Secure and Efficient Decentralized Federated Learning with Data Representation Protection

Zhen Qin, Shuiguang Deng, *Senior Member, IEEE,* Xueqiang Yan, Schahram Dustdar, *Fellow, IEEE* and Albert Y. Zomaya, *Fellow, IEEE*

Abstract—Federated learning (FL) is a promising technical support to the vision of ubiquitous artificial intelligence in the sixth generation (6G) wireless communication network. However, traditional FL heavily relies on a trusted centralized server. Besides, FL is vulnerable to poisoning attacks, and the global aggregation of model updates makes the private training data under the risk of being reconstructed. What's more, FL suffers from efficiency problem due to heavy communication cost. Although decentralized FL eliminates the problem of the central dependence of traditional FL, it makes other problems more serious. In this paper, we propose BlockDFL, an efficient fully peer-to-peer (P2P) framework for decentralized FL. It integrates gradient compression and our designed voting mechanism with blockchain to efficiently coordinate multiple peer participants without mutual trust to carry out decentralized FL, while preventing data from being reconstructed according to transmitted model updates. Extensive experiments conducted on two real-world datasets exhibit that BlockDFL obtains competitive accuracy compared to centralized FL and can defend against poisoning attacks while achieving efficiency and scalability. Especially when the proportion of malicious participants is as high as 40 percent, BlockDFL can still preserve the accuracy of FL, which outperforms existing fully decentralized FL frameworks.

Index Terms—Decentralized Federated Learning, Security, Efficiency, Privacy, Blockchain.

1 Introduction

The sixth generation (6G) wireless communication is under exploration to succeed the fifth generation (5G) [1]. It is widely believed that 6G will be built on the new vision of ubiquitous artificial intelligence (AI) [2]. One of the key challenges to realize ubiquitous AI lies on how to conduct distributed machine learning on a large number of highly distributed end devices. Federated learning (FL) [3] is an emerging distributed solution towards AI that breaks the data island and thus becomes one of the potential technical solutions for ubiquitous AI [4].

6G is envisioned as a distributed and decentralized network to simplify network management and provide better efficiency than the centralized networks which suffer from high delays due to long-distance transmission and uncertainty to service quality due to workload. Unfortunately, traditional FL heavily relies on a trusted centralized server. In addition, FL also faces the problems of 1) vulnerability to poisoning attacks which decrease the accuracy (security problem), 2) relatively insufficient privacy protection because the private training data can be reconstructed from intermediate updates by *model inversion attack* [5], [6], [7] and 3) efficiency problem caused by heavy communication cost for transmitting model updates, which is more severe in a decentralized system. Thus, there is an urgent need for a decentralized solution with high efficiency that protects the FL system from being poisoned and the training data from being reconstructed, prompting us to focus on designing a new decentralized P2P FL framework.

Blockchain, a distributed ledger originated from decentralized currency systems, offers distributed trust that enables the cooperation among participants without mutual trust [8]. It also brings convenience to record stake for monetary reward to motivate honest behaviors, since a decentralized system helps to relieve the burden of maintaining centralized servers for mobile operators and service providers. These characteristics make blockchain an promising basis for designing a decentralized FL framework.

In recent years, there are several FL frameworks based on blockchain, some of which integrate additional protection mechanisms for privacy and security to partially solve the above problems of FL. For example, protect the privacy by using differential privacy (DP) [9], [10], homomorphic encryption [11], [12] and secure aggregation [10], [13], and ensure the security by Krum [14] on local updates [9], [10], [15], threshold-based testing [16] and auditing [11].

However, existing FL frameworks still have some limitations on technical selection. For privacy protection, DP provides provable protection for not only data representation, but also membership inference attacks. However, it lowers the accuracy of models. Homomorphic encryption and secure aggregation brings tremendous computation and communication cost, lowering the efficiency of systems. For security, existing applications of Krum mainly focus on local updates, ignoring that global updates may also being poisoned, threshold-based testing relies on manual thresholds or baseline models that are not easily available in real world, and auditing only provides traceability but fails to defend poisoning. In addition, existing solutions often neglect to optimize efficiency. Moreover, some exist-

- *Z. Qin and S. Deng are with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. E-mail: {zhenqin,dengsg}@zju.edu.cn*
- *X. Yan is with Wireless Technology Lab, Huawei Technologies, Shanghai, China. E-mail: yanxueqiang1@huawei.com*
- *S. Dustdar is with Distributed Systems Group, TUWien, Vienna, Austria. E-mail: dustdar@dsg.tuwien.ac.at*
- *A. Y. Zomaya is with School of Computer Science, The University of Sydney, Sydney, Australia. E-mail: albert.zomaya@sydney.edu.au*

ing blockchain-based FL frameworks that rely on mining for consensus further deteriorates the efficiency [8], [15], [17], since a large number of meaningless hash calculations consume bring heavy computation burden. To the best of our knowledge, there is no existing decentralized P2P FL framework that simultaneously solves the privacy problem in terms of training data, security problem in terms of poisoning attacks and efficiency problem of FL.

To address these issues, we propose BlockDFL, a decentralized fully P2P framework for private and secure FL that integrates several techniques with blockchain. For the filtering of the poisoned updates, we first propose a two-layer scoring mechanism, where local updates are filtered according to the stake verified by median-based testing, and global updates are verified by Krum. Then, we design an efficient voting mechanism based on Practical Byzantine Fault Tolerance (PBFT) [18] algorithm to uniquely select a global update in each round. The two mechanisms work jointly to defend the poisoning attack. BlockDFL reduces the communication cost and protects the training data from being reconstructed by gradient compression. BlockDFL is experimentally demonstrated to maintain the accuracy of FL with efficiency when there are up to 40% malicious participants. The main contributions of this paper are threefold:

- We propose BlockDFL, an efficient decentralized P2P FL framework, which solves the security, efficiency and data representation leakage problems of FL by incorporating several techniques and leveraging blockchain as the foundation. It achieves security, high efficiency and scalability through rapid consensus in a small group of randomly selected participants, where the gradient compression is introduced to protect data representation privacy and further lower the communication cost.
- In order to reach the consensus on the most suitable global update in each round of FL, we propose a PBFT-based voting mechanism for consensus on the finally selected global update among verifiers, which never forks. To take advantage of high-quality model updates, we propose to measure local and global updates by median-based testing and Krum, respectively. The combination of the two mechanisms can effectively prevent poisoning attacks and elect a high-quality global update.
- We implement a prototype of BlockDFL and conduct extensive experiments on two real-world datasets, demonstrating that BlockDFL achieves excellent efficiency and scalability and can effectively resist poisoning attacks when there are up to 40% malicious participants. We also experimentally demonstrate that there exists an appropriate degree of sparsity that protects the data privacy from representation perspective and does not harm the effectiveness of distance-based anti-poisoning algorithms while preserving the accuracy of FL at the same time.

2 PRELIMINARIES

2.1 Data Representation Leakage

Although FL is designed to protect privacy by keeping the local data on the devices, existing researches show that the model updates shared by each participant in FL still contains some important information of training data such that the training data can be reconstructed by model inversion attack [5], [6], [7], [19]. If the model updates of honest participants are leaked, attacker can reconstruct the private training datasets [10], [12]. Thus, FL needs further privacy protection since there is no protection for model updates in vanilla FL. In a decentralized FL system, the leakage of model updates leads to a greater risk of training data being reconstructed, since the model updates are often transmitted among ordinary participants which are more likely to be malicious or controlled by malicious users.

Such kind of risk can be defensed by gradient compression which only transmits the elements with large absolute value in model updates. Attackers will not be able to reconstruct any data from sufficiently sparse updates [6], [7], [20]. However, excessive compression of model updates will result in a drop in accuracy. We experimentally demonstrate that there exists an appropriate degree of sparsity that protects the privacy of data representation while preserving the model accuracy. In BlockDFL, local updates are averagely sparsed to over 90% and 85% respectively on two datasets to ensure the privacy of data from representation perspective, and such degree of sparsity can make model inversion attacks represented by DLG attack [6] unable to obtain any useful information from model updates. More details about protecting data representation by gradient compression are available in Appendix A.

2.2 Poisoning Attack

Since the global model in FL is obtained by averaging several local updates shared by participants, FL is under the risk of poisoning attacks. Malicious participants can conduct poisoning attack by uploading poisoned model updates to lower the accuracy of global model and negatively impact the convergence of FL [21].

In this work, malicious participants will conduct the label flipping attack as in [10], [21]. Specifically, attackers can label the data within one class as another class, and train the model to generate local updates based on the tampered datasets, causing the global model unable to distinguish the data of the two classes correctly.

In a centralized system, there are many existing approaches to defend poisoning attack [14], [21], [22]. However, in a fully decentralized system where there is no mutual trust between participants, there is a problem that which participant should be responsible for detecting poisoned model updates, and how can a participant trust the judgments made by other participants.

BlockDFL solves this problem by introducing Krum into our designed voting mechanism based on PBFT algorithm. Intuitively, since Krum rejects model updates differ heavily from the direction of the majority of the updates. The gradient compression may bring negative impact on it. But we experimentally find that when the gradient compression is introduced, some of the indexes of the transmitted elements with the largest absolute value in different updates may still overlap, enabling to spatially distinguish the normal model updates and the malicious ones. More details about this can be found in Appendix B.

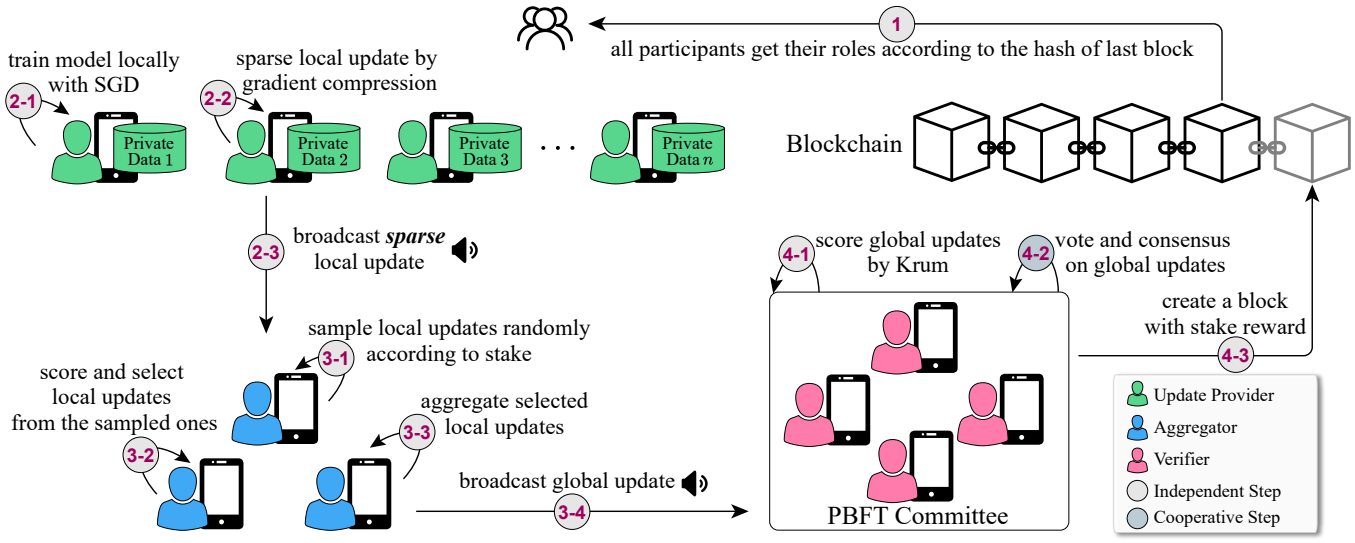


Fig. 1: The detailed processes of BlockDFL in one round of communication, from ① to ④.

3 BlockDFL Overview

The goal of BlockDFL is to build up an efficient decentralized P2P FL framework: 1) to make its accuracy approach that of the centralized FL as much as possible, 2) to prevent the accuracy of global model from being jeopardized by poisoning attacks and 3) to prevent the private training data from being leaked to peer participants, 4) to reduce the communication cost of transmitting model updates.

BlockDFL fulfills all these requirements in a lightweight manner. As illustrated in Fig. 1, there are four processes in BlockDFL during one communication round: 1) *Role Selection*, 2) *Local Training* 3) *Aggregation* and 4) *Verification and Consensus*, some of which contain more than one steps.

BlockDFL is designed based on the following assumptions: 1) Participants in the same FL system use the same model and initialization. They can obtain the public key for verifying digital signatures of others and send information through broadcasting. 2) Participants in BlockDFL hold the *stake* indicating how much they have contributed to the global model. The stake recorded on the blockchain can be tied to monetary reward from mobile operators or AI service providers, since a decentralized FL system relieves them of the heavy burden of setting up and maintaining a centralized server. Thus, we make the same assumption as in [10], [23] that participants holding large amounts of the stake tend to perform obligations honestly, because they can benefit more from the monetary reward.

Participants in BlockDFL are granted three different roles, i.e., *Update Provider*, *Aggregator* and *Verifier*. The update provider is in charge of training model based on its private training data and sharing its local update to aggregators. They work independently. The aggregator is responsible for collecting local updates and selecting a certain number of them to aggregate into a global update. They work independently, too. The verifiers preside over electing a suitable global update together and packaging it with the digital signatures created by the verifiers' private keys and the identity of its aggregator and update providers into a block newly added to the blockchain. They score global update independently, and select one global update collaboratively. The independent steps and cooperative steps are marked with different colors in Fig. 1. In BlockDFL, adding a block means that all participants have conducted a round of communication (equivalent to executing the FedAVG algorithm [3] once in FL). If the block is not empty, all participants will update their model according to the global update contained in the newly added block.

At the start of each communication round, each participant is randomly assigned with a role based on the hash of last block as in [10] (process ①). Then, update providers train local models with stochastic gradient descent (SGD) algorithm on their own training set and sparse the local updates by gradient compression before broadcasting them to aggregators (process ②). Each aggregator continues to receive local updates until a certain number of local updates are obtained, then starts the aggregation independently (process ③). An aggregator first samples a certain number of local updates from the received ones according to the stake of corresponding providers. Then, it scores the sampled local updates and selects some of them to aggregate a global update which is then broadcasted to verifiers. When verifiers receive enough global updates, the verification starts (process ④). Each verifier independently scores the global updates and votes to them based on the scores, so as to select one approved global update. Finally, the approved global update with its relative information are wrapped in a new block which is then broadcasted to all participants.

4 Multiple Processes of BlockDFL

In this section, we will present the detailed steps of the four processes in BlockDFL during one communication round.

4.1 Role Selection

At the start of role selection in BlockDFL, the hash value of last block h_{-1} is mapped to a hash ring where each participant is assigned a space proportional to its stake as in [10]. The participant in whose portion h_{-1} lies is selected

as the first aggregator. Then, the hash value is repeatedly re-hashed to select other aggregators. When a certain number of aggregators are selected, it turns to select verifiers in the same way. When all aggregators and verifiers are selected, the rest participants become update providers. It ensures that participants with more stakes are more likely to be selected as important roles, i.e., aggregators and verifiers. The set of verifiers, aggregators and update providers are represented by $\mathcal{V}$, $\mathcal{A}$ and $\mathcal{U}$, respectively. The number of verifiers $|\mathcal{V}|$ and the number of aggregators $|\mathcal{A}|$ are both hyper-parameters set before FL starts. As illustrated in Section 5.3, the efficiency of BlockDFL is mainly related to the number of aggregators and verifiers, thus, $|\mathcal{V}|$ and $|\mathcal{A}|$ are recommended to be much smaller than $|\mathcal{U}|$.

In BlockDFL, roles are reassigned at the start of each round to give each participant the opportunity to contribute local update to FL system and defend bribery attack.

4.2 Local Training

In round t, update provider u_i will perform local training based on the model parameters of the previous round $w_i(t-1)$ on its private training data with SGD algorithm as:

$$w - \eta \frac{1}{b} \nabla \mathcal{L}(x, w) \rightarrow w \tag{1}$$

where x is a mini-batch with b samples sampled from the training set $\mathcal{X}$ of u_i, $\mathcal{L}$ is the loss function and η is the learning rate (step 2-1 in Fig. 1). Let $w_i(t)$ be the model parameters after several epochs of local training, the local update d_i is obtained as:

$$d_i = w_i(t) - w_i(t-1) \tag{2}$$

To protect the representation of local data private and reduce the communication cost, we sparse the local updates as in gradient compression [24]. Let s be the sparse ratio, i.e., the percentage of zero elements in sparsed d_i, the update provider only transmits the $(1-s)|d_i|$ elements of d_i with the largest absolute value (step 2-2 in Fig. 1). To avoid the loss of accuracy, the rest elements will be kept locally and accumulated to the next local training of the participant. The sparse local update will be digitally signed and broadcasted to aggregators (step 2-3 in Fig. 1).

4.3 Aggregation

When an aggregator has collected enough local updates, the aggregation starts. Let $\mathcal{D}$ denote the set of local updates received by this aggregator and c be the number of local updates which a global update must contain. There are two sampling steps in aggregation. The first step is with the stake filter, in order to discard most of the model updates for relieving the computation cost of later testing. In this step, the aggregator samples $3 \times c$ local updates from $\mathcal{D}$, where the probability of each local update to be selected is proportional to the stake of its update provider (step 3-1 in Fig. 1). The sampled local updates constitute a set $\mathcal{D}^s$. In order to make more honest participants have the opportunity to share their local updates, the stake can be log-scaled in this step.

The second step is based on the median-based testing, which is designed for screening out high-quality, non-poisoned local updates for aggregation. In this step, the aggregator will evaluate the local updates in $\mathcal{D}^s$ by updating the local model based on them one by one and performing inference on its test set (step 3-2 in Fig. 1). Then, we rank the local updates in $\mathcal{D}^s$ in descending order according to their accuracy of inference. Let $\mathcal{DM}^s$ denote the local updates before the median of sorted $\mathcal{D}^s$, the local updates for aggregation are randomly selected from $\mathcal{DM}^s$ and constitute a set $\mathcal{D}^a$ ($|\mathcal{D}^a| = c$). The probability p_i of each local update d_i in ranked $\mathcal{DM}^s$ to be selected is calculated as:

$$p_i = \frac{\exp(q(d_i))}{\sum_{d_j \in \mathcal{DM}^s} \exp(q(d_j))} \tag{3}$$

where $q(d_i)$ is the evaluation accuracy of local update d_i. Local updates in $\mathcal{D}^a$ are aggregated to a global update G as:

$$G = \frac{1}{|\mathcal{D}^a|} \sum_{d \in \mathcal{D}^a} d \tag{4}$$

(step 3-3 in Fig. 1). The aggregated global update G is then digitally signed and broadcasted to verifiers to compete for being packaged on the chain (step 3-4 in Fig. 1).

The stake filter ensures that most of the local updates tested come from honest participants, in order to ensure that the model updates before the median are un-poisoned. Median-based testing determines whether a local update is poisoned by comparing it with the others, instead of relying on a manual threshold [25] or baseline validation model which are hard to obtain in real-world application scenarios to judge whether an update is malicious. Therefore, this process is reliable and applicable.

4.4 Verification and Consensus

In order to uniquely elect one suitable global update in each round, we simplify PBFT [18] algorithm for decentralized FL and design a voting-based verification mechanism based on the simplified PBFT, which has the following advantages: 1) It has high efficiency since the verifiers are a small group of randomly selected participants. 2) It can deal with malicious participants and the disconnection problem, even for the leader of PBFT. 3) It never forks.

The first selected verifier is the leader of verifiers to initiate the verification of global updates one by one. The order of verifying the global update can be decided by the leader. There are three stages in the proposed voting mechanism that each global update needs to go through, i.e., *pre-prepare*, *prepare* and *commit*. Let $\mathcal{G}$ be the set of candidate global updates in this communication round. Assuming that $G_i \in \mathcal{G}$ is the first selected global update to be verified. In the verification of G_i, the leader first sends a *pre-prepare* message with the digital signature of G_i to the other verifiers. When a verifier receives the *pre-prepare* message, it broadcasts a *prepare* message with the digital signature of G_i to all verifiers. When a verifier receives more than $\frac{2}{3}|\mathcal{V}|$ *prepare* messages, it starts the *commit* stage. In *commit*, the verifier scores each G_i by Krum algorithm [14], where a lower score indicates a higher quality. Let f be the percentage of malicious participants and $\mathcal{G}_i^c \subseteq \mathcal{G}$ denote the $(1-f)|\mathcal{G}| - 2$ global updates closest to G_i, Krum scores G_i by calculating the distance of G_i to global updates in $\mathcal{G}_i^c$, as:

$$\mathrm{Krum}(G_i, \mathcal{G}) = \sum_{G_j \in \mathcal{G}_i^c} \|G_i - G_j\|^2 \tag{5}$$

Then, the verifier calculate the score of the other global updates in $\mathcal{G}$ in the same way. Then the verifier sends a signed *commit* message to the leader containing the vote to G_i (step 4-1 in Fig. 1). Only the score of G_i surpasses that of 2/3 global updates will G_i be voted affirmatively, as:

$$\begin{cases} 1 & \text{if } \sum_{G_j \in \mathcal{G} \setminus G_i} \mathbb{I}_{\mathrm{Krum}(G_i,\mathcal{G}) < \mathrm{Krum}(G_j,\mathcal{G})} \geq \frac{2}{3}|\mathcal{G}| \\ 0 & \text{else} \end{cases} \tag{6}$$

where 1 and 0 means the affirmative and negative vote, respectively. $\mathbb{I}$ is a indicator function whose value is 1 when the condition is met otherwise 0.

If the leader has received more than $\frac{2}{3}|\mathcal{V}|$ *commit* messages with the affirmative vote, the verification ends and G_i becomes the approved global update of this communication round (step 4-2 in Fig. 1). Then the leader builds a block containing: 1) the elements of G_i, 2) the identity of the aggregator and update providers of G_i and 3) the identity of the verifiers which vote for support. The block will be signed by the leader and broadcasted to all participants (step 4-3 in Fig. 1). The participants listed in 2) and 3) will be awarded with stake. However, if the number of *commit* messages with the negative vote the leader received has exceeded $\frac{1}{3}|\mathcal{V}|$, the verification of G_i will be finished and the leader will start the verification of another global update G_j. Note that in the verification of the subsequent global updates, the score of them obtained during the verification of the first global update can be directly used. If all global updates in $\mathcal{G}$ are verified but no one is approved, the leader will broadcast an empty block to all participants. When a participant receives a block, it will update the local model if the block contains an approved global update. Then, the next communication round starts.

We apply Krum in verification instead of aggregation because it has the following advantages: 1) The results of Krum are consistent on the same updates as the input, which facilitates the consensus in the result of voting. 2) The complexity of Krum is $\mathcal{O}(n^2)$, where n is the number of updates to be scored, which is larger in aggregation than that in verification of BlockDFL. Intuitively, Krum will be affected by the sparseness of model updates. However, we observe that some of the indexes of the $(1-s)|d_i|$ elements with the largest absolute value in different updates may overlap, enabling to spatially distinguish the normal updates and the malicious ones. More details about this are available in Appendix B.

5 Experiments

5.1 Experimental Setup

We implement BlockDFL with Python 3.8 and PyTorch 1.10 to evaluate its accuracy, poisoning-tolerance, efficiency and scalability. The experiments demonstrate: 1) BlockDFL has a comparative accuracy compared with vanilla FL and can effectively resist poisoning attacks, 2) the reason that BlockDFL can resist poisoning attacks and 3) BlockDFL works efficiently and possesses a good scalability.

TABLE 1: Default Settings of Parameters in Experiments

Parameter Name	Value
# of aggregators & verifiers	8 & 7
Initial stake	Uniformly 10
Stake increment	5
c of global updates	5
# of epochs in local training	5
Sparsity in MNIST	$[90\%, 92.5\%, 95\%, 97.5\%]$ changes every 50 rounds
Sparsity in CIFAR-10	$[85\%, 87.5\%, 90\%, 92.5\%, 95\%]$ changes every 60 rounds

5.1.1 Dataset, Model and Platform

We select two widely-used real-world datasets, i.e., MNIST[1] and CIFAR-10[2] to evaluate BlockDFL. For MNIST, we build a convolutional neural network with 1,662,752 parameters as in [3]. For CIFAR-10, we build the CIFARNET with 1,149,770 parameters[3]. In local training, these models are trained by SGD with learning rate of 0.01, which decays after each round with the coefficient of 0.99. The parameter settings of the experiments are shown in Table 1 unless stated otherwise. Note that as shown in Table 1, all participants start with 10 stake, and if they are awarded with stake, the quantified value of the stake obtained is 5. As shown, the local updates are 93.75% sparse on average on MNIST and 90% on CIFAR-10.

We run 50 participants on a Windows10 platform with an AMD Ryzen 7 5800 3.40GHz CPU, an NVIDIA RTX 3070 GPU and 48GB RAM. The training set and test set are randomly and equally distributed to each participant. The training sets are used by update providers for local training and the test sets are used by aggregators to score local updates.

5.1.2 Strategy of Malicious Participants

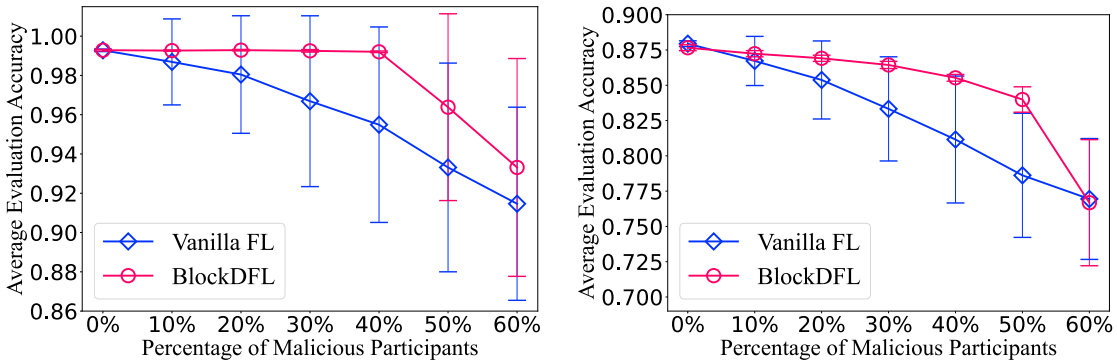
In the experiments, malicious update providers will poison local updates by label-flipping attack as in [10], [21]. They label all 1s as 7s in MNIST, and label all *cats* as *dogs* and all *deer* as *horses* in CIFAR-10, then perform local training on the poisoned training set.

To better evaluate the poisoning-tolerance of BlockDFL, other roles are also granted the ability to conduct poisoning attacks. For example, a malicious aggregator will sample $3c$ local updates uniformly and randomly for evaluation and aggregate c of them with the lowest accuracy. The c local updates from the update providers with lowest stake are not directly selected because such behavior can be easily detected, exposing the malicious participants. A malicious verifier will vote contrarily to an honest one, aiming at a wrong consensus.

5.1.3 Baseline

We select the vanilla federated learning [3] as the baseline, which relies on a trusted centralized server. The settings of relevant parameter and models are exactly the same as in BlockDFL. The model updates transmitted in vanilla FL

1. http://yann.lecun.com/exdb/mnist/
2. https://www.cs.toronto.edu/~kriz/cifar.html
3. 64C3-64C3-MaxPool2-Drop0.1-128C3-128C3-AveragePool2-256C3-256C3-AveragePool8-Drop0.5-256-10



(a) Average evaluation accuracy on MNIST

(b) Average evaluation accuracy on CIFAR-10

Fig. 2: Average evaluation accuracy and standard deviation of the last 50 rounds of vanilla FL and BlockDFL on MNIST and CIFAR-10 with different percentage of malicious participants.

are not compressed by sparsification in order to ensure the performance of vanilla FL as much as possible.

Note that in the scenario without considering the data heterogeneity, vanilla FL can achieve almost the state-of-the-art accuracy as demonstrated in [26], so the goal of BlockDFL is not to surpass vanilla FL in accuracy, but to obtain accuracy as close as possible to vanilla FL in a fully decentralized scenario, and can prevent the FL system being jeopardized by malicious participants.

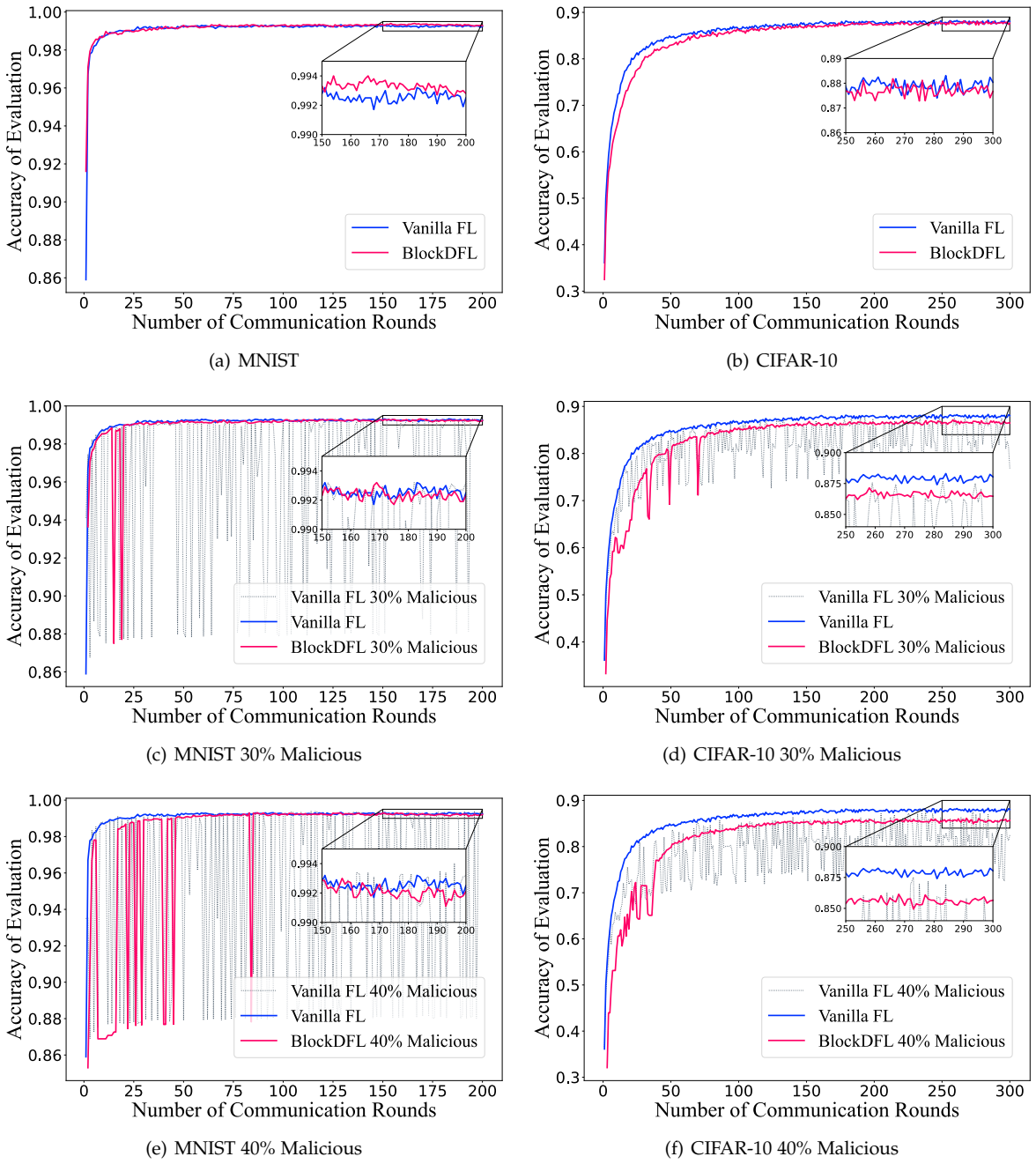
5.2 Performance and Poisoning Tolerance

We iterate BlockDFL and vanilla FL for 200 rounds of communication on MNIST and 300 rounds on CIFAR-10 and subject both of the two approaches to poisoning attacks with the proportions of malicious participants ranged in $[0\%, 60\%]$. We run the two approaches for five times for each proportion of malicious participants. The *average evaluation accuracy* is calculated by averaging the inference accuracy on the whole test set in the last 50 rounds of these runs. Fig. 2 presents the average evaluation accuracy with the corresponding standard deviation of the two approaches. When there is no malicious participant exist, BlockDFL achieves the average accuracy of 99.284% on MNIST while vanilla FL is 99.279%, and BlockDFL achieves the average accuracy of 87.659% on MNIST while vanilla FL is 87.946%. Thus, we can conclude that when all participants are honest, BlockDFL enables a group of peer participants without mutual trust to perform decentralized FL, and can obtain the equivalent performance on accuracy compared with the vanilla FL, which is a centralized scheme.

It is observed that on both datasets, BlockDFL is able to defend poisoning attacks when there are up to 40% malicious participants. When facing malicious participants, BlockDFL keeps relatively steady average accuracy while vanilla FL is severely jeopardized. With the increase of malicious participants, the gap between the average accuracy of BlockDFL and vanilla FL is getting wider. Moreover, BlockDFL converges much more stably when facing malicious participants, showing very low standard deviation of the last 50 rounds. However, we observe that on CIFAR-10, a relatively complex dataset, the average evaluation accuracy of BlockDFL slightly decreases with the increasing ratio of malicious participants, although it is still significantly better than that of vanilla FL. The same phenomenon also appears in [21], because as the ratio of malicious participants increases, they will hold more data, causing a decrease in the amount of data contributed to the global model.

To demonstrate the accuracy of the two approaches after each round of communication, we select 6 runs, each of which corresponds to a different dataset and different ratio of malicious participants, and plot the trends of evaluation accuracy with the rounds go on in Fig. 3. As shown in Fig. 3(a) and (d), the convergence speed of BlockDFL is very close to un-poisoned vanilla FL, indicating that BlockDFL does not require additional communication rounds compared to vanilla FL. And it can be observed that in Fig. 3(b), (c), (e) and (f), when there exists malicious participants, BlockDFL will still gradually converge to a level close to the un-poisoned FL, while the poisoned FL diverges seriously.

Generally, the PBFT-based approach can tolerant f malicious participants when there are $3f + 1$ participants in a system. But we experimentally demonstrate that BlockDFL can tolerant 40% malicious participants. The enhancement origins from the accumulation of stake held by honest participants. From the description in Section 4.4, we can conclude that the voting mechanism described above can only produce a non-empty block if the majority of verifiers (more than $2/3$) are honest or the majority of verifiers are malicious. When a non-empty block is created, all the verifiers voted positively will obtain stake. The situation that over $2/3$ verifiers are honest are more likely to occur than the situation that over $2/3$ verifiers are malicious when the number of honest participants is larger than that of malicious ones. Thus, the proportion of stake held by honest participants will gradually increase as blocks continue to be generated, making the situation that $2/3$ verifiers are honest more and more likely to occur. To demonstrate this process, we present some statistics during running BlockDFL in Fig. 4 for a better comprehending. As shown in Fig. 4(a), when there are less than 40% malicious participants, the proportion of stake held by malicious participants decreases as the rounds go on, meaning that malicious users are increasingly unlikely to be elected as aggregators or verifiers, and thus less likely for a successful poisoning attack to occur. More



(a) MNIST (b) CIFAR-10

(c) MNIST 30% Malicious (d) CIFAR-10 30% Malicious

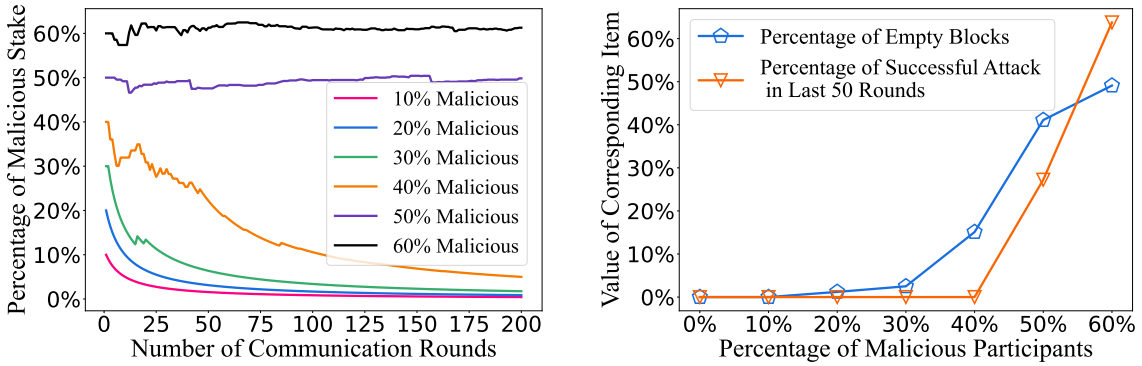
(e) MNIST 40% Malicious (f) CIFAR-10 40% Malicious

Fig. 3: Evaluation accuracy trends of vanilla FL and BlockDFL on MNIST and CIFAR-10 datasets with the increasing of number of communication rounds, where there are different proportions of malicious participants in the federated learning systems.

specifically, the fewer malicious users, the more stable the proportion declines. When the ratio of malicious users is 30% and 40%, there is an upward trend in certain communication rounds, indicating successful poisoning attacks in these rounds. Fig. 4(b) shows the percentage of empty blocks and successful attack with different percentage of malicious participants. An empty block means that neither honest nor malicious participants occupy more than 2/3 among verifiers, resulting in a fail consensus of voting. The percentage of successful attack means that the percentage of global updates containing at least one poisoned local update in the last 50 communication rounds. Both of them raise with the increase of malicious participants.

5.3 Time Consumption and Scalability

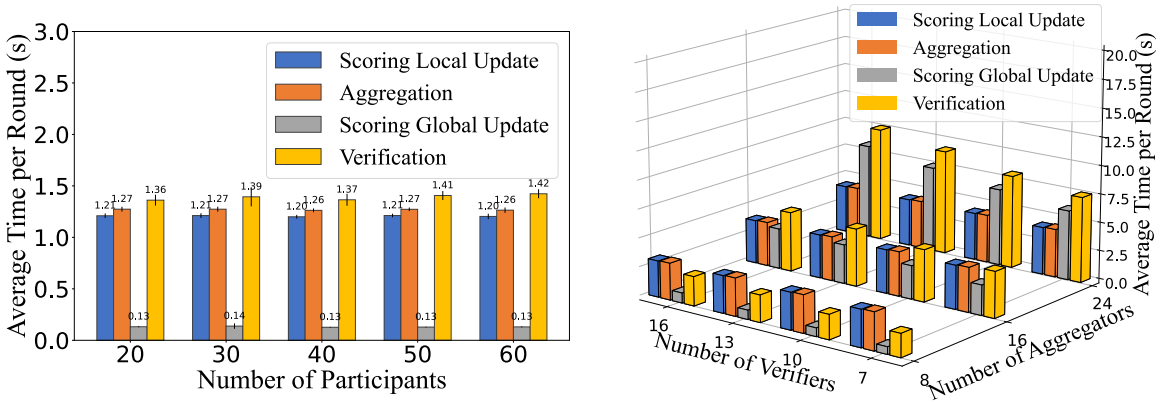
To show the efficiency and scalability of BlockDFL, we run it on MNIST with variant numbers of participants ranged in $[20, 60]$ and record the time consumption of aggregation and verification. Since scoring local update and scoring global update are important steps of aggregation and verification, respectively, we also record the time consumption of the two steps. We fix the numbers of verifiers and aggregators to 4, c of global updates to 3 and scale the number of participants. Since the data on each participant is equally divided from the original dataset, when the number of participants changes, the number of samples in test set on each



(a) Changes in proportion of stake held by malicious participants as the rounds go on.

(b) Percentage of empty blocks and successful attacks versus ratio of malicious participants.

Fig. 4: Statistics during running BlockDFL on MNIST dataset.



(a) Time versus # of participants

(b) Time versus # of $|\mathcal{V}|$ and $|\mathcal{A}|$

Fig. 5: Breakdown of time different processes of BlockDFL take with varying number of participants, aggregators and verifiers.

participant also changes, affecting the time taken for scoring local updates in aggregation. To eliminate the impact, we fix the number of test samples on each participant to 150 by randomly sampling from the split test sets.

Fig. 5(a) presents the time spent by each process and step with varying number of participants. We can find that the time spent by aggregation mainly lies in scoring local updates, and scoring global updates takes much less time than verification, meaning the time of verification is mainly spent by voting cooperatively. With the changes of the number of participants, the time spent by each process keeps steady, implying a good scalability of BlockDFL. The aggregation and verification take less than 3 seconds totally, while Biscotti [10], a similar framework, takes over 30 seconds in the case that the number of model parameters is one-tenth of BlockDFL. The communication cost of BlockDFL is low since less than 10% of the elements in local updates are transmitted on average. Thus, we conclude that BlockDFL obtains excellent efficiency and scalability.

To clarify how the numbers of aggregators and verifiers affect the efficiency of BlockDFL, we run BlockDFL with different numbers of aggregators and verifiers. As shown in Fig. 5(b), the time consumption of verification together with scoring global update are mainly related to the number of aggregators. When the number of aggregators growths, both of them also increase, since the global updates that need to be scored and the average number of votes required to select the final global update also increase. But they are less affected by the number of verifiers. As for aggregation, it almost keeps steady with different numbers of aggregators and verifiers, since the aggregators work independently.

6 Related Work

In recent years, there are many researches about blockchain-based FL. Existing works either rely on a global trust authority or a trusted central server [9], [27], [28], or only solve part of the problems of FL, i.e., addressing poisoning attacks but ignoring the privacy protection [12], [15], [16] and protecting the privacy but failing to prevent poisoning attacks [11], [13]. We investigate some existed blockchain-based frameworks for decentralized FL, and provide comparisons of these decentralized frameworks and our BlockDFL in Table 2. Since BlockDFL is designed for decentralized P2P FL, we only focus on the existed frameworks without the reliance on a global trust authority or trusted servers (including cloud center and edge servers), i.e., FL frameworks designed for the fully decentralized P2P setting. These frameworks are compared on five dimensions as listed below:

TABLE 2: Multi-dimensional Comparisons of Existing Decentralized P2P Federated Learning Frameworks based on Blockchain

Approach	Privacy	Anti-poisoning	Poisoning Tolerance	Communication Optimization	Consensus & Fork-preventing
LearningChain [17]	DP	l-Nearest Aggregation	10% malicious	×	PoW ×
BlockFL [8]	DP	×	×	×	PoW ×
BEMA [15]	×	Krum + Multiparty Multiclass Margin	20% malicious	×	PoW ×
DeepChain [12]	Homomorphic Encryption	× Auditing	0% Only Traceability	×	Algorand ✓
Biscotti [10]	DP + Secure Aggregation	Krum	30% malicious	×	PoF ×
BlockDFL (ours)	Gradient Compression	Median-based Testing and Krum	40% malicious	Gradient Compression	PBFT-based Voting ✓

- **Privacy**: Does the framework provide additional privacy protection on the basis of federated learning? If yes, then how to achieve it.
- **Anti-poisoning**: Whether the framework is able to prevent the global model from being jeopardized by poisoning attacks. If yes, then how to fulfill it.
- **Poisoning Tolerance**: How many malicious participants can the framework tolerate?
- **Communication Optimization**: Whether the framework is able to reduce the communication cost of decentralized FL. If yes, then how to achieve it.
- **Consensus**: Which consensus protocol is introduced by the corresponding framework, and can the framework prevent the forking problems of blockchain.

As we can see, different frameworks introduce different algorithms or mechanisms to partially solve the problems faced by FL, i.e., privacy, security and communication cost. However, existing frameworks do not address these issues uniformly. For example, existing frameworks may address poisoning attacks but ignore the privacy protection or protect the privacy but fail to prevent poisoning attacks. Particularly, existing frameworks often neglect to optimize the communication overhead of decentralized FL. Additionally, there are also some areas that need to be optimized in existing technical solutions. For anti-poisoning, l-nearest aggregation is susceptible to outliers, where a malicious participant can submit a model update that deviates very much from others to destroy the effectiveness of l-nearest aggregation. Auditing provides traceability of aggregation, making the malicious behaviors traceable. However, it can not prevent the system from being poisoned. In some scenarios with high safety requirements, such as autonomous driving, it is not enough to provide traceability cause the catastrophic events may have occurred. The complexity of Krum is $\mathcal{O}(n^2)$ where n is the number of model updates to be evaluated. Thus, Krum may not be appropriate to be directly introduced to score a large number of model updates such as directly applied to verify local updates [10] whose number is relatively large. In other words, it should be applied to a verify a relatively small set of model updates such as the candidate global updates. For privacy preserving, DP is able to protect privacy with the guarantees of mathematical proof from the perspective of data reconstruction and membership inference. Unfortunately, it imposes a significant accuracy loss for protecting complicated models [29]. Homomorphic encryption brings too much computation overhead, making it unsuitable for models with relatively large numbers of parameters. And the secure aggregation brings heavy overhead of computation and communication, which limits the efficiency and scalability of FL frameworks based on it.

As for the ability of poisoning resistance, existing frameworks can only defend against poisoning attacks when the proportion of malicious participants is less than or equal to 30%, while BlockDFL can still effectively defend poisoning attacks when 40% of the participants are malicious. Note that LearningChain [17] is only evaluated on three situations that 10%, 40% and 70% of the participants are malicious, respectively. When 40% of the participants are malicious, the accuracy of LearningChain is seriously jeopardized, although it outperforms Krum. From these comparisons, BlockDFL outperforms all the existing fully decentralized FL frameworks in terms of poisoning resistance.

To the best of our knowledge, BlockDFL is the only framework for decentralized P2P federated learning which uniformly solves the security, efficiency and data representation leakage problems faced by FL. BlockDFL can still effectively defend the poisoning attacks when the proportion of malicious participants is as high as 40 percent. BlockDFL is very efficient because: 1) the verification of local updates is very efficient since the integrated stake-based filtering mechanism filters out most local updates that do not need to be verified, 2) the PBFT-based voting mechanism for global update election works very fast since the verifiers are a small group of participants and 3) the communication cost of transmitting model updates is lowered by gradient compression, It is worth to note that the proposed voting mechanism for the consensus on global updates does not need to perform a lot of meaningless hash calculations like proof-of-working (PoW) consensus and does not fork.

7 Conclusions and Future Work

In this paper, we propose BlockDFL, an efficient fully P2P framework for decentralized FL. To efficiently reach the

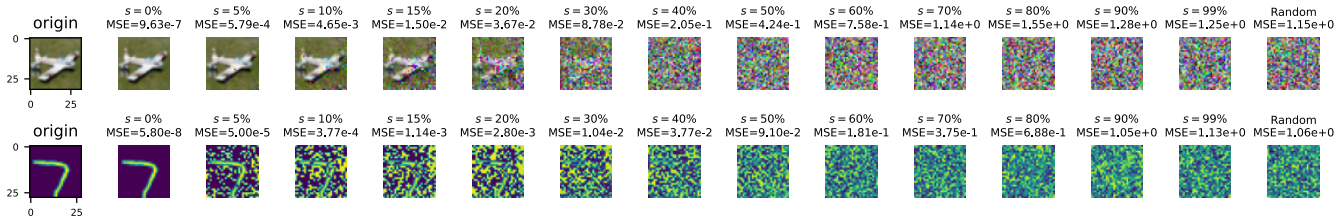


Fig. 6: MSE of images reconstructed by DLG attack [6] with different ratio of sparsity s on CIFAR-10 and MNIST.

consensus on the appropriate global update in each round, we propose a PBFT-based voting mechanism. And to take advantage of high-quality model updates, we propose to measure local and global updates by median-based testing and Krum, respectively. The combination of the two mechanisms helps to uniquely select a high-quality global update in each round, while preventing the FL system from being poisoned. To protect the privacy of data in the representation respect, we introduce the gradient compression, and experimentally demonstrate that gradient compression can be integrated into our framework without affecting the effectiveness of Krum and the accuracy of global model, while protecting privacy and reducing communication overhead. Experiments conducted on two widely-used real-world datasets demonstrate that BlockDFL can defend the poisoning attacks and achieve efficiency and scalability. Specially, when 40% of the participants are malicious, BlockDFL can still defend poisoning attacks, which outperforms existing fully decentralized FL frameworks.

In the future, we will introduce more privacy protection mechanisms for member inference attacks into BlockDFL.

APPENDIX A
GRADIENT COMPRESSION VERSUS DATA RECONSTRUCTION

Gradient compression was proposed to reduce the communication cost among distributed nodes in the distributed training of machine learning models. The gradient compression methods believe that the contribution of each element in the gradient to the model accuracy is different. The elements with smaller absolute values contribute little to the model accuracy, but the elements with larger absolute values usually make more contribution to the model accuracy. For this reason, in the distributed training of the machine learning models, it can greatly reduce communication overhead by only transmitting the elements with relatively large absolute values. The model update in federated learning is similar to the gradient in distributed machine learning, so gradient compression can be applied to FL directly.

Although gradient compression is not proposed for privacy protection, it can effectively prevent the training data from being reconstructed by malicious attackers from gradients [6], [7], [20]. Taking one of the famous model inversion attacks, Deep Leakage from Gradients (DLG) [6], as an example, gradient compression actually destroys the optimization objectives of DLG attack. In other words, gradient compression discards many details of the model updates by sparsifying, making it hard for DLG attack to obtain enough information for reconstructing the training data from sparse gradients.

In [6], a series of experiments are conducted to evaluate the performance of DLG attack under different degree of gradient sparsity (ranged in $[1\%, 70\%]$), drawing a conclusion that DLG can tolerant up to 20% sparsity of gradients. When the sparsity of gradients exceeds this threshold, the images reconstructed by the DLG attack are almost not visually recognizable.

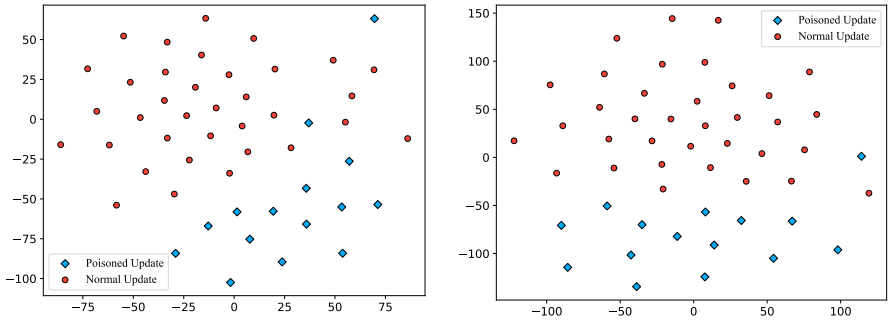
We also conduct detailed experiments to demonstrate that gradient compression can effectively defend the data reconstruction of DLG attack. As in [6], we conduct DLG attack to LeNet on CIFAR-10 and MNIST with different sparsity of model updates, respectively. For each image, we iterate DLG model for 300 rounds and record the result with the lowest Mean Square Error (MSE) between the reconstructed image and the original one. Fig. 6 presents the lowest MSE and the corresponding reconstructed images obtained by DLG attack on MNIST and CIFAR-10 with different sparsity, where the first row and next row show the results on an image of CIFAR-10 and MNIST, respectively. It is observed that as the sparsity gets higher, the visibility of the reconstructed images by DLG attack gets worse. The experimental results are consistent as reported in [6]: on both of the two datasets, when the sparsity exceeds 20%, the reconstructed images are hard to be visually recognized.

However, it cannot fully guarantee that the information will not be leaked if the reconstructed images are only visually unrecognizable. To better illustrate the effectiveness of gradient compression to defense DLG attack, randomly generated images are also introduced as the reference. As shown, for CIFAR-10, when the sparsity exceeds 70%, the MSE of reconstructed image is similar to the randomly generated one, and the corresponding threshold is about 90% for MNIST. Thus, we conclude that gradient compression can effectively defense the DLG attack with the ratio of sparsity over 70% for CIFAR-10 and 90% for MNIST.

In the experiments of BlockDFL, the sparsity of model updates is more than 85% for CIFAR-10 (averaged to 90%) and more than 90% for MNIST (averaged to 93.75%) to ensure the representation privacy of local training data.

APPENDIX B
DISTANCE BETWEEN SPARSED MODEL UPDATES

As introduced in Section 4.4, intuitively, the effectiveness of Krum will be negatively affected by the sparseness of model updates since it depends on the distance between model updates calculated element-wisely. However, we observe that when the gradient compression is introduced,



(a) Model updates without sparsification (b) Model updates with the sparsity of 90%

Fig. 7: 2-dimensional visualization of model updates on MNIST, where each point represents a model update.

some of the indexes of the transmitted elements with the largest absolute value in different updates may still overlap, enabling to spatially distinguish the normal model updates and the malicious ones.

We randomly split the training set of MNIST into 50 subsets with the same number of data samples and poison 15 subsets of them by label-flipping attack as introduced in Section 5.1, in order to simulate the situation that there are 30% malicious participants in a FL system. Then, we train 50 convolutional neural networks introduced in Section 5.1 with 1,662,752 parameters for five epochs, where each network is trained on one of the subsets. Then, we obtain 50 model updates and visualize them by t-SNE [30] on a scatter in Fig. 7(a), where a blue dot indicates a poisoned update and a red dot indicates a normal update. We can observe that the 50 original updates obviously form two clusters that one cluster is composed of normal updates and the other cluster is composed of poisoned updates.

We then sparse the 50 original model updates to 90% sparsity and visualize the sparse ones by t-SNE in Fig. 7(b). As shown, the sparse updates can still form two clusters with clear boundaries according to whether they are poisoned. Therefore, we can conclude that Krum and gradient compression can be simultaneously integrated in a framework without affecting each other, which is also supported by the experimental results in Section 5.2.

ACKNOWLEDGMENTS

This work was supported in part by the Key Research Project of Zhejiang Province under Grant 2022C01145 and in part by the National Science Foundation of China under Grants U20A20173 and 62125206. The work of Schahram Dustdar's was supported in part by the Zhejiang University Deqing Institute of Advanced technology and Industrilization (ZDATI).

REFERENCES

[1] G. Liu, N. Li, J. Deng, Y. Wang, J. Sun, and Y. Huang, "6G mobile network architecture-solids: Driving forces, features, and functional topology," *Engineering*, 2021.

[2] V. Nguyen, P. Lin, B. Cheng, R. Hwang, and Y. Lin, "Security and privacy for 6G: A survey on prospective technologies and challenges," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2384–2428, 2021.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *the International Conference on Artificial Intelligence and Statistics*, vol. 54, 2017, pp. 1273–1282.

[4] Y. Xiao, G. Shi, and M. Krunz, "Towards ubiquitous AI in 6G with federated learning," *CoRR*, vol. abs/2004.13563, 2020.

[5] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

[6] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in Neural Information Processing Systems*, vol. 32, pp. 14 774–14 784, 2019.

[7] J. Sun, A. Li, B. Wang, H. Yang, H. Li, and Y. Chen, "Soteria: Provable defense against privacy leakage in federated learning from representation perspective," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9311–9319.

[8] H. Kim, J. Park, M. Bennis, and S. Kim, "Blockchained on-device federated learning," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1279–1283, 2020.

[9] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, and Y. Liu, "Privacy-preserving blockchain-based federated learning for IoT devices," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1817–1829, 2021.

[10] M. Shayan, C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Biscotti: A blockchain system for private and secure federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1513–1525, 2021.

[11] S. Awan, F. Li, B. Luo, and M. Liu, "Poster: A reliable and accountable privacy-preserving federated learning framework using the blockchain," in *ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2561–2563.

[12] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2438–2455, 2021.

[13] Y. Liu, Z. Ai, S. Sun, S. Zhang, Z. Liu, and H. Yu, "Fedcoin: A peer-to-peer payment system for federated learning," in *Federated Learning - Privacy and Incentive*, ser. Lecture Notes in Computer Science, 2020, vol. 12500, pp. 125–138.

[14] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Annual Conference on Neural Information Processing Systems*, 2017, pp. 119–129.

[15] Q. Wang, Y. Guo, X. Wang, T. Ji, L. Yu, and P. Li, "AI at the edge: Blockchain-empowered secure multiparty learning with heterogeneous models," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9600–9610, 2020.

[16] J. Zhang, Y. Wu, and R. Pan, "Incentive mechanism for horizontal federated learning based on reputation and reverse auction," in *the Web Conference*, 2021, pp. 947–956.

[17] X. Chen, J. Ji, C. Luo, W. Liao, and P. Li, "When machine learning meets blockchain: A decentralized, privacy-preserving and secure design," in *IEEE International Conference on Big Data*, 2018, pp. 1178–1187.

[18] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in

USENIX Symposium on Operating Systems Design and Implementation, 1999, pp. 173–186.
[19] B. Zhao, K. R. Mopuri, and H. Bilen, "iDLG: Improved deep leakage from gradients," *CoRR*, vol. abs/2001.02610, 2020.
[20] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the Conference on Computer and Communications Security*, 2015, pp. 1310–1321.
[21] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu, "Privacy-enhanced federated learning against poisoning adversaries," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4574–4588, 2021.
[22] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.
[23] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *the Symposium on Operating Systems Principles*, 2017, pp. 51–68.
[24] 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.
[25] J. Zhang, Y. Wu, and R. Pan, "Incentive mechanism for horizontal federated learning based on reputation and reverse auction," in *the Web Conference*, 2021, pp. 947–956.
[26] Y. Huang, L. Chu, Z. Zhou, L. Wang, J. Liu, J. Pei, and Y. Zhang, "Personalized cross-silo federated learning on non-iid data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 9, 2021, pp. 7865–7873.
[27] Y. Hu, W. Xia, J. Xiao, and C. Wu, "GFL: A decentralized federated learning framework based on blockchain," *CoRR*, vol. abs/2010.10996, 2020.
[28] A. Z. H. Yapp, H. S. N. Koh, Y. T. Lai, J. Kang, X. Li, J. S. Ng, H. Jiang, W. Y. B. Lim, Z. Xiong, and D. Niyato, "Communication-efficient and scalable decentralized federated edge learning," in *Proceedings of the Joint Conference on Artificial Intelligence*, 2021, pp. 5032–5035.
[29] Y. Wang, C. Wang, Z. Wang, S. Zhou, H. Liu, J. Bi, C. Ding, and S. Rajasekaran, "Against membership inference attack: Pruning is all you need," in *the Joint Conference on Artificial Intelligence*, 2021, pp. 3141–3147.
[30] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. 11, 2008.

Zhen Qin received the M.S. degree in 2021 from School of Computer Engineering and Science, Shanghai University, Shanghai, China. He is currently pursuing the Ph.D. degree with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. His research interests include federated learning, distributed system and machine learning. He has published several papers on international conferences and journals, including IEEE ICWS 2019, ICSOC 2020, IEEE TSC, etc.

Shuiguang Deng is currently a full professor at the College of Computer Science and Technology in Zhejiang University, China, where he received a BS and PhD degree both in Computer Science in 2002 and 2007, respectively. He previously worked at the MIT in 2014 and Stanford University in 2015 as a visiting scholar. His research interests include Edge Computing, Service Computing, Cloud Computing, and Business Process Management. He serves for the journal IEEE Trans. on Services Computing, Knowledge and Information Systems, Computing, and IET CPS as an Associate Editor. Up to now, he has published more than 100 papers in journals and refereed conferences. In 2018, he was granted the Rising Star Award by IEEE TCSVC. He is a fellow of IET and a senior member of IEEE.

Xueqiang Yan is currently a Technology Expert with Wireless Technology Lab, Huawei Technologies. He was a Member of Technical Staff with Bell Labs from 2000 to 2004. From 2004 to 2016, he was the Director of Strategy Department, Alcatel-Lucent Shanghai Bell. His current research interests include future mobile network architecture, edge AI, data analytics, Blockchain and Internet of Things.

Schahram Dustdar is a Full Professor of Computer Science (Informatics) with a focus on Internet Technologies heading the Distributed Systems Group at the TU Wien. He is Chairman of the Informatics Section of the Academia Europaea (since December 9, 2016). He is elevated to IEEE Fellow (since January 2016). From 2004-2010 he was Honorary Professor of Information Systems at the Department of Computing Science at the University of Groningen (RuG), The Netherlands.

From December 2016 until January 2017 he was a Visiting Professor at the University of Sevilla, Spain and from January until June 2017 he was a Visiting Professor at UC Berkeley, USA. He is a member of the IEEE Conference Activities Committee (CAC) (since 2016), of the Section Committee of Informatics of the Academia Europaea (since 2015), a member of the Academia Europaea: The Academy of Europe, Informatics Section (since 2013). He is recipient of the ACM Distinguished Scientist award (2009) and the IBM Faculty Award (2012). He is an Associate Editor of IEEE Transactions on Services Computing, ACM Transactions on the Web, and ACM Transactions on Internet Technology and on the editorial board of IEEE Internet Computing. He is the Editor-in-Chief of Computing (an SCI-ranked journal of Springer).

Albert Y. Zomaya is the Peter Nicol Russell Chair Professor of Computer Science and Director of the Centre for Distributed and High-Performance Computing at the University of Sydney. To date, he has published more than 700 scientific papers and articles and is (co-)author/editor of 30 books. A sought-after speaker, he has delivered 250 keynote addresses, invited seminars, and media briefings. He is currently the Editor in Chief of the ACM Computing Surveys and served in the past as Editor in Chief of the IEEE Transactions on Computers (2010-2014) and the IEEE Transactions on Sustainable Computing (2016-2020).

Professor Zomaya is a decorated scholar with numerous accolades including Fellowship of the IEEE, the American Association for the Advancement of Science, and the Institution of Engineering and Technology. He is a Fellow of the Royal Society of New South Wales, Foreign Member of Academia Europaea, and Member of the European Academy of Sciences and Arts. Some of Professor Zomaya's recent awards include the New South Wales Premier's Prize of Excellence in Engineering and Information and Communications Technology (2019) and the Research Innovation Award, IEEE Technical Committee on Cloud Computing (2021). His research interests lie in parallel and distributed computing, networking, and complex systems.