Digital Communications and Networks 8 (2022) 614–624

Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.keaipublishing.com/dcan

A blockchain-based audit approach for encrypted data in federated learning

Zhe Sun [a], Junping Wan [a], Lihua Yin [a,*], Zhiqiang Cao [a], Tianjie Luo [a], Bin Wang [b]

[a] *Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou, 510006, China*
[b] *College of Electrical Engineering, Zhejiang University, Hangzhou, 310058, China*

ARTICLE INFO

Keywords:
Audit
Data quality
Blockchain
Secure aggregation
Federated learning

ABSTRACT

The development of data-driven artificial intelligence technology has given birth to a variety of big data applications. Data has become an essential factor to improve these applications. Federated learning, a privacy-preserving machine learning method, is proposed to leverage data from different data owners. It is typically used in conjunction with cryptographic methods, in which data owners train the global model by sharing encrypted model updates. However, data encryption makes it difficult to identify the quality of these model updates. Malicious data owners may launch attacks such as data poisoning and free-riding. To defend against such attacks, it is necessary to find an approach to audit encrypted model updates. In this paper, we propose a blockchain-based audit approach for encrypted gradients. It uses a behavior chain to record the encrypted gradients from data owners, and an audit chain to evaluate the gradients' quality. Specifically, we propose a privacy-preserving homomorphic noise mechanism in which the noise of each gradient sums to zero after aggregation, ensuring the availability of aggregated gradient. In addition, we design a joint audit algorithm that can locate malicious data owners without decrypting individual gradients. Through security analysis and experimental evaluation, we demonstrate that our approach can defend against malicious gradient attacks in federated learning.

1. Introduction

Benefiting from the development of cyber-physical systems, Internet-of-Things (IoT) devices have been widely used in daily life. These devices generate a large amount of data, which is conducive to the development of big data applications. Data application providers generally tend to obtain the most comprehensive data from devices to improve applications [1]. However, due to the high correlation between data and the physical world, most data owners are typically reluctant to share data for security and privacy reasons [2]. To solve such problems, researchers have proposed a series of multi-party data sharing schemes. In 2016, Google [3] proposed federated learning, in which multiple data owners jointly train a global model by exchanging encrypted gradients with the application provider. It prevents the user from raw data leak threats, but it cannot resist privacy attacks on gradients. Subsequent research shows that attacks such as model inversion attack [4], attribute inference attack [5], and membership inference attack [6] can infer privacy in gradients, putting privacy at risk of leakage.

To deal with this privacy leakage, researchers have introduced cryptographic methods in gradient protection, such as homomorphic encryption [7], obfuscation circuits [8], and secret sharing [9]. These cryptographic methods ensure that the application provider obtains the final aggregation gradient, but does not know any individual gradient from data owners. Cryptographic methods have successfully solved the problem of the privacy leakage, but also bring new problems. Due to the confidentiality of the encrypted gradient, malicious data owners can easily launch attacks such as data poisoning [10] and free-riding [11], thus hindering the fair operation of multi-party data sharing systems. The global model may be corrupted and the honest data owners may be discouraged from collaboration.

Researchers have devised several approaches to maintain the fairness of data sharing among participants. They can be roughly divided into two categories: game theory-based incentive methods [12] and audit-based penalty methods [13]. Game theory-based incentives aim to induce all participants to fairly maximize their own benefits based on a reward mechanism. Kang et al. [14] design an incentive mechanism based on contract theory to encourage good participants to contribute more actively. Zhan et al. [15] formulate a Stackelberg game model as an incentive mechanism in federated learning. They derive a Nash equilibrium that allows participants to adjust their strategies based on the

* Corresponding author.
E-mail address: yinlh@gzhu.edu.cn (L. Yin).

https://doi.org/10.1016/j.dcan.2022.05.006
Received 30 June 2021; Received in revised form 6 May 2022; Accepted 10 May 2022
Available online 13 May 2022

training cost and rewards. However, these approaches only encourage good participants to train, rather than prevent malicious participants from intentionally corrupting the collaboration, such as launching a data poisoning attack. Another type of approach, the audit-based penalties is to locate and remove harmful gradients from malicious participants and then penalize these participants. Current work mainly involves gradient feature analysis and gradient utility evaluation. Blanchard et al. [16] propose a Multi-Krum algorithm to calculate the feature distances between gradients, distinguishing the malicious gradients from others. However, this method is not available for an encrypted data audit. Lu et al. [17] propose a method to evaluate the utility of gradients. They introduce blockchain to schedule the requests and interactions of each participant. Consensus nodes of the blockchain generate training quality proofs for all gradients, but this scheme only detects malicious behavior in aggregation, but cannot precisely locate the harmful gradient. The above methods do not work well for an encrypted data audit.

In this paper, we propose a blockchain-based audit approach for encrypted gradients. Based on consortium blockchain, our audit approach can precisely locate malicious data owners in cryptography-based federated learning. Specifically, we use a blockchain called behavior chain to record the interactions of gradients between data owners and the blockchain. Behavior chain does not decrypt the received gradients, but it uses the homomorphism of the BCP (Bresson-Catalano-Pointcheval) [18] algorithm to add noise to each encrypted gradient. The noise mechanism is designed to make the aggregated gradient noiseless. All encrypted gradients are submitted to another blockchain called audit chain, which performs an aggregation of different encrypted gradients and analyses the aggregated gradient using our joint audit approach. In this way, the audit chain can judge whether malicious data owners have intentionally provided low-quality gradient. Put it briefly, the main contributions of this paper can be summarized as follows:

- We propose a blockchain-based audit approach for encrypted data, using the behavior chain and audit chain to record data owners' gradients and their contributions to collaborative training, respectively. Our approach can audit whether there is a malicious gradient without decrypting any individually encrypted gradient.
- We design a BCP-based noise mechanism for encrypted gradients, which ensures the noise in each group of gradients sums to zero after aggregation. Encrypted gradients from any data owner are not decrypted during training, ensuring that unauthorized parties cannot access these gradients.
- We design a joint audit algorithm that maps multiple gradients into a square matrix. It enables our audit approach to locate malicious gradients by evaluating the aggregated gradient in rows and columns of the matrix.

The rest of the paper is organized as follows. In Section 2, we present related works about blockchain and data audit. We briefly describe the assumptions and threat models in Section 3. Section 4 presents our approach in detail. Section 5 provides a security analysis of our audit approach. We describe the performance evaluation in Section 6. In Section 7, we conclude our audit approach.

2. Related works

2.1. Blockchain for gradient collecting in federated learning

In federated learning, data owners' contributions are generally different because of their varying training data. How to reliably record their contributions to collaborative training is widely studied among researchers. Blockchain [19] is an excellent technology for recording contributions because it is tamper-resistant and traceable [20]. Current blockchain-based methods in federated learning can be briefly classified as gradient recording [21] and gradient consistency verification [13].

Blockchain can impartially record the behavior of participants during training, such as uploading gradients. Lu et al. [17] design a blockchain-based federated learning architecture in which a committee is responsible for assigning task requests and collecting gradients from participants. The committee manages the blockchain and uses it to record all the gradients. However, it does not consider whether the participants are honest for a long time in federated learning. In subsequent work, some researchers take participants' reputations into account. Zhao et al. [19] propose a framework that miners in the blockchain use the Multi-Krum algorithm to evaluate whether a gradient is malicious or not. They set up a reputation mechanism to continuously reward the non-malicious parties. Kang et al. [14] propose a blockchain-based recording method based on contract theory to manage the participants' reputations. The participants with a higher reputation are prone to be selected to participate in training. The above approaches collect the participants' gradients through the blockchain in federated learning. However, if participants encrypt gradients for privacy reasons, gradients will be invisible and tampering threats will emerge.

To ensure the correctness of encrypted gradients, researchers use blockchain-based methods to detect errors in the aggregation of gradients. Xu et al. [22] design an algorithm based on bilinear pairing to help participants verify the correctness of the aggregated gradient from the parameter server. However, the server may obtain privacy in gradients during the aggregation process by deceptively claiming that some participants are offline in collaboration. To supervise the aggregation process, Jiang et al. [23] introduce the blockchain to record membership proofs, thus checking whether the server correctly aggregates gradients from all participants. As for regulating the participants, Weng et al. [13] design an audit algorithm based on the Σ protocol and introduce a blockchain to record audit information accessible to third parties. Though the approach can ensure the legality of encrypted gradients, it still cannot audit the quality of gradients without decrypting individual gradients.

The existing blockchain-based methods in federated learning cannot be directly applied to the quality audit of encrypted gradients. They mainly focus on using blockchain for data recording. To better defend the data poisoning in gradients, we propose an audit method for encrypted gradients using two blockchains, enabling both privacy-preserving gradient recording and trusted quality auditing. Due to the different sizes of the behavior chains and audit chains, our framework is easy to manage and avoid waste that occurs during block generation and consensus.

2.2. Quality audit in federated learning

In federated learning, incentives such as reputation mechanisms can prevent the model from long-term poisoning attacks, but not immediately defend against attacks. It is difficult to accurately evaluate individual participants' data in federated learning for privacy reasons. To audit participants' contributions, researchers propose quality evaluation approaches on participants' data, which can be divided into two groups: training data detection and gradient evaluation.

Training data detection mainly uses statistical methods to determine whether malicious training data exists in the dataset. Shen et al. [24] use *k*-means clustering algorithm on the features of training data to discriminate between malicious and honest parties. Rubinstein et al. [25] propose an improved PCA-GRID algorithm that uses the Laplace threshold as the cutoff threshold to detect anomalous data. Such methods require that participants provide features of training data. It does not work if the parameter server only obtains gradients in federated learning.

Another approach is to identify malicious attackers by evaluating gradients. The gradient has plaintext [26] and ciphertext forms [27]. The plaintext gradients are mainly evaluated by specific clustering algorithms. Blanchard et al. [16] propose a Multi-Krum algorithm that calculates the Euclidean distances between gradients, and the gradient with abnormal distance from others is identified as malicious. However, Shayan et al. [28] point out that the Multi-Krum algorithm has

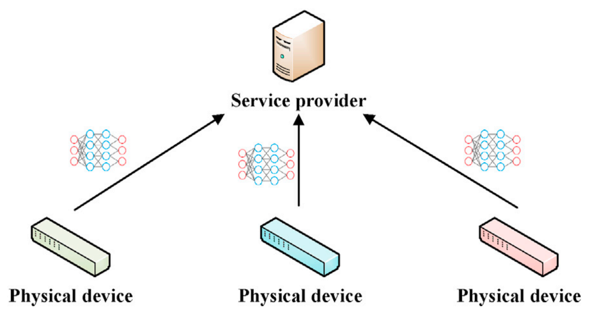

Fig. 1. System model of federated learning.

implementation limitations because it requires many gradient samples, which is not always achievable in distributed systems. Based on this, Fung et al. [29] consider that each participant's data is unique in distribution, leading to differences in gradient features. Therefore, they propose a method to locate the malicious attackers by calculating gradient similarity. The above method works for plaintext gradients. As similarity detection is not applicable in ciphertexts, these methods do not work for encrypted gradients. To address this problem, Lu et al. [17] propose a method to defend against malicious attacks by assessing the overall utility of encrypted gradients. Participants are requested to submit the local loss of their gradient to the committee. The committee checks the error between the aggregated gradient loss and each local loss and then determines whether malicious gradients exist. A disadvantage of this scheme is that once the malicious gradient does appear, it cannot efficiently and accurately locate the gradient. Qu et al. [30] propose an accuracy verification mechanism that uses a homomorphic cryptography-based neural network and a garbled circuit for label prediction. However, this mechanism is not practical in some federated learning scenarios, as the cost of label prediction can increase rapidly as the model becomes complex.

Most existing work of quality audit cannot effectively and accurately locate encrypted malicious gradients. We aim to efficiently identify malicious attackers without decrypting any individual gradient through our joint audit algorithm.

3. Assumptions and threat models

Our audit approach can be applied in cryptography-based multi-party data sharing schemes. We take the homomorphic-based federated learning as an example. We assume there is a service provider that trains a service model. It does not have enough training data and cannot access data on other physical devices for privacy issues. The service provider launches a federated learning process with multiple physical devices to leverage enough data for training, as shown in Fig. 1. We describe the design assumptions of our approach and the threat models in this section.

3.1. Design assumptions

Federated learning: two roles work together to train the global model, which can be described as follows:

Service Provider (*SP*): The entity that assigns the model to each physical device. It aggregates encrypted gradients from physical devices and uses the aggregated gradient to update the global model. The training process continues until the model has good enough performance.

Physical Devices (*PDs*): IoT devices use local data to train the model. They submit the encrypted gradient to the service provider.

We assume that *SP* needs to train a machine learning model *M*. The data required for training is collected and stored by multiple *PDs*. We assume that these *PDs* are reluctant to share their private data directly with *SP* because of privacy concerns. To use these training data, *SP* assigns the model to *PDs* and requests *PDs* to train *M*. Each *PD* uses private data to calculate the model gradient, encrypt and submit it to *SP*. Then, *SP* aggregates and decrypts the gradients for the model update. Generally, *SP* wants to obtain high-quality data but pays less attention to the privacy of participants, while *PDs* are the opposite in federated learning. That is, the interests of *SP* and *PDs* are inconsistent. Based on this, we assume that they do not trust each other. To defend against data poisoning attacks, *SP* needs to audit the quality of gradients. Specifically, *SP* can calculate the loss value of the updated model on the test dataset. To conduct federated learning, we assume each *PDs* has enough computing power to support local training and model updating, and they are also capable of running various cryptographic algorithms for authentication, etc.

Hyperledger: The blockchains involved in this paper are based on Hyperledger [31], which is a consortium blockchain. Unlike the PoW and PoS [32] policies in the public blockchain, the endorsing policy in Hyperledger enables all nodes to reach a consensus faster. The number of endorsing peers in Hyperledger is set to be small to improve the efficiency of consensus. We assume that Hyperledger and internal nodes are set up in advance and that each endorsing peer supports the deployment of the smart contract. The smart contract in our scheme is a piece of code that is honestly executed by endorsing peers. The code and data it manipulates are subject to external supervision. Once certain conditions are triggered, the code is automatically executed to produce running results. We use smart contracts to perform processes such as parameter initialization, gradient collection, and quality audit in federated learning.

When the majority of the endorsing peers reach a consensus on the output of the smart contract, the output will be written in a new block. We refer to this process as the endorsement process for short. The generation of new blocks indicates that the blockchain has collected or audited a batch of gradients. We omit details of how participants join Hyperledger, such as authentication, etc.

Homomorphic encryption: We assume that the service provider is honest but curious. It honestly collects and aggregates the gradients but may try to obtain private information from the gradients. It is not safe to allow such an aggregator to obtain any individual gradient directly.

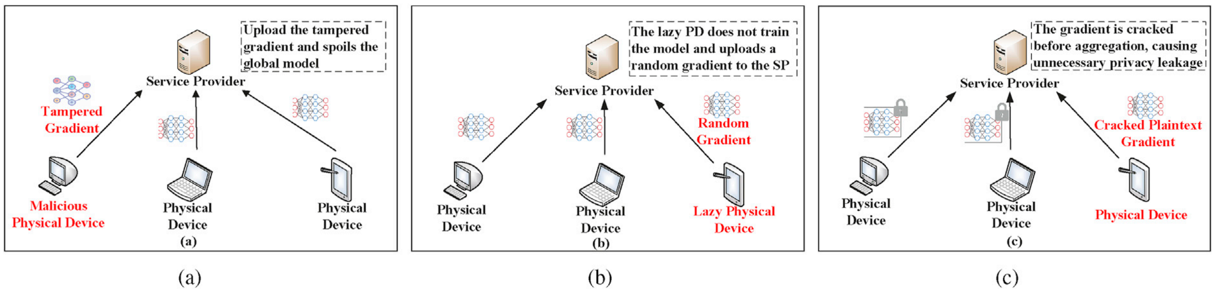
Upload the tampered gradient and spoils the global model
Service Provider
Tampered Gradient
Malicious Physical Device
Physical Device
(a)
Physical Device
(a)
The lazy PD does not train the model and uploads a random gradient to the SP
Service Provider
Random Gradient
Physical Device
Physical Device
(b)
Lazy Physical Device
(b)
The gradient is cracked before aggregation, causing unnecessary privacy leakage
Service Provider
Cracked Plaintext Gradient
Physical Device
Physical Device
(c)
Physical Device
(c)

Fig. 2. Threat models. (a) Data poisoning; (b) Free-riding attacks; (c) Privacy leakage during quality auditing.

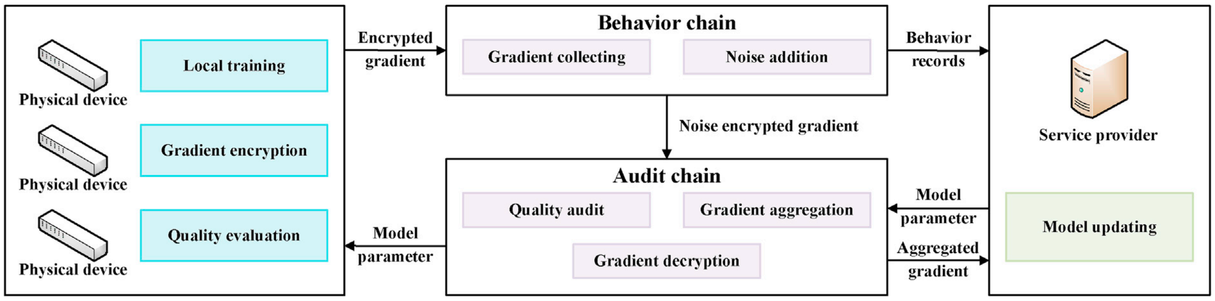

Fig. 3. System architecture of our quality audit approach.

Accordingly, we choose homomorphic encryption to encrypt the gradients [33]. We denote m_i as the gradient from physical devices P_i, and C_i as a cipher encrypted from m_i. They satisfied the relation $\prod_i C_i = Enc(\sum_i m_i)$ due to homomorphism, which enables the aggregator to obtain aggregated gradient without knowing any individual gradient m_i. During the initialization process, we assume that the participants and endorsing peers only publish public parameters, but do not disclose other secret parameters.

3.2. Threat models

When we use cryptographic methods to protect the privacy of gradients, the service provider only has access to the encrypted gradients. Because the encrypted gradients cannot be evaluated directly, the quality of gradients from each physical device is not easy to identify. Physical devices may perform malicious attacks on gradients. As shown in Fig. 2, we discussed the threats to the federated learning system and our security goals.

Threat 1: Data poisoning. Although cryptographic methods can effectively prevent devices' gradients from being obtained by third parties, it also poses other challenges. Once gradients are encrypted, malicious devices can tamper with the gradients undetectably. Malicious devices can intentionally use mislabeled data to train models and provide poisoning gradients to the service provider, thus corrupting the global model.

Security Goal 1: Preventing the global model from being compromised. We assume that at least 2/3 of the participants are honest and do not maliciously tamper with the gradient. We do not limit the complicity of malicious attackers here. To prevent the training model from being corrupted, all gradients are aggregated separately with other different gradients and tested for data quality. In each gradient aggregation, malicious gradients generally leave the global model optimization stagnant or even worse. By evaluating different aggregated gradients multiple times, the aggregator can locate one or even multiple malicious gradients.

Threat 2. Free-riding attacks. Another challenge in cryptography-based federated learning is fairness. A “lazy” physical device may provide a historical gradient or useless gradient to the service provider during the training process. In this way, they can profit from other honest participants without paying any cost, such as the consumption of data collection and local training, etc. This type of attack frustrates honest participants and makes devices participate in the training positively.

Security Goal 2: Guaranteeing fairness of federated learning. To defend against such free-riding attacks, we request physical devices to evaluate the gradients locally and provide the evaluation results along with the encrypted gradients. The aggregator will use the local evaluation results of all devices to predict the performance of the updated model. If the actual quality of the gradient does not match the quality claimed by the physical device, the performance of the aggregated model will be anomalous, which can be detected.

Threat 3: Privacy leakage during quality auditing. It is generally not easy to precisely locate malicious attackers when auditing the aggregated gradient. When malicious gradients are detected, the general solution is to delete all gradients in the batch [14] or require participants to provide unencrypted data for quality evaluation. Abandoning the honest participants' gradients will result in waste for honest participants, and requiring the unencrypted gradient of honest participants to prove their innocence will put their data privacy at risk of unnecessary leakage.

Security Goal 3: Privacy protection in audit. We need to locate malicious participants and provide evidence without decrypting other participants' data. To achieve this, our audit approach maps the encrypted gradient into a square matrix. We aggregate encrypted gradients into groups divided by rows and columns of the matrix. The common gradient that appears in multiple abnormal aggregations is a suspect malicious gradient: If performance anomalies are detected in multiple aggregations, there may exist a data poisoning attack; if the quality of the aggregated gradient differs significantly from the participants' statement, there may exist a free-riding attack. In either case, our audit approach can locate the malicious participants without decrypting individual gradients. The remaining honest participants can be reorganized into groups to continue training. As a result, we neither need to give up any contributions from honest participants nor decrypt their gradients.

4. System design

In this section, we present a blockchain-based quality audit approach for encrypted data in federated learning. We use a smart contract to manage the behavior of each participant and a joint audit approach to achieve a privacy-preserving quality audit on gradients.

4.1. System overview

We implement blockchain to conduct federated learning and audit the quality of encrypted gradients. The blockchain mainly achieves two tasks: collecting devices' gradients and evaluating aggregated gradients. As shown in Fig. 3, we separately deploy the two tasks on two blockchains of different sizes, called behavior chain and audit chain. The two blockchains are built on Hyperledger. Other details of them are listed as follows:

Behavior chain: It processes transactions of encrypted gradients from physical devices and records the transaction information. Besides, it adds noise to each encrypted gradient and then submits it to the audit chain.

Audit chain: It responds to training requests from service providers and distributes models to be trained to physical devices. During the training process, it obtains encrypted gradients from the behavior chain, and then aggregates the gradients to audit the quality. After the audit process is completed, quality audit information for each gradient is written to a new block. The globally aggregated gradient is sent to the service provider for the model update.

Our quality audit approach is applied in each iteration of training composed of *N* steps. In the following description, we use *PD* and *SP* to denote Physical Device and Service Provider, respectively. We use *AC* and *BC* to denote Audit Chain and Behavior Chain, respectively. We

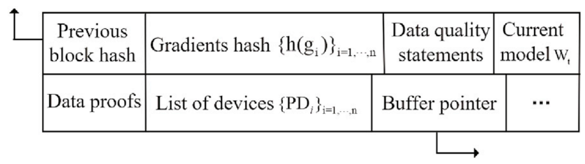

Fig. 4. Structure of blocks in behavior chain.

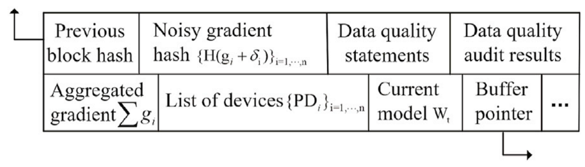
Previous block hash
Noisy gradient hash $\{H(g_i+\delta_i)\}_{i=1,\cdots,n}$
Data quality statements
Data quality audit results
Aggregated gradient $\sum g_i$
List of devices $\{PD_i\}_{i=1,\cdots,n}$
Current model W_t
Buffer pointer
...

Fig. 5. Structure of blocks in audit chain.

simplify *AC* and *BC* as two entities, omitting the detailed description of the transaction process in the blockchain. Each iteration is comprised of five steps.

Step 1 (*PD* gets the model from *AC* and trains it): *SP* provides the model parameter and a public test dataset with *AC* at the beginning of the federated learning. *AC* distributes the model to all registered *PDs* and provides the dataset with *BC*. Then, each *PD* trains the model to obtain the gradient using local data.

Step 2 (*PD* submits the gradient and local evaluation results): Each *PD* encrypts the gradient using the BCP algorithm and submits a data proof of the encrypted gradient to *BC*. Subsequently, each *PD* updates the local model and calls smart contract to calculate the loss of model on the test dataset. Then, it submits the encrypted gradient to *BC* along with the loss.

Step 3 (*BC* collects gradients and adds noise): *BC* collects messages containing encrypted gradients and losses from *PD* and then generates a new block to store all gradients and the corresponding transaction information. After that, *BC* uses the BCP algorithm to add noise to each encrypted gradient and send them to *AC*.

Step 4 (*AC* obtains noisy gradient and audits its quality): *AC* obtains noisy gradients and losses from *BC*. Then, it aggregates different encrypted gradients to obtain noise-free aggregation results and performs quality audits on them. Specifically, *AC* evaluates each aggregated gradient and compares the result to the loss stated by *PDs*. If inconsistencies occur, there may be malicious gradients in the current batch of gradients.

Step 5 (*SP* gets the globally aggregated gradient and updates the model): When the quality audit is completed, *AC* records the audit information in a new block. Then, *AC* aggregates all the gradients and sends the aggregation to *SP*. *SP* uses the gradient to update the global model and judge whether it converges. If it does not converge, the updated model and another test dataset are sent to *AC* again for the next iteration of training.

4.2. The workflow of blockchain

The behavior chain and audit chain are designed as ledgers to deliver and store training information in federated learning. In behavior chain, each block stores encrypted gradients and other information. The audit chain is used to audit the quality of gradients and record the results in new blocks.

The service provider first launches a training request to the audit chain and provides the model parameter. After each physical device has registered with the audit chain and behavior chain, the audit chain will request each device to participate in federated learning. The initialization of a federated learning system includes: a) Audit chain initializes the BCP algorithm and publishes public parameters. b) Each physical device generates its private key and public key of the BCP algorithm and publishes the public key. c) Behavior chain initializes the noise mechanism based on the BCP algorithm.

First, physical devices respond to the audit chain and obtain the model parameter. Then each device uses its private data to train the model. When devices have finished training, they launch a transaction to behavior chain to provide gradient. Behavior chain consists of several independent endorsing peers and orderers. Endorsing peers will respond to the devices and collect the encrypted gradients from them, which is called an endorsement process. The endorsing peers will sign the result of transactions and send the signed results back to the devices.

When a device receives at least $t(k \geq 3)$ signatures from endorsing peers, it sends the signed transaction result to an orderer in the behavior chain. The orderer will sort the transactions and generate a new block to record them. The new block will be synchronized in each endorsing peer's ledger if the signatures are verified as legal. The structure of blocks in the behavior chain is shown in Fig. 4.

The block stores the hash of encrypted gradients and a pointer to a private buffer that stores encrypted gradients. The block also contains a list of devices and data quality statements, etc. After a new block is generated, the endorsing peers that previously signed the transactions send the gradients to the audit chain. They respectively add noises in gradients using the BCP algorithm and send the noisy gradients to the endorsing peers in the audit chain.

The audit chain also has several independent endorsing peers and orderers. All endorsing peers that receive the gradients will aggregate the gradients to remove the noise. Specifically, each endorsing peer aggregates noisy gradients they received. If all aggregated gradients in each endorsing peer are the same, the aggregated gradient is noiseless. The audit chain will calculate the average value of each noisy gradient and stores them. After the audit chain uses our audit approach to check whether malicious gradients exist, the globally aggregated gradient will be decrypted and sent to the service provider.

The audit results will be written to a new block along with the noisy gradients, as shown in Fig. 5. If any malicious gradient is detected, the audit chain will inform the behavior chain and relevant participants for review and punishment. The whole process is also shown in Fig. 6.

4.3. BCP algorithm for gradient collection

When a physical device provides a gradient to behavior chain, it uses the addictive homomorphic BCP algorithm to encrypt the gradient. BCP algorithm [18] is a variant of the Paillier algorithm [34] with a double trapdoor decryption mechanism. We have revisited the algorithm to make it suitable for our framework. The algorithm in our audit approach contains five parts as follows:

- ***Setup***(1^k): given a security parameter k, select large prime number $p = 2p' + 1$, $q = 2q' + 1$, where p' and q' are also prime numbers and $gcd(pq, (p-1)(q-1)) = 1$. Set $N = pq$ as a modulus, and then compute $\lambda(N) = lcm(p-1, q-1) = 2p'q'$, where $lcm(p-1, q-1)$ means the least common multiple of $p-1$ and $q-1$. Set a group $G = \{y \in Z^*_{N^2} | \exists x \in Z^*_{N^2}\ s.t\ y = x^2 \bmod N^2\}$, where $ord(G) = pp'qq'$. Select an element $g \in G$ of order N such that $g^{p'q'} \equiv 1 + kN \bmod N^2$, $k \in \{1, 2, \ldots, N-1\}$. Output the public parameter $pp = (N, k, g)$ and master key $mk = (p', q')$.
- ***KeyGen***(pp): given public parameter $pp = (N, k, g)$, select a random element $a \in [1, ord(G)]$ as secret key sk and compute public key $pk = h = g^a \bmod N^2$. Output a pair of keys (pk, sk).
- ***Encrypt***(pk): given a message $m \in Z_N$, choose a random element $r \in Z_{N^2}$ to compute components $A = g^r \bmod N^2$, $B = h^r(1 + mN) \bmod N^2$. Output (A, B) as an encrypted message $enc(m)$.
- ***Decrypt***(sk): given encrypted message $enc(m) = (A, B)$, use $sk = a$ to compute message

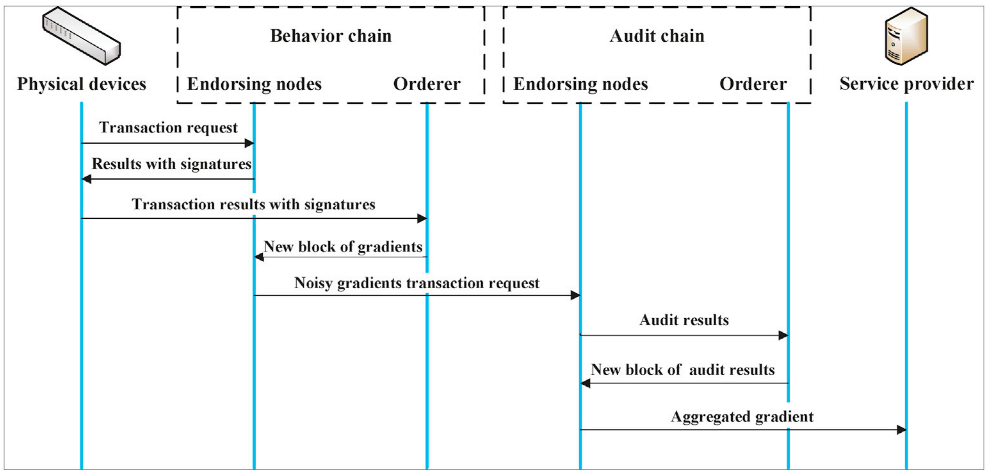

Fig. 6. The interactive flow of gradient auditing.

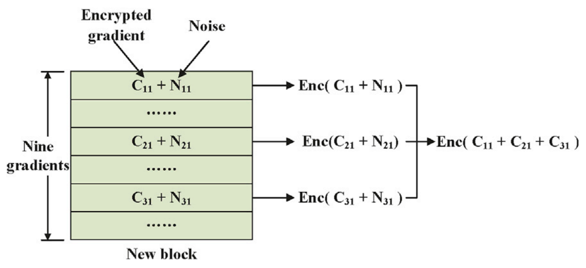

Fig. 7. Noise mechanism.

$$m=\frac{\frac{B}{A^a}-1 \bmod N^2}{N} \tag{1}$$

- ***Decrypt***$(\boldsymbol{mk})$: given encrypted message $enc(m) = (A, B)$, use $mk = (p', q')$ to compute the secret key

$$a=\frac{h^{p'q'}-1 \bmod N^2}{N}k^{-1} \bmod N \tag{2}$$

Then compute $\pi = p'q' \bmod N^2$. Finally, compute the message

$$m=\frac{\left(\frac{B}{A^a}\right)^{p'q'}-1 \bmod N^2}{N}\pi^{-1} \bmod N \tag{3}$$

At the beginning of federated learning, the audit chain runs ***Setup***$(\boldsymbol{1^k})$ to initialize the BCP algorithm and then shares public parameter $pp = (N, k, g)$ to physical devices and behavior chain. Then, each physical device P_i runs ***KeyGen***$(\boldsymbol{pp})$ to generate a pair of keys (pk_i, sk_i), where pk_i is published to the audit chain and the behavior chain.

When the physical device P_i launches a transaction to provide a gradient, it encodes the gradient to get message m, and runs ***Encrypt***$(\boldsymbol{pk_i})$ to get the encrypted message (A_i, B_i). Then, it submits encrypted message (A_i, B_i) to behavior chain. Due to the behavior chain does not have sk_i or mk_i to decrypt the encrypted message (A_i, B_i), the gradient will not be disclosed to others during the transaction.

The homomorphic BCP algorithm enables the federated learning system to achieve two goals: a) Secure aggregation. The aggregator obtains the aggregated results without knowing any individual gradients. b) Noise elimination. Noises in gradients can protect them from being disclosed. The BCP algorithm makes the noises harmless because they will be eliminated in the final aggregation. It does not spoil the correctness of aggregated gradients. We can evaluate the noiseless aggregated gradient to detect malicious gradients.

4.4. Noise mechanism

The audit chain is designed to possess the master key mk to decrypt any gradient. To prevent any individual gradient from being exposed to the audit chain and other parties, the behavior chain is supposed to add noise to each encrypted gradient before submitting it. To make our joint audit approach work well, the behavior chain groups and maps the gradients to an n-order square matrix. In this matrix, the behavior chain adds noise to each element and ensures that the noises in each row and column are summed to zero. Hence, the audit chain can get the correct noiseless aggregated gradient in each matrix.

As shown in Fig. 7, we design a noise mechanism based on the BCP algorithm to achieve this goal. We assume that a block contains 9 gradients that are mapped to a 3-order square matrix C. The encrypted gradients are noted as C_{ij}, and the corresponding gradients are noted as m_{ij}. We mark the row number as $i \in \{1, 2, 3\}$ and the column number as $j \in \{1, 2, 3\}$. The matrix can be represented as $C = (C_{ij})_{3\times 3}$ where $C_{ij} := (A_{ij}, B_{ij}) = (g^{r_{ij}} \bmod n^2, h_{ij}{}^{r_{ij}}(1+n)^{m_{ij}} \bmod n^2)$, and h_{ij} is the public key of user P_{ij}. We use Σ to denote the aggregation of all BCP-based ciphers in the following parts.

The behavior chain then follows the encoding rules of the BCP algorithm to generate a size-similar random value to encoded gradients. Take C_{ij} as an example, the behavior chain generates a noise n_{ij}, then uses P_{ij}'s public key to encrypts n_{ij}, and gets $N_{ij} := (A'_{ij}, B'_{ij}) = (g^{r'_i} \bmod n^2, h^{r'_i}(1+n)^{n_i} \bmod n^2)$. Then, it aggregates C_{ij} and N_{ij} to get $C_{ij} + N_{ij} = \left(A_{ij}, A'_{ij}, B_{ij} \times B'_{ij}\right)$. The matrix composed of noisy gradients is represented as $C' = (C'_{ij})_{3\times 3}$.

During noise generation, the behavior chain adjusts each n_{ij} to ensure that the noises in rows and columns are summed to zero. Because the BCP algorithm is homomorphic, all N_{ij} in rows and columns are also summed to zero, which makes the aggregated gradients in each row and column noiseless. That is, $\sum_{j=1}^{3}(C_{ij} + N_{ij}) = \sum_{j=1}^{3} C_{ij}, i \in \{1,2,3\}$ and $\sum_{i=1}^{3}(C_{ij} + N_{ij}) = \sum_{i=1}^{3} C_{ij}, j \in \{1,2,3\}$. In this way, the audit chain can get noiseless gradients after aggregation.

Compared to other noise mechanisms, the BCP-based noise mechanism brings two benefits to our audit approach: a) the audit chain can run the audit approach in rows and columns, even if it cannot obtain any individual gradient m_{ij}. Malicious gradients are easy to be accurately located. b) The noisy encrypted gradients can be correctly decrypted to noisy gradients. The BCP-based noise mechanism does not destroy the original data. When any device appeals to the audit result, it can recover the original gradient and noise from the encrypted gradient and initiate a review process in the audit chain.

4.5. The joint audit algorithm

Before providing gradients to the behavior chain, each physical device generates a hash of encrypted gradients and submits it to the behavior chain. Behavior chain will provide them with an interface to evaluate the gradients on public test dataset. Then, the behavior chain collects the encrypted gradients along with evaluation results and uploads the noisy gradients to the audit chain for quality audit. The audit aims to achieve two security goals: a) check whether any gradient poisons the global model. b) verify whether each gradient's quality is consistent with the claims of the physical device.

Each physical device updates the model locally and then calls the smart contract to calculate the loss of the local model. A lower loss value means a higher quality gradient. Each physical device needs to upload the loss L_{ij} as a data quality claim.

To audit each gradient's quality, the audit chain first aggregates all the gradients to obtain $\sum_{j=1}^{3}\sum_{i=1}^{3}C_{ij}$, and then decrypts it to $\sum_{j=1}^{3}\sum_{i=1}^{3}m_{ij}$. It then calculates the global aggregated gradient $g_{global} = \frac{1}{9}\sum_{j=1}^{3}\sum_{i=1}^{3}m_{ij}$ and uses it to update the model. After that, the audit chain evaluates the model on the public test dataset and calculates the loss, which is denoted as L_{global}. The loss L_{global} will be set as a baseline in the current audit process.

If $|L_{global} - \frac{1}{9}\sum_{j=1}^{3}\sum_{i=1}^{3}L_{ij}| < \epsilon_1, \epsilon_1 > 0$, that is, the loss error is within the acceptable range, and each loss L_{ij} can be seen as honestly stated. Otherwise, it means that the data quality between devices is significantly different. There may exist dishonest devices which provide bad gradients. The audit chain will audit the gradien in groups. Given $C^{'} = (C^{'}_{ij})_{3\times3} = (C_{ij} + N_{ij})_{3\times3}$, the audit chain aggregates gradients in rows or columns. To explain the joint audit approach in Algorithm 1, we divide the possible scenario into two parts for discussion:

***Case 1*. Single malicious physical device.** In this scenario, there is only one malicious individual gradient which damages aggregated gradients. The audit chain aggregates gradients in rows and columns, respectively. If the audit chain finds only two abnormal aggregated gradients between different rows and columns, it identifies their common, gradients as suspected malicious gradients. For example, the audit chain aggregates and decrypts the gradients in a row to $m^{\mathrm{r}}_{i} = decrypt(\sum_{j=1}^{3}C^{'}_{ij})$, where $C^{'}_{ij}$ is the gradient in the *i-th* row and the *j-th* column. Then, the audit chain repeats the process in column *j* to get $m^{\mathrm{c}}_{j} = decrypt(\sum_{i=1}^{3}C^{'}_{ij})$. If m^{r}_{i} and m^{c}_{j} are evaluated as abnormal, that is, $L^{r}_{i} - L_{global} > \epsilon_2$ and $L^{c}_{j} - L_{global} > \epsilon_2$, it means that the common gradient $C^{'}_{ij}$ may be the malicious gradient which spoils the global gradient.

***Case 2*. Multiple malicious physical devices.** We assume that malicious devices are less than 1/3 of the devices. If the audit chain finds the globally aggregated gradients abnormal but can not locate a malicious gradient as in *Case 1*, it considers that there are multiple malicious gradients in C'. In this case, the audit chain checks whether the quality statements are consistent with the quality of the gradients. Specifically, the audit chain checks whether the errors between statements L_{ij} and evaluation results L^{c}_{j}, L^{r}_{i} are less than the acceptable threshold. The abnormal gradients will be classified as suspect gradients. The audit chain can return the result to the behavior chain to initiate a gradient resubmission. The behavior chain needs to regroup C_{ij} according to the audit results, and resubmits a new noisy gradients matrix.

If a device is found to be suspected more than twice, behavior chain can identify it as a malicious device unless it appeals to the audit result. In this situation, the device can interact with the audit chain to remove the noises in gradients and request individual audits to prove innocence. After the audit process, any gradient which spoils the model will be discovered. Data poisoning and free-riding attacks can be well defended by further mechanisms such as penalties.

Algorithm 1: Joint audit algorithm

```
Input  : Global loss $L_{global}$;
Nosiy gradient martrix $C^{'} = (C^{'}_{ij})_{3\times3}$;
Quality statement matrix $(L_{ij})_{3\times3}$;
Threshold $\varepsilon_1 > 0, \varepsilon_2 > 0, \varepsilon_3 > 0$;
Output: Malicious Poisoning device set $MP$;
Malicious Free-riding device set $MF$;
Initialzation:$MP = \emptyset\ MF = \emptyset$;
if $|L_{global} - \frac{1}{9}\sum_{j=1}^{3}\sum_{i=1}^{3}L_{ij}| < \varepsilon_1$ then
    Audit pass;
else
    Aggregate $C^{'}_{ij}$ in each row and column;
    Decrypt each aggregated gradient to obtain
    $\{m^{r}_{i}\}_{i=1,2,3}$ and $\{m^{c}_{j}\}_{j=1,2,3}$;
    Evaluate each $m^{r}_{i}$, $m^{c}_{j}$ to obtain $\{L^{r}_{i}\}_{i=1,2,3}$,
    $\{L^{c}_{j}\}_{j=1,2,3}$;
    for each $L^{r}_{i}$, $L^{c}_{j}$ do
        if $L^{r}_{i} - L_{global} > \varepsilon_2$ and $L^{r}_{j} - L_{global} > \varepsilon_2$
        then
            $MP = MP \cup C_{ij}$
        end
    end
    for each $L^{r}_{i}$, $L^{c}_{j}$ do
        if $L^{r}_{i} - L_{ij} > \varepsilon_3$ and $L^{c}_{j} - L_{ij} > \varepsilon_3$ then
            $MF = MF \cup C_{ij}$
        end
    end
end
return $MP$, $MF$
```

5. Security analysis

In this section, we discuss how our approach enables privacy-preserving gradient auditing in federated learning. This is also the security goal presented in Section 3.

5.1. Quality auditability of encrypted gradient

Our goal is to locate malicious gradients that corrupt the global model or find devices that contribute little to the training. Based on the following lemmas, we can ensure that our proposed approach can identify malicious low-quality gradients.

Lemma 1. *In epoch t, the model's parameter w^t is updated by $w^{t+1} \leftarrow w^t - \frac{\rho}{t}\sum_{i=0}^{t}g_i^{(t)}$, where ρ is learning rate and t is the number of devices. $g_i^{(t)}$ denotes the gradient from device P_i.*

In data poisoning attacks, a malicious device may use the wrong data labels to train a model. According to Lemma 1, a malicious gradient $g_{malicious}^{(t)}$ differs significantly from other gradients $g_i^{(t)}$. It makes the aggregated gradient unable to update the model parameters w^t positively and even makes the model performance worse. In our joint audit algorithm, each gradient needs to be aggregated together with different gradients of the current batch. In different aggregations, since the gradients of honest devices generally make the model converge in a similar positive optimization direction, the aggregation results also produce similar model optimization. If the aggregation does not optimize the model positively, it indicates that there may be malicious gradient in that aggregation. We assume that at least 2/3 of the devices are honest. The gradients within each abnormal aggregation are also involved in other aggregations. Consequently, if a gradient appears in at least two abnormal aggregations, it will be defined as a suspicious malicious gradient.

In free-riding attacks, malicious devices profit by disguising low-quality gradients (such as zero gradients) as high-quality gradients. Such gradients from malicious devices do not fully affect the optimization direction of the model, but reduce the efficiency of the optimization. To identify such malicious gradients, we add a quality statement mechanism to our audit approach. We set a public test dataset for each iteration, thereby unifying the measurement standard for each gradient. If a device provides a malicious low-quality gradient $g_{malicious}^{(t)}$ with actual loss $L_i = t$ but states a lower loss value $L_{i_{fake}}$, it *will result in aggregation's loss $L_{aggregation}$ much higher than $L_{i_{fake}}$. When this situation also occurs within other aggregations, it indicates that the source device of this gradient may be dishonest. If a device is detected as dishonest multiple times, it will be defined as a free-riding attacker.*

Given that different devices have different data collection capabilities in varying environments, the data they hold varies in quality. The damage to the global model may not be subjective to a device. If a suspect device wants to prove its innocence, it can apply for a review. Specifically, it obtains the encrypted gradient with noise from the audit chain and recovers the noise using the private key and the local gradient. Next, the device returns the noise to the audit chain. The audit chain will use this noise to recover the individual gradient and evaluate whether the suspect gradient is misidentified.

5.2. Privacy protection in auditing

To prevent a single gradient from being exposed to others, we utilize the additive homomorphism of the BCP algorithm in gradient aggregation. We assume that each device does not disclose its private key. Based on the following lemmas, we make full use of the double trapdoor decryption mechanism of the BCP algorithm and the double blockchain architecture to ensure the confidentiality of each gradient before aggregation.

Lemma 2. *If there is a number $y \in Z^*_{n^2}$ such that $z = y^n \bmod n^2$, there is no polynomial-time approach to find the exact value z* [34]. *Thus, the BCP algorithm is difficult to be cracked without a secret key or master key.*

Lemma 3. *According to the BCP algorithm, if two ciphers C_1 and C_2 are encrypted with the same public key pk_i, their aggregation can be decrypted by the private key sk_i or by the master key mk. For two ciphers C_1' and C_2' encrypted with different public keys pk_i and pk_j, respectively, their aggregation can only be decrypted by the master key mk.*

Lemma 2 *demonstrates the security of the BCP algorithm. It states that a cipher can only be decrypted by the party holding the private key sk_i or master key mk. Based on the assumption that each participant does not reveal its private key pk_i, we propose a structure that decomposes the gradient collection and gradient aggregation work into two different blockchains. Since the behavior chain does not hold any sk_i and mk, the encrypted gradient is secure during the collection process based on* Lemma 3.

Lemma 4. *Suppose there exist n variables in a matrix, and they are constrained by at most $n - 1$ homogeneous linear equations. There are infinitely non-zero solutions for those n variables. In other words, there exist values that make the sum of rows and columns to zero in the matrix.*

Because the audit chain holds the master key mk to decrypt any ciphers, the behavior chain cannot directly submit the individual encrypted gradients to the audit chain for aggregation and audit. To solve this problem, we design a noise addition mechanism based on the BCP algorithm on the behavior chain. The behavior chain adds encrypted noise N_i to encrypted gradient C_i to obtain $C_i + N_i$. The audit chain decrypts the encrypted noisy gradient $C_i + N_i$ to obtain the noisy gradient $m_i + n_i$, but not the gradient m_i. In addition, the behavior chain needs to ensure the noises n_i in each row and column are summed to zero, which ensures the correctness of gradients. According to Lemma 4, *it is not difficult to find the noises to ensure correctness. The audit chain can obtain noiseless aggregated gradients for audit without knowing each gradient for privacy reasons.*

6. Experiments evaluation

In this section, we implement a prototype for our audit approach. To simulate a federated learning scenario, we choose 1 GPU server (8NVIDIA TELSA T4 GPUs 16 GB) and 9 PCs to simulate the service provider and physical devices. The functions of the behavior chain and the audit chain are deployed on 6 CPU servers (Hygon C86 7159 16-core Processor). We use image recognition as our training task and construct our learning model based on Python (version 3.83) and Pytorch (version 1.6.0). The face dataset CelebA[1] and handwritten digits dataset MNIST[2] were chosen for training and testing.

Table 1
The overhead of the BCP algorithm.

Size of group (bit)	Number of group (s)		
	1×10^3	2×10^3	3×10^3
30	5.03	10.17	15.22
300	5.69	10.98	16.40

6.1. The overhead of encryption algorithm

The BCP algorithm plays a very important role in our auditing approach. In this section, we examine the feasibility of the BCP algorithm by testing the encryption and decryption overhead for different group sizes and message sizes. Our BCP algorithm is implemented based on the Charm-crypto library in Python and deployed on CPU servers (Hygon C86 7159 16-core Processor). Charm-crypto is a Python library developed in 2013 for fast cryptographic operations. Its bottom layer converts Python to C for execution, thus improving the execution efficiency of cryptographic algorithms.[3]

Specifically, we set the modulus N in the BCP algorithm to 1024 bits and test its performance on the CPU. Since the encryption algorithm has a limit on the message size, we encode the gradient and divide it into multiple vectors for batch encryption. First, we measure the impact of group size on the overhead of the BCP algorithm, as shown in Table 1. When the message packet size is 30 bits and the number of packets is 1×10^3, the encryption overhead of the BCP algorithm is 5.03 s. And when the message size increases to 300 bits, the encryption overhead does not change significantly. This shows that the message group size has no significant effect on the encryption and decryption overhead when the message group size is maintained within the limits of the BCP algorithm.

To maintain the correctness of the aggregated gradient, we choose 30 bits as our group size of the message. We test the overhead of our revised BCP decryption at different gradient scales and compare the overhead with Paillier decryption and original BCP decryption.

As shown in Fig. 8(a), when the number of parameters grows linearly, the overhead of the BCP decryption maintains a similar linear growth. While the total number of parameters is 1.5×10^5, the overhead of Paillier decryption and original BCP decryptions is 24.29 s, 25.44 s and 51.17 s, respectively. The overhead of our BCP decryption is 38.33 s. This shows that our BCP algorithm has a similar performance to the original additional homomorphic encryption. It will not cause much extra burden in gradient aggregation. Fig. 8(b) shows the decryption overhead in our approach. We adjust the number of encrypted gradients to observe the overhead changes of our BCP decryption and original BCP decryption. When the number of gradients increases, our decryption method costs much less time and the total time consumption is maintained at a low level. This means that our BCP algorithm can be well adapted to our blockchain-based federated learning in terms of performance.

6.2. The performance of our joint audit algorithm

To prove that our audit approach is feasible, we simulate the federated learning on datasets CelebA and MNIST, respectively. We evenly divide the datasets into several subsets. We distributed them to our 9 PCs. In each epoch of training, each PC randomly selects a subset to participate in the federated learning. The number of global epochs and local

[1] Large-scale CelebFaces Attributes (CelebA) Dataset. http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html.

[2] The MNIST database of handwritten digits Dataset. http://yann.lecun.com/exdb/mnist/.

[3] Charm-crypto. https://github.com/JHUISI/charm.

iterations is 50 and 1. The learning rate is 0.0001, which we halve every 10 global epochs. As for CelebA, we set the size of the data subset to 1280 and the batch size to 64. As for MNIST, we set the size of the data subset to 800 and the batch size to 50.

In each epoch, the behavior chain collects the encrypted gradients from 9 PCs. Then, it generates a new block containing a 3 × 3 matrix of gradients and sends it to the audit chain. We first evaluated the gradients in the model aggregation without any attacks and set it as the baseline. The result is shown in Fig. 9.

We can see that the gradient aggregated in each row and column performs similarly to the global aggregated gradient. The accuracy can reach 92.28% on CelebA and 95.52% on MNIST. We then set the first PC as a malicious physical device and named it PD 1. We record the performance of PD 1 to help us analyze the experimental results. We first evaluated the impact of data poisoning attacks on gradient aggregation in the audit method. PD 1 uploads a large negative gradient instead of the normal gradient to the service provider. The result can be seen in Fig. 10. When PD 1 performs a data poisoning attack, its model keeps a low accuracy in each epoch. The convergence rate of the global model slows down significantly. The final accuracy only reaches 90.75% on CelebA and 94.47% on MNIST, lower than the baseline. We can also see that Row 1 and Column 1 perform unstable and lower than others. It means that the common malicious PD 1 can be discovered in our audit approach.

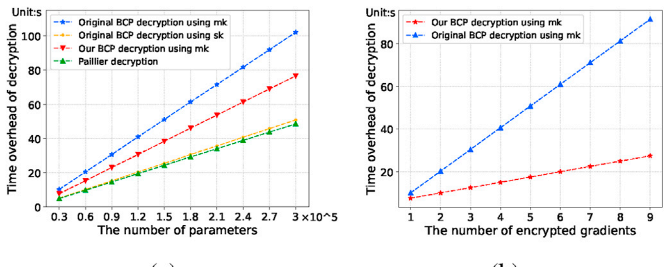

Fig. 8. The overhead of decryptions. (a) Different lengths of single gradient; (b) Different numbers of gradients in aggregation.

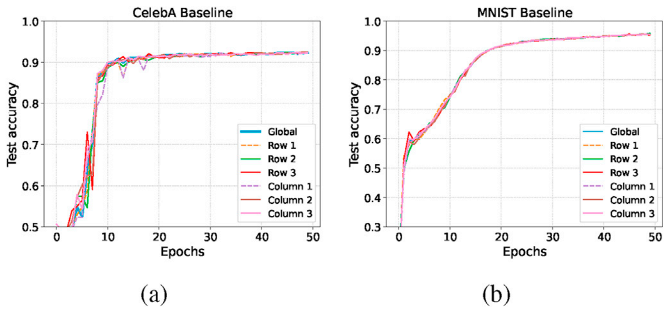

Fig. 9. Federated learning without attacks. (a) on CelebA; (b) on MNIST.

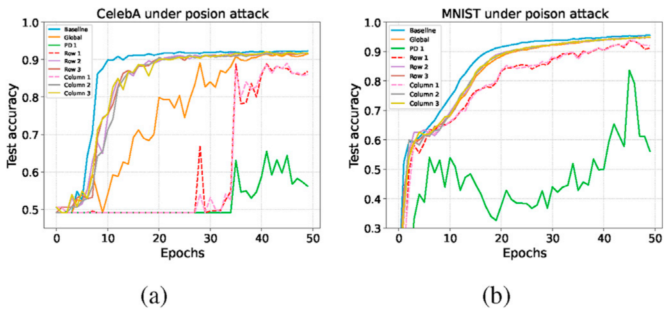

Fig. 10. Test accuracy of model under poison attack. (a) on CelebA; (b) on MNIST.

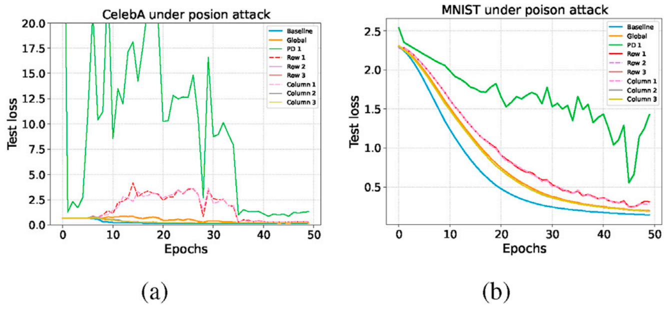

Fig. 11. Loss of model under posion attack. (a) on CelebA; (b) on MNIST.

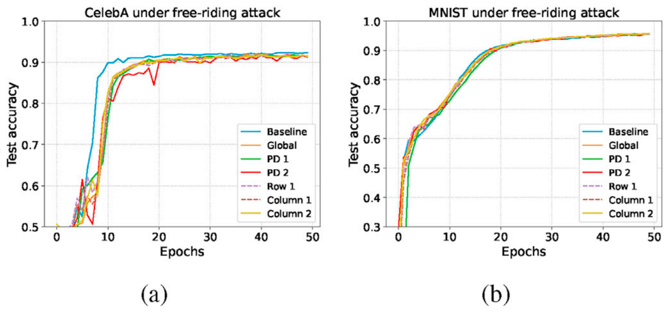

Fig. 12. Accuracy of model under free-riding attack. (a) on CelebA; (b) on MNIST.

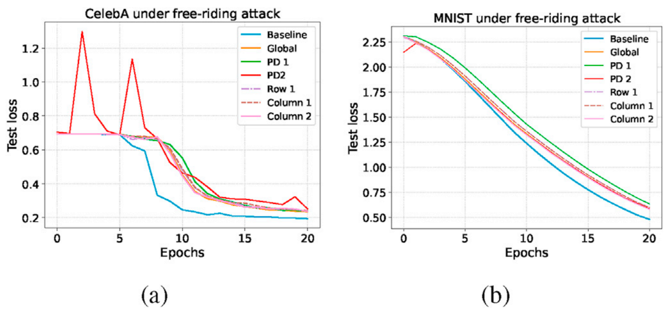

Fig. 13. Loss of model under free-riding attack. (a) on CelebA; (b) on MNIST.

We implement our audit approach using the loss as our reference. The loss value changes are shown in Fig. 11. On both CelebA and MNIST, the losses of Row 1 and Column 1 are much higher than those of other rows and columns. Our audit approach can easily locate the malicious device PD 1, which provides the common gradient in Row 1 and Column 1.

As for the free-riding attack, we set the malicious PD 1 not to conduct local training but only to upload a zero gradient in each epoch. The accuracy changes can be seen in Fig. 12. A free-riding attack also slows down the convergence speed. We especially show the difference between PD 1 and another honest PD 2. In the early epoch of training, the accuracy of PD 1 is always lower than the global accuracy. It makes the accuracy of Row 1 and Column 1 continually lower than global accuracy, especially between epoch 10 and epoch 20.

We show the gradient loss in Fig. 13. PD 1 always has a significantly poor performance between epoch 10 and epoch 15. When our audit chain finds that Row 1 and Column 1 consistently underperform the others, it considers PD 1 as a free-riding attacker. As for PD 2, the loss value sometimes rises rapidly. This may be due to the poor training data in some epochs. Its loss does not show a consistently higher than global loss, so it won't be identified as a free-riding attacker. When honest PDs perform well and the global accuracy rises fast in an epoch, our approach will work better.

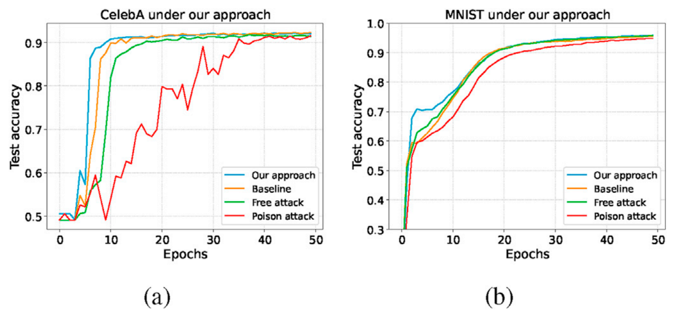

Fig. 14. The performance of our audit approach. (a) On CelebA; (b) On MNIST.

To better demonstrate the performance of our audit approach, we assume that if a malicious gradient is detected, the relevant malicious devices will not appeal for a review. It will resubmit a normal gradient to replace the malicious gradient, so the audit chain can aggregate gradients and update the model as normal. The result is shown in Fig. 14. As we can see, our approach makes the model performance under attacks similar to the baseline. Due to the discrepancy in training data, some honest gradients may damage the global model unintentionally. Our approach will also identify this behavior, and even make the training better than baseline. If the device has honestly reported the true gradient quality, it will not be identified as malicious. Our approach can well defend against data poison attacks and free-riding attacks.

7. Conclusion

In this paper, we propose a blockchain-based quality audit approach for encrypted data in federated learning. The gradient collection and secure aggregation are assigned to two separate blockchains, which prevent individual gradients from being accessed by third parties. Specifically, we use the homomorphic BCP algorithm to protect the gradients and add specially designed noises to encrypted gradients. The noises will be eliminated after model aggregation, which ensures the correctness of aggregated gradient. To identify the malicious participants, we map the gradients into different groups. If a gradient spoils the aggregated gradient in multiple groups, we can define it as a malicious gradient. Based on the audit results, we can penalize malicious devices to maintain the fairness of federated learning. Devices judged to be malicious can also be reviewed by recovering the original gradient through noise elimination. Through security analysis and experimental evaluation, we demonstrate that our approach can achieve our security goals: effective resistance to data poisoning and free-rider attacks in federated learning.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research is sponsored by the National Key R&D Program of China (No. 2018YFB2100400), the National Natural Science Foundation of China (No. 62002077, 61872100), the Guangdong Basic and Applied Basic Research Foundation (No. 2020A1515110385), Strategic Research and Consultation Project of the Chinese Academy of Engineering (No. 2021-HYZD-8-3), the China Postdoctoral Science Foundation (No. 2020M682657), Zhejiang Lab (No. 2020NF0AB01).

References

[1] M.K. Hasan, S. Islam, S. Khan, A.H.A. Hashim, S. Habib, S. Alyahya, K. Samar, Lightweight encryption technique to enhance medical image security on internet of medical things applications, IEEE Access 9 (2021) 47731–47742.
[2] W. Ding, X. Jing, Z. Yan, L.T. Yang, A survey on data fusion in internet of things: towards secure and privacy-preserving fusion, Inf. Fusion 51 (2019) 129–144.
[3] J. Konečnỳ, H. B. McMahan, D. Ramage, P. Richtárik, Federated Optimization: Distributed Machine Learning for On-Device Intelligence, arXiv preprint arXiv: 1610.02527.
[4] M. Fredrikson, S. Jha, T. Ristenpart, Model inversion attacks that exploit confidence information and basic countermeasures, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, ACM, 2015, pp. 1322–1333.
[5] L. Melis, C. Song, E. De Cristofaro, V. Shmatikov, Exploiting unintended feature leakage in collaborative learning, in: 2019 IEEE Symposium on Security and Privacy (SP), IEEE, 2019, pp. 691–706.
[6] R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership inference attacks against machine learning models, in: 2017 IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 3–18.
[7] M. Naehrig, K. Lauter, V. Vaikuntanathan, Can homomorphic encryption be practical?, in: Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop ACM, 2011, pp. 113–124.
[8] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, B. Waters, Candidate indistinguishability obfuscation and functional encryption for all circuits, SIAM J. Comput. 45 (3) (2016) 882–929.
[9] A. Beimel, Secret-sharing schemes: a survey, in: International Conference on Coding and Cryptology, Springer, 2011, pp. 11–46.
[10] X. Chen, C. Liu, B. Li, K. Lu, D. Song, Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning, arXiv preprint arXiv:1712.05526.
[11] H. Kim, J. Park, M. Bennis, S.-L. Kim, Blockchained on-device federated learning, IEEE Commun. Lett. 24 (6) (2019) 1279–1283.
[12] Z. Sun, L. Yin, C. Li, W. Zhang, A. Li, Z. Tian, The qos and privacy trade-off of adversarial deep learning: an evolutionary game approach, Comput. Secur. 96 (2020), 101876.
[13] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, W. Luo, Deepchain: auditable and privacy-preserving deep learning with blockchain-based incentive, IEEE Trans. Dependable Secure Comput. 18 (5) (2019) 2438–2455.
[14] J. Kang, Z. Xiong, D. Niyato, S. Xie, J. Zhang, Incentive mechanism for reliable federated learning: a joint optimization approach to combining reputation and contract theory, IEEE Internet Things J. 6 (6) (2019) 10700–10714.
[15] Y. Zhan, P. Li, Z. Qu, D. Zeng, S. Guo, A learning-based incentive mechanism for federated learning, IEEE Internet Things J. 7 (7) (2020) 6360–6368.
[16] P. Blanchard, E.M. El Mhamdi, R. Guerraoui, J. Stainer, Machine learning with adversaries: byzantine tolerant gradient descent, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 118–128.
[17] Y. Lu, X. Huang, Y. Dai, S. Maharjan, Y. Zhang, Blockchain and federated learning for privacy-preserved data sharing in industrial iot, IEEE Trans. Ind. Inf. 16 (6) (2019) 4177–4186.
[18] E. Bresson, D. Catalano, D. Pointcheval, A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications, in: International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2003, pp. 37–54.
[19] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, Y. Liu, Privacy-preserving blockchain-based federated learning for iot devices, IEEE Internet Things J. 8 (3) (2020) 1817–1829.
[20] M. Westerkamp, F. Victor, A. Küpper, Tracing manufacturing processes using blockchain-based token compositions, Digit. Commun. Netw. 6 (2) (2020) 167–176.
[21] Y. Qu, L. Gao, T.H. Luan, Y. Xiang, S. Yu, B. Li, G. Zheng, Decentralized privacy using blockchain-enabled federated learning in fog computing, IEEE Internet Things J. 7 (6) (2020) 5171–5183.
[22] G. Xu, H. Li, S. Liu, K. Yang, X. Lin, Verifynet: secure and verifiable federated learning, IEEE Trans. Inf. Forensics Secur. 15 (2019) 911–926.
[23] C. Jiang, C. Xu, Y. Zhang, Pflm: privacy-preserving federated learning with membership proof, Inf. Sci. 576 (2021) 288–311.
[24] S. Shen, S. Tople, P. Saxena, Auror: defending against poisoning attacks in collaborative deep learning systems, in: Proceedings of the 32nd Annual Conference on Computer Security Applications, ACM, 2016, pp. 508–519.
[25] B.I. Rubinstein, B. Nelson, L. Huang, A.D. Joseph, S.-h. Lau, S. Rao, N. Taft, J.D. Tygar, Antidote: understanding and defending against poisoning of anomaly detectors, in: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, ACM, 2009, pp. 1–14.
[26] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and Open Problems in Federated Learning, arXiv preprint arXiv:1912.04977.
[27] Y. Aono, T. Hayashi, L. Wang, S. Moriai, et al., Privacy-preserving deep learning via additively homomorphic encryption, IEEE Trans. Inf. Forensics Secur. 13 (5) (2017) 1333–1345.

[28] M. Shayan, C. Fung, C.J. Yoon, I. Beschastnikh, Biscotti: a blockchain system for private and secure federated learning, IEEE Trans. Parallel Distr. Syst. 32 (7) (2020) 1513–1525.

[29] C. Fung, C. J. Yoon, I. Beschastnikh, Mitigating Sybils in Federated Learning Poisoning, arXiv preprint arXiv:1808.04866.

[30] X. Qu, S. Wang, Q. Hu, X. Cheng, Proof of federated learning: a novel energy-recycling consensus algorithm, IEEE Trans. Parallel Distr. Syst. 32 (8) (2021) 2074–2085.

[31] C. Cachin, et al., Architecture of the hyperledger blockchain fabric, in: Workshop on distributed cryptocurrencies and consensus ledgers 310, 2016, pp. 1–4.

[32] B. Cao, Z. Zhang, D. Feng, S. Zhang, L. Zhang, M. Peng, Y. Li, Performance analysis and comparison of pow, pos and dag based blockchains, Digit. Commun. Netw. 6 (4) (2020) 480–485.

[33] M. Van Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan, Fully homomorphic encryption over the integers, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2010, pp. 24–43.

[34] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 1999, pp. 223–238.