Computing and Informatics, Vol. 41, 2022, 1401–1424, doi: 10.31577/cai_2022_6_1401

# BVFB: TRAINING BEHAVIOR VERIFICATION MECHANISM FOR SECURE BLOCKCHAIN-BASED FEDERATED LEARNING

Zhaohui Zhang

*School of Computer Science and Technology, Donghua University*
*Shanghai, 201620, China*
*&*
*Shanghai Engineering Research Center of Network Information Services*
*Shanghai, 201804, China*
*e-mail:* zhzhang@dhu.edu.cn

Jiawei Hu, Lina Ma, Ruoxuan Pei, Pengwei Wang

*School of Computer Science and Technology, Donghua University*
*Shanghai, 201620, China*
*e-mail:* hjw012029@163.com

**Abstract.** There are still two problems of the existing methods of defending against poisoning attacks of the blockchain-based federated learning: 1) It is difficult to accurately identify the nodes under attack; 2) The effect of the model is greatly affected when the number of malicious nodes exceeds a half. So, an innovative secure mechanism is proposed for blockchain-based federated learning, which is called the training behavior verification mechanism. The mechanism describes the consistent training behavior rules of nodes by constructing the training behavior model, and distinguishes honest nodes from malicious nodes by comparing the differences in training behavior models on the training behavior verification algorithm. Experiments show that the new mechanism can effectively resist more than half of the label-flipping attacks and backdoor attacks, and has the advantages of higher stability and higher accuracy than methods such as Krum, Trimmed Mean, and Median.

**Keywords:** Federated learning, blockchain, poisoning attack, behavior verification, secure aggregation

## 1 INTRODUCTION

Federated learning (FL), as a distributed learning framework [1, 2], allows participants to train models in a distributed manner while safeguarding their data privacy [3], effectively solving the problem of "data silos" due to data privacy and other issues [4]. In traditional FL, a centralized server is used to aggregate, distribute, and update models [5]. This centralized server does not have a direct access to the participants' datasets and training process, and a malicious node can perform a poisoning attack by modifying the classification boundaries of the local model [6], which often affects the entire training process and leads to the failure of the aggregated model [7]. In this paper, we focus on two common poisoning attacks, one is a label flipping attack [8], which causes misclassification of the model by flipping the labels of normal samples to the target labels, and the second one is a backdoor attack [9], which triggers the classification effect of a specific sample by adding hidden triggers with strong links to the target labels in normal samples.

Poisoning attacks can lead the malicious node to submit local models containing malicious parameters. To address this situation, existing research is mainly based on analyzing the discrepancy between malicious and honest parameters to eliminate malicious parameters. The research solutions are broadly divided into two categories, one is to construct specific secure aggregation rules, such as *Trimmed Mean* [11], *Median* [11], *Krum* [10] and other aggregation rules. The idea of mean or median is used to weaken the effect of malicious parameters on the model during aggregation; the other category is to prevent the malicious parameters from participating in aggregation by comparing them with honest parameters in certain dimensions such as data distribution through the anomaly detection [12, 13] to identify them. For the data with obvious poisoning effect, the above scheme can be used for safe model aggregation, but for some backdoor attacks which are not particularly effective [14], it is difficult to carry out the safe model aggregation.

Traditional centralized servers are often managed by a third party, however, the reliability of such third parties cannot be guaranteed. A server attacked by a malicious node can easily cause leakage of important private data, such as financial data [15]. Driven by some illegal interests, malicious node may return wrong models to participants in the process of updating models and distributing them after conspiring with third parties [16]. Moreover, under traditional FL, nodes are non-anonymous to each other, which allows malicious actors to conspire with each other, resulting in larger scale attacks. Blockchain, as a distributed shared ledger maintained by multiple parties, establishes multi-party trust relationships through cryptographic techniques and has the characteristics of decentralization, immutability, and anonymity. Existing research combines blockchain to build trusted FL and use blockchain instead of centralized servers to complete FL tasks such as model aggregation, which effectively solves the third-party trust problem in FL [17, 18, 19].

However, combining blockchain only ensures the reliability of model parameters in distribution and aggregation, and poisoning attacks still exist. Existing

research prevents the problem of poisoning attacks in FL on the basis of blockchain, which are broadly classified into three categories. The first category is by establishing a committee mechanism in the blockchain [20, 21, 22], which identifies the authenticated nodes based on the keys issued by the committee, and it is able to prevent malicious nodes from submitting local models provided that the authenticated nodes are all honest, however, it is also possible for malicious node to execute attacks by controlling the authenticated nodes. The second category is to maintain the reliability of model updates by establishing a reputation mechanism [23, 24], which is often based on a voting mechanism or subjective judgment, which may lead to subjective judgment errors or malicious node intentionally messing up. The third category is to combine existing secure aggregation rules or anomaly detection [25, 26], but these aggregation and detection still have the problems mentioned above.

In addition, the schemes mentioned above are difficult to accurately identify malicious nodes after a poisoning attack, and many of them will lose their effectiveness when the number of malicious nodes exceeds a half. Therefore, if malicious nodes can be identified and eliminated, and thus malicious parameters involved in aggregation can be eliminated, secure aggregation by legitimate nodes can be ensured. In this paper, we propose a blockchain-based FL based on training behavior verification mechanism, which can effectively identify malicious nodes by using the behavioral commonality of nodes in local training models to construct training behavior models and combining with training behavior verification algorithms. When the number of malicious nodes exceeds a half, it can effectively detect the poisoning attacks that occur during the training process and identify malicious nodes. The main contributions of this paper are as follows:

- We propose a new secure mechanism (BVFB, Training Behavior Verification Mechanism for Secure Federated Learning on Blockchain), which effectively ensures the reliability of local model aggregation of participating nodes during the federation learning training process by combining blockchain and federation learning.
- We propose a training behavior model, which is used to characterize the training behavior of nodes during the local training process.
- We propose a verification aggregation algorithm based on the training behavior model, which ensures the security of model aggregation by verifying out the malicious models that appear during the model aggregation process.

The rest of this paper is organized as follows. Section 2 describes the related work. Section 3 describes the composition and workflow of the proposed framework, and outlines the training behavior validation mechanism. Sections 4 and 5 describe training behavior modeling and training behavior verification. Section 6 conducts experiments and performance evaluation of the proposed framework. Finally, the work of this paper is summarized in Section 7.

## 2 RELATED WORK

Research on model poisoning attacks in FL has always attracted much attention. The poisoning attack defense methods under traditional FL are mainly divided into two categories. One is the use of aggregation rules to ensure the reliability of model parameters, and these aggregation rules mitigate the damage of malicious models to the aggregated model by using specific rules in aggregation to reduce the aggregation participation of malicious parameters. Yin et al. [11] proposed *Trimmed Mean* and *Median*. The former aggregates the model by removing the maximum and minimum values of the model parameters and calculating the mean of the remaining parameters, and the latter aggregates by calculating the median of the model parameters. Blanchard et al. [10] proposed *Krum* by selecting one of the models from the model that is similar to all other models for aggregation. With no more than half of the malicious nodes, *Krum* is able to select the honest nodes for aggregation with high probability. While these aforementioned schemes are able to eliminate the effect of malicious components on the model with a relatively obvious attack, they show poor results for some backdoor attacks [14]. Another category is to use anomaly detection to guarantee the reliability of model parameters, which are distinguished by the differences between malicious models and honest models in certain aspects, such as data distribution and accuracy. Bhagoji et al. [12] proposed an accuracy checking method and a weight update statistics method, which were detected by comparing the effects of different model combinations and comparing the differences between the histograms of model parameter updates. However, these anomaly detections were shown to be vulnerable to backdoor attacks.

Traditional FL is vulnerable to malicious attack or illegal control due to the use of centralized servers for aggregation. The combination of blockchain and FL can well solve the problem of third-party trust in traditional FL. However, combining only blockchain still can not effectively defend against poisoning attacks, and the existing solutions to combine blockchain and solve poisoning attacks are roughly divided into three categories. The first category is to establish a trust committee mechanism to guarantee the reliability of model parameters. Li et al. [20] proposed a committee consensus mechanism and Weng et al. [21] proposed the *DeepChain*, which guarantees the reliability of model training by an elected committee. The *VFChain* [22] proposed by Peng et al. identifies legitimate participants through the authentication key issued by the committee. These methods need to be performed under the premise that the authentication node is not controlled, but the reality is that even the authentication node may lose control over the node and training data. The second category is to establish a reputation mechanism to maintain the reliability of participating nodes. Kang et al. [23] proposed a combination of reputation and contract theory, using a multi-weighted subjective logic model to select legitimate nodes. Chen et al. [24] combined reputation and reward mechanism to select honest nodes by voting. These methods can effectively decide legal nodes on the premise that all legal nodes abide by the rules, but in the case of illegal nodes maliciously disrupting, it will affect the effect of the final model. The third category

is secure aggregation by combining aggregation rules or anomaly detection. Shayan et al. [25] proposed a *Biscotti* to prevent poisoning attacks by using blockchain instead of centralized servers and combining *Krum*. Chen et al. [27] proposed a FL approach with model verification to detect the reliability of the model by verifying the accuracy of the model. But a malicious node can perform a backdoor attack by maintaining a certain accuracy threshold.

In addition to this, the abovementioned schemes lack effective methods to prevent attacks and identify malicious nodes effectively in the case of more than half of the malicious nodes. In order to effectively identify malicious nodes, this paper introduces a training behavior verification mechanism in the blockchain-based FL. This new secure federated learning ensures that only the local parameters of honest nodes participate in model aggregation to ensure the security and accuracy of subsequent model aggregation.

## 3 NEW SECURE MECHANISM FOR FEDERATED LEARNING ON BLOCKCHAIN

### 3.1 Work Flow of BVFB

As shown in Figure 1, the blockchain-based FL with training behavior verification mechanism (BVFB) is mainly composed of three parts, namely the publisher, the participant and the blockchain. Task publishers mainly publish FL tasks in the blockchain. Participants are usually distributed on different nodes of the blockchain, each node has independent local data, and participates in model training on the premise that the local data is consistent with the publisher's requirements. The blockchain replaces the parameter server in traditional FL, is responsible for node verification and model aggregation. The operation under a global iteration includes the following steps:

**Step 1: Publish the FL task.** The task publisher publishes the FL task according to the requirements. The FL task generally includes the model convergence requirements, training data requirements, initialization model and corresponding hyperparameters. When publishers publish tasks, they usually encourage participants to actively participate in FL through rewards. Tasks are published in the genesis block of the blockchain.

**Step 2: Update the local model.** The node downloads the latest global model from the latest block of the blockchain. The node then uses local training samples to train on the global model. The gradient is continuously updated by the optimization algorithm until the value of the loss function is minimized, and the final local model is obtained.

**Step 3: Generate the training behavior model.** The node generates training behavior pace and training behavior direction in the process of local training, and then build the training behavior model (see Section 4 for details).

**Step 4: Upload to the blockchain.** After the node finish local training, the generated final local model and the training behavior model are packaged and uploaded to the blockchain network.

**Step 5: Verify the training behavior model.** Miners in the blockchain verify the collected training behavior models, and mark the nodes corresponding to the verified training behavior models as honest nodes (see Section 5 for details).

**Step 6: Update the global model.** The blockchain aggregates the local models marked as honest nodes to update the global model.

In the whole FL process, the above steps 2–6 are repeated continuously in each global iteration until the global model reaches convergence.

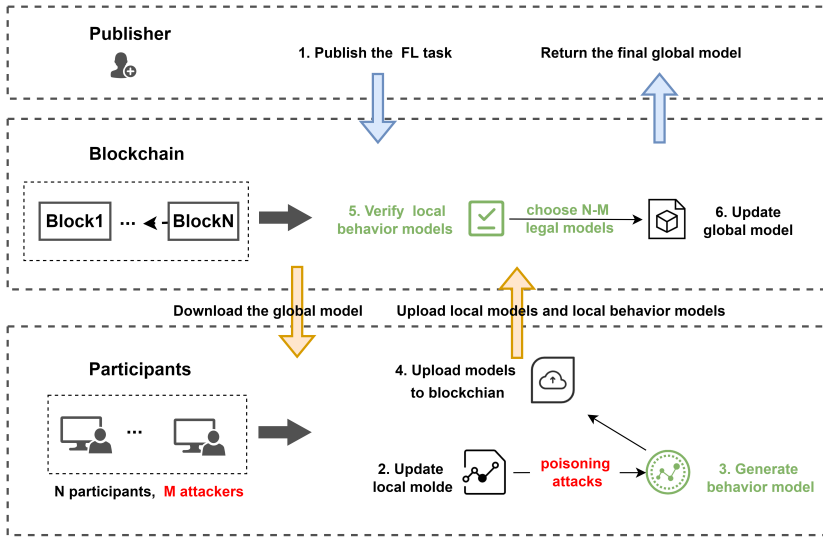



Figure 1. The work flow of BVFB

### 3.2 Training Behavior Verification Mechanism

In the above new framework, the training behavior verification mechanism must satisfy the following assumptions:

1. Participants have their own local datasets, and the data distribution of training samples under the same label in these datasets is consistent. Most participants will honestly follow the design specifications, and some participants may be maliciously attacked, resulting in erroneous calculation results.
2. After the participants are attacked, they can manipulate their local data sets to carry out poisoning attacks. The attack methods are divided into label-flipping attacks and backdoor attacks.

3. In each global iteration, the number of nodes participating in the aggregation is $N$, and the number of malicious actors who initiate poisoning attacks is $M$. We assume that there is at least one honest node in the aggregation in each iteration, and the honest node has the highest local model accuracy.

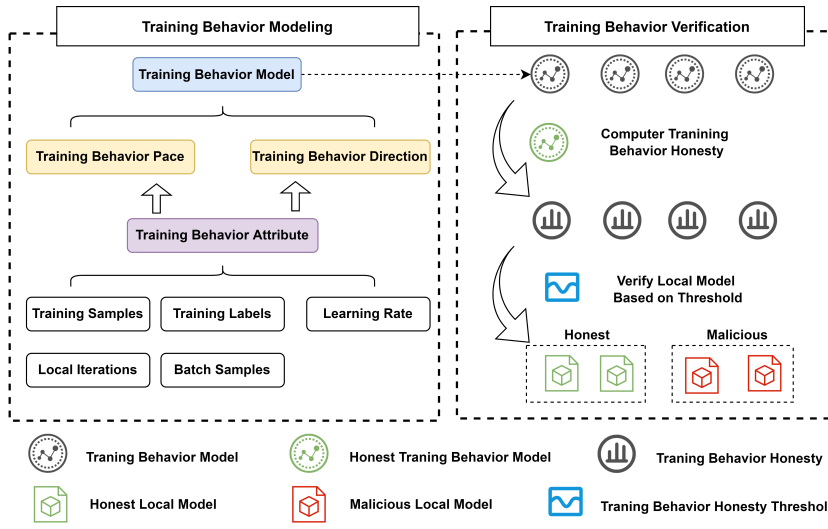



Figure 2. Training behavior verification mechanism

The training behavior verification mechanism includes two parts: training behavior modeling and training behavior verification, as shown in Figure 2. For the process of node training behavior modeling: first, the participating nodes generate the training behavior pace according to the local training behavior attribute and the gradient value updated by the local model, where the local training behavior attribute is composed of the node's training samples, local iteration times and learning rate. Second, the change direction of the training behavior pace is calculated, that is, the training behavior direction. Finally, a training behavior model is constructed according to the training behavior pace and training behavior direction, which is used to describe the consistent behavior rules of nodes during local training.

The training behavior verification is responsible for the identification of legal nodes and ensures the safe aggregation of the model. First, honest nodes are determined by the highest accuracy at each global iteration, based on assumptions. Secondly, the similarity of other nodes in the training behavior model is calculated through the honest node, that is, the training behavior honesty. The higher the honesty, the more honest the training behavior of the node is. Finally, after obtaining the training behavior honesty of these nodes, they are classified according to the corresponding thresholds, and the nodes larger than the corresponding thresholds are honest nodes, otherwise they are malicious nodes.

Through training behavior modeling and training behavior verification, honest nodes can be effectively selected to participate in model aggregation. Even when the number of malicious nodes exceeds a half, the malicious nodes can be effectively screened out before aggregation, thereby maximizing the use of local model updates of honest nodes and accelerating the convergence of the global model. The two core parts of the training behavior mechanism are described in detail below: training behavior modeling and training behavior verification.

## 4 TRAINING BEHAVIOR MODELING

In the $t^{\text{th}}$ global iteration of FL task, after receiving the global model $\Phi_G^{(t-1)}$ of the previous iteration, the participating node $D_i$ starts local iterative training based on the local dataset $S_i$. The initial local model $\Phi_i^{(t,0)}$ that the node $D_i$ trains locally iteratively is the global model $\Phi_G^{(t-1)}$. Usually, in order to improve the efficiency of the next global model aggregation, the node will iteratively train $E_i$ iterations locally. The purpose of local iterative training is to continuously reduce the local model loss value [5].

**Definition 1** (Local Loss Minimization)**.** Local loss minimization refers to minimum error value between the model predicted value and true values after the local model $\Phi_i^{(t,e)}$ iteratively trained on the local dataset $S_i$ in global iteration $t$ and local iteration $e$. It can be calculated as follows:

$$\min_{\Phi_i^{(t,e)}} \left\{ \frac{1}{|S_i|} \sum_{j=1}^{|S_i|} Loss\left(y, \hat{y}\right) \right\}, i = 0, 1, 2, \ldots, N \tag{1}$$

where $\Phi_i^{(t,e)}$ represents the local model of node $D_i$ in global iteration $t$ and local iteration $e$. $Loss\left(y, \hat{y}\right)$ represents the loss function used to calculate the error between the true value y and the predicted value $\hat{y}$, the cross-entropy loss function [28] is used here. $|S_i|$ represents the size of the training sample. $N$ represents the number of participating nodes.

Usually, in each local iteration, an optimizer is needed to calculate the optimal solution of the model parameters. Here, the stochastic gradient descent (SGD) optimization algorithm [29] is used for the local model update, which usually uses mini-batch data samples $B_i \in S_i$ to participate in the calculation.

**Definition 2** (Local Model Update)**.** Local model update refers to the process of updating the local model by performing SGD on the mini-batch data samples $B_i$ in the global iteration $t$ and the local iteration $e$. It can be calculated as follows:

$$\Phi_i^{(t,e)} = \Phi_i^{(t,e-1)} - \eta_i \delta \left( \Phi_i^{(t,e-1)}, B_i \right), i = 0, 1, 2, \ldots, N \tag{2}$$

where $\delta\left(\Phi_i^{(t,e-1)}, B_i\right)$ represents the gradient of the local model in global iteration $t$ and local iteration $e$. $B_i$ represents the mini-batch data samples used for SGD and $\delta\left(\Phi_i^{(t,e-1)}, B_i\right) = \frac{1}{|B_i|}\sum_{j=1}^{|B_i|}\frac{\partial Loss(y,\hat{y})}{\partial \Phi_i^{(t,e-1)}}$. $\eta_i$ represents the learning rate of model training for node $D_i$.

Since the local iterations $E_i$, the learning rate $\eta_i$, the data samples $X_i$, the data labels $Y_i$ and the mini-batch data samples $B_i$ involved in SGD are different for nodes to be used in local training. These different attributes can be regarded as the behavioral attributes of the nodes trained locally. The tuple consisting of these attributes is referred to as the training behavior attribute.

**Definition 3** (Training Behavior Attribute)**.** Training behavior attribute refers to a quintet composed of training data samples, training data labels, mini-batch data samples, learning rate and local iterations when the node $D_i$ is trained locally in the $t^{\text{th}}$ global iteration. It can be represented as follows:

$$BA_i^{(t)} = \left(X_i, Y_i, E_i, \eta_i, B_i\right), i = 0, 1, 2, \ldots, N \tag{3}$$

where $X_i$ represents the training data samples, $Y_i$ represents the training data labels, $E_i$ represents the local iterations, $\eta_i$ represents the learning rate, and $B_i$ represents the mini-batch data sample used for SGD.

The essence of local training is to continuously update the local model parameters to minimize the model loss, that is, the behavior of nodes in local training is reflected in the local gradient values. The change rule of the gradient value can reflect the training behavior rule of the node, so as to construct the training behavior portrait of the node. The local model gradient values between nodes have similarity when the data distribution and the initial local model are consistent, provided that the nodes are all honest. In order to prevent the weakening of this similarity due to the different number of local iterations $E_i$, the average calculation of the gradient value is used here. Since it reflects the common behavior of nodes in local training, it is called the training behavior pacing here.

**Definition 4** (Training Behavior Pace)**.** The training behavior pace refers to the average gradient value of node $D_i$ after local iteration $E_i$ in the $t^{\text{th}}$ global iteration. It can be calculated as follows:

$$BP_i^{(t)} = \frac{\sum_{e=1}^{E_i}\delta\left(\Phi_i^{(t,e-1)}, B_i\right)}{E_i}, i = 0, 1, 2, \ldots, N \tag{4}$$

where $\delta\left(\Phi_i^{(t,e-1)}, B_i\right)$ is the gradient of the local model $\Phi_i^{(t,e-1)}$ in the global iteration $t$ and the local iteration $e$.

From Equation (3), the training behavior pace $BP_i^{(t)}$ can be obtained by calculating the difference between the local model of node $D_i$ before and after the local

iteration divided by the learning rate $\eta_i$ and the number of local iterations $E_i$, that is, ${BP_i}^{(t)} = \frac{\left(\Phi_i^{(t,E_i)} - \Phi_i^{(t,0)}\right)}{\eta_i E_i}$.

Only through the training behavior pace cannot fully characterize the training behavior rules of nodes. In addition to the similarity of gradient values between honest nodes, there is also similarity in the direction of gradient change between honest nodes, which reflects the consistency of honest nodes in the direction of training behavior. Here, the *Sgn* function [32] is used to calculate the gradient change direction to reflect the direction of the training behavior, which is called the training behavior direction here.

**Definition 5** (*Sgn* Function)**.** *Sgn* function, also known as sign function, refers to returning an integer variable, indicating the sign of the parameter. It can be calculated as follows:

$$Sgn(x) = \begin{cases} -1, & x < 0, \\ 0, & x = 0, \\ 1, & x > 0. \end{cases} \tag{5}$$

**Definition 6** (Training Behavior Direction)**.** The training behavior direction refers to the change direction of the gradient of the node $D_i$ after the local iteration $E_i$ in the $t^{\text{th}}$ global iteration. It can be calculated as follows:

$${BD_i}^{(t)} = Sgn\left({BP_i}^{(t)}\right), i = 0, 1, 2, \ldots, N. \tag{6}$$

The commonalities in training behavior among honest nodes can be reflected by the training behavior pace and training behavior direction. These commonalities can describe the consistent behavior rules of nodes in local training, through which honest nodes and malicious nodes can be effectively distinguished. Here, the training behavior model is constructed to represent these commonalities.

**Definition 7** (Training Behavior Model)**.** The training behavior model refers to the two-tuple composed of the training behavior pace and the training behavior direction of the node $D_i$ in the $t^{\text{th}}$ global iteration. It can be represented as follows:

$$H_i^{(t)} = \left({BP_i}^{(t)}, {BD_i}^{(t)}\right), i = 0, 1, 2, \ldots, N. \tag{7}$$

In summary, the process of the training behavior modeling of the node $D_i$ can be obtained. First, the node $D_i$ is trained based on the global model $\Phi_G^{(t-1)}$ downloaded from the blockchain to obtain the final local model $\Phi_i^{(t)}$. Secondly, the corresponding training behavior pace ${BP_i}^{(t)}$ and training behavior direction ${BD_i}^{(t)}$ are obtained according to the gradient value calculated during the local iteration process. Finally, ${BP_i}^{(t)}$ and ${BD_i}^{(t)}$ together to build the training behavior model $H_i^{(t)}$. The specific modeling process is shown in Algorithm 1.

**Algorithm 1** Training Behavior Modeling

**Input:** $D_i, X_i, Y_i, \Phi_G^{(t-1)}, \eta_i, B_i, t$
**Output:** $\Phi_i^{(t)}, H_i^{(t)}$

// 1. Generate local model update
Download the global model $\Phi_G^{(t-1)}$ from blockchain
Initialize the local model $\Phi_i^{(t,0)} \leftarrow \Phi_G^{(t-1)}$
**for** $e \in E_i$ **do**
    Compute $\delta\left(\Phi_i^{(t,e-1)}, B_i\right) = \frac{1}{|B_i|}\sum_{j=1}^{|B_i|}\frac{\partial Loss(y,\hat{y})}{\partial \Phi_i^{(t,e-1)}}$
    Compute $\Phi_i^{(t,e)} = \Phi_i^{(t,e-1)} - \eta_i \delta\left(\Phi_i^{(t,e-1)}, B_i\right)$
**end for**
Get final local model $\Phi_i^{(t)} = \Phi_i^{(t,\ E_i)}$
// 2. Generate training behavior pace and training behavoir direction
Compute $\Delta\Phi_i^{(t)} = \Phi_i^{(t,E_i)} - \Phi_i^{(t,0)}$
Computer ${BP_i}^{(t)} = \frac{\Delta\Phi_i^{(t)}}{\eta_i E_i}$
Computer ${BD_i}^{(t)} = Sgn({BP_i}^{(t)})$
// 3. Generate training behavior model
Generate ${H_i}^{(t)} = \left({BP_i}^{(t)}, {BD_i}^{(t)}\right)$
**return** $\Phi_i^{(t)}, H_i^{(t)}$

## 5 TRAINING BEHAVIOR VERIFICATION

After collecting the local models and training behavior models of the participating nodes, the blockchain needs to verify the training behavior models first. Since the training behavior model reflects the commonality of nodes' local training behaviors, the training behavior models among honest nodes have similarity. By calculating the similarity between the train behavior models of nodes, honest nodes and malicious nodes can be divided into two groups with different similarities. In order to calculate the similarity between nodes and improve the calculation efficiency, we select the training behavior model $H_h$ of the honest node $D_h$ with the highest accuracy in each global iteration and use it as the object of comparison for calculating the similarity between other nodes. In calculating the similarity between the nodes' training behavior models, the Pearson correlation [30] is used here.

**Definition 8** (Pearson Correlation)**.** The Pearson correlation is a linear correlation coefficient used to reflect the degree of linear correlation between two random variables. It can be calculated as follows:

$$\rho(x,y) = \frac{\sum_{i=0}^{n}(X_i - \mu_X)(Y_i - \mu_Y)}{\sqrt{\sum_{i=0}^{n}(X_i - \mu_X)^2}\sqrt{\sum_{i=0}^{n}(Y_i - \mu_Y)^2}} \tag{8}$$

where $\mu$ is the mean and $\mu_X$ refers to the mean of $X$.

**Definition 9** (Training Behavior Pace Honesty)**.** Since there are multiple weight parameters $w_1, w_2, \ldots, w_b$ in the training behavior pace of the node training behavior model, for each parameter there are multidimensional data corresponding to it. The similarity between the node $D_i$ and the honest node $D_h$ in the parameters corresponding to the training behavior pace can be calculated, and then these similarities are accumulated and averaged. The result of the calculation is referred to here as the training behavior pace honesty, which is calculated as follows:

$$BPH_i^{(t)} = \frac{1}{b}\sum_{j=0}^{b} \rho\left(BP_{i,w_j}^{(t)}, BP_{h,w_j}^{(t)}\right), \quad 0 \le i, h \le N \tag{9}$$

where $BP_{h,w_j}^{(t)}$ is the node training behavior pace for weight parameter $w_j$ of honest node $D_h$ in the $t^{\text{th}}$ global iteration, $b$ is the number of weight parameters in the model.

**Definition 10** (Training Behavior Direction Honesty)**.** In the same way, the similarity between node $D_i$ and honest node $D_h$ in the parameters corresponding to the training behavior direction can be calculated, and then these similarities are accumulated and averaged to obtain the training behavior direction honesty, which is calculated as follows:

$$BDH_i^{(t)} = \frac{1}{b}\sum_{j=0}^{b} \rho\left(BD_{i,w_j}^{(t)}, BD_{h,w_j}^{(t)}\right), \quad 0 \le i, h \le N \tag{10}$$

where $BD_{h,w_j}^{(t)}$ is the node training behavior direction for weight parameter $w_j$ of honest node $D_h$.

**Definition 11** (Training Behavior Honesty)**.** The cumulative average of the above two honesty degrees can be used as the similarity of the training behavior model between the node $D_i$ and the honest node $D_h$, which is used as an evaluation metric at each iteration of verification to assess the similarity of behavior between the node $D_i$ and the honest node $D_h$, referred to here as the training behavior honesty. It can be calculated as follows:

$$BH_i^{(t)} = \frac{1}{2}\left(BPH_i^{(t)} + BDH_i^{(t)}\right), \quad 0 \le i \le N \tag{11}$$

where $BPH_i^{(t)}$ and $BDH_i^{(t)}$ represent the training behavior pace honesty and the training behavior direction honesty of the node $D_i$. The result range of $BH_i^{(t)}$ is $[0, 1]$, and the larger the value, the more similar the training behavior of honest node $D_h$, that is, the more honest.

**Algorithm 2** Training Behavior Verification

**Input:** $D = \bigcup_{i=1}^{N} D_i, \Phi^{(t)} = \bigcup_{i=1}^{N} \Phi_i^{(t)}, H^{(t)} = \bigcup_{i=1}^{N} H_i^{(t)}, t, b$
**Output:** $D_H = \bigcup_{i=1}^{M} D_i$

// 1. Get the honest node with the highest accuracy
**for** $\Phi_i^{(t)} \in \Phi^{(t)}$ **do**
    Compute the accuracy of $\Phi_i^{(t)}$ on the global test dataset: $Acc_i^{(t)}$
    Add $Acc_i^{(t)}$ into $Set_{acc}^{(t)}$ (the set of $Acc_i^{(t)}$)
**end for**
Get the honest node $D_h$ with the highest accuracy in $Set_{acc}^{(t)}$
**for** $H_i^{(t)} \in H^{(t)}$ **do**
    // 2. Computer training behavior honesty
    Computer $BPH_i^{(t)} = \frac{1}{b} \sum_{j=0}^{b} \rho \left( BP_{i,w_j}^{(t)},\ BP_{h,w_j}^{(t)} \right)$
    Computer $BDH_i^{(t)} = \frac{1}{b} \sum_{j=0}^{b} \rho \left( BD_{i,w_j}^{(t)},\ BD_{h,w_j}^{(t)} \right)$
    Computer $BH_i^{(t)} = \frac{1}{2} \left( BPH_i^{(t)} + BDH_i^{(t)} \right)$
    // 3. Discriminate node by training behavior honesty
    **if** $BH_i^{(t)} > \theta$ **then**
        Add $D_i$ into $D_H$
    **end if**
**end for**
**return** $D_H$

Before each iteration of local model aggregation, whether the node is maliciously attacked, it is determined by calculating the training behavior honesty. When the training behavior honesty is greater than a certain threshold $\theta$, it can be considered that the node and its local model is honest and can participate in the model aggregation. Otherwise, the node and its local models are malicious and should be removed before aggregation, thus effectively ensuring the security of model aggregation.

In summary, the training behavior verification process of the node can be obtained: Firstly, the accuracy of the local models of the participating nodes is calculated, and the node with the highest accuracy is selected as the honest node. Secondly, the honesty corresponding to the training behavior pace and training behavior direction in the training behavior model of each node is calculated, and the training behavior honesty is obtained after accumulating and averaging these honesties. Finally, determine whether the training behavior honesty of each node exceeds the threshold $\theta$, if it does, it is an honest node, otherwise it is a malicious node. The specific algorithm process is shown in Algorithm 2.

## 6 IMPLEMENTATION AND EVALUATION

To verify the effectiveness of the blockchain-based FL with the training behavior validation mechanism, this paper conducts experiments on the public dataset MNIST dataset [31].

### 6.1 Experimental Setup

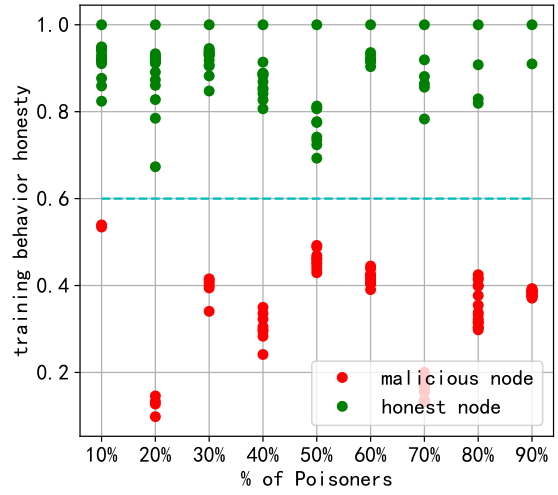
The MNIST dataset consists of 60 000 training data samples and 10 000 test dataset samples. Each sample is a grayscale handwritten digital image composed of $28 \times 28$ pixels, and the number range is $0 \sim 9$. The MNIST dataset is randomly assigned into 20 mutually disjoint subsets and randomly assigned to 20 participant nodes in the blockchain. To simulate the distributed scenario, we used four servers equipped with 8 GB RAM and 2.40 GHz Intel(R) Core (TM) i5-1135G7 processors for the experiment, and each server started 5 processes to simulate the participating nodes.

The training model that nodes train locally employs the multi-layer perceptron (MLP) [33] and is iteratively trained using the stochastic gradient descent (SGD) optimization algorithm [29]. MLP includes a input layer, two hidden layers and a output layer, and the corresponding number of neurons is 784, 256, 128 and 10. In the model, the Sigmoid function [34] is used as the activation function, and the cross-entropy loss function [28] is used as the loss function. The learning rate of the node $\eta = 0.1$, the number of local model weight parameters $b = 6$, the number of global iterations $T = 20$, and the number of local iterations $E_i = 5$. The aggregation method between local models adopts the Federated Average (FedAvg) [35] algorithm. Proof of Work (PoW) [36] is used in the blockchain as the consensus mechanism. The global model is stored in the block header, and the honest local model uploaded by the node is stored in the block body. A block in the blockchain represents a global iteration, and the training behavior verification mechanism is executed through the smart contract.

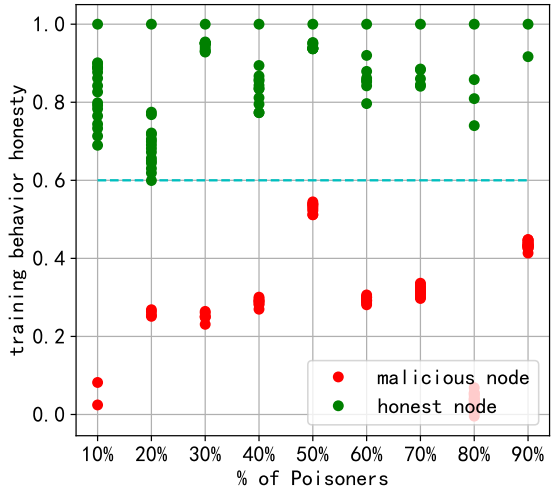
The poisoning attacks involved in the experiments include label-flipping [8] and backdoor attacks [9]. In order to verify the effectiveness of the training behavior verification mechanism for node verification, we set up two groups of experiments, respectively using label-flipping and poisoning attacks to attack BVFB with poisoning rate $0\,\% \sim 90\,\%$. In order to verify that the BVFB has more advantages over other frameworks, we set up four groups of experiments, namely *Median* [11], *Trimmed Mean* [11], *Krum* [10] and BVFB, and the poisoning rate of label-flipping and backdoor attack is $0\,\% \sim 90\,\%$.

### 6.2 Experimental Result

The experimental results of node training behavior honesty of nodes under label-flipping (left) and backdoor attack (right) in BVFB are shown in Figure 3. As can be seen from the figure, there is a big difference between the honesty of honest




a)




b)

Figure 3. Training behavior honesty under label-flipping a) and backdoor attack b) in BVFB

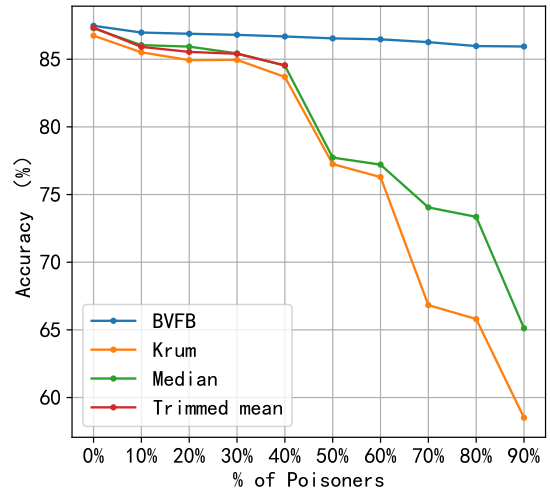
nodes and malicious nodes, the honesty between honest nodes is relatively close, the honesty between malicious nodes is also relatively close. There is an obvious dividing line threshold between these two kinds of nodes. Here, $\theta = 0.6$ is selected as the threshold for judging whether a node is an honest node. As can be seen from the figure, the threshold $\theta = 0.6$ can identify more than 90 % of honest nodes. In the subsequent comparative experiments, the threshold $\theta = 0.6$ is selected as the judgment basis for verifying honest nodes.

The accuracy of BVFB under label-flipping attack and backdoor attack with different poisoning rates is shown in Figure 4, and *Krum*, *Median*, and *Trimmed Mean* are compared. Among them, *Trimmed Mean* cannot be used when more than half of the nodes are attacked, so it cannot be counted when the poisoning rate exceeds half. We can see that as the poisoning rate of the poisoning attack increases, especially after the poisoning exceeds 50 %, the accuracy of other frameworks drops sharply. The principle of *Krum* is to select one of the most similar local models, but when the number of malicious nodes exceeds half, *Krum* will select one of the malicious nodes for aggregation with a high probability, thus affecting the effect of the global model. *Median* adopts the method of taking the median of the parameters, but when the number of malicious nodes exceeds half, the median will tend to the malicious parameter value. However, using the training behavior verification mechanism, when there is at least one honest node, it can accurately determine whether other nodes are honest, so as to eliminate malicious nodes, and only aggregate the local models of honest nodes to ensure the reliability of the global model.

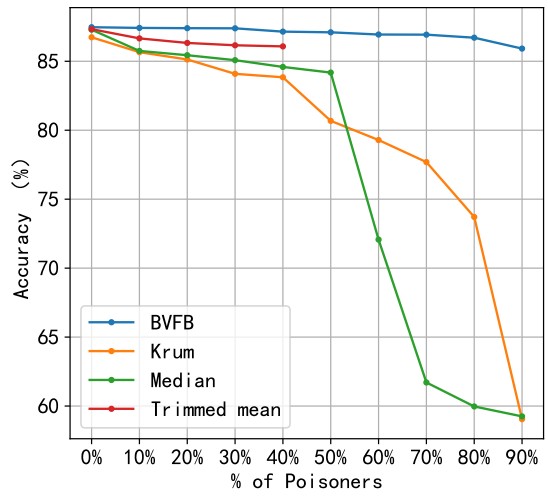
| Frame / Poisoning Rate | Krum | Trimmed Mean | Median | **BVFB** |
|---|---|---|---|---|
| 0 % | 86.74 % | 87.33 % | 87.29 % | **87.47** % |
| 10 % | 85.51 % | 85.91 % | 86.04 % | **86.97** % |
| 20 % | 84.93 % | 85.54 % | 85.93 % | **86.88** % |
| 30 % | 84.95 % | 85.40 % | 85.43 % | **86.80** % |
| 40 % | 83.69 % | 84.55 % | 84.52 % | **86.68** % |
| 50 % | 77.25 % | | 77.73 % | **86.54** % |
| 60 % | 76.29 % | | 77.21 % | **86.47** % |
| 70 % | 66.83 % | | 74.05 % | **86.26** % |
| 80 % | 65.80 % | | 73.35 % | **85.97** % |
| 90 % | 58.51 % | | 65.12 % | **85.94** % |

Table 1. Accuracy of different frameworks under label-flipping attack

The data in Tables 1 and 2 show the accuracy of BVFB as well as other frameworks under two poisoning attacks. It can be seen from the data in the table that after label-flipping and backdoor attacks occur in BVFB, the accuracy of the model has been maintained at around 86 %, and the average accuracy under 50 % and over 50 % poisoning rate has improved by about 2 % and 10 % compared to the other




a)




b)

Figure 4. Accuracy of different frameworks under label-flipping a) and backdoor attack b)

| Frame / Poisoning Rate | Krum | Trimmed Mean | Median | **BVFB** |
|---|---|---|---|---|
| 0 % | 86.74 % | 87.33 % | 87.29 % | **87.47** % |
| 10 % | 85.66 % | 86.66 % | 85.75 % | **87.42** % |
| 20 % | 85.13 % | 86.33 % | 85.44 % | **87.40** % |
| 30 % | 84.09 % | 86.16 % | 85.08 % | **87.39** % |
| 40 % | 83.84 % | 86.08 % | 84.59 % | **87.15** % |
| 50 % | 80.68 % | | 84.18 % | **87.10** % |
| 60 % | 79.29 % | | 72.07 % | **86.94** % |
| 70 % | 77.69 % | | 61.71 % | **86.93** % |
| 80 % | 73.71 % | | 59.97 % | **86.71** % |
| 90 % | 59.05 % | | 59.20 % | **85.92** % |

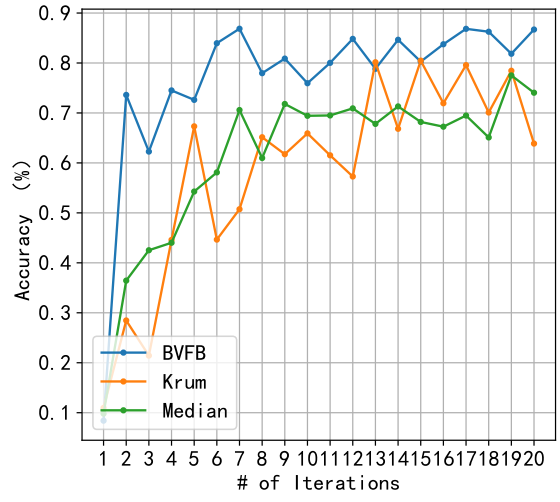
Table 2. Accuracy of different frameworks under backdoor attack

frameworks, which indicates that BVFB has higher accuracy than other frameworks. As the poisoning rate increases, the accuracy of the model in the BVFB decreases between 0 % and 1.5 %, and the effect of the model is not much affected by poisoning attack. The accuracy of the models of other frameworks decreases between 1 % and 10 %, and the change is relatively large. This indicates that the BVFB has a more stable effect than other frameworks.

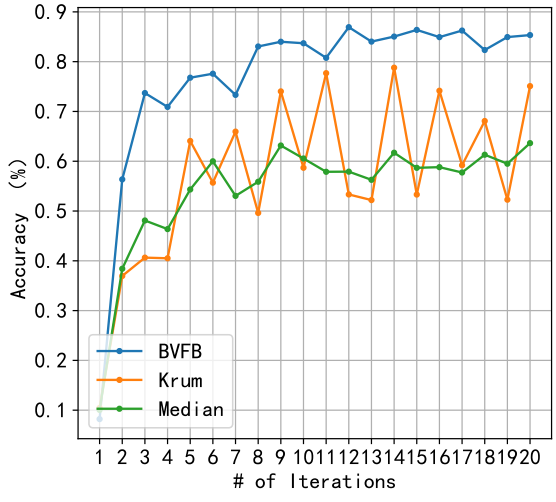
Since the change in accuracy with the number of iterations is roughly the same for different frameworks subjected to poisoning attacks when the poisoning rate is more than half, the poisoning rate of 70 % is chosen here for analysis. Figure 5 shows the trend of the accuracy of different frameworks with the number of iterations under the poisoning rate of 70 % label-flip attack and backdoor attack. It can be seen from the figure that when the number of malicious nodes exceeds half, the accuracy of the global model of the *Krum* and the *Median* after each iteration of aggregation is almost lower than that of BVFB, and the global models trained under the BVFB converge faster than the other frameworks. This is due to the fact that BVFB involves almost all honest nodes in model aggregation in each iteration and integrates the data features of all honest nodes, with almost no malicious nodes messing up, so the model convergence speed and model accuracy are better than other frameworks.

## 7 CONCLUSIONS

In this paper, we created a training behavior verification mechanism for secure blockchain-based federated learning (BVFB) that can effectively distinguish honest nodes from malicious nodes to ensure secure aggregation of the model, while maintaining a stable high aggregation accuracy even when more than half of the nodes are maliciously attacked. In BVFB, training behavior model can characterize the common behaviors between honest and malicious nodes. The training behavior ver-




a)




b)

Figure 5. Accuracy of different iterations under 70 % poisoning rate label flipping-attack a) and backdoor attack b)

ification can effectively distinguish honest nodes from malicious nodes even if only one honest node exists. The security mechanism proposed in this paper takes the highest accuracy rate as a judgment basis of the seed honesty node, but the model aggregation accuracy of honest nodes may not be the highest in practice. So, we will further consider the case where an honest node cannot be determined by the highest accuracy.

**Acknowledgment**

This work was supported by the Shanghai Science and Technology Innovation Action Plan Project (No. 22511100700) and the Natural Science Foundation of Shanghai (No. 19ZR1401900).

## REFERENCES

[1] Yang, Q.—Liu, Y.—Cheng, Y.—Kang, Y.—Chen, T.—Yu, H.: Federated Learning. Springer, Cham, Synthesis Lectures on Artificial Intelligence and Machine Learning, 2020, doi: 10.1007/978-3-031-01585-4.

[2] Bonawitz, K.—Eichner, H.—Grieskamp, W.—Huba, D.—Ingerman, A.—Ivanov, V.—Kiddon, C.—Konečný, J.—Mazzocchi, S.—McMahan, B.—Van Overveldt, T.—Petrou, D.—Ramage, D.—Roselander, J.: Towards Federated Learning at Scale: System Design. Proceedings of Machine Learning and Systems 1 (MLSys 2019), 2019, pp. 374–388, doi: 10.48550/arXiv.1902.01046.

[3] Yang, Q.—Liu, Y.—Chen, T.—Tong, Y.: Federated Machine Learning: Concept and Applications. ACM Transactions on Intelligent Systems and Technology (TIST), Vol. 10, 2019, No. 2, Art. No. 12, doi: 10.1145/3298981.

[4] Li, Q.—Diao, Y.—Chen, Q.—He, B.: Federated Learning on Non-IID Data Silos: An Experimental Study. 2021, doi: 10.48550/arXiv.2102.02079.

[5] Konečný, J.—McMahan, H. B.—Yu, F. X.—Richtárik, P.—Suresh, A. T.—Bacon, D.: Federated Learning: Strategies for Improving Communication Efficiency. 2016, doi: 10.48550/arXiv.1610.05492.

[6] Tolpegin, V.—Truex, S.—Gursoy, M. E.—Liu, L.: Data Poisoning Attacks Against Federated Learning Systems. In: Chen, L., Li, N., Liang, K., Schneider, S. (Eds.): Computer Security – ESORICS 2020. Springer, Cham, Lecture Notes in Computer Science, Vol. 12308, 2020, pp. 480–501, doi: 10.1007/978-3-030-58951-6_24.

[7] Nasr, M.—Shokri, R.—Houmansadr, A.: Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-Box Inference Attacks Against Centralized and Federated Learning. 2019 IEEE Symposium on Security and Privacy (SP), 2019, pp. 739–753, doi: 10.1109/SP.2019.00065.

[8] Biggio, B.—Nelson, B.—Laskov, P.: Poisoning Attacks Against Support Vector Machines. Proceedings of the 29th International Conference on Machine Learning (ICML 2012), 2012, pp. 1467–1474, doi: 10.48550/arXiv.1206.6389.

[9] Bagdasaryan, E.—Veit, A.—Hua, Y.—Estrin, D.—Shmatikov, V.: How to Backdoor Federated Learning. In: Chiappa, S., Calandra, R. (Eds.): Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics (AISTATS 2020). Proceedings of Machine Learning Research (PMLR), Vol. 108, 2020, pp. 2938–2948, doi: 10.48550/arXiv.1807.00459.

[10] Blanchard, P.—El Mhamdi, E. M.—Guerraoui, R.—Stainer, J.: Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In: Guyon, I., Von Luxburg, U., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.): Advances in Neural Information Processing Systems 30 (NIPS 2017). Curran Associates, Inc., 2017, pp. 119–129, doi: 10.48550/arXiv.1703.02757.

[11] Yin, D.—Chen, Y.—Kannan, R.—Bartlett, P.: Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. In: Dy, J., Krause, A. (Eds.): Proceedings of the 35th International Conference on Machine Learning (ICML 2018). Proceedings of Machine Learning Research (PMLR), Vol. 80, 2018, pp. 5650–5659, doi: 10.48550/arXiv.1803.01498.

[12] Bhagoji, A. N.—Chakraborty, S.—Mittal, P.—Calo, S.: Analyzing Federated Learning Through an Adversarial Lens. In: Chaudhuri, K., Salakhutdinov, R. (Eds.): Proceedings of the 36th International Conference on Machine Learning (ICML 2019). Proceedings of Machine Learning Research (PMLR), Vol. 97, 2019, pp. 634–643, doi: 10.48550/arXiv.1811.12470.

[13] Li, S.—Cheng, Y.—Liu, Y.—Wang, W.—Chen, T.: Abnormal Client Behavior Detection in Federated Learning. 2019, doi: 10.48550/arXiv.1910.09933.

[14] Fang, M.—Cao, X.—Jia, J.—Gong, N. Z.: Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. Proceedings of the 29th USENIX Security Symposium (SEC '20), 2020, Art. No. 92, pp. 1623–1640, doi: 10.48550/arXiv.1911.11815.

[15] Ma, C.—Li, J.—Ding, M.—Yang, H. H.—Shu, F.—Quek, T. Q. S.—Poor, H. V.: On Safeguarding Privacy and Security in the Framework of Federated Learning. IEEE Network, Vol. 34, 2020, No. 4, pp. 242–248, doi: 10.1109/MNET.001.1900506.

[16] Lyu, L.—Yu, H.—Yang, Q.: Threats to Federated Learning: A Survey. 2020, doi: 10.48550/arXiv.2003.02133.

[17] Kim, H.—Park, J.—Bennis, M.—Kim, S. L.: Blockchained On-Device Federated Learning. IEEE Communications Letters, Vol. 24, 2020, No. 6, pp. 1279–1283, doi: 10.1109/LCOMM.2019.2921755.

[18] Lu, Y.—Huang, X.—Dai, Y.—Maharjan, S.—Zhang, Y.: Blockchain and Federated Learning for Privacy-Preserved Data Sharing in Industrial IoT. IEEE Transactions on Industrial Informatics, Vol. 16, 2020, No. 6, pp. 4177–4186, doi: 10.1109/TII.2019.2942190.

[19] Bao, X.—Su, C.—Xiong, Y.—Huang, W.—Hu, Y.: FLChain: A Blockchain for Auditable Federated Learning with Trust and Incentive. 2019 5th International Conference on Big Data Computing and Communications (BIGCOM), 2019, pp. 151–159, doi: 10.1109/BIGCOM.2019.00030.

[20] Li, Y.—Chen, C.—Liu, N.—Huang, H.—Zheng, Z.—Yan, Q.: A Blockchain-

Based Decentralized Federated Learning Framework with Committee Consensus. IEEE Network, Vol. 35, 2021, No. 1, pp. 234–241, doi: 10.1109/MNET.011.2000263.

[21] Weng, J.—Weng, J.—Zhang, J.—Li, M.—Zhang, Y.—Luo, W.: DeepChain: Auditable and Privacy-Preserving Deep Learning with Blockchain-Based Incentive. IEEE Transactions on Dependable and Secure Computing, Vol. 18, 2021, No. 5, pp. 2438–2455, doi: 10.1109/TDSC.2019.2952332.

[22] Peng, Z.—Xu, J.—Chu, X.—Gao, S.—Yao, Y.—Gu, R.—Tang, Y.: VFChain: Enabling Verifiable and Auditable Federated Learning via Blockchain Systems. IEEE Transactions on Network Science and Engineering, Vol. 9, 2022, No. 1, pp. 173–186, doi: 10.1109/TNSE.2021.3050781.

[23] Kang, J.—Xiong, Z.—Niyato, D.—Xie, S.—Zhang, J.: Incentive Mechanism for Reliable Federated Learning: A Joint Optimization Approach to Combining Reputation and Contract Theory. IEEE Internet of Things Journal, Vol. 6, 2019, No. 6, pp. 10700–10714, doi: 10.1109/JIOT.2019.2940820.

[24] Chen, X.—Wang, T.—Zhang, S.: The Design of Reputation System for Blockchain-Based Federated Learning. 2021 International Conference on Artificial Intelligence and Blockchain Technology (AIBT), 2021, pp. 114–120, doi: 10.1109/AIBT53261.2021.00026.

[25] Shayan, M.—Fung, C.—Yoon, C. J. M.—Beschastnikh, I.: Biscotti: A Blockchain System for Private and Secure Federated Learning. IEEE Transactions on Parallel and Distributed Systems, Vol. 32, 2021, No. 7, pp. 1513–1525, doi: 10.1109/TPDS.2020.3044223.

[26] Preuveneers, D.—Rimmer, V.—Tsingenopoulos, I.—Spooren, J.—Joosen, W.—Ilie-Zudor, E.: Chained Anomaly Detection Models for Federated Learning: An Intrusion Detection Case Study. Applied Sciences, Vol. 8, 2018, No. 12, Art. No. 2663, doi: 10.3390/app8122663.

[27] Chen, H.—Asif, S. A.—Park, J.—Shen, C. C.—Bennis, M.: Robust Blockchained Federated Learning with Model Validation and Proof-of-Stake Inspired Consensus. AAAI 2021 Workshop – Towards Robust, Secure and Efficient Machine Learning, 2021, doi: 10.48550/arXiv.2101.03300.

[28] Tewari, A.—Bartlett, P. L.: On the Consistency of Multiclass Classification Methods. Journal of Machine Learning Research, Vol. 8, 2007, No. 36, pp. 1007–1025.

[29] Zhang, T.: Solving Large Scale Linear Prediction Problems Using Stochastic Gradient Descent Algorithms. Proceedings of the Twenty-First International Conference on Machine Learning (ICML '04), 2004, doi: 10.1145/1015330.1015332.

[30] Sedgwick, P.: Pearson's Correlation Coefficient. BMJ, Vol. 345, 2012, Art. No. e4483, doi: 10.1136/bmj.e4483.

[31] Li, L.—Fan, Y.—Tse, M.—Lin, K. Y.: A Review of Applications in Federated Learning. Computers and Industrial Engineering, Vol. 149, 2020, Art. No. 106854, doi: 10.1016/j.cie.2020.106854.

[32] Yun, B. I.: A Non-Iterative Method for Solving Non-Linear Equations. Applied Mathematics and Computation, Vol. 198, 2008, No. 2, pp. 691–699, doi: 10.1016/j.amc.2007.09.006.

[33] Gardner, M. W.—Dorling, S. R.: Artificial Neural Networks (The Multi-

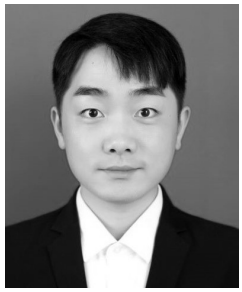
layer Perceptron) – A Review of Applications in the Atmospheric Sciences. Atmospheric Environment, Vol. 32, 1998, No. 14–15, pp. 2627–2636, doi: 10.1016/S1352-2310(97)00447-0.

[34] Han, J.—Moraga, C.: The Influence of the Sigmoid Function Parameters on the Speed of Backpropagation Learning. In: Mira, J., Sandoval, F. (Eds.): From Natural to Artificial Neural Computation (IWANN 1995). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 930, 1995, pp. 195–201, doi: 10.1007/3-540-59497-3_175.

[35] McMahan, B.—Moore, E.—Ramage, D.—Hampson, S.—Agüera y Arcas, B.: Communication-Efficient Learning of Deep Networks from Decentralized Data. In: Singh, A., Zhu, J. (Eds.): Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS 2017). Proceedings of Machine Learning Research (PMLR), Vol. 54, 2017, pp. 1273–1282, doi: 10.48550/arXiv.1602.05629.

[36] Gervais, A.—Karame, G. O.—Wüst, K.—Glykantzis, V.—Ritzdorf, H.—Capkun, S.: On the Security and Performance of Proof of Work Blockchains. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 3–16, doi: 10.1145/2976749.2978341.

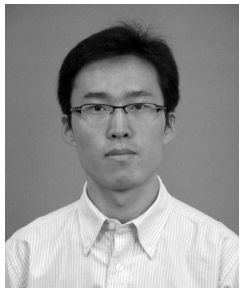
**Zhaohui Zhang** recived his B.Sc. degree in computer science from the Anhui Normal University, Wuhu, China in 1994. He obtained his Ph.D. degree in computer science from the Tongji University, Shanghai, China in 2007. From 1994 to 2015, he worked in Anhui Normal University and became Professor in 2009. Since 2015 he has been working as Professor in the School of Computer Science and Technology, Donghua University, Shanghai, China. His research interests include big data intelligent processing and behavior analysis.



**Jiawei Hu** received his B.Sc. degree in software engineering from the Yangtze University, Jingzhou, China, in 2020, and now he is pursuing his M.Sc. degree in computer science and technology from the Donghua University, Shanghai, China. His current research interests include blockchain, machine learning and federated learning.

**Lina Ma** received her B.Sc. degree in computer science and technology from the Yanshan University, Qinhuangdao, China, in 2021, and now she is pursuing her M.Sc. degree in computer science and technology from the Donghua University, Shanghai, China. Her current research interests include big data, machine learning and financial fraud.

**Ruoxuan Pei** received her B.Sc. degree in computer science and technology from the Anhui Normal University, Wuhu, China, in 2021, and now she is pursuing her M.Sc. degree in computer science and technology from the Donghua University, Shanghai, China. Her current research interests include big data, machine learning and financial fraud.



**Peiwei Wang** received his B.Sc. and M.Sc. degrees from the Shandong University of Science and Technology, Qingdao, China, in 2005 and 2008, respectively, and his Ph.D. degree from the Tongji University, Shanghai, China, in 2013, all in computer science. He finished his postdoctoral research work at the Department of Computer Science, University of Pisa, Italy, in 2015. He is currently serving as Associate Professor in the School of Computer Science and Technology, Donghua University, Shanghai. His research interests include cloud computing, data mining, and service computing.