs of Theorems 1, 2, 3, 4, 5 and Corollary 1 are in the Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2022.3167434.

Theorem 1. *For $n \in \mathbb{N}, \epsilon \in (0,1)$, $\delta \in (0,1)$, and $\gamma = max\{\frac{42\log(2/\delta)}{(n-1)\epsilon^2}, \frac{81}{(n-1)\epsilon}\} < 1$, $\mathcal{P}_{n,\gamma}$ is (ϵ, δ)-differentially private.*

Corollary 1. *For $n \in \mathbb{N}, \epsilon \in (0,1)$, $\delta \in (0, 2e^{-27/14})$, and $\gamma = \frac{42\log(2/\delta)}{(n-1)\epsilon^2} < 1$, $\mathcal{P}_{n,\gamma}$ is (ϵ, δ)-differentially private.*

Concerning the precision of SPS, the upper bound of the mean squared error is

Theorem 2. *For $n \in \mathbb{N}, \epsilon \in (0,1)$, $\delta \in (0, 2e^{-27/14})$, and $n \geq 1 + \frac{42\log(2/\delta)}{(n-1)\epsilon^2}$, and $X = (x_1, \ldots, x_n) \in \{-1, 0, 1\}^n$*

$$MSE(\mathcal{P}_{n,\gamma}(X)) = O\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right).$$

Note that the estimation is unbiased, in which case the mean square error is equal to the variance. When we preset the $\gamma > \frac{8}{3n}\log\frac{2}{\beta}$, we can obtain the upper bound of the probability so that the deviation is higher than a small threshold:

Theorem 3. *For $n \in \mathbb{N}, \epsilon, \delta \in (0,1)$, $n \geq 1 + \frac{84\log(2/\delta)}{(n-1)\epsilon^2}$, $X = (x_1, \ldots, x_n) \in \{-1, 0, 1\}^n$, $\gamma > \frac{8}{3n}\log\frac{2}{\beta}$, and $\beta \in (0,1)$,*

$$\mathbb{P}\left[\left|\mathcal{P}_{n,\gamma}(X) - \sum_{i=1}^{n} x_i\right| > \frac{6}{\epsilon}\sqrt{22\log\frac{2}{\delta}\log\frac{2}{\beta}}\right] < \beta.$$

In DPBFL, we utilize shuffle protocol $\mathcal{P}_{n,\gamma}$ to realize privacy and accuracy improvements by a shuffler. Similarly, LDP can be directly utilized in constructing a local differentially private Byzantine-robust federated learning scheme (LDPBFL). But, we will prove that the utility of the shuffle-based scheme (DPBFL) is superior to the LDP-based scheme (LDPBFL).

We define the LDP-based protocol as $\mathcal{P}'_{n,\gamma} = (\mathcal{R}, \mathcal{A})$, in which the shuffler $\mathcal{S}$ is removed so that the worker's output is no longer anonymized. We formally analyze the privacy budget of the LDP-based scheme and draw the following conclusion.

Theorem 4. *For $\gamma \in (0,1)$, $\mathcal{P}'_{n,\gamma}$ is ϵ-local differentially private, where $\varepsilon = \log\left(\frac{1-\frac{2}{3}\gamma}{\frac{1}{3}\gamma}\right)$.*

In Corollary 1, we proved that $\mathcal{P}_{n,\gamma}$ is (ϵ, δ)-differentially private, where $\epsilon = \sqrt{\frac{42\log(2/\delta)}{(n-1)\gamma}}$ and $\delta \in (0, 2e^{-27/14})$. The result shows that the privacy budget of $\mathcal{P}_{n,\gamma}$ will drop down when the number of the worker grows. However, Theorem 4 shows that the privacy of LDPBFL is not affected by the number of the workers. In the following numerical experiments, we compare the impact of different privacy budgets on the accuracy of the models obtained from these two schemes in detail.

Convergence Analysis. The main challenge of ensuring model convergence is that the Byzantine workers can collude and send arbitrary malicious values to bias the optimization process. We assume the proportion of Byzantine workers is $b < 1 - \frac{1}{2-\gamma}$ throughout, that is, the proportion of honest workers sending their truthful values is always greater than the proportion of Byzantine workers to ensure the model is updated in the correct direction. Then, we further prove the convergence of the model in the scheme.

Assumption 1. *(Strong convexity and Lipschitz continuity of gradients) The local loss functions $\mathbb{E}[L(\omega_i, \xi_i)]$ and the regularization term $f_0(\omega_0)$ are μ_i and μ_0-strongly convex, respectively, and have Lipschitz continuous gradients with constants L_i and L_0, respectively.*

Assumption 2. *(Bounded variance) For every worker i, the data sampling is i.i.d. across time such that $\xi_i^k \sim \mathcal{D}_i$. The variance of $\nabla L(\omega_i^k, \xi_i^k)$ is upper bounded by δ_i^2. That is, $\mathbb{E}[||\nabla \mathbb{E}[L(\omega_i, \xi_i)] - \nabla L(\omega_i, \xi_i)||^2] \leq \delta_i^2$.*

Assumptions 1 and 2 are standard for performance analysis of stochastic gradient-based methods [38], the former is standard in convex analysis, and the other bounds the variation of gradients. We show our scheme's convergence under Assumptions 1 and 2 against the Byzantine workers.

Theorem 5. *Under Assumptions 1 and 2, we have*

$$\mathbb{E}||\omega_0^{k+1} - \omega_0^*||^2 \leq (1 - \eta\alpha^{k+1})\mathbb{E}||\omega_0^k - \omega_0^*||^2 + \alpha^{2(k+1)}\Delta_1 + \Delta_2, \tag{13}$$

where

$$\Delta_1 = \left(\frac{2\lambda^2 d}{(1-\gamma)^2}\left(4r^2 + b^2 n^2\right) + 8\lambda^2 r^2 d\right)$$

$$\Delta_2 = \frac{198\alpha^{k+1}\lambda^2 d}{\xi\epsilon^2}\log\frac{1}{\delta}$$

TABLE 1
Accuracy Comparison Between RSA [31] and DPBFL

	γ	$b = 0\%$	$b = 30\%$	$b = 40\%$	$b = 50\%$
RSA [31]	0	89.96%	89.92%	89.88%	5.08%
DPBFL	0.283	89.92%	89.61%	87.55%	0.18%
	0.383	89.90%	89.51%	0.16%	0.17%
	0.483	89.89%	89.07%	0.13%	0.17%

$$\eta = \left(\frac{2\mu_0 L_0}{\mu_0 + L_0} - \bar{\xi}\right).$$

Theorem 5 indicates that the local iterative sequence sublinearly converges to the approximate optimal solution of (1) and is quadratically dependent on the proportion of Byzantine workers b and the number of honest workers r.

5 Efficiency Analysis and Experimental Evaluations

In this section, we first report the results showing that our scheme is comparable to Byzantine-robust stochastic aggregation one (RSA) [31] in terms of efficiency and then provide a systematic comparison with several representative cryptographic approaches. Finally, we test the performance and robustness of the model and the utility of SPS through experiments on the MNIST, FashionMNIST, CI- FAR10 datasets.

5.1 Efficiency Analysis and Comparison

DPBFL is better than RSA [31] in the communication efficiency and computation efficiency theoretically. On the one hand, unlike worker in RSA proposed by Li *et al.* [31] directly transfers the model parameters computed from (3) to the master server, in our scheme, the worker transfers only a one-bit sign computed from (8) to the master server. Therefore, under the same scene, the number of bits transferring to the mater server in our scheme is much less, which makes our scheme is more efficient than RSA [31] in communication. On the other hand, since the values received by the master server are model parameters and signs respectively in the two schemes, resulting in the computation efficiency for aggregation of the master server in DPBFL is better than that in RSA [31]. In addition, the random output required by the shuffle protocol is only a simple random selection within the worker, and the consumption of computing resources and time is negligible. Therefore, the efficiency in the communication and computation of DPBFL is better than RSA [31]. Furthermore, as shown in Table 1, the model accuracy of DPBFL is close to RSA [31] when the Byzantine workers are less than 40%, and according to Theorem 2, once b is close to $(1 - \frac{1}{2-\gamma})$, the model may be controlled by Byzantine workers and the model accuracy decreases sharply.

In Section 4.3, we present the privacy analysis and efficiency analysis of DPBFL. In Table 2, we additionally provide a systematic comparison with other representative cryptographic approaches. BatchCrypt [39] is a system solution for cross-silo FL that substantially reduces the encryption and communication overhead caused by homomorphic encryption (HE), but HE will still bring a large computation overhead. Bonawitz *et al*. [40] design a communication-efficient, failure-robust protocol for secure aggregation using secure multiparty computation (SMC), but they did not verify whether their scheme can defend against inference attacks. Hybrid [41] utilizes both DP and SMC to defend against inference attacks and improve model accuracy. However, the SMC method is inefficient. LDP-Fed [26] is an efficient federated learning system with a formal privacy guarantee using local differential privacy (LDP). Moreover, none of the above-mentioned schemes takes effective means to defend against Byzantine attacks.

However, DPBFL does not require computation-intensive cryptographic protocols and can defend against inference attacks because the data exchanged throughout the entire training phase is differentially private. At the same time, DPBFL also reduces the negative impact on Byzantine workers.

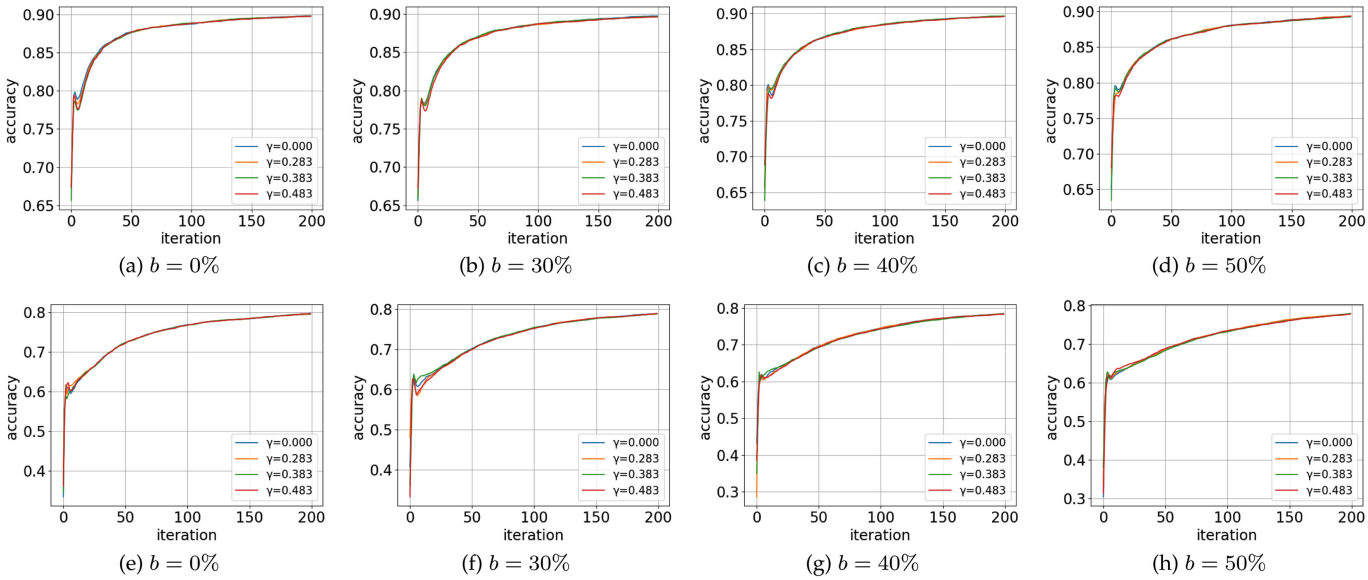
5.2 Experimental Evaluations

In this section, we first evaluate the threshold of our SPS to ensure that our scheme satisfies differential privacy and guarantees convergence. Then, we evaluate the robustness of our model under the presence of Byzantine workers and compare our scheme with RSA[31] and other schemes that combine different aggregation rules (Median[18], Krum [20]). Finally, we experiment on the influence of the worker scale on the accuracy of our model and do a comparative experiment to compare the accuracy of DPBFL using shuffle protocol and LDPBFL using LDP under the same privacy budget.

Experiment Settings. (1) In the experiments, we used the MNIST and FashionMNIST datasets. The MNIST dataset contains 60,000 training samples and 10,000 testing samples with 10 classes, and each data sample is a one-dimensional matrix with a length of 784 containing a 28×28-pixel handwriting image. FashionMNIST dataset also covers a total of 70,000 samples and has the same size, format, and training/test set partition as the MNIST dataset. Unlike the MNIST handwriting dataset, the samples in FashionMNIST are images of different products from 10 categories. (2) To

TABLE 2
Comparison of Cryptography Tools for Private Federated Learning

Privacy-Preserving Federated Learning Scheme	Cryptography Tools	Efficient	Defenses Inference Attacks	Byzantine-robust
DPBFL	DP	✓	✓	✓
BatchCrypt [39]	HE	✗	✓	✗
Practical Secure Aggregation [40]	SMC	✗	–	✗
Hybrid [41]	DP+SMC	✗	✓	✗
LDP-Fed [26]	DP	✓	✓	✗



(a) $b = 0\%$ (b) $b = 30\%$ (c) $b = 40\%$ (d) $b = 50\%$

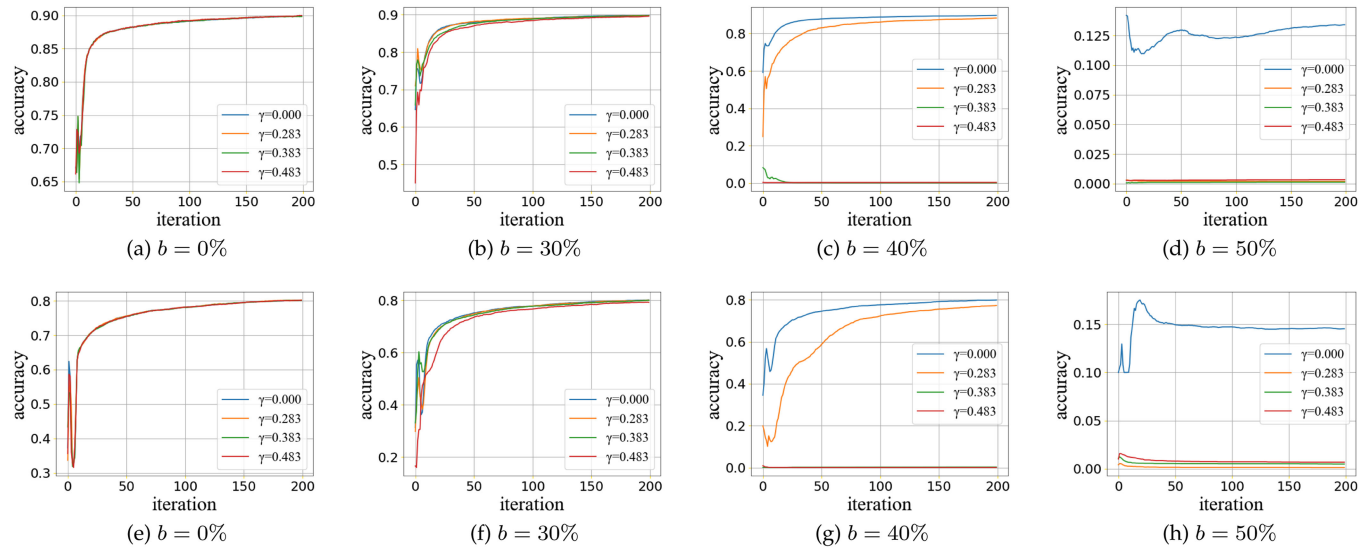
(e) $b = 0\%$ (f) $b = 30\%$ (g) $b = 40\%$ (h) $b = 50\%$

Fig. 3. Threshold Experiments(Gaussian attack). (a-d) and (e-h) shows the accuracy of DPBFL in different γ at $b = 0\%,\ 30\%,\ 40\%,$ and 50% on MNIST and FashionMNIST dataset, respectively.

simulate non-i.i.d. training data, we use Label skewing [42] in which the data is distributed based on the Dirichlet distribution with a specific parameter $\alpha = 0.2$. We also show the data distribution for 10 random workers in Fig. 5. (3) We simulated 1,000 worker processes and 1 master server process on a computer with an AMD Ryzen 9 4900H with a Radeon Graphics CPU @ 3.30 Mhz. (4) We used softmax regression with a batch size of 32 and used test accuracy as the performance metric. In all experiments, unless otherwise specified, the Byzantine attack is the sign-flipping attack, which is the most aggressive Byzantine attacks against DPBFL.

Threshold Experiments. We show how the percentage of Byzantine workers (b) and the probability of the honest worker sending random value in SPS (γ) impact the performance of learning. Figs. 3 and 4 show how fast the learning process converges and the maximum accuracy can be achieved.

Fig. 3 shows that under Gaussian attack, the model utility of DPBFL has little effect at 50% Byzantine workers. Fig. 4 shows the model utility of DPBFL under Gaussian attack. When $\gamma = 0$, DPBFL is essentially RSA without differential privacy. In figure (a) with $b = 0\%$ on the MNIST dataset, we can see with larger γ, i.e., with stronger DP, the convergence speed and the maximum accuracy are similar to RSA, and DPBFL introduces random noise to protect privacy, the initial convergence speed and accuracy are slightly lower than that of RSA. Figures (b)– (d) show the results with larger b. We can see the learning performance is not significantly affected when $b = 30\%$. However, when b is larger than 40%, we can observe a significant drop in the maximum accuracy. The specific values are recorded in Table 1. In



(a) $b = 0\%$ (b) $b = 30\%$ (c) $b = 40\%$ (d) $b = 50\%$

(e) $b = 0\%$ (f) $b = 30\%$ (g) $b = 40\%$ (h) $b = 50\%$

Fig. 4. Threshold Experiments(sign-flipping attack). (a-d) and (e-h) shows the accuracy of DPBFL in different γ at $b = 0\%,\ 30\%,\ 40\%,$ and 50% on MNIST and FashionMNIST dataset, respectively.

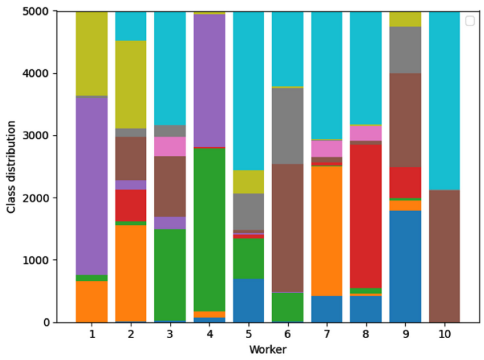


Fig. 5. Synthetic populations generated from Dirichlet distribution with concentration parameters $\alpha = 0.2$. Distribution among classes is represented with different colors and populations.

figures (e)– (h), we can observe similar results using the FashionMNIST dataset.

Therefore, as a result of the threshold experiment, when $\gamma = 0.283$ (minimum γ at n = 1000) and the percentage of Byzantine workers exceeds 50%, the accuracy of our model decreases with the number of iteration rounds, and the model does not converge. Even so, the DPBFL scheme guarantees convergence of the federated model when less than 40% of Byzantine workers participate, so it has enhanced privacy protection while superior Byzantine robustness.

Robustness Comparison. In this experiment, we compare the performance of DPBFL, RSA, Krum, federated average (FedAvg), Median.

Fig. 6 shows that under Gaussian attack, the accuracy of DPBFL is slightly lower than that of RSA but superior to Median, Krum, and FedAvg. Fig. 7 shows the model utility of these Byzantine-robust schemes under the Sign-flipping attack, note that when $b = 0\%$ in figure (a), DPBFL, RSA, and Median have approximate accuracy, they are slightly less accurate than FedAvg and higher than Krum. In figure (b), we can observe that the accuracy of FedAvg, DPBFL, RSA, and Median are similar and significantly higher than the Krum. In figure (c), the accuracy of DPBFL is lower than RSA and similar to Median, and at this time, the FedAvg scheme failed. In the case of $b = 50\%$, the DPBFL's model no longer converges, and other robust methods cannot guarantee the robustness of their model. Figures (e)– (h) show the test results on the FashionMNIST dataset, all the patterns are consistent with the MNIST dataset.

Therefore, compared with other schemes, ours still has good Byzantine robustness under the premise of increased privacy protection.

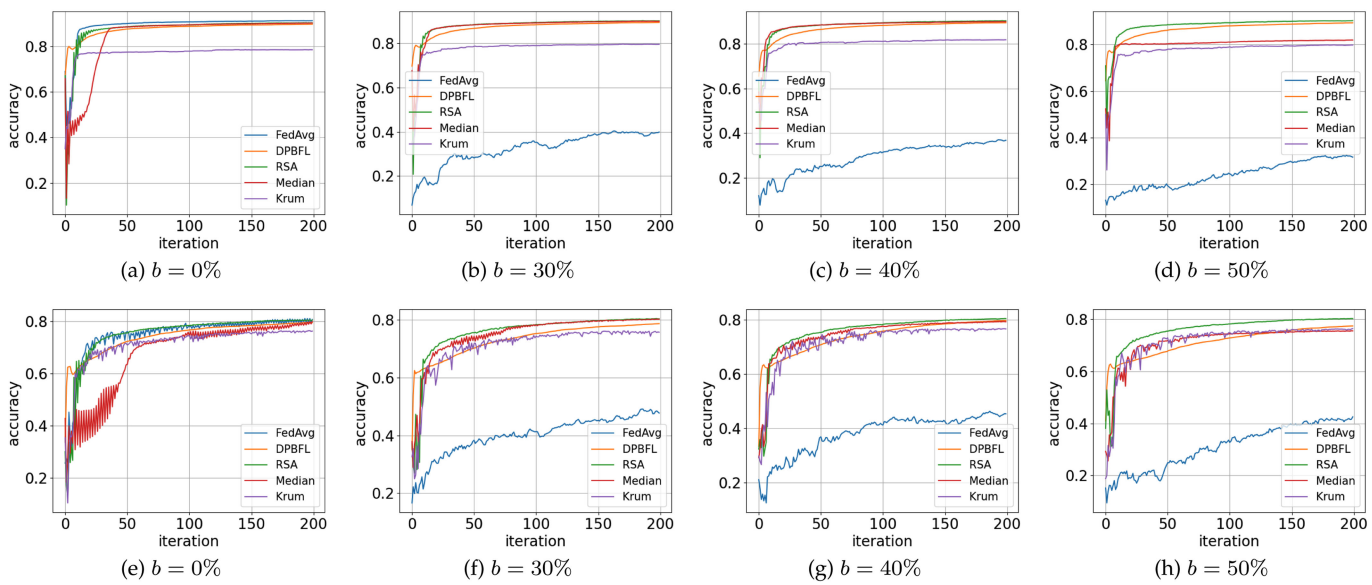
Runtime Comparison. Fig. 8 shows the runtime of the schemes when $b = 30\%$. The total number of iterations for every scheme is 100. As shown in Fig. 8, DPBFL has a faster per-iteration runtime than does RSA and Median and slightly slower than does FedAvg, and the accuracy of DPBF is only lower than RSA.

In conclusion, due to the additional computational costs incurred when disposing of Byzantine workers, DPBFL is only less efficient than the benchmark FedAvg but much more efficient than other robust aggregation schemes, and in the same number of rounds, our scheme maintains good accuracy. Therefore, DPBFL achieves an effective trade-off between efficiency and availability.

Impact of Worker Scale. We also tested our scheme in settings with 1,000, 2,000, 3,000, 4,000, and 5,000 workers and $\gamma = 0.283$ to evaluate the change in accuracy under different numbers of workers. The experimental results in Fig. 9 compare the accuracies of RSA and DPBFL on the FashionMNIST dataset, showing that the accuracy of both schemes increases steadily as the number of workers increases, and the accuracy of DPBFL is close to RSA.

Therefore, the experimental results show that in the large-scale worker application scenario, our scheme can achieve similar performance to the RSA scheme, and at the same time is better than the RSA scheme in terms of privacy.

Utility Comparison. We do this comparative experiment on the MNIST dataset with 150 iterations and 20% of byzantine workers to compare the utility of shuffle protocol and LDP by comparing the accuracy of the models under the different $\varepsilon \in (0, 1)$ in the DPBFL and LDPBFL schemes.



(a) $b = 0\%$ (b) $b = 30\%$ (c) $b = 40\%$ (d) $b = 50\%$

(e) $b = 0\%$ (f) $b = 30\%$ (g) $b = 40\%$ (h) $b = 50\%$

Fig. 6. Robustness Comparison(Gaussian attack). (a-d) and (e-h) shows the accuracy of DPBFL, RSA, FedAvg, Median, and Krum at $b = 0\%,\ 30\%,\ 40\%,$ and 50% on MNIST and FashionMNIST dataset, respectively.

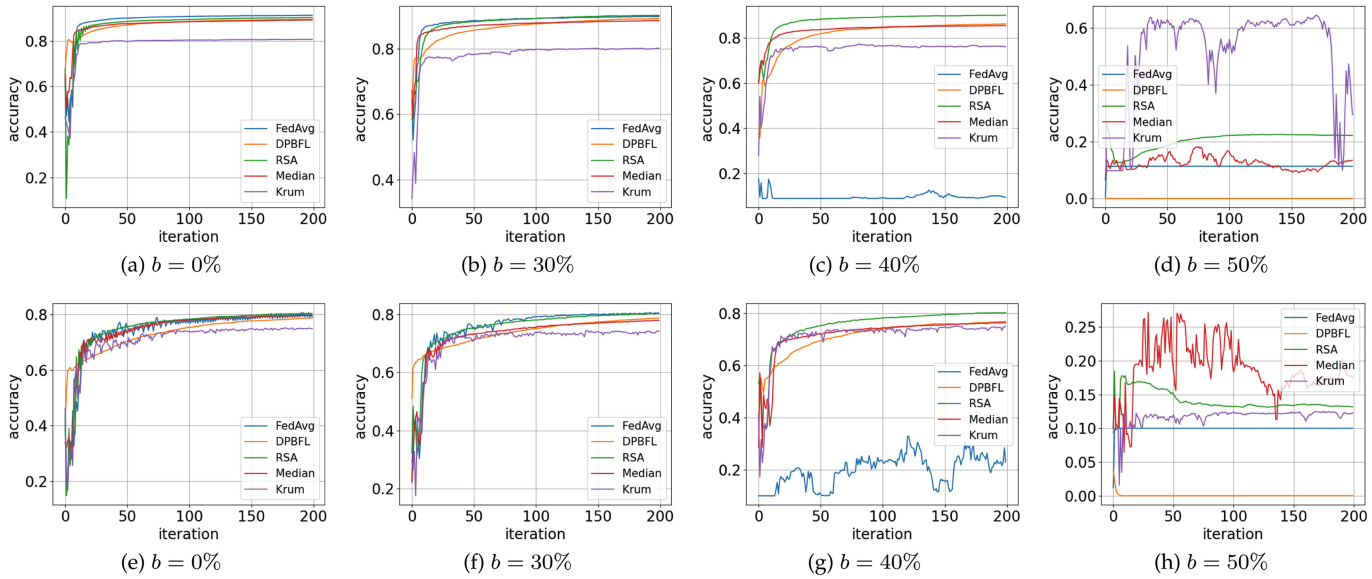


Fig. 7. Robustness Comparison(sign-flipping attack). (a-d) and (e-h) shows the accuracy of DPBFL, RSA, FedAvg, Median, and Krum at $b = 0\%,\ 30\%,\ 40\%,\ \text{and}\ 50\%$ on MNIST and FashionMNIST dataset, respectively.

In DPBFL, since the shuffle protocol has a constraint $\frac{42\log(2/\delta)}{(n-1)\epsilon^2} < 1$ with $n \in \mathbb{N},\ \epsilon \in (0,1),\ \delta \in (0, 2e^{-27/14})$, and DPBFL needs to satisfy the definition of robustness, i.e., the number of Byzantine workers is less than $(1 - \frac{1}{2-\gamma})$. Therefore, with fixing $\delta = 10^{-6}$, we can compute the lower bound of the ε under n= 10000, 50000, and 100000 respectively in Table 3, and then we compare the accuracy difference between the two schemes under the same privacy budget in Fig. 10.

We can observe through all the figures and tables that although the accuracy of DPBFL model is lower than LDPBFL at $n = 10000$ and $\varepsilon < 0.2$, once the number of workers is up to 50,000 or a lower value, DPBFL can achieve a smaller ε while having the higher accuracy than LDPBFL, especially when $\varepsilon < 0.4$, the difference in accuracy is more obvious.

Therefore, compared with LDP, our shuffle protocol can achieve higher utility and privacy in large-scale worker application scenarios.

Performance on DNN and CIFAR10. We aim to develop a convergence theory that is relevant for real problems in federated learning. For this reason, we use logistic regression functions which abide by strong convexity assumptions to perform experimental verification. Moreover, we also use DNN whose loss function is usually not convex to verify the effectiveness of our scheme.

Due to the complex parameters of the DNN, we use the model accuracy of a single worker as the test metric and every worker in FL trains Alexnet on the CIFAR10 dataset. (1)AlexNet [43] is a convolutional neural network architecture, designed by Alex Krizhevsky in collaboration with Ilya Sutskever and Geoffrey Hinton. In PyTorch, the model's subpackage torchvision.models contains definitions for the AlexNet model architectures [44] for image classification. Specifically, AlexNet contained eight layers: the first five were convolutional layers, some of them followed by max-pooling layers, and the last three were fully connected layers. It used the non-saturating ReLU activation function, which showed improved training performance over tanh and sigmoid. (2) The CIFAR10 dataset consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. The test batch contains exactly 1,000 randomly-selected images from each class. (3) This is epoch training where each

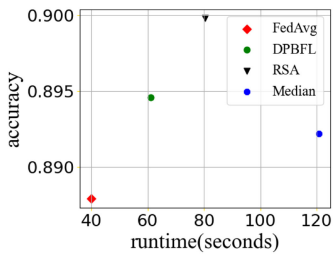


Fig. 8. Runtime comparison with different schemes under $b = 30\%$ and MNIST dataset.

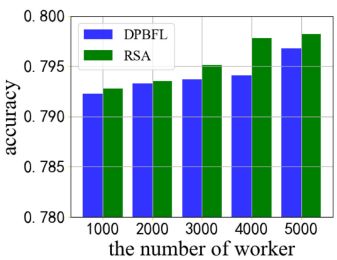


Fig. 9. Accuracy comparison with the increase of workers under $b = 30\%$ and FashionMNIST dataset.

TABLE 3
Minimum Privacy Budget Required for Shuffle Protocol Under Different Number of Workers in DPBFL

n	ε
10000	0.28505544898604424
50000	0.12747557282703412
100000	0.09013839128179175

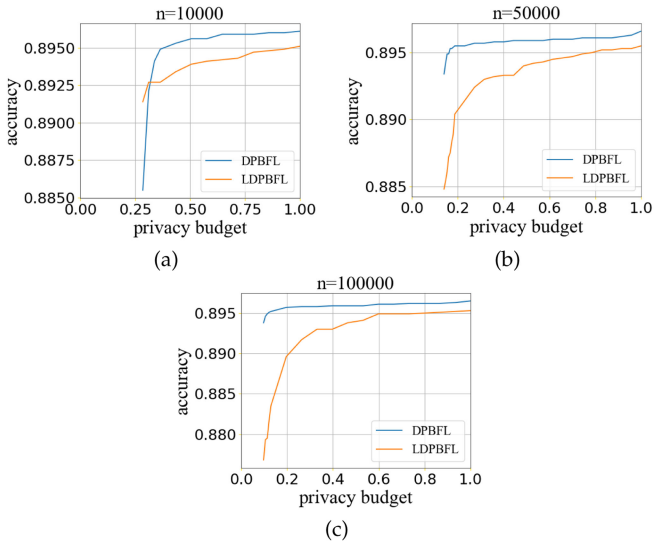


Fig. 10. Utility Comparison. With $b = 20\%$ and $\delta = 10^{-6}$, the utility comparison of shuffle protocol in DPBFL and LDP in LDPBFL with the increase of the number of participant workers on MNIST dataset.

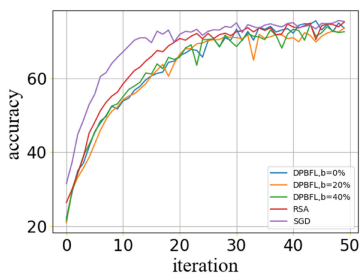


Fig. 11. Performance on DNN and CIFAR10: The accuracy of SGD, RSA and DPBFL with b=0%, 20% and 40%.

worker takes 200 local updates on its samples, and at every epoch, every worker estimates its local gradient on the batch of 64 samples. (4) We built the federated learning framework in Pytorch and simulated 100 worker processes and 1 master server process.

In Fig. 11, we can observe that the DPBFL scheme converges a little slower than the SGD and RSA schemes, and the accuracy of the DPBFL scheme at the 50th iteration is slightly lower than that of the SGD and RSA schemes. In the presence of 40% Byzantine workers, DPBFL can still converge to good accuracy. As long as the DPBFL scheme still converges on AlexNet and CIFAR10, all conclusions can be extended to DNN and more complex datasets.

6 Conclusion

In this paper, by integrating an efficient shuffle protocol SPS with federated aggregation method RSA, we develop a differentially private Byzantine-robust federated learning scheme (DPBFL) under the presence of an honest-but-curious master server and Byzantine adversaries. To ensure DPBFL's privacy and robustness, we also provide theoretical proof that our scheme can resist privacy leakage caused by model parameters exchanged during the learning process, and verify its high performance with the experiments on the MNIST, FashionMNIST, CIFAR10 datasets.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[2] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, "The secret revealer: Generative model-inversion attacks against deep neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 253–261.

[3] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 739–753.

[4] X. Zhang, X. Chen, H. Yan, and Y. Xiang, "Privacy-preserving and verifiable online crowdsourcing with worker updates," *Inf. Sci.*, vol. 548, pp. 212–232, 2021.

[5] X. Chen *et al.*, "Publicly verifiable databases with all efficient updating operations," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 12, pp. 3729–3740, Dec. 2021.

[6] X. Ma *et al.*, "Secure multiparty learning from the aggregation of locally trained models," *J. Netw. Comput. Appl.*, vol. 167, 2020, Art. no. 102754.

[7] X. Zhang, X. Chen, J. K. Liu, and Y. Xiang, "DeepPAR and DeepDPA: Privacy preserving and asynchronous deep learning for industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 2081–2090, Mar. 2020.

[8] X. Ma, X. Chen, and X. Zhang, "Non-interactive privacy-preserving neural network prediction," *Inf. Sci.*, vol. 481, pp. 507–519, 2019.

[9] X. Ma, F. Zhang, X. Chen, and J. Shen, "Privacy preserving multiparty computation delegation for deep learning in cloud computing," *Inf. Sci.*, vol. 459, pp. 103–116, 2018.

[10] N. Wang *et al.*, "Collecting and analyzing multidimensional data with local differential privacy," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2019, pp. 638–649.

[11] Y. Zhao *et al.*, "Local differential privacy-based federated learning for Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 8836–8853, Jun. 2021.

[12] L. Sun and L. Lyu, "Federated model distillation with noise-free differential privacy," 2020, *arXiv:2009.05537*.

[13] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, "Protection against reconstruction and its applications in private federated learning," 2018, *arXiv:1812.00984*.

[14] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta, "Amplification by shuffling: From local to central differential privacy via anonymity," in *Proc. 30th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2019, pp. 2468–2479.

[15] H. Chang, V. Shejwalkar, R. Shokri, and A. Houmansadr, "Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer," 2019, *arXiv:1912.11279*.

[16] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2938–2948.

[17] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," 2018, *arXiv:1803.01498*.

[18] C. Xie, O. Koyejo, and I. Gupta, "Generalized byzantine-tolerant SGD," 2018, *arXiv:1802.10116*.

[19] Z. Wu, Q. Ling, T. Chen, and G. B. Giannakis, "Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks," *IEEE Trans. Signal Process.*, vol. 68, pp. 4583–4596, 2020.

[20] P. Blanchard *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 119–129.

[21] M. El El Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in byzantium," 2018, *arXiv:1802.07927*.

[22] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory Cryptogr. Conf.*, 2006, pp. 265–284.

[23] O. Goldreich, "General cryptographic protocols: The very basics," *Secure Multi-Party Comput.*, vol. 10, no. 1, pp. 1–27, 2013.

[24] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2017, *arXiv:1712.07557*.

[25] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," 2017, *arXiv:1610.05755*.

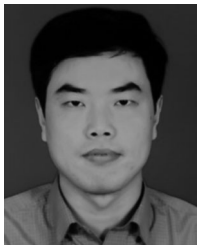
[26] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei, "LDP-Fed: Federated learning with local differential privacy," in *Proc. 3rd ACM Int. Workshop Edge Syst. Anal. Netw.*, 2020, pp. 61–66.
[27] A. Cheu, A. Smith, J. Ullman, D. Zeber, and M. Zhilyaev, "Distributed differential privacy via shuffling," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2019, pp. 375–403.
[28] A. Bittau *et al.*, "Prochlo: Strong privacy for analytics in the crowd," in *Proc. 26th Symp. Oper. Syst. Princ.*, 2017, pp. 441–459.
[29] B. Balle, J. Bell, A. Gascón, and K. Nissim, "The privacy blanket of the shuffle model," in *Proc. Adv. Cryptol.*, 2019, pp. 638–667.
[30] A. R. Elkordy and A. S. Avestimehr, "Secure aggregation with heterogeneous quantization in federated learning," 2020, *arXiv:2009.14388*.
[31] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 1544–1551.
[32] X. Ma, Y. Zhou, L. Wang, and M. Miao, "Privacy-preserving byzantine-robust federated learning," *Comput. Standards Interfaces*, vol. 80, 2022, Art. no. 103561.
[33] R. Guerraoui, N. Gupta, R. Pinot, S. Rouault, and J. Stephan, "Differential privacy and byzantine resilience in SGD: Do they add up?," in *Proc. ACM Symp. Princ. Distrib. Comput.*, 2021, pp. 391–401.
[34] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2168–2181, Jul. 2021.
[35] L. He, S. P. Karimireddy, and M. Jaggi, "Secure byzantine-robust machine learning," 2020, *arXiv:2006.04747*.
[36] B. Ghazi, N. Golowich, R. Kumar, P. Manurangsi, R. Pagh, and A. Velingker, "Pure differentially private summation from anonymous messages," 2020, *arXiv:2002.01919*.
[37] N. Li, M. Lyu, D. Su, and W. Yang, "Differential privacy: From theory to practice," *Synth. Lectures Inf. Secur. Privacy Trust*, vol. 8, pp. 1–138, 2016.
[38] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM J. Optim.*, vol. 19, no. 4, pp. 1574–1609, 2009.
[39] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proc. USENIX Annu. Tech. Conf.*, 2020, pp. 493–506.
[40] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191.
[41] S. Truex *et al.*, "A hybrid approach to privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Secur.*, 2019, pp. 1–11.
[42] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," 2019, *arXiv:1909.06335*.
[43] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[44] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," 2014, *arXiv:1404.5997*.

Xu Ma received the bachelor's degree of computer science from Ludong University, China, in 2008, and the master's and PhD degrees of information security and cryptography from Sun Yat-sen University, China, in 2010 and 2013, respectively. He is also a postdoctor of the Department of Cyberspace Security, Xidian University, China. He is currently an associate professor with the School of Cyber Science and Engineering, Qufu Normal University, China. His research focuses on applied cryptography, outsourcing computation, and privacy preserving machine learning.

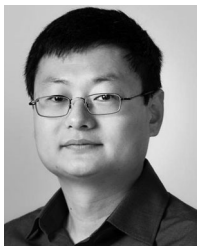
Xiaoqian Sun received the bachelor's degree of software engineering from Qufu Normal University, China, in 2020. She is currently working toward the master's degree in software engineering with Qufu Normal University, China. Her research interests include differential privacy and privacy-preserving machine learning.

Yuduo Wu received the bachelor's degree of information security and law from Nankai University, China, in 2019. She is currently working toward the master's degree in computer science with Nankai University, China. Her research interests include differential privacy and data privacy protection.

Zheli Liu received the BSc and MSc degrees in computer science from Jilin University, China, in 2002 and 2005, respectively, and the PhD degree in computer application from Jilin University, China, in 2009. After a postdoctoral fellowship with Nankai University, China, he joined the College of Computer and Control Engineering, Nankai University, China, in 2011. Currently, he works with Nankai University, China as a professor. His current research interests include applied cryptography and data privacy protection.



Xiaofeng Chen (Senior Member, IEEE) received the BS and MS degrees on mathematics from Northwest University, China, in 1998 and 2000, respectively, and the PhD degree in cryptography from Xidian University, China, in 2003. Currently, he works with Xidian University, China as a professor. His research interests include applied cryptography and cloud computing security. He has published more than 200 research papers in refereed international conferences and journals. His work has been cited more than 10000 times with Google Scholar. He is in the editorial board of *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Knowledge and Data Engineering*, *International Journal of Foundations of Computer Science* etc. He has served as the program/general chair or program committee member in more than 30 international conferences.



Changyu Dong received the PhD degree from Imperial College London, U.K. He is currently a senior lecturer with the School of Computing, Newcastle University, U.K. He has authored more than 30 publications in international journals and conferences. His research interests include applied cryptography, trust management, data privacy, and security policies. His recent work focuses mostly on designing practical secure computation protocols. The application domains include secure cloud computing and privacy preserving data mining.