FL-MFGM: A Privacy-preserving and High-accuracy Blockchain Reliability Prediction Model

Jianlong Xu[1], Weiwei She[1], Jian Lin[1], Zhuo Xu[1], Hao Cai[1], Zhi Xiong[1], and Changsheng Zhu[1]

[1]Colleage of Engineering, Shantou University, Shantou 515061, China
{xujianlong, 17wwshe,20jlin3,20zxu3,haocai,zxiong,cszhu}@stu.edu.cn

Abstract. Reliability prediction mechanism plays an important role in selecting reliable peers for blockchain users. Traditionally, reliability prediction approaches are performed by collecting a large amount of user data and conducting centralized training. Although these methods can achieve high accuracy, they may lead to malicious attacks by third-party services on users' private data. As a privacy-preserving approach to addressing privacy leakage caused by centralized training, federated learning can enable users to learn a shared prediction model by providing gradients to a central server instead of raw data. However, uploading complete gradients directly may be exposed to a gradient attack and cause privacy leakage. To address the privacy data leakage from centralized training and prevent the attack threat from uploading gradients, we propose a privacy-preserving and high-accuracy blockchain reliability prediction model, namely FL-MFGM. Based on federated learning architecture, the model protects user privacy by uploading the gradients of matrix factorization. Specifically, in order to enhance the privacy-preserving capability of our model, we employ gradient compression with momentum correction for federated matrix factorization. To validate the effectiveness of our approach, sufficient experiments are conducted, which indicate that our approach can achieve higher prediction accuracy than other approaches with privacy-preserving based on federated learning architecture.

Keywords: blockchain; reliability prediction; privacy protection; federated learning; matrix factorization

1 Introduction

Blockchain technology has received wide attention from industry and academia because of its decentralized, persistent, and auditable features. These features are derived from the working mechanism of blockchain system. Blockchain is a shared digital ledger and each transaction is public[1]. The communication technology of the system is Peer-to-Peer (P2P), where communication between peers does not need to rely on a third party. Each participant peer in such a system maintains a blockchain and acts in the roles of both client and server, i.e., each peer can request content or services from other peers and act as a server to provide content or services to other peers[2]. The way to maintain the blockchain is through a consensus mechanism. Once a peer finds a proof-of-work, it broadcasts the block to all peers and receives a reward[1].

In order to get rewards[3] or verify transactions[4], users in the blockchain network need to request the latest block information from neighboring peers. However, not all the peers would return the latest block information because of the network latency, malicious attacks, etc. If a peer returns incorrect or outdated block information, it would waste the users' time and resources. The users may even be vulnerable to being attacked as a result[3].

In order for users to select more secure and reliable blockchain peers, one simple solution is to use known reliability information (e.g., successful request rates from users to peers) to predict the unknowns. Many existing approaches are designed for centralized training, which collects users' raw data in the central servers. However, large-scale collection of sensitive data entails risks[5]. For example, malicious third parties can steal sensitive data by attacking the server and untrustworthy data collectors may disclose users' private data[6]. Therefore, it is necessary to enhance the privacy protection of reliability prediction.

Federated learning is an effective method to protect user privacy. Current researches based on federated learning include edge computing, Internet of Things, and smart healthcare[7][8][9]. The main idea of federated learning is to enable users to leave the training data distributed on the mobile devices and learn a shared model by aggregating locally-computed updates (e.g., gradient information)[10]. However, the canonical federated learning involves two big challenges: First, shared gradients publicly may lead to leakage of private training data[11]; Second, communication in federated networks can be much more expensive than that in classical data center environments because of comprised of a massive number of devices (e.g., millions of smartphones)[12].

In this paper, we propose a privacy-preserving and high-accuracy blockchain reliability prediction model based on federated matrix factorization with gradient compression and momentum correction (FL-MFGM). FL-MFGM achieves privacy preservation in blockchain reliability prediction by federated matrix factorization and gradient compression. The main contributions of this paper are summarized as follows:

(1) We propose a blockchain reliability prediction model based on federated matrix factorization to protect user privacy.
(2) We design a gradient compression method to reduce the uploaded gradients, which enables our model to defend against gradient attacks.
(3) We adopt momentum correction to ensure the prediction accuracy.
(4) We verify the effectiveness of our model through sufficient experiments.

The rest of the paper is structured as follows: Section 2 introduces the related work. Section 3 includes details of the approach in this paper. Section 4 elaborates on the experiments and discusses the results of the experiments. Section 5 concludes the work of this paper and gives the vision for the future work.

2 Related Work

This section focuses on reviewing current approaches to blockchain reliability prediction and privacy protection based on federated learning, which are relevant to this paper.

Blockchain Reliability Prediction Current approaches to blockchain reliability prediction are based on studies of software reliability prediction. Zheng et al.[4] proposed a hybrid blockchain reliability prediction (H-BRP), which takes blockchain-related factors (e.g., block hash, block height) into consideration and finds the similarity among users and among peers to do the prediction. However, H-BRP is a centralized training model that must collect the data of all users to calculate the similarity between peers, which may cause sensitive data leakage. Xu et al.[13] proposed a high-accuracy reliability prediction approach for block-chain services under BaaS (BSRPF), using matrix factorization. BSRPF is also a centralized training model, but it inspired our work in this paper that we can use matrix factorization based on federated learning to predict reliability effectively and protect user privacy by not sharing raw data.

Privacy Protection Based on Federated Learning Federated learning can significantly reduce privacy and security risks by sharing model updates instead of uploading a local dataset[10]. However, it may still reveal sensitive information to third parties or central servers[12]. Zhu et al.[11] demonstrated that attackers can extrapolate the local data from the high-density gradient communication. They also summarize the practical methods to defend against this kind of attack effectively without a dropped performance, including gradient compression and homomorphic encryption.

Homomorphic encryption is a kind of cryptology that is the most secured defense. But it has limitations: the gradients required should be an integer[5], homomorphic encryption is only for parameter servers[14]. Gradient compression can be used universally because it has no limitations on gradients or servers and is easy to be implemented. Many studies have focused on two aspects to optimize gradient compression: the choice of sparsification threshold and the problem of accuracy loss and low convergence rate.

Strom Nikko[15] developed a gradient compression method with a fixed threshold. But it is difficult to choose an available threshold due to the variation of gradients. Tao et al.[16] proposed an approach to solving this problem. They set a threshold according to gradient average value and designed a momentum residual accumulation for tracking the residual gradients. It can also recover the low convergence caused by gradient sparsification. However, the convergence comes with nearly a thousand iterations, which was exceeding our expectations. Lin et al.[17] proposed a deep gradient compression (DGC), using warm-up training and gradient sparsification with momentum correction and gradient clipping to solve the problem of low accuracy and low convergence rate. DGC achieves a gradient compression ratio of about 600× without losing accuracy.

Although the existing reliability prediction methods of blockchain have achieved relatively high accuracy, these methods are based on centralized training and lack consideration for user privacy protection, which may be attacked by third-party services. To address the privacy data leakage from centralized training and prevent the attack threat, inspired by federal learning, we focus on the privacy protection of blockchain reliability prediction in this paper, hoping to ensure the accuracy of prediction while protecting users' privacy.

3 Privacy-preserving Blockchain Reliability Prediction Model

In this section, we first present a framework of our FL-MFGM model in Section 3.1. Then, We illustrate the federated matrix factorization approach for our model in Section 3.2 and introduce how to achieve gradient compression with momentum correction for our model in Section 3.3. Finally, we show the implementation in Section 3.4.

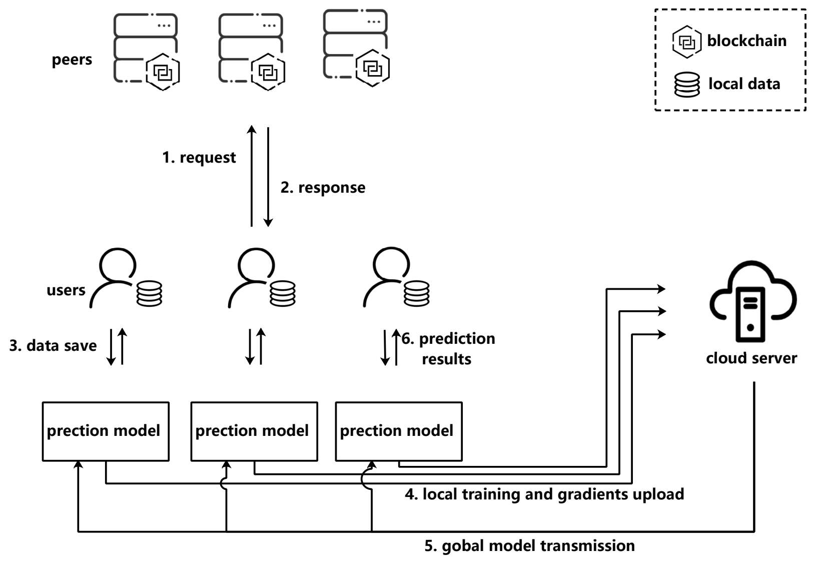


Fig. 1. The framework of FL-MFGM model

3.1 Model Framework

The main idea of FL-MFGM is that users train the model respectively using the data stored locally, and upload the gradients generated by the local model update to the central server, which updates the global model and transmit it to users. As the iterations proceed, the model gradually converges, and users can use the model to predict the reliability of candidate peers for selecting a more reliable peer. As shown in Fig. 1, the model framework mainly consists of six steps:

(1) Users randomly request blockchain data from candidate peers over a period of time.
(2) The requested peers respond and return data (e.g., block hash, block height).
(3) Users compile the information responded from peers and save the data locally.
(4) Users take a certain number of iterations to update the local model based on their local data. After local training, they compute gradients and send them to the central server.
(5) The central server aggregates the gradients from users and updates the global model. The improved global model will be transmitted to users for the next training.
(6) After the model becomes converged, users can predict the reliability of candidate peers and select the ones that meet their reliability requirements.

It is worth noting that during the model training, the fourth and fifth steps will conduct for multiple rounds, where users and the central server will keep in communication until the model becomes converged.

3.2 Federated Matrix Factorization for Blockchain Reliability Prediction

To predict the reliability of blockchain while protecting user privacy, we propose a matrix factorization model based on federated learning. Matrix factorization is a typical factor analysis model. Its main idea is to predict the unknowns in the matrix by mapping both users and peers into a joint latent factor space of a low dimensionality d. And federated learning enables users to have a shared model without uploading raw data, which achieves protecting users' privacy. Thus, our federated matrix factorization model incorporates the features of matrix factorization and federated learning.

The proposed model training uses the successful request rates from users. After finishing the request testing mentioned in Section 3.1, the SuccessRate matrix which represents successful request rates to peers is generated. It's worth noting that, the higher successful request rate indicates that the peer is more reliable to the user[4].

We update the matrix factorization model, including user profile matrix and peer profile matrix by the SuccessRate matrix. To prevent the server from directly knowing users' raw data or learned profiles, users will update the user profile matrix and peer profile matrix only locally using their SuccessRate matrix, and share gradients of peer profile matrix to the server for updating the global model[18].

We assume there are N users and M peers. The SuccessRate matrix for user i is denoted as $R^i \in \mathbb{R}^{1\times M}$, and the element of column j of R^i denotes the successful request rate from user i to peer j. For user i, the local matrix factorization model consists latent user factors $U^i \in \mathbb{R}^{1\times d}$ and latent peer factors $P^i \in \mathbb{R}^{M\times d}$. Thus, we aim to minimize the objective function with regularization terms[19]:

$$\mathcal{L}^i = \frac{1}{2}\sum_{j=1}^{M} I_j^i \left(R_j^i - U^i \left(P_j^i\right)^T\right)^2 + \frac{\lambda_U}{2}\|U^i\|_F^2 + \frac{\lambda_P}{2}\|P^i\|_F^2, \tag{1}$$

where I_j^i acts as an indicator that equals to 1 if R_j^i is observed, and 0 otherwise. λ_u and λ_p are the regularization coefficients to control the degree of regularization, and $\|\cdot\|_F^2$ denotes the Frobenius norm[19]. We employ the stochastic gradient descent method (SGD)[20] to train our model. For each successful rate from user i to requested peer j, we have the following pairwise loss function:

$$\ell\left(U^i, P_j^i\right) = \frac{1}{2}\left(R_j^i - U^i\left(P_j^i\right)^T\right)^2 + \frac{\lambda_u}{2}\left\|U^i\right\|_2^2 + \frac{\lambda_p}{2}\left\|P_j^i\right\|_2^2, \tag{2}$$

where $\|\cdot\|_2^2$ denotes the Eulerian norm. Instead of directly minimizing $\mathcal{L}$, SGD relaxes to minimize the pairwise loss function ℓ [21]. We compute gradients of U^i and P_j^i respectively:

$$\frac{\partial \ell}{\partial U^i} = \left(U^i\left(P_j^i\right)^T - R_j^i\right)P_j^i + \lambda_u U^i \tag{3}$$

$$\frac{\partial \ell}{\partial P_j^i} = \left(U^i\left(P_j^i\right)^T - R_j^i\right)U^i + \lambda_p P_j^i \tag{4}$$

Then the local matrix U^i and P^i would update using Eq.(5) and Eq.(6) respectively:

$$U^i = U^i - \alpha \cdot \frac{\partial \ell}{\partial U^i} \tag{5}$$

$$P^i_j = P^i_j - \alpha \cdot \frac{\partial \ell}{\partial P^i_j}, \tag{6}$$

where α is the learning rate of our model.

The local training will perform for a certain number of iterations to reduce the rounds of communication needed between the server and users. After local updates, the user computes gradients for P^i and sends them to the server. To distinguish the global model from the local model, we use $\hat{P}$ to denote the global model and P^i to denote the local model of user i. The gradients generated by the local training of user i are denoted as:

$$g^i = \hat{P} - P^i \tag{7}$$

The central server aggregates the gradients uploaded by users and updates the global model $\hat{P}$:

$$\hat{P} = \hat{P} - \frac{1}{N} \sum_{i=1}^{N} g^i \tag{8}$$

After the global model has been updated, the central server broadcasts it to users for the next training round. During the training process, the server keeps providing the global model for all the users to download. The training rounds and model updates will keep going until the model becomes converged. As a result, all the users can gain an effective blockchain reliability prediction model without sharing their raw data, which will achieve the preservation of private data.

3.3 Gradient Compression

To protect users' privacy, we transmit the gradients for model updating instead of users' raw data during the training process. However, sharing gradients is considered insecure in federated learning. For example, malicious third parties can deduce the user's raw data from the shared gradients[11]. To enhance the privacy preservation of our model and defend against gradient attacks, we proposed gradient compression with momentum correction.

Gradient Sparsification Gradient compression only uploads the important gradients which are larger than a threshold[17]. Due to the uncertainty of data, it is difficult to set a fixed threshold. Thus, we set a sparsity to choose the larger gradients instead of a threshold. We assume that the gradient matrix generated by user i is g^i, and set a sparsity $ratio \in (0, 1]$. The proportion of uploaded gradients to the overall number is the $ratio$, whose values are all larger than the gradients that do not need to be uploaded. Especially, when $ratio$ fetches 1, all the gradients will be uploaded.

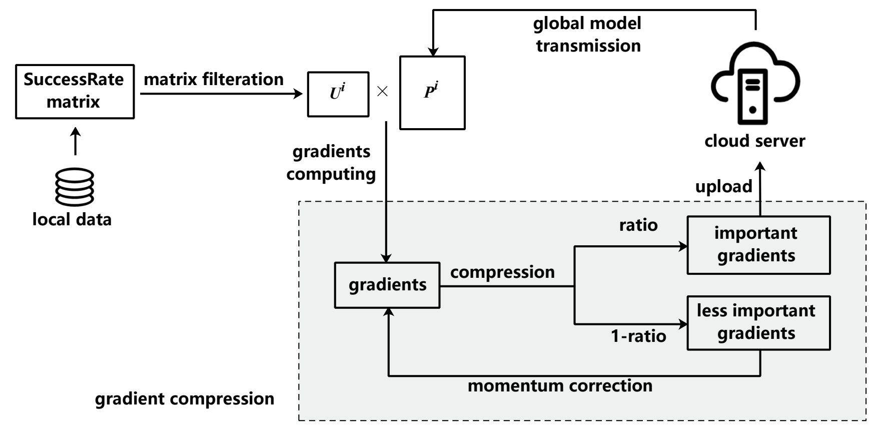


Fig. 2. FL-MFGM model training for user i

Momentum correction With the increasing of sparsity, the model becomes tougher and tougher to converge[17], because gradient sparsification only retains the important gradients. To solve this problem, we design a momentum correction method. The gradients which are not available to be uploaded in the previous iteration will be saved on the local device and accumulated with the newly gradients using momentum correction method. We perform momentum correction locally before gradient sparsification. The momentum correction method in iteration t is shown in Eq.(9) and Eq.(10).

$$u^{(i,t)} = m \cdot u^{(i,t-1)} + g^{(i,t)},\ \ v^{(i,t)} = v^{(i,t)} + u^{(i,t)},\ \ g^{(i,t)} = \text{Sparsification}\left(v^{(i,t)}\right), \tag{9}$$

where m is the momentum coefficient, $u^{(i,t)}$ denotes the result of momentum correction, $g^{(i,t)}$ is the gradients of this iteration, $v^{(i,t)}$ is used for subsequent sparsification and accumulation, and $Sparsification(v^{(i,t)})$ is for gradient compression. The details of gradient compression is shown in Fig. 2. After determining the gradients to be uploaded, $v^{(i,t)}$ will store the gradients that have not been uploaded:

$$v^{(i,t)} = v^{(i,t)} - g^{(i,t)} \tag{10}$$

Fig. 2 is mainly a more detailed explanation of the model training. The "gradient compression" part shows the workflow of the proposed gradient compression method in one iteration. Based on the existing SuccessRate matrix and the global model, we use matrix factorization to update model and compute gradients. We perform gradient compression: According to the sparsity, the larger gradients will be uploaded, and the remaining will be saved locally and accumulated with the newly computed gradients by the momentum correction method. As to reduce the times of communication, the local model training and update will perform a certain of rounds and then compute gradients to upload. After the central server updates the model and transmits it to user i, user i can update the training model and move on to the next iteration. Model training and communication between the user and the central server will keep on until the model converges.

3.4 FL-MFGM Algorithm

The pseudo code of our FL-MFGM is shown in Algorithm 1.

Algorithm 1 FL-MFGM Algorithm

```
Input:
    The dataset and parameters of training: $\{R, N, iteration, m, ratio\}$
Output: A convergence model $P$
 1: Divide R into a training set and a test set $\{trainMatrix, testMatrix\}$
 2: Initialize the user matrix $U^i$ and peer matrix $P^i$ for user $i$
 3: $u \leftarrow 0, v \leftarrow 0$
 4: for $t = 1$ to $iteration$ do
 5:     for $i = 1$ to $N$ do
 6:         $g^{(i,t)} \leftarrow 0$
 7:         $U^i, \quad P^i \leftarrow \mathrm{UserUpdate}\left(U^i, P^i, \mathrm{R}^i\right)$
 8:         $g^{(i,t)} \leftarrow P - P^i$
 9:         $u^{(i,t)} \leftarrow m \cdot u^{(i,t-1)} + g^{(i,t)}$
10:         $v^{(i,t)} \leftarrow v^{(i,t-1)} + u^{(i,t)}$
11:         $g^{(i,t)} \leftarrow \mathrm{Sparsification}\left(v^{(i,t)}, ratio\right)$
12:         $v^{(i,t)} \leftarrow v^{(i,t)} - g^{(i,t)}$
13:     end for
14:     $\mathrm{P} \leftarrow \frac{1}{N} \sum_{i=1}^{N} g^{(i,t)}$
15: end for
16: return $P$
```

Specially, the iterative updating process of the global model decomposes into two parts that are performed on the user side (Line 5 $\sim$ 13) and the server side (Line 14), respectively. The global model updating will keep on until convergence. Especially, the model updating operations on the user side are defined as a function $UserUpdate\left(U^i, P^i, \mathrm{R}^i\right)$. After the local model updating, users will compute the gradients which are required for the model updating. Note that for newly computed gradients, we first accumulate them with the remaining gradients of the previous iteration by momentum correction (Line 9 $\sim$ 10) and then conduct gradient sparsification (Line 11). The operations of gradient sparsification are defined as a function $Sparsification\left(v^{(i,t)}, ratio\right)$, according to the details described in the gradient sparsification part in Section 3.3. Another important point is that the user would save the remaining gradients locally after gradient sparsification (Line 12). The uploaded gradients from users will be aggregated on the server side and used for the global model updating (Line 14). As such, our model can protect users' privacy as it enables users to keep their raw data on device and compress the uploaded gradients.

4 Experiments and Analysis

In this section, sufficient experiments are conducted in real-world datasets to answer the following questions:

Q1: Is our proposed approach effective for blockchain reliability prediction?

Q2: What is the effect of gradient compression and momentum correction on prediction accuracy?

Q3: What is the impact of different sparsity on prediction accuracy?

Our experiments were conducted on an AMD Ryzen 5 3600 6-core processor and 8GB of RAM, running on Python 3.7 and windows 10 (64-bit). In the following subsections, we will introduce the experimental settings, and then conduct experiments and analysis according to the above three questions.

4.1 Experiment Settings

Dataset and parameter setting In this paper, we use the real dataset proposed by Zheng et al.[4]. It is a 100 × 200 SuccessMate matrix between 100 blockchain users and 200 blockchain peers. It is worth noting that, the successful request rate range from 0 to 1 and the higher successful request rate indicates that peer j is more reliability to user i. We use *MaxRTT* to indicate the maximum tolerable request round-trip time and use *MaxBlockBack* to indicate the maximum tolerable return block backward to the latest block[4]. We set four different combinations of *MaxBlockBack* and *MaxRTT* for our experiments: <*MaxBlockBack=0, MaxRTT=1000*>, <*MaxBlockBack=12, MaxRTT=1000*>, <*MaxBlockBack=12, MaxRTT=2000*>, <*MaxBlockBack=100, MaxRTT=5000*>, and the density of the training matrix is set as *Density = 30%, 50%, 65%, 95%.*

Evaluation Metric We use the root mean square error (RMSE) to measure the difference between the predicted and real values. A smaller RMSE value represents higher prediction accuracy. The formula for calculating the root mean square error is:

$$RMSE = \frac{\sum_{(i,j,R_{i,j})\in T} \left|\hat{R}_{i,j} - R_{i,j}\right|}{|T|}, \tag{11}$$

where $R_{i,j}$ is the known real record value, which indicates the success rate of user i to the peer j, $\hat{R}_{i,j}$ denotes the success rate predicted by FL-MFGM, T indicates the records in training set.

4.2 Prediction Accuracy Comparison (RQ1)

To verify the prediction accuracy of FL-MFGM, we compare it with some baseline methods for reliability prediction: the user similarity-based prediction method UPCC[22], the item similarity-based prediction method IPCC[23], the mixed user and item similarity-based prediction method UIPCC[24], and the mixed block information-based prediction method H- BRP[4]. We set the sparsity of FL-MFGM is 99% which means users can only upload the top 1% of the large gradients. The results are shown in Table 1.

From Table 1, we can observe that RMSE of H-BRP is the lowest among the baseline method and the performance of our FL-MFGM model is even better than H-BRP. Concretely, for <*MaxBlockBack=12, MaxRTT=1000*>, the RMSE of FL-MFGM is 27.9%∼51.4% lower than that of H-BRP at different matrix densities, which means

Table 1. Comparison of RMSE among FL-MFGM and baseline approaches

Parameter	Method	Density=30%	Density=50%	Density=65%	Density=95%
MaxBlockBack=0 MaxRTT=1000	UPCC	0.2823	0.2791	0.2769	0.2758
	IPCC	0.0821	0.0777	0.0764	0.0748
	UIPCC	0.0851	0.0806	0.0791	0.0773
	H-BRP	0.0791	0.0716	0.0711	0.0670
	FL-MFGM	**0.0637**	**0.0565**	**0.0399**	**0.0397**
	Improve.(%)	19.5%	21.1%	43.9%	40.7%
MaxBlockBack=12 MaxRTT=1000	UPCC	0.3627	0.3591	0.3566	0.3559
	IPCC	0.1009	0.0949	0.0919	0.0920
	UIPCC	0.1053	0.0994	0.0963	0.0961
	H-BRP	0.1017	0.0908	0.0901	0.0834
	FL-MFGM	**0.0718**	**0.0655**	**0.0438**	**0.0466**
	Improve.(%)	29.4%	27.9%	51.4%	44.1%
MaxBlockBack=12 MaxRTT=2000	UPCC	0.3793	0.3775	0.3758	0.3760
	IPCC	0.0810	0.0785	0.0765	0.0751
	UIPCC	0.0887	0.0864	0.0845	0.0831
	H-BRP	0.0641	0.0546	0.0528	0.0451
	FL-MFGM	**0.0461**	**0.0439**	**0.0330**	**0.0328**
	Improve.(%)	28.1%	19.6%	37.5%	27.3%
MaxBlockBack=100 MaxRTT=5000	UPCC	0.4595	0.4585	0.4569	0.4574
	IPCC	0.0921	0.0887	0.0842	0.0774
	UIPCC	0.1022	0.0993	0.0954	0.0895
	H-BRP	0.0635	0.0547	0.0524	0.043
	FL-MFGM	**0.051**	**0.0478**	**0.0328**	**0.0327**
	Improve.(%)	19.7%	12.6%	37.4%	24.0%

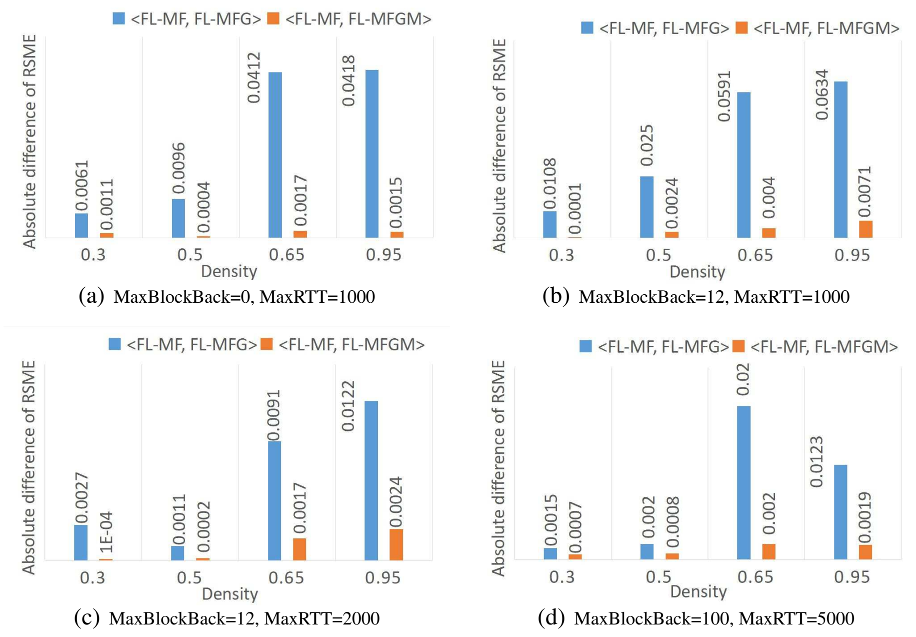
that FL-MFGM achieves 27.9%$\sim$51.4% improvement on prediction accuracy. And for <*MaxBlockBack=100, MaxRTT=5000*>, FL-MFGM achieves 12.6%$\sim$37.4% improvement on prediction accuracy. According to the results, our FL-MFGM approach significantly outperforms the other approaches over RMSE. And as the density of the training set increases, the RMSE of FL-MFGM decreases correspondingly indicating that its prediction accuracy also increases. We verify that our FL-MFGM model can achieve higher accuracy than existing approaches for blockchain reliability prediction.

4.3 Effect of Gradient Compression and Momentum Correction (RQ2)

Gradient compression is an important method to defend against gradient attacks. However, it would reduce the accuracy of the prediction due to the reduction of the uploaded gradients. As discussed in Section 3.3, we recover the accuracy by momentum correction. To study the performance of our gradient compression method which contains gradient sparsification and momentum correction, we compare the prediction accuracy of FL-MFGM with two methods, named FL-MF and FL-MFG respectively. FL-MF and FL-MFG are both blockchain reliability prediction approaches based on federated matrix factorization. Compared with FL-MFGM, FL-MF performs without gradient compression or momentum correction, and FL-MFG performs gradient compression without momentum correction. And we set sparsity, $ratio = 99\%$. The experimental results are shown in Table 2.

Table 2. Effect of gradient compression and momentum correction on prediction accuracy

Parameter	Method	Density=30%	Density=50%	Density=65%	Density=95%
MaxBlockBack=0 MaxRTT=1000	FL-MF	0.0626	0.0569	0.0382	0.0382
	FL-MFG	0.0687	0.0665	0.0794	0.0800
	FL-MFGM	**0.0637**	**0.0565**	**0.0399**	**0.0397**
MaxBlockBack=12 MaxRTT=1000	FL-MF	0.0717	0.0679	0.0398	0.0395
	FL-MFG	0.0825	0.0929	0.0989	0.1029
	FL-MFGM	**0.0718**	**0.0655**	**0.0438**	**0.0466**
MaxBlockBack=12 MaxRTT=2000	FL-MF	0.0462	0.0441	0.0313	0.0304
	FL-MFG	0.0489	0.0452	0.0404	0.0426
	FL-MFGM	**0.0461**	**0.0439**	**0.0330**	**0.0328**
MaxBlockBack=100 MaxRTT=5000	FL-MF	0.0503	0.0470	0.0308	0.0308
	FL-MFG	0.0518	0.0490	0.0508	0.0431
	FL-MFGM	**0.0510**	**0.0478**	**0.0328**	**0.0327**



(a) MaxBlockBack=0, MaxRTT=1000

(b) MaxBlockBack=12, MaxRTT=1000

(c) MaxBlockBack=12, MaxRTT=2000

(d) MaxBlockBack=100, MaxRTT=5000

Fig. 3. Performance comparison of gradient compression and momentum correction on prediction. Note: <*FL-MF, FL-MFG*> means the absolute value of the difference between the RMSE of FL-MF and FL-MFG. <*FL-MF, FL-MFGM*> means the absolute value of the difference between the RMSE of FL-MF and FL-MFGM.

From Table 2, we find that FL-MF without gradient compression achieves highest accuracy in most cases. The prediction accuracy of FL-MFGM is very close to that of FL-MF. It is worth noting that, the RMSE values of FL-MFGM are lower than those of FL-MF when $Density = 50\%$ and *MaxBlockBack* $= 12$, which means that the prediction accuracy of FL-MFGM can be higher than that of FL-MF. With the increase of $Density$, the RMSE values of FL-MF and FL-MFGM show a tendency of decreasing which means the accuracy is improving, while the accuracy of the FL-MFG method without a decreasing trend is the lowest among these three methods. It reflects that FL-MFGM and FL-MF are much more effective than FL-MFG.

To observe the effect of compression and momentum correction on the prediction accuracy more clearly, we compute the absolute value of the RMSE difference between FL-MF and FL-MFG and also compute it for FL-MF and FL-MFGM. The results are shown in Fig. 3.

From Fig. 3, we can find that the difference between FL-MF and FL-MFG becomes larger with the increasing *Density*, as well as between FL-MF and FL-MFGM. While the maximum RMSE difference between FL-MF and FL-MFGM is 0.0071 and the maximum difference between FL-MF and FL-MFG is 0.0634. Notably, with a fixed *Density*, *MaxRTT*, and *MaxBlockBack*, the difference between FL-MF and FL- MFGM is smaller than that between FL-MF and FL-MFG. It indicates that FL-MFGM which employs gradient compression with momentum correction can approach the effect of FL-MF with a very small difference (roughly less than 0.01 difference in RMSE values), while the RMSE values of FL-MFG without momentum correction are larger than those of FL-MF.

In view of the above experiments, we can conclude that gradient compression would reduce prediction accuracy while momentum correction can rescover prediction accuracy. Therefore, our FL-MFGM model can achieve privacy preservation without losing accuracy.

4.4 Impact of Sparsity (RQ3)

Sparsity plays an important role in our gradient compression method. As discussed in Section 3.3, gradient sparsification can protect privacy from gradient attacks while it would lead to the loss of prediction accuracy. To discuss the impact of sparsity on the prediction accuracy of our FL-MFGM model, in this experiment, we set *MaxBlockBack* $=$ 12, *MaxRTT* $= 2000$ and set the sparsity ratio = 0, 10%, 30%, 65%, 99%. The results are shown in Table 3.

Table 3. Impact of sparsity ratio on model performance

Sparsity ratio	Density=30%	Density=50%	Density=65%	Density=95%
0	0.0462	0.0441	0.0313	0.0304
10%	0.0460	0.0442	0.0337	0.0320
35%	0.0462	0.0440	0.0342	0.0331
65%	0.0465	0.0440	0.0336	0.0326
99%	0.0464	0.0442	0.0340	0.0330

From Table 3, we can observe that when the sparsity ratio is fixed, the RMSE of FL-MFGM shows a gradually decreasing trend with *Density* increasing. When we keep *Density* constant, the prediction error is not significantly different when sparsity ratio is changed. Specifically, when *Density* $= 65\%$, the maximum RMSE difference between different ratios obtains the largest value (about 0.0029), while *Density* $= 10\%$, the maximum RMSE difference between different ratios obtains the smallest value (about

0.0002). The results show that the variation of sparsity has little effect on the prediction accuracy of our model. It turns out that FL-MFGM can achieve high accuracy at different sparsity even when the degree of sparsity ratio reaches 99%.

5 Conclusion and Future Work

In this paper, we propose a privacy-preserving and high-accuracy blockchain reliability prediction approach based on federated matrix factorization with gradient compression and momentum correction. Firstly, we propose a matrix factorization model based on federated learning, which enables users to learn an effect global prediction without sharing private data. Secondly, we use gradient compression method to defend against gradient attacks by sparsifying the gradient matrix. Thirdly, we design the gradient compression with momentum correction which fixes the accuracy degradation caused by compression. Finally, the experimental results verify that our approach can improve prediction accuracy while achieving privacy preservation. And the proposed approach achieves a high gradient compression rate and reduces the communication bandwidth without compromising the prediction accuracy.

In our future research, we plan to improve our model performance by considering the impact of other block information, such as block height and block hash. We will also explore ways to improve the model's ability to resist attacks. In addition, our proposed approach is based on the condition of trusted users and trusted servers, and further work will consider the credibility of users and servers.

Acknowledgments. This research was financially supported by 2021 Guangdong Province Special Fund for Science and Technology ("major special projects + task list") Project (No.STKJ2021201), Guangdong Province Basic and Applied Basic Research Fund (No.2021A1515012527) and in part by 2020 Li Ka Shing Foundation Cross-Disciplinary Research Grant (No. 2020LKSFG08D).

References

1. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review p. 21260 (2008)
2. Bhutta, M.N.M., Khwaja, A.A., Nadeem, A., Ahmad, H.F., Khan, M.K., Hanif, M.A., Song, H., Alshamari, M., Cao, Y.: A survey on blockchain technology: Evolution, architecture and security. IEEE Access **9**, 61048–61073 (2021)
3. Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H., Capkun, S.: On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. pp. 3–16 (2016)
4. Zheng, P., Zheng, Z., Chen, L.: Selecting reliable blockchain peers via hybrid blockchain reliability prediction. arXiv preprint arXiv:1910.14614 (2019)
5. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for federated learning on user-held data. arXiv preprint arXiv:1611.04482 (2016)
6. Liu, J., Meng, X.: Survey on privacy-preserving machine learning. Journal of Computer Research and Development **57**(2), 346 (2020)

7. Lu, Y., Huang, X., Dai, Y., Maharjan, S., Zhang, Y.: Differentially private asynchronous federated learning for mobile edge computing in urban informatics. IEEE Transactions on Industrial Informatics **16**(3), 2134–2143 (2019)
8. Ren, J., Wang, H., Hou, T., Zheng, S., Tang, C.: Federated learning-based computation offloading optimization in edge computing-supported internet of things. IEEE Access **7**, 69194–69201 (2019)
9. Wang, X., Han, Y., Wang, C., Zhao, Q., Chen, X., Chen, M.: In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning. IEEE Network **33**(5), 156–165 (2019)
10. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. pp. 1273–1282. PMLR (2017)
11. Zhu, L., Liu, Z., Han, S.: Deep leakage from gradients. Advances in Neural Information Processing Systems **32** (2019)
12. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine **37**(3), 50–60 (2020)
13. Xu, J., Zhuang, Z., Wang, K., Liang, W.: High-accuracy reliability prediction approach for blockchain services under BaaS. In: International Conference on Blockchain and Trustworthy Systems. pp. 648–660. Springer (2020)
14. Aono, Y., Hayashi, T., Wang, L., Moriai, S., et al.: Privacy-preserving deep learning via additively homomorphic encryption. IEEE Transactions on Information Forensics and Security **13**(5), 1333–1345 (2017)
15. Strom, N.: Scalable distributed DNN training using commodity GPU cloud computing. In: Sixteenth Annual Conference of the International Speech Communication Association (2015)
16. Tao, Z., Li, Q.: {eSGD}: Communication Efficient Distributed Deep Learning on the Edge. In: USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18) (2018)
17. Lin, Y., Han, S., Mao, H., Wang, Y., Dally, W.J.: Deep gradient compression: Reducing the communication bandwidth for distributed training. arXiv preprint arXiv:1712.01887 (2017)
18. Chai, D., Wang, L., Chen, K., Yang, Q.: Secure federated matrix factorization. IEEE Intelligent Systems **36**(5), 11–20 (2020)
19. Salakhutdinov, R., Mnih, A.: Probabilistic Matrix Factorization. In: Proceedings of the 20th International Conference on Neural Information Processing Systems. pp. 1257–1264 (2007)
20. Shapiro, A., Wardi, Y.: Convergence analysis of gradient descent stochastic algorithms. Journal of optimization theory and applications **91**(2), 439–454 (1996)
21. Zhu, J., He, P., Zheng, Z., Lyu, M.R.: Towards online, accurate, and scalable qos prediction for runtime service adaptation. In: 2014 IEEE 34th International Conference on Distributed Computing Systems. pp. 318–327. IEEE (2014)
22. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. arXiv preprint arXiv:1301.7363 (2013)
23. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web. pp. 285–295 (2001)
24. Zheng, Z., Lyu, M.R.: Collaborative reliability prediction of service-oriented systems. In: 2010 ACM/IEEE 32nd International Conference on Software Engineering. vol. 1, pp. 35–44. IEEE (2010)