FIFL: A Fair Incentive Mechanism for Federated Learning

Liang Gao
National University of Defense Technology
China

Li Li*
State Key Laboratory of IoTSC, University of Macau
China

Yingwen Chen*
National University of Defense Technology
China

Wenli Zheng*
Shanghai Jiaotong University
China

ChengZhong Xu
State Key Laboratory of IoTSC, University of Macau
China

Ming Xu
National University of Defense Technology
China

ABSTRACT

Federated learning is a novel machine learning framework that enables multiple devices to collaboratively train high-performance models while preserving data privacy. Federated learning is a kind of crowdsourcing computing, where a task publisher shares profit with workers to utilize their data and computing resources. Intuitively, devices have no interest to participate in training without rewards that match their expended resources. In addition, guarding against malicious workers is also essential because they may upload meaningless updates to get undeserving rewards or damage the global model. In order to effectively solve these problems, we propose FIFL, a fair incentive mechanism for federated learning. FIFL rewards workers fairly to attract reliable and efficient ones while punishing and eliminating the malicious ones based on a dynamic real-time worker assessment mechanism. We evaluate the effectiveness of FIFL through theoretical analysis and comprehensive experiments. The evaluation results show that FIFL fairly distributes rewards according to workers' behaviour and quality. FIFL increases the system revenue by 0.2% to 3.4% in reliable federations compared with baselines. In the unreliable scenario containing attackers which destroy the model's performance, the system revenue of FIFL outperforms the baselines by more than 46.7%.

KEYWORDS

federated learning; incentive mechanism; attack detection

ACM Reference Format:
Liang Gao, Li Li*, Yingwen Chen*, Wenli Zheng*, ChengZhong Xu, and Ming Xu. 2021. FIFL: A Fair Incentive Mechanism for Federated Learning. In *50th International Conference on Parallel Processing (ICPP '21), August 9–12, 2021, Lemont, IL, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3472456.3472469

1 INTRODUCTION

Traditionally, the training data are collected and stored in the cloud to train various machine learning models. However, data privacy [10, 15] becomes increasingly important for users and governments. Gathering the sensitive user data to a central server can cause serious privacy issues. In order to well protect data privacy in the training process, Federated learning (FL) [30] gains popularity as it allows workers (e.g., mobile devices) to train models collaboratively by only uploading the model gradients instead of raw data. Thus, data privacy is well preserved in the whole training process.

In a typical commercialization scenario of FL, data owners get rewards for sharing their data resources and computing resources. However, if workers' rewards do not match their corresponding utility and expenses, they would quit the current training or join other high-yield federation [31]. In addition, it is difficult to prevent attackers and free-riders [6, 20] without punishments. Existing research has already suggested that attacks such as sign-flipping attack [29] and data-poison attack [22] greatly damage model performance. The free-riders and low-quality workers can bring less revenue to the system but get larger rewards, which leads to a decline in the system revenue of the whole system. Even so, the previous incentive mechanisms [26, 32] assume all the workers to be honest, which ignore attackers and unstable workers [12] though they have a huge impact on the system revenue and expenditure. As attracting high-quality workers and guarding against potential attackers are important for FL, a fair incentive mechanism that tolerates Byzantine attack [28, 29] is a must-have.

There are two challenges in the FL incentive mechanisms. First, how can we accurately and efficiently identify the workers' utilities without heavy computation overhead? Second, how to ensure fairness and reliability of the incentive mechanism under attacks and deceptions? Since the workers' utilities determine their rewards, an efficient assessment method is essential for the incentive mechanism. However, workers' raw data is strictly protected in FL, thus the traditional assessment approaches that require direct contact with the original data are not practical in FL scenario. It is unreliable even if workers run the assessment procedure themselves to avoid privacy leakage because the results may be tampered [6]. Designing an efficient and reliable incentive mechanism for FL in unreliable scenarios is challenging.

We propose FIFL, an incentive mechanism that characterizes the assessment results of workers based on two indicators: 1) contribution and 2) reputation. Contribution measures workers' current utility to the system, and reputation is the probability of workers producing helpful updates in one period. FIFL combines contribution and reputation to decide how much to reward honest workers (or punish attackers). Besides, FIFL adopts a blockchain-based audit

ICPP '21, August 9–12, 2021, Lemont, IL, USA

ACM ISBN 978-1-4503-9068-2/21/08…$15.00
https://doi.org/10.1145/3472456.3472469

method and a reputation-based server selection method to prevent malicious nodes from manipulating the assessment results.

To assess the reputation, we first design an attack detection module to detect attackers. Honest workers train and upload gradients based on the correctly labelled local dataset, while unreliable attackers use forged or tampered data [1]. The attackers' local gradients are different from the honest workers' gradients which reflect the differences in their dataset [29]. We separate the attackers by checking the abnormal gradients and then filter them out to prevent model damage. Moreover, to avoid attackers consistently spamming the model, we design a long-term reputation assessing module. We extend the subjective logic model (SLM) method [9] with a time decay factor that calculates workers' reputation based on their historical updates. We verify that the reputation reflects the trustworthiness of workers through theoretical analysis and experiments. Although reputation identifies whether workers are trustworthy, it can not indicate workers' utilities. High-quality workers can better accelerate the convergence speed and enhance the performance of FL models [18], thus determining each worker' effectiveness in detail with contributions indicator is a prerequisite for a fair incentive. Zhan et al. [33] calculate contributions with a logarithmic fitting function about sample size and model accuracy, in which they take the number of workers' training samples as inputs. However, workers may deliberately exaggerate the number of their samples to obtain excess rewards. Xie et al.[28] estimate the contribution of workers by calculating the value loss caused by workers. We define workers' contributions based on the deviation distance of workers' local gradients to the global gradient to distinguish their utilities in real time. We analyze the theoretical basis for the gradient similarity-based contribution mechanism. The analysis indicates that FIFL is more reliable and lightweight than the previous methods which are based on inference loss [28].

As contribution and reputation respectively measure workers' utilities and trustworthiness, FIFL uses their product to decide the share of workers' rewards fairly. Thus, workers' rewards are proportional to their reputations and contributions. High-quality and reliable workers get more rewards, and malicious attackers are punished. To the best of our knowledge, FIFL is the first profit-sharing scheme that is robust against attackers. The evaluation shows that FIFL is the most attractive for high-quality workers as FIFL pays them more, and it achieves the highest system revenue than existing incentive mechanisms. In the experiment that simulates a real-world scenario with 38.5% of unreliable workers [21], FIFL's system revenue outperforms the baselines by more than 46.7%.

To summarize, the contributions of this paper are as follows:

(1) We develop an attack detection module that removes the potential malicious updates in advance to prevent model damage. We quantitatively assess the trustworthiness and utilities of workers in real time.
(2) We propose a fair incentive mechanism FIFL based on workers' contribution and reputation indicators. We conduct comprehensive experiments to evaluate FIFL's effectiveness.
(3) FIFL is the first solution of profit-sharing that works in unreliable federations that contains unstable attackers. FIFL maintains a constant system revenue in unreliable scenarios, while the system revenue of the baselines drops sharply.

The rest of this paper is organized as follows: Section 2 introduces related work; Section 3 introduces the background; Section 4 introduces the proposed FIFL; Section 5 evaluates system performance; Section 6 summarizes this paper.

2 RELATED WORK

Incentive Mechanism. Previous research mainly focuses on the payoff-sharing schemes to improve system revenue. In traditional distributed machine learning scenarios, the task publisher pays workers equal rewards [31], where all workers are assumed to be honest and equally important. Sreenivas et al. consider workers' heterogeneity and propose the Union incentive [8] mechanism which determines the reward-sharing based on the system's marginal gain. Wang and Tseng [26] present a payoff-sharing scheme with the Shapley game theory, which enhances the fairness of workers' marginal gain by traversing all possible combinations of workers. Yu et al. [32] propose the FL Incentivizer (FLI) method, which divides a given budget among workers to maximize the collective utility and minimize inequality. However, the above methods do not consider the potential risks from attackers, and they are all expensive in terms of communication and storage.

Worker Assessment. The influence of the utilities and trustworthiness of workers on FL has received much attention. The research on the impact of workers' utilities demonstrates that attackers significantly damage the FL model's performance [23]. Peng et al. [19] analyze the correlation between data quality and the FL model's performance. They determine that high-quality workers could accelerate the convergence speed. In addition, reputation is another widely used indicator to evaluate the trustworthiness of nodes. Josang et al. [11] advise that the reputation indicator should be adopted to keep the FL system stable and reliable. Wang et al. [24] propose a reputation scheme to encourage devices to play useful roles in the industrial Internet of Things, which solves the security and efficiency problems. Ghasempouri et al. [7] use a Trust and Reputation Interaction Model (TRIM) method to avoid malicious agents who do sequences of evil actions in eBay, Beta, and CORE TCMs. Most of the above assessment methods verify the worker's status and original data directly. However, the devices in FL can not satisfy the huge computational consumption needed in general assessment methods. The data of workers in federated learning cannot be shared for assessment. The worker assessment in FL is still under exploration as the nodes are flexible and their data cannot be accessed.

3 BACKGROUND ABOUT FEDERATED LEARNING

In this section, we first introduce the background about FL, then discuss three representative architectures of FL.

3.1 Federated Learning Framework

FL trains models with decentralized data. Assuming there are N workers, the local training data on worker i is D_i, which contains n_i training samples. Thus, there are totally $\sum_{i=1}^{N} n_i$ training samples. Let (x, y) represent the corresponding feature and label for a specific training sample. The objective function of FL is as follows:

$$\min_{\theta} \lim L(F(\theta), D) = \sum_{i=1}^{N} \frac{n_i}{\sum_{j=1}^{N} n_j} L_i(F(\theta), D_i), \quad (1)$$

where $D = \bigcup D_i, \forall i \in [N]$, θ is the parameter of model F. L indicates the global loss and L_i indicates the local loss of worker i. $L_i(F(\theta), D_i) = \frac{1}{n_i} \sum_{(x,y) \in D_i} l(F(\theta, x), y)$, where l is a loss function that measures the distance between the prediction label $F(\theta, x)$ and the ground truth label y.

In the training process of FL, at the beginning of local training iteration t each worker sets its local parameter as the global parameter $\theta_{i,0} = \theta_t$, and then computes its local gradients as: $G_i = \sum_{k=1}^{K} \frac{\partial L_i(\theta_{i,k}, D_i)}{\partial \theta_{i,k}}$, where K is local training iterations. After that, workers upload their local gradients to the server. The server aggregates local gradients to get the global gradient as follows:

$$\tilde{G} = \sum_{i=1}^{N} (\frac{n_i}{\sum_{j=1}^{N} n_j} G_i), \quad (2)$$

where $\frac{n_i}{\sum_{j=1}^{N} n_j}$ is the weight for worker i. In iteration t, the parameters of global model F is updated as follows:

$$\theta_{t+1} = \theta_t - \eta \tilde{G}_t, \quad (3)$$

where η is the learning rate, θ_t is the parameter in communication iteration t and $\tilde{G}_t$ is the global gradient. In the next iteration, workers train model from θ_{t+1}. The above steps iterates till the model converges.

3.2 Federated Learning Architectures

There are three representative architectures for FL: 1) centralized architecture, 2) decentralized architecture, 3) polycentric architecture. In the centralized architecture [34], all the workers interact with a central server. However, the bottlenecks of communication and computing on the central server usually hinder the deployment of FL on a large scale [5]. Decentralized FL [16] reduces the bottlenecks through sharing communication and computing overhead. Specifically, there are communication connections between any two workers, and each device acts as a parameter server for the gradient of $1/N$ partition. However, decentralized architecture lacks fault tolerance in which any node failure will cause the system to crash. Polycentric FL [14] combines the advantages of centralized and decentralized architecture.

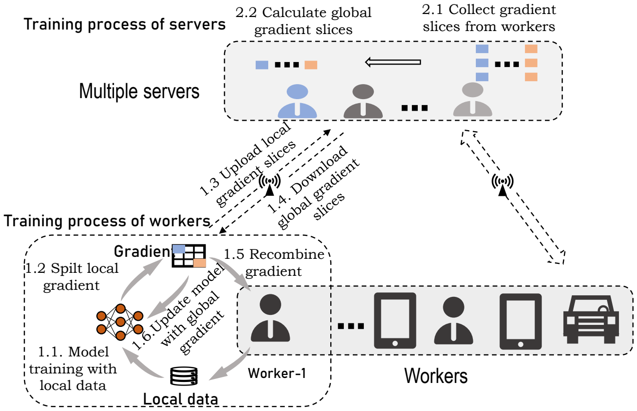
Training process of servers
2.2 Calculate global gradient slices
2.1 Collect gradient slices from workers
Multiple servers
1.3 Upload local gradient slices
1.4. Download global gradient slices
Training process of workers
Gradien
1.2 Spilt local gradient
1.5 Recombine gradient
1.6.Update model with global gradient
1.1. Model training with local data
Local data
Worker-1
Workers

Figure 1: The framework for polycentric FL.

The framework of polycentric FL is shown in Figure 1, which contains a server cluster $\mathbb{S}$ and a worker cluster $\mathbb{W}$. Suppose there are N devices, all devices being workers, and M of them are also selected as servers, which means $\mathbb{S} \sqsubset \mathbb{W}$. The iterative training process of polycentric FL is shown in Figure 1: **1.1)** local training to get the local gradient; **1.2)** local gradient is divided into M slices: $Spilt(G_i) = (g_i^1, g_i^2, ..., g_i^M)$; **1.3)** workers send in parallel local gradient slices to servers (for example worker i send its gradient slices g_i^j to corresponding servers j where g_i^j is the gradients produced by worker i and aggregated by server j, $\forall i \in [N]$ and $\forall j \in [M]$); **2.1)** each server aggregates one slice of gradients; **2.2)** servers calculate in parallel the global gradient slice as $\tilde{g}^j = \sum_{i=1}^{N} (\frac{n_i}{\sum_{t=1}^{N} n_t} g_i^j)$, $\forall j \in [M]$; **1.4)** servers broadcast global gradient slices, and workers download them; **1.5)** workers recombine global gradient slices to get the complete global gradient: $\tilde{G} = Recombine(\tilde{g}^1, \tilde{g}^2, ... \tilde{g}^M)$; 1.6) workers update local models with the global gradient.

The polycentric FL has two advantages: 1) the unreliable devices are divided into the set $\mathbb{W} \setminus \mathbb{S}$ (as a worker but not a server) to ensure system flexibility and fault tolerance; 2) each server aggregates a slice of gradients via parallel communication to reduce overhead. In this paper, we design FIFL based on polycentric architecture. We can easily generalize FIFL to the centralized ($M = 1$) or decentralized ($M = N$) architecture through controlling the number of servers.

4 FIFL MECHANISM

This section discusses the system design of FIFL. Figure 2 shows the system architecture and workflow of FIFL, FIFL can be mainly divided into the following four components: attack detection module, reputation module, contribution module and incentive module. Firstly, the attack detection module receives workers' gradients and eliminates harmful ones. Secondly, the reputation module measures workers' reputation based on the sequences of their historical detection results. After that, the contribution module computes workers' utilities. Finally, the incentive module determines the share of workers' rewards based on both reputation indicator and contribution indicator. All intermediate results and the signature of executors are stored in a blockchain [25] to prevent fraud and denial. In the following, we describe each component in FIFL, respectively.

4.1 Attack Detection Module

The attack detection module aims to identify and eliminate attackers. FL relies on the assumption that the averaging parameters of multiple local models yield a good estimator for the desired global model [34]. Due to parameter averaging characteristics, the offset injected by the attacker will be linearly weakened with respect to the number of workers. A small random noise does not affect model converge [17]. Hence, attackers have to upload gradients hugely different from the normal gradients to damage the global model, and this leads to the attacker's gradient deviation much greater than the deviation caused by non-iid data [3, 6]. Although multiple conspiring attackers can hide the backdoor in small changed gradients [2], in this paper, we follow most existing research on defenses [3, 6], we focus on disorganized attack scenarios with not colluding attackers.

In the attack detection module of FIFL, we identify attackers through detecting their abnormal gradients. We express the local

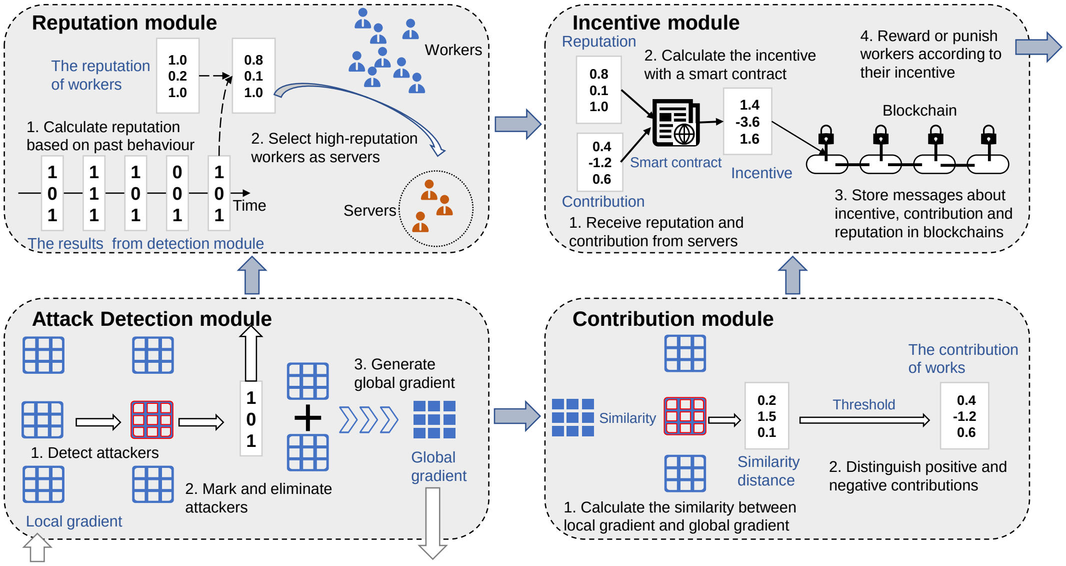
Reputation module
The reputation of workers
1.0 0.2 1.0
0.8 0.1 1.0
Workers
1. Calculate reputation based on past behaviour
2. Select high-reputation workers as servers
1 0 1
1 1 1
1 0 1
0 0 1
1 0 1
Time
Servers
The results from detection module
Incentive module
Reputation
0.8 0.1 1.0
2. Calculate the incentive with a smart contract
4. Reward or punish workers according to their incentive
Blockchain
1.4 -3.6 1.6
0.4 -1.2 0.6
Smart contract
Incentive
Contribution
1. Receive reputation and contribution from servers
3. Store messages about incentive, contribution and reputation in blockchains
Attack Detection module
1. Detect attackers
1 0 1
3. Generate global gradient
Global gradient
2. Mark and eliminate attackers
Local gradient
Contribution module
The contribution of works
Similarity
0.2 1.5 0.1
Threshold
0.4 -1.2 0.6
Similarity distance
2. Distinguish positive and negative contributions
1. Calculate the similarity between local gradient and global gradient

Figure 2: The workflow of FIFL.

gradient of worker i as:

$$G_i = \begin{cases} \frac{1}{n_i}\sum_{k=1}^{K}\frac{\partial l(F(\theta_{i,k}),D_i)}{\partial \theta_{i,k}} & \text{Honest worker} \\ \text{arbitrary or tampered value} & \text{Byzantine attacker} \end{cases} \quad (4)$$

To detect attackers, we define the detection score $S(\theta, G_i)$ as following loss difference as in [28]:

$$S(\theta, G_i) = L_t(\theta) - L_t(\theta - G_i) \quad (5)$$

where L_t is the test loss, G_i is the gradient to be detected. A more significant value of $S(\theta, G_i)$ suggests that G_i is useful for optimization. A negative $S(\theta, G_i)$ indicates that the attackers' gradient prevents model convergence. Howerver, this method is costly because calculating the marginal loss needing much inference. We use the the first Taylor's first-order expansion to simplify calculation: $S = L_t(\theta) - L_t(\theta - G_i) = \langle \nabla L_t(\theta), G_i\rangle - \rho||G_i||^2$, where $L_t(\theta - G) = L_t(\theta) - \langle \nabla L_t(\theta), G\rangle + \rho||G||^2$, ρ is a positive values close to 0. Discarding the second item of $\rho||G||^2$ which has less impact on the result, the detection score of worker i becomes $S_i = L_t(\theta) - L_t(\theta - G_i) \approx \langle G, G_i\rangle$, where $G = \nabla L_t(\theta)$ is the benchmark gradient from honest workers, G_i is the local gradient of worker i, $\langle G, G_i\rangle$ is the gradients' similarity distance. Intuitively, if we can ensure that servers are reliable, then their local gradients can be adopted as the benchmark gradients. In Section 4.5 we will discuss the reliability of servers.

In the polycentric architecture of FIFL, considering that each server aggregates one gradient slice, we aggregate detection scores from servers to generate unbiased detection results. Specifically, the global detection score of worker i is:

$$S_i = \sum_{j=1}^{M} S_i^j, \quad where \quad S_i^j = \langle g_j^j, g_i^j\rangle, \forall j \in [M] \quad (6)$$

where S_i^j is the detection score of server j assessing worker i, g_i^j is the gradient slice produced by worker i and aggregated by server j.

Then we set a hyperparameter threshold S_y to screen out the attackers in which:

$$r_i = \begin{cases} 1 & if \quad S_i \geq S_y & \text{honest worker} \\ 0 & if \quad S_i < S_y & \text{Byzantine worker} \end{cases} \quad (7)$$

Based on Equation 7, servers identify and reject attackers' gradients, the aggregated global gradient becomes $\tilde{G} = \sum_{i=1}^{N}(\frac{n_i r_i}{\sum_{j=0}^{N} r_i n_j} G_i)$. The hyperparameter S_y is the boundary to distinguish attackers from workers. S_y controls the trade-off between the detection accuracy and the false alarm rate in detecting attackers. As the difference in network structures and tasks cause different trade-off points, we advise that the task publisher should determine S_y through a short test process on a validation set before FL training. We analyze the impact of S_y in the experiments of Section 5.3.1.

4.2 Reputation Module

The reputation module measures the trustworthiness of workers, which is defined as the probability to provide valuable gradients. Any harmful update will reduce the accuracy and convergence speed of FL models. Therefore, a reputation assessing mechanism is required for identifying long-term stable devices. The reputation is calculated from historical events. In previous work, the subjective logic model (SLM) [4, 13] is widely used to estimate the reputation of a specific worker. SLM uses a triple $S = (St, Sd, Su)$ which represents (trust, distrust, uncertainty) to measure workers' reputation. Applying the SLM method to FL, we divide the gradients of workers into three types of events: positive events, negative events and uncertain events. If a local gradient is assessed to be useful by the attack detection module, it will be regarded as a positive event ($r_i = 1$); otherwise, it is a negative event ($r_i = 0$). Uncertain events include network transmission failures and unidentifiable gradients. In SLM method, the server counts the number of positive events, negative events, and uncertain events in each period. We use Pn to represent the number of negative events, and Pt to represent the number of positive events. Specifically, if the detection result of worker i's gradient G_i is $r_i = 0$ then $Pn_i = Pn_i + 1$, otherwise $Pt_i = Pt_i + 1$. In the SLM method, the scores of trust and distrust for worker i are calculated as:

$$St_i = (1 - Su_i)\frac{Pt_i}{Pt_i + Pn_i}, \quad \text{and} \quad Sn_i = (1 - Su_i)\frac{Pn_i}{Pt_i + Pn_i}, \tag{8}$$

where St indicates the trusted score, and Sn indicates the untrusted score, Su is the gradient transmission error rate. The reputation of worker i in an time interval is calculated as:

$$R_i = \alpha_t St_i - \alpha_b Sn_i - \alpha_u Su_i, \tag{9}$$

where $\alpha_t, \alpha_n, \alpha_u$ are weighted hyperparameters. In order to measure workers' reputation dynamically and objectively, we extend SLM through considering the time decay. Specifically, older events carry smaller weights while recent events are given larger weights. To this end, we set the reputation's time decay factor as γ. The reputation of worker i in time $t+1$ is calculated as follows:

$$R_i(t+1) = (1-\gamma)R_i(t) + \gamma r_i(t+1), \tag{10}$$

where $R_i(t)$ is the reputation of worker i at time t, γ controls the sensitivity of the reputation to current event $r_i(t+1)$.

Theorem 1: The value of a worker's reputation is consistent with its probability of producing useful updates.

Theoretical analysis: Assuming that the probability of a worker to be honest is $r(t)$ at time t, then according to Equation 10, expanding the formula step by step, we get:

$$\begin{aligned}\mathbb{E}R(t) &= (1-\gamma)R(t-1) + \gamma r(t)\\ &= (1-\gamma)((1-\gamma)R(t-2) + \gamma r(t-1)) + \gamma r(t)\\ &= (1-\gamma)^2 R(t-2) + (1-\gamma)\gamma r(t) + (1-\gamma)\gamma r(t-1)\\ &= (1-\gamma)^t R(0) + \sum_{i=1}^{t}(1-\gamma)^{t-i}\gamma r(0),\end{aligned} \tag{11}$$

since $0 < 1-\gamma < 1$, $(1-\gamma)^t \approx 0$ with the increase of t, the value of the first term of the equation approaches $(1-\gamma)^t R(0) \approx 0$. If the worker maintains a constant training strategy, that means the evil probability p is a constant, $r(t) = (1-p), \forall i$. The second term meets $\sum_{i=1}^{t}(1-\gamma)^{t-i}\gamma(1-p) = (1-p)\gamma\sum_{i=1}^{t}(1-\gamma)^{t-i}$, which is a geometric sequence with t. With $0 < \gamma < 1, 0 < (1-\gamma) < 1$, if the value of t is big enough then $(1-\gamma)^t \approx 0$. According to the summation formula of the geometric sequence, the reputation at time t becomes:

$$\begin{aligned}\mathbb{E}R(t) &= (1-\gamma)^t R(0) + \sum_{i=1}^{t}(1-\gamma)^{t-i}\gamma r(0)\\ &\approx (1-p)\gamma\frac{(1-\gamma)^{t-1}(1-(1-\gamma)^t)}{1-\gamma^{-1}}\\ &\approx (1-p).\end{aligned} \tag{12}$$

The result proves that the reputations are approximately equal to the probabilities of workers producing useful gradients.

4.3 Contribution Module

The contribution module aims to measure the utilities of workers. We first introduce the theoretical basis for using gradient similarity as a contribution measure. Without loss of generality, a high-quality worker's local gradient leads to a small loss $L(F(\theta - G_i))$. We denote $\tilde{G}$ as the unbiased gradient which makes the smallest loss $L(F(\theta - \tilde{G}))$. Assuming $L()$ is a β-smooth function, we get: $L(y) \leq L(x) + \langle \nabla L(x), y - x\rangle + \frac{\beta}{2}||y-x||^2$. With the μ convex assumption and quadratic upper bound assumption, we get: $L(y) \geq L(x) + \nabla L(x)(y-x) + \frac{\mu}{2}||y-x||^2$. If we take $\theta + G_i$ as y, take the fixed $\theta + \tilde{G}$ as x, and take the function $L(F())$ as L, then both the upper bound and the lower bound of $L(F(\theta + G_i))$ are positive correlation terms of $||G_i - \tilde{G}||^2$. That indicates that high-quality workers' local gradients are close to $\tilde{G}$, while the low-quality workers' gradients have more significant deviations from the $\tilde{G}$.

Inspired by the above analysis, we calculate workers' contribution based on the distance between the local gradient and global gradient. Specifically, we calculate the gradient distance for worker i as follows:

$$b_i = Dis(\tilde{G}, G_i) = \sum_{j=1}^{M} Dis(\tilde{g}^j, g_i^j), \tag{13}$$

where $Dis()$ is square of the Euclidean norm $||\tilde{G}, G_i||^2$. Compared with calculating contribution based on loss reduction [29], our method is computing-efficient because no inference procedure is needed.

Assuming G_0 is the gradient in which all values are 0, hence having no utility to the system, to calculate the relative contribution, we set a threshold $b_h = Dis(\tilde{G}, G_0)$ to distinguish the positive contribution from negative contribution. The contribution of worker i is:

$$C_i = \frac{b_h - b_i}{b_h} = 1 - \frac{b_i}{b_h}, \tag{14}$$

where if $b_i \leq b_h$ the contribution is positive, and if $b_i > b_h$, the contribution is negative. Furthermore, if we use worker i as baseline and set $b_h = ||\tilde{G}, G_i||^2$, any worker j whose utility is higher than that of worker i ($||\tilde{G}, G_j||^2 > ||\tilde{G}, G_i||^2$) could get profit, and the others ($||\tilde{G}, G_j||^2 \leq ||\tilde{G}, G_i||^2$) can not get profits. The threshold b_h effectively prevent free-riders and workers whose utility is lower than required from joining the federation.

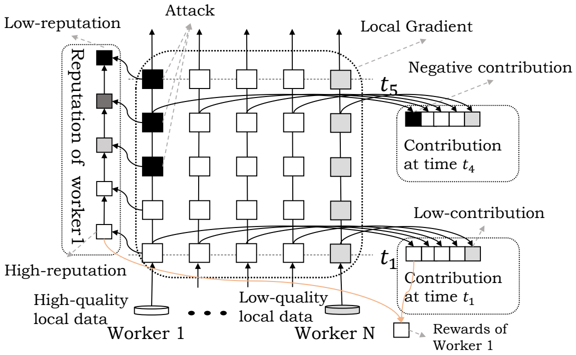

Figure 3: Contribution and reputation diagram.

4.4 Incentive Module

In FIFL, we calculate workers' reward share through jointly considering both contribution and reputation to achieve fairness. Figure 3 abstractly expresses the role of contribution, reputation and reward in FIFL. Contribution is the indicator to measure workers' utilities in each iteration. Reputation is the indicator that reflects the workers' trustworthiness of producing useful gradients. The task publisher determines the reward share as the product of contribution and reputation, the worker i's reward is:

$$I_i = R_i\frac{C_i}{\sum_{j, C_j > 0} C_j} = R_i\frac{b_h - b_i}{\sum_{j, b_j \leq b_h}(b_h - b_j)}, \tag{15}$$

where the sign of $(b_h - b_i)$ decides whether to reward or punish the worker. $\sum_{j,C_j>0} C_j$ is the total positive contribution in a training iteration. $\frac{C_i}{\sum_{j,C_j>0} C_j}$ is the weight of worker i's contribution.

Theorem 2: FIFL distributes profits fairly according to workers' contribution and reputation.

Theoretical analysis: We analyze the fairness for honest workers in FIFL, and the theoretical proof of fairness for attackers is similar. The rewards of honest worker i is: $I_i = R_i \frac{C_i}{\sum_{j,C_j>0} C_j}$. It is easy to get that $\frac{\partial I_i}{\partial R_i} = \frac{C_i}{\sum_{j,C_j>0} C_j} > 0$, and $\frac{\partial I_i}{\partial C_i} = \frac{R_i}{\sum_{j,C_j>0} C_j} > 0$. Therefore, the rewards of honest workers is positively correlated with contribution and reputation. If the contribution of worker i is greater than that of worker j (that is $C_i > C_j$) then $I_i > I_j$.

We use the correlation coefficient between the utilities and the rewards of workers to quantify the fairness of FIFL. Considering fairness among workers in a training iteration, we assume that all workers are honest, that is $R_i = R_j, \forall i, j$. That means the reputations of different workers are equal. Thus high-quality workers with greater utilities should get rewards according to their contribution indicators in FIFL. We take the contributions as the X-axis coordinates and the workers' rewards as the Y-axis coordinates, $X = C = (C_1, C_2, .., C_N)$ and $Y = I = (I_1, I_2, ..., I_N)$. Let $\hat{x}$ and $\hat{y}$ be the average of X and Y. The fairness coefficient is the co-correlation coefficient of X and Y, which is calculated as follows:

$$C_s = \frac{\sum_{i=1}^{N}(x_i - \hat{x})(y_i - \hat{y})}{N\delta(X)\delta(Y)}, \tag{16}$$

where $\delta(X)$ and $\delta(Y)$ are the standard deviations of X and Y. The range of the co-correlation coefficient is $[-1, 1]$. The larger the co-correlation coefficient, the higher the fairness, and vice versa. For honest workers, bringing $y_i = I_i = R_i \frac{C_i}{\sum C_j} = R_i \frac{x_i}{\sum x_j}$ and $R_i = R_j$ into the Equation 16, we get:

$$\begin{aligned} C_s &= \frac{\sum_{i=1}^{N}(x_i - \hat{x})(y_i - \hat{y})}{N\delta(X)\delta(Y)} \\ &= \frac{\sum_{i=1}^{N}(x_i - \frac{1}{N}\sum x_i)(R_i \frac{x_i}{\sum x_j} - \frac{1}{N}\sum R_i \frac{x_i}{\sum x_j})}{N\delta(X)\delta(Y)} \\ &= \frac{\frac{R_d}{\sum x_j}\frac{1}{N}\sum_{i=1}^{N}(x_i - \frac{1}{N}\sum x_i)^2}{N\delta(X)\delta(R_d \frac{X}{\sum x_j})} \\ &= 1. \end{aligned} \tag{17}$$

The fairness coefficient of contributions and rewards equals 1, indicating that the FIFL method achieves high fairness among workers. Similarly, we can prove that the fairness coefficient between rewards and reputations also equals 1.

4.5 Reliability of The Server Cluster

Obviously, FIFL relies on reliable servers to supply reliable attack detection results, reputation indicators and contribution indicators. We introduce our server selection method and audit strategy to ensure the reliability of servers, which also makes the system transparent and effective. Before the FL training, all workers conduct a small amount of training process and upload their models. The task publisher selects the devices with higher verification accuracy to join the initial server cluster. During the FL training process, we store the signatures of servers executing FIFL and their results about reputation and contribution in a blockchain. To prevent servers from manipulating reputation, we re-identify suspected result and compare it with the result in blockchain. For example, if the worker i suspects its reputation has been tampered by servers, the task publisher would recalculate the reputation value for worker i based on the global gradient. Comparing the task publisher's results with the blockchain's records, the server that manipulates the reputation can be traced by its signature and then removed. At the end of each iteration, the task publisher re-selects devices with high reputations to form a new server cluster for the next iteration.

With the blockchain-based audit mechanism and flexible high-reputation server selecting mechanism, we ensure the server cluster's reliability to enhance the FIFL's effectiveness and feasibility.

5 EVALUATION

5.1 Experiment Setup

We implement FIFL based on PyTorch [1] and build a simulator consisting of different agents with different data qualities in order to emulate the corresponding participating workers. In this work, we mainly emulate the following three types of workers in a federated learning system.

- **Honest workers.** Honest workers always calculate and upload local gradients honestly during the training process.
- **Sign-flipping workers [29].** The sign-flipping workers flip the sign of local gradients and multiply a coefficient p_s to undermine the global model. p_s indicates the attack intensity.
- **Data-poison workers [22].** The data-poison workers learn with a dataset in which part of the labels are incorrect. The percentage of incorrect labels p_d is used to represent the degree of unreliability.

Baselines. We use the following baselines for comparison purpose to evaluate the effectiveness of FIFL from different perspectives. Before presenting the corresponding baselines, we first introduce the related notations. We assume that the total profit for all the workers in a communication iteration t is $I_{sum}(t)$, where $I_{sum}(t) = \sum_{i=1}^{N} I_i(t)$. Worker i's reward at iteration t is expressed as follows:

$$I_i(t) = \frac{\omega_i(t)}{\sum_{j=1}^{N}\omega_j(t)} I_{sum}(t), \tag{18}$$

where $\omega_i(t)$ is worker i's rewards weight, and $\frac{\omega_i(t)}{\sum_{j=1}^{N}\omega_j(t)}$ is the normalized weight of worker i's rewards in t-th communication iteration. The relationship between the amount of training data and system revenue is expressed as the following utility function: $\Psi = log(1 + n)$ [33], where n is the amount of training data and Ψ is the revenue. The description of each adopted baseline is as follows:

- **Individual incentive [31]:** A worker' reward share is proportional to the its independent utility:

$$\omega_i(t) = \Psi_i, \tag{19}$$

  where $\Psi_i = \Psi(n_i)$ is worker i's utility of independent training.

[1] https://pytorch.org/

- **Equal incentive [31]:** This mechanism is widely adopted in traditional distributed machine learning, where the participating workers get the egalitarian payoff. Among N workers, the proportion of reward for worker i is:
$$\omega_i(t) = 1/N. \tag{20}$$
- **Union incentive [8]:** The Union incentive sets the worker's profit as the marginal utility which is defined as the revenue the federation gained while the worker joins. In Union incentive, worker i's reward share is:
$$\omega_i(t) = \Psi(\mathbb{A}) - \Psi(\mathbb{A} \setminus \{i\}), \tag{21}$$
where $\mathbb{A}$ is the worker set in the federation and $\mathbb{A} \setminus \{i\}$ represents the federation with worker i removed from $\mathbb{A}$.
- **Shapley incentive [26]:** Compared with the Union method, the Shapley incentive mechanism eliminates the influence of different combinations by traversing every workers' combinations. The Shapley method calculates the average marginal utility:
$$\omega_i(t) = \frac{(|\mathbb{P}_i| - 1)!(N - |\mathbb{P}_i|)!}{N!} \sum (\Psi(\mathbb{P}_i) - \Psi(\mathbb{P}_i \setminus \{i\})), \tag{22}$$
where $\mathbb{P}_i \subset \mathbb{A}$, $|\mathbb{A}| = N$ and $\{i\} \in \mathbb{P}_i$. The Shapley value is the average utility of worker i in different combinations $\mathbb{P}_i$, $(|\mathbb{P}_i| - 1)!(N - |\mathbb{P}_i|)!$ is the amount of combinations which contain i, and $N!$ is the total amount of workers' combinations.

Discussion: The above baselines' utility functions rely on the samples' number reported by workers. Although FIFL's contribution indicator and the Union's marginal utility have similar definitions, FIFL's gradient-based contribution can avoid fraud from workers. Besides, the above baselines have no way to defence against attackers who seek to damage the system. FIFL adopts the reputation indicator to measure the trustworthiness of workers and removes attackers by the attack detection module to obtain stable system revenue.

5.2 Comparison with Baselines

We first establish a federated learning system that contains 20 workers. In experiments, the total rewards I_{sum} which are distributed to workers are equal in all incentive mechanisms. The amount of local training samples for each worker is randomly generated, which is uniformly distributed in $[1, 10000]$. We divide workers into ten groups according to their qualities, where the i-th group's worker owns samples in the range of $[1000 * (i-1), 1000 * (i)]$. The workers greedily join a federated learning system with different incentive mechanisms to maximize their benefits. A worker's probability of entering a federation is the relative proportion of rewards received in this federation. We regard the reward proportion as the attractiveness of different incentive mechanisms to the worker. All experiments are repeated 100 times to reduce the impact of experimental deviation. Each experiment contains 500 communication iterations.

5.2.1 *Effectiveness of FIFL in reliable federations.*

Figure 4 shows the attractiveness of different incentive mechanisms to different types of workers. Figure 4 (a) shows the reward distribution of different incentive mechanisms. The x-axis represents workers owning different amount of training samples; the y-axis represents workers' rewards obtained from different incentive mechanisms.

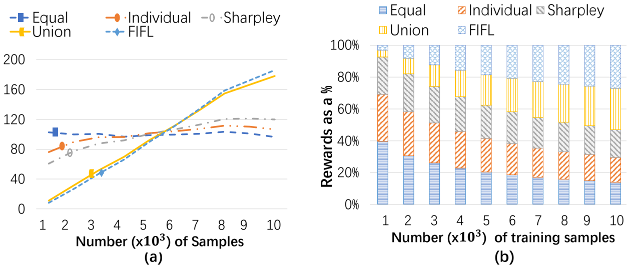

Figure 4: (a) The reward distribution for workers with different qualities of training samples. (b) The relative proportion of rewards (incentives' attractiveness) for workers with different incentive mechanisms.

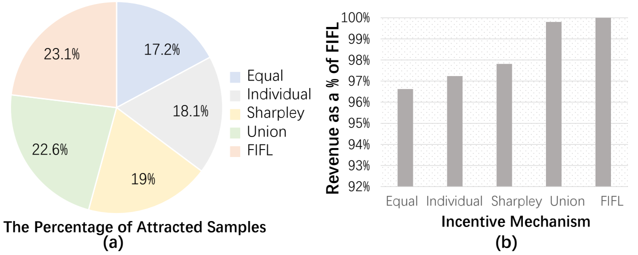

Figure 5: (a) The percentage of data attracted by different incentive mechanisms. (b) The relative system revenue in different incentive mechanisms compared with FIFL.

The Equal incentive mechanism rewards workers equally while the other methods tend to reward more to high-quality ones. The Union mechanism and FIFL mechanism pay more attention to high-quality workers than the Individual and Shapley methods. The FIFL mechanism spends the least on low-quality workers and the most on high-quality workers. Figure 4 (b) shows the attractiveness (the relative proportion of rewards) of each incentive to workers; the x-axis represents workers owning different amount of samples; the y-axis represents the attractiveness, which is proportional to the relative proportion of benefits that workers receive. The Equal mechanism attracts most low-quality workers who own less than 1000 samples (39.7%), while other methods tend to reduce investment in low-quality workers (less than 29%). The Individual and Shapley mechanisms get a similar tendency. The FIFL attracts most (27.1%) high-quality workers who own more than 9000 samples while the Union mechanism attracts 25.9%, the Equal mechanism attracts 14.0% and the Shapley mechanism attracts 17.4%.

Figure 5 shows the market attractiveness and system revenue of different incentive mechanisms. Figure 5 (a) shows the proportion of samples attracted by different incentives. The FIFL method receives the best market recognition, and it attracts 23.1% samples, followed by Union (22.6%), Shapley (19%), Individual (18.1%) and Equal (17.2%). Figure 5 (b) shows the relative system revenue of federations using different incentive mechanisms compared with FIFL. The x-axis represents incentives; the y-axis represents the relative revenue. The Equal method gets the lowest revenue (which is 3.4% lower than FIFL) because it indiscriminately incentivizes workers. The FIFL method achieves the best performance, and Union is only 0.2% lower than FIFL.

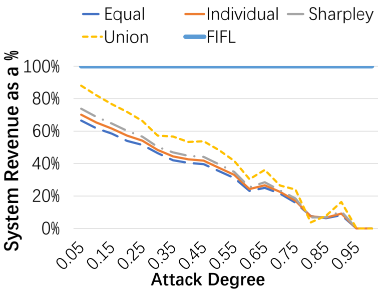

Figure 6: The system revenue in different incentive mechanisms under attacks relative to FIFL.

5.2.2 Advantage of FIFL in unreliable federations. Figure 6 shows the system revenues of different incentive mechanisms under attacks relative to FIFL. In the experiments, if an attacker a decreases the federation $\mathbb{A}$'s system revenue to $\Psi(\mathbb{A})-\Psi(\mathbb{A}\bigcup\{a\}) = \mho*\Psi(\mathbb{A})$, we define its attack degree as $\mho$. Actually, the ratio of unreliable workers in real-world is in the range of 8.0% to 38.5% [21, 27]. We take $\mho = 0.385$ as the representative real-world scenario. From the results, we demonstrate that FIFL's relative advantage expands with the increase of the attack degree. While the attack degree is 0.15, FIFL outperforms Union by 23.3%, Individual by 38.3%, Sharpley by 36.4%, Equal by 41.6%. In addition, in the unreliable scenario simulating the real-world, where the highest attack degree is 0.385 [7], the system revenue of FIFL outperforms Union by 46.7%, Individual by 57.4%, Sharpley by 55.3%, Equal by 60%. Attackers can not deceive FIFL because the attack detection module identifies and eliminates them. By contrast, the other methods have no defence against attacks, leading to a decline in their system revenue. In summary, FIFL effectively improves system revenue compared with the baselines. FIFL shows a huge advantage over baselines in the scenarios that include attackers.

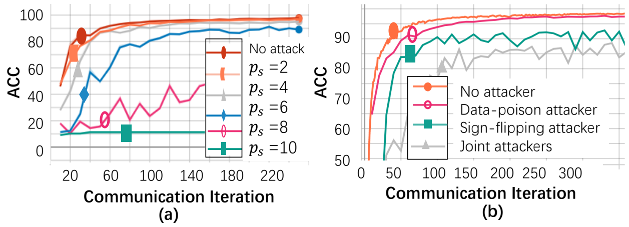

Figure 7: The damages of attackers on MNIST data set. (a) The accuracy (ACC) of training with sign-flipping attackers. (b) The accuracy of training with differents types of attackers. p_s is the Sign-flipping attackers' attack intensity. Joint attackers means both Sign-flipping attacker and Data-poison attacker.

5.3 Effectiveness of Each Module

5.3.1 Effectiveness of the Attack Detection Module. The attack detection module marks attackers and excludes them to protect the FL model. In this experiment, we first explore the impact of attackers on the model performance. We build a federated learning system consisting of 10 workers. The training data are uniformly distributed to each worker. We test the performance of the global model under various scenarios. Specifically, for MNIST data set, each of the 10 workers is given 6000 samples randomly sampled from the training set; for the CIFAR10 data set, each worker has 5000 samples. We use LeNet for MNIST and ResNet for CIFAR10.

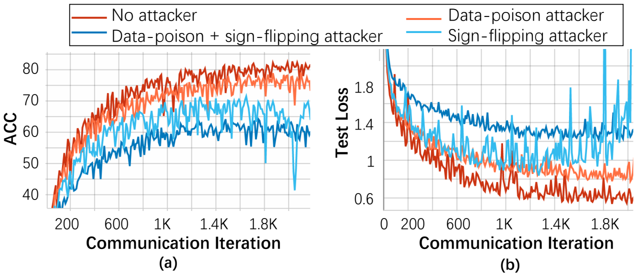

Figure 8: The damages to the global model training with attackers on CIFAR10 data set. (a) The accuracy (ACC) and (b) the test loss of global models training with attackers.

Figure 7 shows the attackers' damage. Figure 7 (a) shows the impact of workers' attack intensity p_s. The x-axis represents the communication iteration, and the y-axis represents the global model's accuracy (ACC). We can draw the following conclusions: the damage of sign-flipping attackers increases as the attack intensity increases. The model accuracy decreased more than 30% while training with the attacker $p_s = 8$, while the accuracy only decreased 3% when $p_s = 4$. As the figure shows, attackers also slow down the convergence speed, the model training with attacker $p_s = 6$ spends 2× communication iterations to reach convergence as without attackers. A strong aggressive attacker ($p_s \geq 10$) thoroughly crashes the model where the loss becomes Not a Number (NAN). Figure 7 (b) shows the accuracy of models on MNIST which is trained with different types of attackers. Figure 8 shows accuracy (ACC) and test loss of FL model on CIFAR10 data set. We get the same conclusion from them. The sign-flipping attacker caused more damage to the model than the data-poison attacker. The combination of attackers causes most damage to the global model on both two data sets.

Figure 9 (a) shows the impact of hyperparameter S_y in the detection module; the x-axis represents the attack intensity, and the y-axis represents the detection accuracy. The detection accuracy increases with the rise of attack intensity. The results indicate that a higher p_s leads to larger deviations on attackers' gradients, which is easier to be detected. A smaller detection threshold S_y would increase the detection accuracy, which increases by 26% (from 0.63 to 0.89) as S_y reduces from 0.15 to 0.09 for the attacker ($p_s = 2$). However, the greatest S_y is not necessarily the best. As shown in Figure 9 (b), where x-axis represents different values of S_y, left y-axis represents the accuracy of detecting positive events (TP), and right y-axis represents the true negative (TN) rate. TP increases but TN decreases as S_y increases. That demonstrates that S_y is the hyperparameter to trade off TN and TP. Figure 10 shows the effectiveness of the attack detection module; the x-axis is the communication iteration; the y-axis is the global model's test accuracy. The results indicate that the attack detection module is useful to prevent attackers from damaging the global model. The model with the attack detection module maintains high performance, and the model without the attack detection module crashed under high-intensity attacks.

5.3.2 Effectiveness of the Reputation Module. Reputation indicates workers' probabilities to produce useful updates, which is calculated based on their historical behaviour. We set the probabilities for

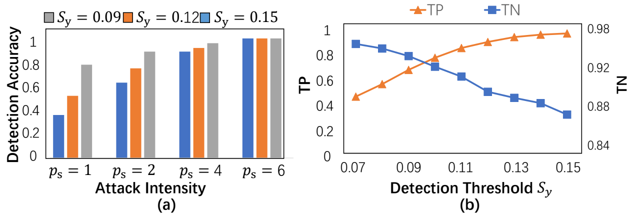

Figure 9: (a) The influence of detection threshold S_y and attack intensity p_s for detection module. (b) The tradeoff of True Negative (TN) rate and True Positive (TP) rate with different detection threshold S_y.

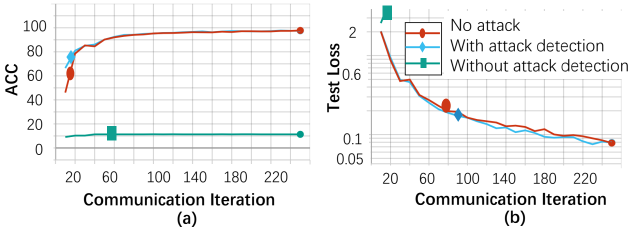

Figure 10: The effectiveness of the attack detection module. (a) The accuracy (ACC) (b) the test loss of models.

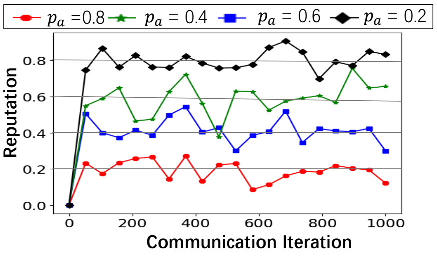

Figure 11: The reputation indicator reflects workers' probabilities to produce useful gradients. p_a is an attacker's probability to attack in an iteration.

four attackers doing evil as $(0.2, 0.4, 0.6, 0.8)$, the corresponding trustworthiness is $(0.8, 0.6, 0.4, 0.2)$, respectively. We set the initial reputation value as 0. Figure 11 shows the reputation module's effectiveness; the x-axis represents the communication iteration; the y-axis represents the reputation value. We can observe that a worker's reputation fluctuates around its trustworthiness, and the reputation does not converge to a fixed value because the reputation still keeps sensitive to current events. The experiment indicates that the reputation accurately reflects workers' probabilities to generate useful gradients.

5.3.3 *Effectiveness of the Contribution Module.*

To analyze different workers' contributions, we set several data-poison attackers with incorrectly labelled data sets. The error rate of labels indicates attackers' unreliability degree p_d. Figure 12 shows workers' contributions, where the x-axis is the communication iteration, and the y-axis is workers' contributions. As attackers' mislabeled data increases, the distances between their local gradients and the global gradient also increases. Therefore, we can distinguish workers of different qualities and calculate their contributions. We set the threshold $b_h = ||G_{0.2}, \tilde{G}||$, where $G_{0.2}$ is the local gradients of the worker whose data labeling error rate $p_d = 0.2$. Only the workers better than the threshold make positive contributions; the others make negative contributions. The qualities of workers and their contributions are positively correlated.

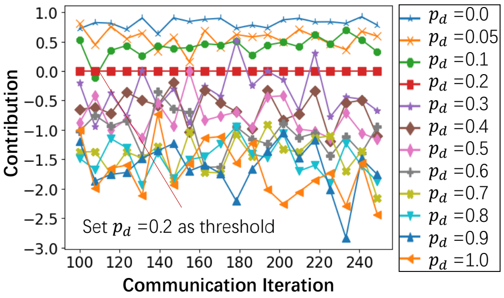

Figure 12: The contribution of workers. p_d is the proportion of workers' mislabeled data (p_d=0.2 is the threshold).

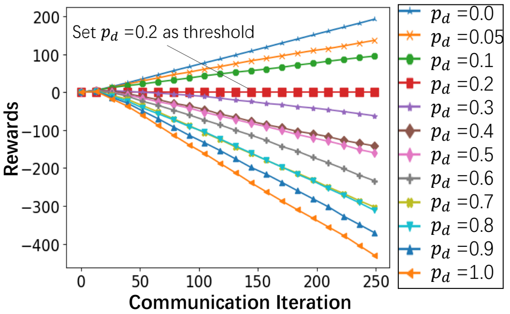

Figure 13: The cumulative rewards (or punishments) of workers owning with different data while $b_h = ||G_{0.2}, \tilde{G}||$. p_d is the proportion of workers' mislabeled data.

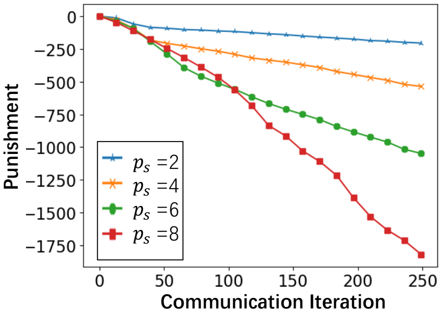

Figure 14: The punishments for sign-flipping attackers. p_s is the attack intensity.

5.3.4 *Effectiveness of the Incentive Module.*

FIFL fairly distributes rewards to honest workers and punishes malicious workers. Figure 13 compares the cumulative rewards of different workers using FIFL. The x-axis represents the communication iteration, and the y-axis represents workers' cumulative rewards. p_d is the proportion of workers' mislabeled data. We still set $p_d = 0.2$ as the threshold. From the results, we can conclude that only workers whose data qualities exceed the threshold can be judged as collaborators and receive rewards from FIFL. The workers whose qualities are lower than the threshold will be punished. The cumulative rewards are positively related to data labelling quality. FIFL punishes more to workers those workers more who own less reliable data. We also evaluate the fairness of FIFL for sign-flipping attackers. Figure 14 shows the punishments for sign-flipping attackers; the x-axis represents the communication iteration; the y-axis represents attackers' cumulative punishments. The punishment is positively related to the attack intensity. The results indicate that FIFL fairly rewards (or punishes) workers according to workers' utilities and behaviours.

6 CONCLUSION

In this paper, we propose FIFL, an incentive mechanism for federated learning in order to share profit with workers according to their behaviours and utilities. FIFL is the first profit-sharing scheme for federated learning that works in unreliable scenarios. We prevent the attackers from damaging the model by detecting abnormal updates in FIFL. For all the participating workers, we use contribution indicators to measure their utilities and reputation indicators to measure their probabilities to make reliable updates. Combining the reputation and contribution, the task publisher fairly determines the rewards for honest workers and punishments for attackers.

ACKNOWLEDGMENTS

This research was partially supported by The NUDT Research Grants (No. ZK19-38) and The Research on Privacy-Preserving Issues Within Smart City Truth Discovery Process (No. 61872372). Please contact Dr. Yingwen Chen (ywch@nudt.edu.cn), Dr. Wenli Zheng (zheng-wl@cs.sjtu.edu.cn) and Dr. Li Li (LLiLi@um.edu.mo) for correspondence.

REFERENCES

[1] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2938–2948.

[2] Gilad Baruch, Moran Baruch, and Yoav Goldberg. 2019. A Little Is Enough: Circumventing Defenses For Distributed Learning. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/ec1c59141046cd1866bbbcdfb6ae31d4-Paper.pdf

[3] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 118–128.

[4] Tong Cheng, Guangchi Liu, Qing Yang, and Jianguo Sun. 2019. Trust assessment in vehicular social network based on three-valued subjective logic. *IEEE Transactions on Multimedia* 21, 3 (2019), 652–663.

[5] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. 2020. Distributionally Robust Federated Averaging. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 15111–15122. https://proceedings.neurips.cc/paper/2020/file/ac450d10e166657ec8f93a1b65ca1b14-Paper.pdf

[6] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Louis Alexandre Rouault. 2018. The Hidden Vulnerability of Distributed Learning in Byzantium. (2018), 13. http://infoscience.epfl.ch/record/256124 camera ready version available also on ICML proceedings (open access).

[7] Seyed Asgary Ghasempouri and Behrouz Tork Ladani. 2020. Model checking of robustness properties in trust and reputation systems. *Future Generation Computer Systems* 108 (2020), 302–319.

[8] Sreenivas Gollapudi, Kostas Kollias, Debmalya Panigrahi, and Venetia Pliatsika. 2017. Profit Sharing and Efficiency in Utility Games. In *ESA*.

[9] Seunghan Han, Bonjung Koo, and Walter Stechele. 2010. Subjective Logic Based Approach to Modeling Default Reasoning for Visual Surveillance. *IEEE Internet Computing - INTERNET*, 112–119. https://doi.org/10.1109/ICSC.2010.55

[10] Kai Huang, Ximeng Liu, Shaojing Fu, Deke Guo, and Ming Xu. 2021. A Lightweight Privacy-Preserving CNN Feature Extraction Framework for Mobile Sensing. *IEEE Transactions on Dependable and Secure Computing* 18, 3 (2021), 1441–1455. https://doi.org/10.1109/TDSC.2019.2913362

[11] Audun Josang, Roslan Ismail, and Colin Boyd. 2007. A survey of trust and reputation systems for online service provision. 43, 2 (2007), 618–644.

[12] Jiawen Kang, Zehui Xiong, Dusit Niyato, Shengli Xie, and Junshan Zhang. 2019. Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory. *IEEE Internet of Things Journal* 6, 6 (2019), 10700–10714.

[13] Guangchi Liu, Qing Yang, Honggang Wang, and Alex X Liu. 2019. Three-valued subjective logic: A model for trust assessment in online social networks. *IEEE Transactions on Dependable and Secure Computing* (2019).

[14] Lumin Liu, Jun Zhang, S. H. Song, and Khaled B. Letaief. 2019. Client-Edge-Cloud Hierarchical Federated Learning. arXiv:1905.06641 [cs.NI]

[15] Yuchuan Luo, Xiaohua Jia, Shaojing Fu, and Ming Xu. 2019. pRide: Privacy-Preserving Ride Matching Over Road Networks for Online Ride-Hailing Service. *IEEE Transactions on Information Forensics and Security* 14, 7 (2019), 1791–1802. https://doi.org/10.1109/TIFS.2018.2885282

[16] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*. PMLR, 1273–1282.

[17] Arvind Neelakantan, Luke Vilnis, Quoc Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. 2015. Adding Gradient Noise Improves Learning for Very Deep Networks. (11 2015).

[18] Takayuki Nishio and Ryo Yonetani. 2019. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 1–7.

[19] Yanghua Peng, Yixin Bao, Yangrui Chen, Chuan Wu, and Chuanxiong Guo. 2018. Optimus: an efficient dynamic resource scheduler for deep learning clusters. In *Proceedings of the Thirteenth EuroSys Conference*. 1–14.

[20] Shiqi Shen, S. Tople, and P. Saxena. 2016. Auror: defending against poisoning attacks in collaborative deep learning systems. *Proceedings of the 32nd Annual Conference on Computer Security Applications* (2016).

[21] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. 2021. Learning from Noisy Labels with Deep Neural Networks: A Survey. arXiv:2007.08199 [cs.LG]

[22] Gan Sun, Yang Cong, Jiahua Dong, Qiang Wang, and Ji Liu. 2020. Data Poisoning Attacks on Federated Machine Learning. arXiv:2004.10020 [cs.CR]

[23] Bo Tang, Hongjuan Kang, Jingwen Fan, Qi Li, and Ravi Sandhu. 2019. Iot passport: A blockchain-based trust framework for collaborative internet-of-things. In *In the 24th ACM symposium on access control models and technologies*. 83–92.

[24] Eric Ke Wang, Zuodong Liang, Chienming Chen, Saru Kumari, and Muhammad Khurram Khan. 2020. PoRX: A reputation incentive scheme for blockchain consensus of IIoT. *Future Generation Computer Systems* 102 (2020), 140–151.

[25] Huaimin Wang, Zibin Zheng, Shaoan Xie, Hong-Ning Dai, and Xiangping Chen. 2018. Blockchain challenges and opportunities: a survey. *International Journal of Web and Grid Services* 14 (10 2018), 352 – 375. https://doi.org/10.1504/IJWGS.2018.10016848

[26] Yue Wang and Mitchell M. Tseng. 2011. Adaptive attribute selection for configurator design via Shapley value. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 25, 2 (2011), 185–195. https://doi.org/10.1017/S0890060410000624

[27] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. 2015. Learning from massive noisy labeled data for image classification. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2691–2699. https://doi.org/10.1109/CVPR.2015.7298885

[28] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. 2018. Zeno: Byzantine-suspicious stochastic gradient descent. *arXiv preprint arXiv:1805.10032* 24 (2018).

[29] Cong Xie, Sanmi Koyejo, and Indranil Gupta. 2020. Zeno++: Robust fully asynchronous SGD. In *International Conference on Machine Learning*. PMLR, 10495–10503.

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

[31] S. Yang, F. Wu, S. Tang, X. Gao, B. Yang, and G. Chen. 2017. On Designing Data Quality-Aware Truth Estimation and Surplus Sharing Method for Mobile Crowdsensing. *IEEE Journal on Selected Areas in Communications* 35, 4 (2017), 832–847. https://doi.org/10.1109/JSAC.2017.2676898

[32] Han Yu, Zelei Liu, Yang Liu, Tianjian Chen, Mingshu Cong, Xi Weng, Dusit Niyato, and Qiang Yang. 2020. A Fairness-Aware Incentive Scheme for Federated Learning. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society* (New York, NY, USA) *(AIES '20)*. Association for Computing Machinery, New York, NY, USA, 393–399. https://doi.org/10.1145/3375627.3375840

[33] Yufeng Zhan, Peng Li, Zhihao Qu, Deze Zeng, and Song Guo. 2020. A learning-based incentive mechanism for federated learning. *IEEE Internet of Things Journal* (2020).

[34] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582* (2018).