A Secure and Verifiable Outsourcing Scheme for Machine Learning Data

Cheng Li, Li Yang(✉), and Jianfeng Ma

Xidian University, Xi'an 710071, Shaanxi, China
15991727802@sina.cn, yangli@xidian.edu.cn, jfma@mail.xidian.edu.cn

Abstract. In smart applications, such as smart medical devices, in order to prevent privacy leaks, more data needs to be processed and trained locally or near the local end. However, the storage and computing capabilities of smart devices are limited, so some computing tasks need to be outsourced; concurrently, the prevention of malicious nodes from accessing user data during outsourcing computing is required. Therefore, this paper proposes EVPP (efficient, verifiable, and privacy-preserving), a machine learning method based on a collaboration of edge computing devices. In this solution, the computationally intensive part of the model training process is outsourced. Meanwhile, a random encryption perturbation is performed on the outsourced training matrix, and verification factors are introduced to ensure the verifiability of the results. In addition, when a malicious service node is found, verifiable evidence can be generated to build a trust mechanism. Through the analysis of theoretical and experimental data, it can be shown that the scheme proposed in this paper can effectively use the computing power of the equipment.

Keywords: Machine learning · Edge computing · Privacy-preserving · Mobile devices · Outsourced computing

1 Introduction

With the development of the Internet of Things, 5G communication networks, AI technology and the construction of intelligent facilities that promote the development of mobile devices, connected cars and smart wearable devices have been developed. Concurrently, a large amount of data has also been generated that is processed by different companies and different servers. Data are collected on various cloud computing platforms for various data analyses and mining. It is expected that by 2020, an average person will generate approximately 250 million bytes of data per day [1], which may come from mobile phone sensors, smart wearable devices, and so on.

We would like to thank the anonymous reviewers for their careful reading and useful comments. This work was supported by the National Key Research and Development Project (2017YFB0801805), the National Natural Science Foundation of China (61671360).

D. Wang et al. (Eds.): SPNCE 2020, LNICST 344, pp. 301–320, 2021.
https://doi.org/10.1007/978-3-030-66922-5_21

Abundant data require intelligent terminal processing, calculations, storage, etc. [2]; however, the storage and computing capabilities of smart devices are limited, and more data is being continuously collected, transmitted, and calculated. The transmission capabilities and data storage capabilities have become increasingly powerful, but in the face of a geometrically increasing amount of data, it is also difficult to meet users' requirements for data processing capabilities and transmission quality. Furthermore, the transmission of these data in the network will definitely apply a great pressure to the network.

The traditional centralized computing architecture based on a cloud centre [3–5] has been unable to meet the requirements of modern devices and applications for low latency, high efficiency, and low cost applications. In some special scenarios, such as smart healthcare [6,7], identity recognition [9], smart homes [10], all have high requirements on time and accuracy. Transferring data to cloud servers will raise latency, but running artificial intelligence algorithms such as machine learning and deep learning locally will bring an additional consumption of computing and power to the device.

Therefore, the application of edge computing [11] technology is used to outsource the calculation of data to edge nodes that are close and satisfy the computing power to reduce the computing and processing pressure of the device and reduce the delay in data transmission. At the same time, to reduce the pressure of network transmission, some data needs to be processed locally, such as the basic operations including simple data cleaning and partial data processing; simultaneously, in order to avoid a lengthy time, it is necessary to seek auxiliary computing nodes in the model training process on the near device side due to the limitation of the network with a high delay and high network pressure.

When applying edge computing to model training for local devices and nodes, data security and privacy issues cannot be ignored [12,13]. For example, in the application scenarios of user data collection such as smart medical devices and smart bracelets, the local device continuously accesses the user's geographic location, physical characteristics (including heart rate, stride, voiceprint, and other characteristics) or medical characteristics, which is apart from the collection and processing data by these devices that include a large number of user's privacy characteristics. As mentioned earlier, the local device's computing and processing capabilities cannot process and return results in a timely manner. To avoid the leakage of users' private data and to ensure that the calculation results are obtained in a timely and effective manner, advancements are needed.

Therefore, as shown in Fig. 1, this paper uses edge computing to solve data processing and computing problems in the construction of intelligent facilities, such as the Internet of Things, ensuring a high availability of data, effectively reducing network pressure and network delays. Concurrently, it will combine existing artificial intelligence and machine learning algorithms; the machine learning algorithm training process is "local + edge" for effective and safe training, and finally, the machine learning algorithm model is obtained. EVPP (efficient, verifiable, and privacy-preserving) is proposed: an outsourcing algorithm for device-to-edge machine learning model training. This algorithm is a

good compromise between privacy-preserving and execution efficiency. For example, deep learning is adjusted and compressed to reduce complexity, and high-complexity computing tasks are deployed at the edge of the network. The device only needs to perform some relatively simple operations to complete the entire model training process to appropriately reduce the network time delay via the effective use of computing resources, While to ensure the security of the data and the correctness of the calculation results, the outsourced data is encrypted and replaced, and the existence of malicious service nodes is taken into consideration to ensure the correctness of the calculation results of rational computing nodes. This paper also adds a trust mechanism to further increase the security of the system.

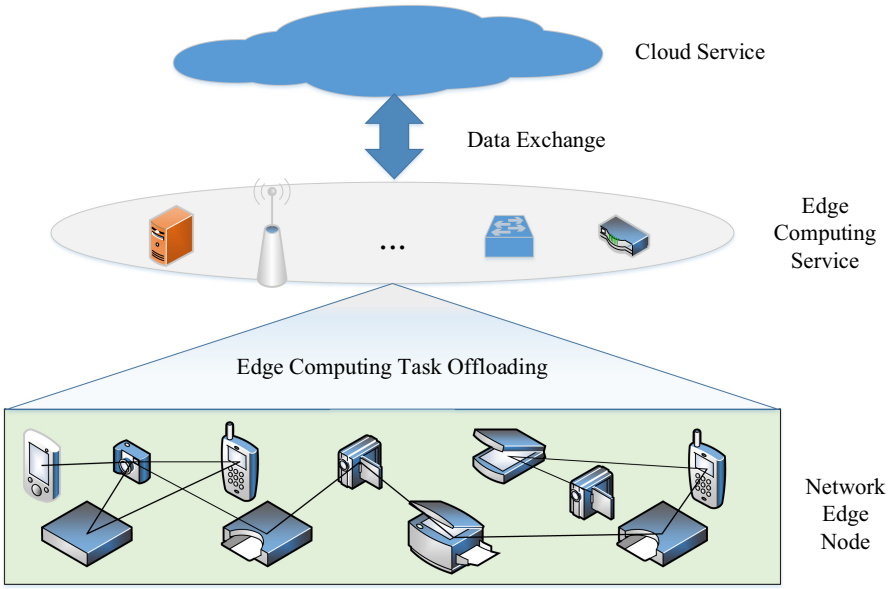

Fig. 1. Schematic diagram of data processing and calculation using edge computing.

The contributions of this article are as follows:

1. To solve the high calculation and high storage pressure caused by local machine learning algorithms on the device (especially mobile devices), a method called EVPP is proposed to outsource the computing part of the training process.
2. To solve the problems of high latency and network transmission pressure in outsourced computing, a near-local outsourcing algorithm is proposed in conjunction with edge computing, and concurrently, a cryptographic device is designed to solve the privacy and security problems brought by data outsourcing, a random matrix calculation scheme is introduced to randomly perturb the calculation data.

3. To prevent the dishonest outsourced computing nodes from affecting the training process, a trust mechanism with the arbitration function is proposed, which can guarantee the correctness of the calculation results of rational outsourced computing nodes.

The organizational structure of this paper is as follows: the first section will briefly introduce the related research, the second section will further describe the problems and challenges studied in this paper, and the third section will discuss the scheme and its algorithms in detail. The fourth section will focus on the security and performance proofs of the proposed goals in this paper. In the last section, the scheme will be summarized, and future research directions will be discussed.

2 Related Work

Smarter healthcare [6–8], urban transportation [14,15], connected cars [16], social networks [17] and other scenarios are increasingly applying machine learning algorithms for prediction and analysis. In these application scenarios, the data are outsourced to the cloud, which has a strong computing power, so that resource-constrained devices can use the cloud centre to complete various complex computing tasks and better serve users [18].

However, with the advent of the Internet of Everything, the edge of the network no longer generates abundant data, which is not suitable for processing in the cloud centre from the perspective of computing or network transmission. Applying edge computing technology is a good choice to solve this problem. At the same time, machine learning algorithms with a higher computational complexity can be applied in the cloud, but it is not realistic to apply them directly on the edge of the network. Zhang et al. [19] proposed the "OpenEI" open-edge intelligent framework to "marginalize" the model training process.

At the edge of the network, in order to meet the requirements of delay and efficiency, complex computing tasks must be outsourced to edge nodes with strong computing capabilities, but these nodes are often untrustworthy. To ensure the security and privacy of the data, the data must be encrypted and calculated in the ciphertext domain. This work has become more mature in cloud computing. For example, secure multiparty computing can be applied [20–22], including homomorphic encryption [23], differential privacy [24], and attribute-based encryption [12], but these solutions are not friendly to edge devices with low computing and storage capabilities.

Since determinant and matrix calculations are widely used in the fields of science and engineering, especially in various AI algorithms, there have been many studies on outsourcing calculations of determinants and matrixes [25–28]. Salinas et al. [25], a large-scale deterministic secure outsourcing computing solution was proposed. The client can effectively verify the correctness of the calculation results of the outsourced data. Chen et al. [29], a scheme for scrambling the original matrix data using diagonal matrix multiplication was proposed to ensure the security of the data; subsequently, it was improved in Zhou et al. [30],

and it must be further improved in terms of security and result verification. Hu et al. [31], a matrix outsourcing inversion matrix scheme that can be applied to cloud computing and other scenarios was proposed, which effectively reduces the computational complexity of the client.

3 Problem Description and Research Goals

3.1 Research Goals and Challenges

In this paper, to better solve the computing and privacy issues of edge devices and ensure the security and accuracy of computing, four research goals are proposed.

- Privacy: These data contain tremendous user identity information, privacy data, etc. The collection and processing of these data are extremely prone to leakage of information. Therefore, the outsourcing of data computing needs to ensure the privacy of the data.
- Verifiability: Due to the instability of the system, network or computing nodes, the nodes to which the data are outsourced should be assumed to be incompletely trusted or even malicious. They may steal or peek at the user's data; furthermore, the operation may not be performed in accordance with the protocol at all, and the wrong calculation result for the user is returned, leading to the failure of the entire data training. Therefore, the calculation result of the data should be verifiable.
- High efficiency: In the whole process, the user's calculation amount in the outsourced calculation process should be lower than the entire operation performed by the device itself, otherwise the outsourced operation will be useless.
- Accuracy: This requires the design of the entire system to ensure that the calculation results can be guaranteed under the premise of the correct operation at each stage.

3.2 System Model

To ensure the security and availability of the system and achieve the research goals proposed in Sect. 3.1, this paper designs an outsourcing model training scheme based on edge computing, as shown in Fig. 2:

In the solution, the system is divided into three layers (the cloud computing problem is not considered here), which are the sensor node layer (or data acquisition layer), the edge node layer, and the edge service layer.

The sensor node layer is responsible for collecting data, but because of its poor computing and storage capabilities, the collected data cannot be calculated, organized, and stored. The collected data must be transmitted to the edge node layer of other networks for processing. For example, smart phones are implanted with several sensors, and these sensors transmit data to the mobile phone's computing unit for processing and storage. To ensure the availability of data and the security of users' data, the edge node layer mainly guarantees the data cleaning,

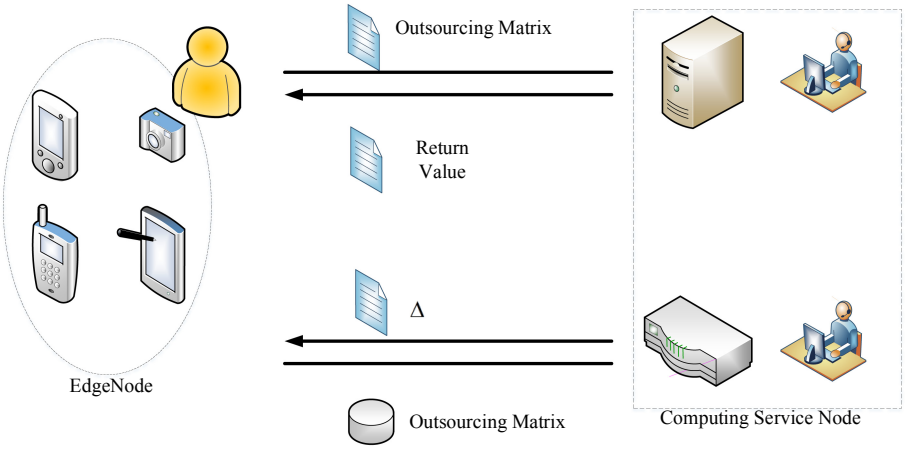

Fig. 2. Training model of the outsourcing model based on edge computing.

calculation, and storage tasks of sensor devices. Concurrently, in order to ensure the timeliness and accuracy of calculations, it also performs some calculation tasks to offload the edge service layer. The main task of the edge service layer is to assist the devices in the edge node layer to perform collaborative computing. However, due to the difficulty in ensuring security, there are the following risks: (1) the layer may peep and steal data, leading to information leakage; (2) it may not complete the computing task as agreed, causing the computing task of the edge node to fail.

As shown in Fig. 2, the solution proposed in this paper includes edge service nodes that need to outsource computing tasks and edge service nodes that assist edge nodes to outsource the computing tasks. Edge service nodes include two parts that assist users in key generation calculations, and an edge server that assists users in computing tasks.

3.3 Linear Regression and Gradient Descent

There are many optimization and learning algorithms in machine learning and deep learning. Most of these algorithms are based on matrix calculations and training models. During the training phase, the device performs a large number of matrix multiplications and additions. Through the analysis of the corresponding algorithm, it is not difficult to find that the number of multiplication operations is higher than that of addition operations. At the same time, the computational complexity of multiplication is higher than addition. Linear regression and gradient descent methods are more common methods. Therefore, in order to facilitate the description of the scheme, this paper uses gradient descent to optimize the model in the linear regression problem and finally obtain the training model. At the same time, describe the main ideas of the scheme.

Given the n sample set (X, Y) where the i-th sample X_i contains d features, that is, $X_i = (x_1, x_2, ..., x_d)$, adjust the objective function $h(X_i) = x_1 \cdot w_1 + x_2 \cdot w_2 + ... + x_d \cdot w_d = X_i \cdot w^T$, where $w = (w_1, w_2, ..., w_d)$.
Adjust the parameters through the training process to yield the appropriate to make $J_i(w) = (\frac{1}{2n}\sum_{i=1}^{n}(h(X_i - Y_i)^2))$ the smallest, that is, $Y_i \approx h(X_i)$.

The gradient descent method is widely used in machine learning to solve optimization problems. When targeting linear regression problems, for the j-th feature of X_i, the weight is $w_j = w_j - \alpha\frac{\partial J_i(w)}{\partial w_i} = w_j - \alpha\frac{1}{n}\sum_{i=1}^{n}(f(X_i) - Y_i)X_i^j$, and by representing vectors as a matrix, w can be expressed as:

$w := w - \alpha X^T \times (X \times w - Y)$ [21].

Among them, α is the learning rate or step size, which is a fixed value, and this parameter determines the convergence degree of the algorithm; Y is a vector of $n \times l$ dimensions, that is, a given data tag set.

In the gradient descent method, because all the samples are used for training at one time, it will cause pressure on the memory and calculations, so $|B|$ samples are selected for small batch training. In the formula, $|B|$ is the amount of data, and $w := w - \alpha\frac{1}{|B|}X_B^T \times (X_B \times w - Y_B)$ [21].

Therefore, this paper decomposes the matrix calculations with abundant calculations. Among them, suppose

$$\begin{aligned} \Delta &= X_B^T \times (X_B \times x - Y_B) \\ &= X_B^T \times (X_B \times x + (-Y_B)) \\ &= X_B^T \times (X_B \times x) + X_B^T \times (-Y_B) \end{aligned} \tag{1}$$

That is, the update formula can be expressed as:

$$w := w - \alpha\frac{1}{|B|}\Delta \tag{2}$$

3.4 System Framework

The main idea of our solution is shown in Fig. 3, which includes the following five parts:

Step 1 (outsourced data generation algorithm): The client constructs a reversible matrix D for scrambling the data matrix, generating a random matrix, and randomly generating a verification matrix. Concurrently, the client has a training data set. The sample set X contains n samples x_i. Each sample set can be represented as an m-dimension vector, and the tag set is represented as Y.

The client calculates the confusion matrix forms C_1, C_2, C_3, and C_4 corresponding to the sample set X and its transposed matrix X^T, tag set Y, and initialization weight matrix w, and sends the data and corresponding calculation rules $f(C')$ to the edge service layer for calculation. At the same time, the matrix verification block is calculated and saved to facilitate subsequent result verification work.

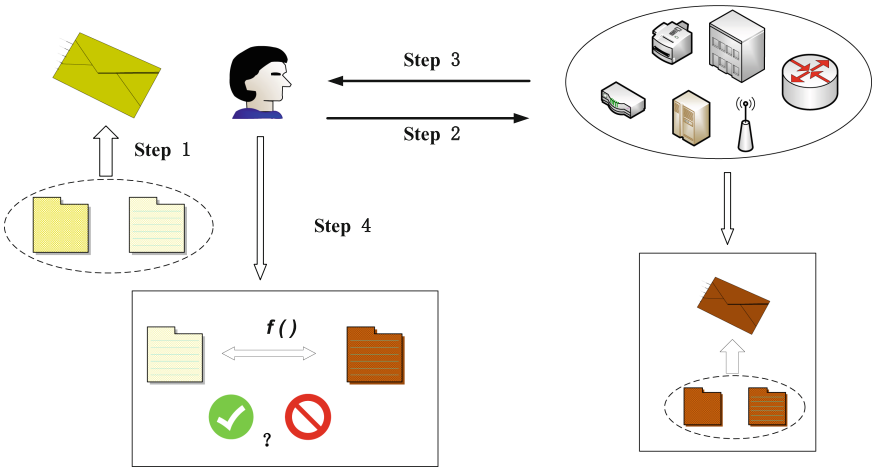

Fig. 3. The main design ideas of the scheme.

Step 2 (outsourcing data calculation algorithm): The edge service layer node outputs the calculation result Δ^* according to the outsourcing calculation rule $f(C')$ sent by the client and sends the calculated result back to the client.

Step 3 (training result generation algorithm): The client receives the calculation result Δ^* sent back by the edge server layer node and performs a recovery operation based on the information held locally to obtain the calculation result Δ. Then, the result is brought into Formula (1) and calculated, obtaining w_t. By comparison, $w_t < w_{t-1}$ indicates that the function has not reached the convergence value, the scrambling operation is performed, and return to Step 2 to continue training; $w_t > w_{t-1}$ indicates that the function has reached the convergence value, which terminates this calculation task and enters Step 5.

Step 4 (data verification algorithm): The client receives the calculation result Δ^* returned by the edge server layer node, extracts the verification matrix block V^* from it, and compares it with $V? = V^*$. When they are equal, the result indicates that the edge service layer node has performed the calculation operation correctly, otherwise it indicates that it has not faithfully calculated the outsourcing task; the client retains the test evidence, publishes the evidence and the identity of the edge service layer node, and executes Step 5 to find new computing-nodes.

Step 5 (End the calculation task): When the function reaches the convergence value, the client sends W^0 to the edge service layer node. When the edge service layer node receives the message, it knows that the calculation task is terminated, and the edge service layer node clears all relevant data. (This step may be performed in the following two situations: (1) The protocol execution process is normally completed, and (2) it is found that the edge server does not faithfully execute the calculation protocol. Therefore, it is not identified in Fig. 3).

4 System Solutions

In this section, the application scenario of EVPP is introduced in detail. The solution takes the gradient descent method as an example to achieve the four goals required by the previously described.

4.1 Encryption and Decryption Methods for Outsourced Data

In this section, we will describe the construction, encryption, and decryption processes of Formula (1) and Formula (2) in the scheme.

To ensure the security of the data and the simplicity of the result verification, the training data is encrypted:

The edge node encrypts $m \times n$ data X, $m \times l$ data Y, and $n \times l$ data w to ensure data security and then operates the matrix according to the following methods:

The edge nodes randomly generate $m \times t$ order matrixes M_1 and M_3; $n \times t$ order matrixes M_2 and M_4 ; four randomly generated t order matrixes V_1, V_2, V_3, and V_4; randomly select the diagonal matrix R; and construct the reversible matrix D. Finally, we can obtain the outsourcing matrix:

$$(X^T)'_{(n+t)\times(m+t)} = \begin{bmatrix} X^T D^{-1} & M_2 \\ 0 & V_2 \end{bmatrix} \tag{3}$$

$$w'_{(n+t)\times(i+t)} = \begin{bmatrix} R(w)^T & M_3 \\ 0 & V_3 \end{bmatrix} \tag{4}$$

$$Y'_{(n+t)\times(i+t)} = \begin{bmatrix} R(Y)^T & M_4 \\ 0 & V_4 \end{bmatrix} \tag{5}$$

$$X'_{(m+t)\times(n+t)} = \begin{bmatrix} DX & M_1 \\ 0 & V_1 \end{bmatrix} \tag{6}$$

The construction process of the invertible matrix D will be described in detail [30]:

The invertible matrix $D = D_1 + D_2$ and the matrixes D_1, D_2 and D are all square matrixes of order $m \times m$. The matrix D_1 is a random diagonal matrix, in which $a_{11} = 0$, and other diagonal position element values are randomly selected from the edge nodes. In the matrix $D_2 = P^T Q$, where $P = (p_1, p_2, ..., p_m)$ and $Q = (1, q_1, q_2, ..., q_{m-1})$. D will be shown below as an invertible matrix.

Proof. First, the matrixes D_1,D_2, and D are all square matrixes of $m \times m$.

Second, D_1 is a diagonal matrix, $D_2 = P^T Q$, and $D = D_1 + D_2$. Therefore,

$$D = \begin{bmatrix} p_1 & q_1 p_1 & q_2 p_1 & \cdots & q_{m-1} p_1 \\ p_2 & q_1 p_2 + d_2 & q_2 p_2 & \cdots & q_{m-1} p_2 \\ p_3 & q_1 p_3 & q_2 p_3 + d_3 & \cdots & q_{m-1} p_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_m & q_1 p_m & q_2 p_m & \cdots & q_{m-1} p_m + d_m \end{bmatrix} \tag{7}$$

Subtract the first row from 2 to m of the matrix D to yield

$$D = \begin{bmatrix} p_1 & q_1 p_1 & q_2 p_1 & \cdots & q_{m-1} p_1 \\ 0 & d_2 & 0 & \cdots & 0 \\ 0 & 0 & d_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & d_m \end{bmatrix} \tag{8}$$

Finally, the unit matrix I can be obtained from the above matrix through a further elementary transformation, so it can be easily obtained that the matrix D must be an invertible matrix.

The matrix inversion process can be performed with reference to [31].

Next, we will demonstrate how to ensure that the calculation results are recoverable and verifiable in the matrix.

For the formula $\Delta = X_B^T \times (X_B \times w) + X_B^T \times (-Y_B)$, the method of matrix block construction can be obtained: $\Delta' = (X^T)' \times ((X)' \times (w)') + (X^T)' \times (-Y')$. Multiply two block matrixes, for example:

$$\begin{aligned} C_2 \times C_1 &= \begin{bmatrix} X^T D^{-1} & M_2 \\ 0 & V_2 \end{bmatrix}^T \times \begin{bmatrix} DX & M_1 \\ 0 & V_1 \end{bmatrix} \\ &= \begin{bmatrix} X^T D^{-1} DX & X^T D^{-1} M_1 + M_2 V_1 \\ 0 & V_2 V_1 \end{bmatrix} \end{aligned} \tag{9}$$

It is not difficult to see from the result of the matrix multiplication that the upper left part of the matrix is equivalent to solving $X^T \times X$; the edge node can calculate $\det(V) = \det(V_1 V_2 V_3 - V_2 V_4)$ as the verification factor for the outsourced calculation, which indirectly proves the correctness of the calculation result.

4.2 Description of Scheme

In this section, Algorithm 1 and Algorithm 2 will be described according to the algorithm described in Sect. 4.1, and the initialization process of the scheme has been completed. Use $f()$ to represent the calculation rules for outsourced data. In this paper, $f()$ uses the Δ of Sect. 3.4 to define the calculation rules.

Step 1 (outsourced data generation algorithm): The client constructs a reversible matrix D for scrambling the data matrix, generates a random matrix M_1, M_2, M_3, M_4, and randomly generates a verification matrix V_1, V_2, V_3, V_4; R is a diagonal matrix. Concurrently, the client has a training data set. The sample set X contains n samples x_i. Each sample set can be represented as a d-dimension vector, and the tag set is represented as Y.

Outsource the invertible matrix D to obtain D^{-1}.

The client can obtain the sample set X and its transposed matrix X^T, the tag set Y, and the initial w confusion matrixes C_2, C_1, C_3, and C_4 through

Algorithm 1. Outsourced data generation algorithm

Input: Key k, Invertible matrix D_1, P^T, Q, Random matrixes (M_1, M_2, M_3, and M_4), Random verification matrixes (V_1, V_2, V_3, and V_4), Sample set X, Tag set Y, and the initialization vector w.

Output: Outsourcing matrix C_1, C_2, C_3, C_4.

1: Calculate $D = D_1 + D_2$;
2: Goto the inverse matrix algorithm is outsourced to obtain the inverse matrix D^{-1};
3: The edge computing node calculates the key matrix $K \leftarrow kI$;
4: Initialize the vector for scrambling (perturbation) $w' \leftarrow Kw$ and tag set $Y' \leftarrow KY$;
5: Design calculation rules $f()$;
6: According to the calculation rules, solve the verification factor $\det V$;
7: Construct the sample set $X' \leftarrow DX$ and its transpose matrix $(X^T)' \leftarrow X^T D^{-1}$;
8: **return** $C_2 \leftarrow ((X^T)'||M_2||V_2)$, $C_1 \leftarrow (X'||M_1||V_1)$, $C_3 \leftarrow (w'||M_3||V_3)$, $C_4 \leftarrow (Y'||M_4||V_4)$;

Algorithm 2. Outsourcing data calculation algorithm

Input: Key k, Matrix C_1, C_2, C_3, C_4, and Calculation Rule $f()$.

Output: Calculation result Δ^*.

1: Calculation $\Delta^* = C_2 C_1 C_3 + C_2(-C_4)$
2: Send the calculated result Δ^* to the edge computing node.

Algorithm 1. The constructed calculation rules and confusion matrix is sent to the edge service node for outsourced calculation.

Step 2 (outsourcing data calculation algorithm): The edge service layer node outputs the calculation result Δ^* according to the outsourcing calculation rule $f(C')$ sent by the client (the calculation rule is based on the gradient descent method for linear regression) and send the calculation result to the client.

Step 3 (training result generation algorithm): The client receives the calculation result δ^* returned by the edge server layer node and executes Algorithm 3. In this section, to better explain the application process of the algorithm, Formula (1) will be taken as an example.

Step 4 (data verification algorithm): After the client receives the calculation result Δ^* returned by the edge server layer node, it verifies the calculation result.

Step 5 (end the calculation task) When the function reaches the convergence value or the edge service node calculation task fails, the edge calculation node executes this step algorithm.

When the edge computing node checks and finds that there is an error in the calculation result returned by the edge service node, Algorithm 6 is executed. This algorithm is used to generate evidence that the edge service node has not faithfully performed the model training task according to the protocol, thereby announcing that the node is untrustworthy and building a system trust mechanism.

When other nodes verify the security of the edge service node, the verification matrix is extracted from the evidence $E_{u \to s}$, and the corresponding results are obtained according to the calculation rules to determine whether the evidence is

Algorithm 3. Training result generation algorithm

1: $w_t^* \leftarrow \Delta^*$, extract submatrix part of the matrix w_t^*;
2: Goto Algorithm 4;
3: **if** $w_t' = w_t^*$ **then**
4: The results returned by outsourced calculations are true;
5: **if** $w_t' < w_{t-1}'$ **then**
6: The function does not reach the convergence value and generates a new validation factor V_t to replace the validation factor in Δ^*;
7: Perform scrambling operation to generate $\Delta^* \leftarrow w_t'$;
8: Goto Step2;
9: **else**
10: The function reaches a convergence value;
11: **end if**
12: **end if**
13: $w_t = K^{(}-1)\, W_t'$;
14: Goto Step 5.

Algorithm 4. Data verification algorithm

Input:Key: k, Calculation result Δ^*.
Output: Validation result $V? = V^*$.

1: Extract validation matrix blocks $V^* \leftarrow \Delta^*$;
2: Calculation $\det(V^*)$;
3: **if** $\det(V)$=$\det(V^*)$ **then**
4: **return** "*True*."
5: **else**
6: Goto Step5 and Step6;
7: Find new computing nodes and perform model training tasks;
8: **end if**

valid. When the verification node records the verification result, a trust record is built locally.

5 System Analysis

5.1 Security

The security of the solution is considered from the following aspects: data security and privacy, the correctness of the calculation results, and the trust mechanism of edge service nodes.

1. The correctness of calculation results

When calculating a block matrix, it can be divided into two parts: matrix addition and matrix multiplication for analysis and consideration.

Algorithm 5. End the calculation task

1: The client will send W^0 to the edge server node;
2: When the edge service layer node receives W^0, it learns that the computing task is terminated;
3: The edge service layer node deletes the computing data related to the training task.

Algorithm 6. Evidence generation and adjudication algorithm

Input:Key: k, Validation matrixes V_1, V_2, V_3, V_4, and Calculation Rule $f()$.
Output: Evidence E.

1: Generate evidence's signature $S \leftarrow H(V_1||V_2||V_3||V_4||V^*||H_u(ID)||H_s(ID))$;
2: Generate evidence $E_{u\rightarrow s} \leftarrow (V_1||V_2||V_3||V_4||V^*||f()||S)$.

First, verify the correctness of matrix multiplication:
In the calculation of $(X^T)' \times X'$, the result is

$$\begin{bmatrix} X^T D^{-1} DX & X^T D^{-1} M_1 + M_2 V_1 \\ 0 & V_2 V_1 \end{bmatrix}. \tag{10}$$

And the result in the upper right corner of the matrix is not difficult to see as $X^T \times X$. For a further description of the calculation with w, it can be simplified to $\begin{bmatrix} X^T D^{-1} DX(Rw^T) & \cdots \\ 0 & V_2 V_1 V_3 \end{bmatrix}$.

Since R is a diagonal matrix, it is assumed that k is randomly selected as the diagonal element value of the matrix. The matrix element of $X^T \times X$ is x_{ij}. The calculation result of Formula (1) indicates

$$\begin{bmatrix} \sum_{j=1}^{n} \sum_{i=1}^{n} x_{1j} k w_{i1} & \cdots & \sum_{j=1}^{n} \sum_{i=1}^{n} x_{1j} k w_{il} \\ \vdots & \ddots & \vdots \\ \sum_{j=1}^{n} \sum_{i=1}^{n} x_{nj} k w_{i1} & \cdots & \sum_{j=1}^{n} \sum_{i=1}^{n} x_{nj} k w_{il} \end{bmatrix} \tag{11}$$

.

It is not difficult to see that multiplying the matrix by $\frac{1}{k} I$ can restore the original data.

The correctness of the matrix addition is relatively simple and will not be repeated; the reader can prove it by himself.

2. Proof of algorithm security

In this section, the security of this scheme is demonstrated under five assumptions of insecurity.

– Hypothesis 1: Malicious users obtain data through intermediate parameters.

In the embodiment, the data used for the matrixes X, Y, and w are used for the matrixes randomly perturbing invertible matrix operation, and the elements in the invertible matrix D are randomly generated and have no correlation.

Therefore, the service edge node cannot guess any information data from the matrix, to ensure the privacy of the data. At the same time, both the target matrix and the intermediate parameters are the result of adding K to the disturbance, and only users who master K can restore the data.

For $\Delta = X_B^T \times (X_B \times w) + X_B^T \times (-Y_B)$, there is a transformation of $X^T D^{-1} DX$, which guarantees data security and recovery.

Meanwhile because of the learning rate (step), use the data to set the size of the initial parameters, and other related parameters of the system are performed locally. Therefore, the program also ensures the safety system model in the training process, the iterative training process is completed, and the edge service node only grasps the intermediate value of, thus, ensuring the safety training model.

This paper only describes the solution with a linear regression model. When other machine learning algorithms are needed, only the calculation process needs to be improved.

– Hypothesis 2: The malicious edge server can recover the intermediate parameter through the inverse matrix.

In the solution described in this paper, all user data is added with disturbance and encapsulation, and edge servers cannot know the true meaning of their calculation data. Here, we assume a more powerful enemy that can conspire with the edge server that computes the inverse matrix $(D^{-1})'$ (this is not truly an inverse matrix). In this case, although the malicious server obtains the intermediate value of the relevant inverse matrix D^{-1}, it cannot recover the accurate inverse matrix information, and it also cannot restore the original data. After generating the inverse matrix, the inverse matrix returned by the server contains the data disturbance. Therefore, except for the owner of the data, it is difficult for others to restore the original matrix and its inverse matrix.

– Hypothesis 3: The two edge servers conspire to get user data.

In this method, the biggest threat is that the two edge service providers recover the data by obtaining the inverse matrix D^{-1} information of the user, and other methods cannot recover the data. Similar to Hypothesis 2, due to a matrix calculated by one of the servers. The matrix contains inverse matrix and disturbance information, and to restore the true inverse matrix, adversary need to grasp the parameters of its disturbance information. Therefore, the final inverse matrix D^{-1} cannot be obtained between these two edge servers, so this scheme is already safe under this assumption.

– Hypothesis 4: Trust mechanism of edge service nodes.

In the system, this paper introduces an arbitration mechanism. When the edge node detects that the calculation result returned by the edge service node is abnormal, that is, the returned result is inconsistent with the verification result, then the edge node considers labelling the edge service node as malicious.

The edge node verifies the information, including the returned result v, the raw data used to generate the verification results (V_1, V_2, V_3, and V_4), and the relevant identity information $H(ID_u)|H(ID_s)$ of the edge node and the service node, and then generates evidence and publishes it to other nodes. When each node in the system receives the evidence, it performs a test. If it is indeed proved that the edge service node has not performed calculation tasks according to the agreement, it will be added to the blacklist, and the data will no longer be outsourced to the node. Concurrently, to ensure the long-term effectiveness of the system, you can look for trusted authorities (for example, government agencies) to store evidence and maintain node information in the system.

– Hypothesis 5: Adversary obtains enough ciphertext for analysis.

The scheme in this paper is to protect the security and privacy of the sample data X, tag data Y and parameter (weight) information w. Ensure that the adversary can assist the user in the calculation task, and does not obtain any data information.

In the scheme, the invertible matrix D and its inverse matrix D^{-1} are the key to ensure the computability and safety of X and X^T. It has been proved in Hypothesis 1 and 2 that it can guarantee the security of the data. However, long-term use of the same set of matrices (including the invertible matrix D and its inverse matrix D^{-1}) is a security risk. The adversary may recover the original data contained in the disturbance data by collecting a large number of matrices (for example, C_1, C_2). In turn, threatens the security of the program. Therefore, the frequency of use of D and D^{-1} can be adjusted to achieve better data security. The safest solution is to use different D and D^{-1} each time. Of course, this will increase the computational cost of the initialization phase (Algorithm 1 and Algorithm 2). The computational cost of the initialization phase will be discussed in the experimental part.

5.2 System Performance Analysis

For the performance of the system, this paper uses a real data set for comparison experiments (It is not difficult to see through a theoretical analysis that the calculation results generated by using this solution are consistent with the original calculation results, so the comparison of the calculation results will not be performed here).

The experimental parameters are as follows: CPU I5-2450M (2.5 GHz), 8 GB of memory, and Windows 10 64-bit operating system. The system is implemented using Python language, and because some edge devices may not be equipped with a GPU, this paper does not use a GPU to accelerate processing.

In this paper, we first use the Boston house price prediction data set for experiments. The data set is 507 * 13. During the experiment, 400 pieces of data are selected for training. The average of 15 test results is used to determine the final experimental data results. The experimental test results are shown in Fig. 4: the method proposed in this paper takes 24.76 ms, and the general training

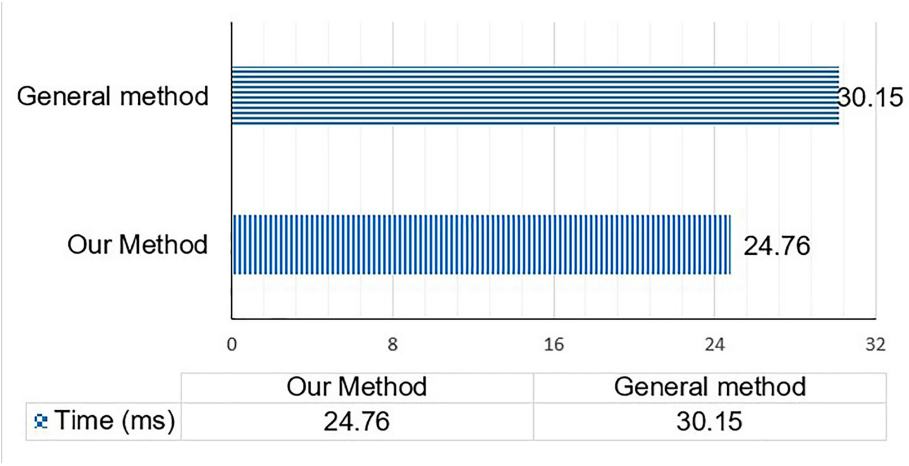

Fig. 4. Comparison of time consumptions for the Boston house price prediction dataset.

process takes 30.15 ms. From the results, it can be seen that the method in this paper is approximately 18% faster than the general method (In this paper, it refers to the general machine learning algorithm, that is, linear regression algorithm).

The feature dimensions of the Boston house price prediction data set are small, as there are only 13 features. Afterwards, this paper randomly generates a data set of size 500, 600, 700, etc., which has 300 features and 200 training rounds. The experimental results obtained are shown in Fig. 5:

It can be seen from Fig. 5 that the solution proposed in this paper can effectively reduce the calculation amount of edge computing nodes. Among devices with limited resources, the system will effectively reduce the computing pressure of the device and alleviate the consumption of local resources.

From the experimental results, the scheme in this paper saves time by approximately 20%, when compared with the unoptimized scheme. The machine learning algorithm used in this paper is relatively basic and has a small number of calculations. The results have a certain advantage in terms of time consumption. Since the main computing tasks in the training process are outsourced to other service nodes, appropriate adjustments can be applied to machine learning algorithms with more complex training processes. In general, the solution in this article is suitable for application scenarios where the edge device cannot execute or is difficult to handle.

At the same time, as shown in Fig. 6, this article also compares with homomorphic encryption. By using a homomorphic encryption scheme to perform addition operations and multiplication operations of the same order of magnitude, this method can be seen to be more suitable for mobile devices in terms of efficiency. It is worth noting that, because the addition homomorphic encryption

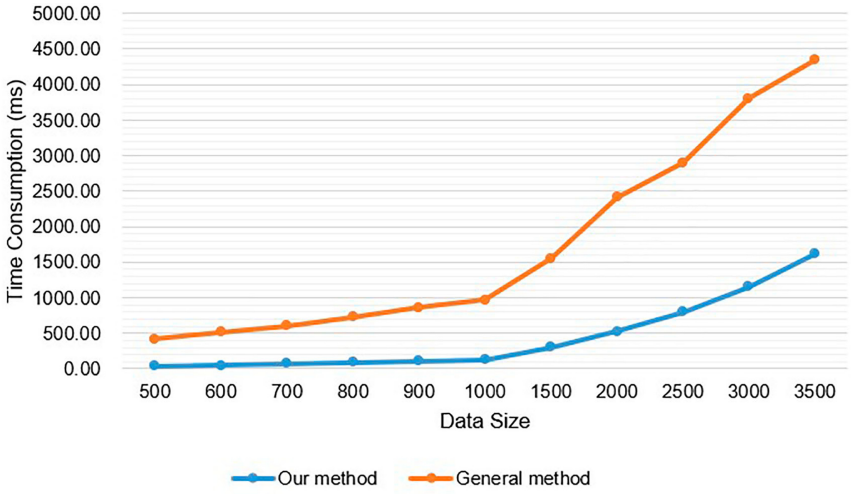

Fig. 5. Comparison of the efficiencies of the proposed scheme and the general method.

time based on Paillier cryptography runs far longer than does the scheme in this paper during the addition operation of the same order of magnitude (data is not shown in the figure).

In the initialization phase of the algorithm, the initialization time of the multiplicative homomorphic encryption based on RSA is approximately 45.4 ms, while the initialization time of the additive homomorphic encryption based on Paillier is approximately 1687.8 ms; the initialization time of the method in this paper varies with the amount of data formed by the matrix. When processing the same amount of data, the scheme in this paper takes the shortest time of time at this stage.

Through the experimental comparison and theoretical analysis, it can be seen that, by comparing with the scheme without any encryption and data perturbation, the scheme in this paper significantly improves the execution efficiency and other aspects, as well as ensured the security and accuracy of the data. Compared with the homomorphic encryption system, the security of this solution is weaker than the homomorphic encryption technology, but the execution efficiency is more suitable for devices with lower computing capabilities. This solution can enable low computing power equipment to process larger machine learning algorithms, while ensuring their safety and accuracy.

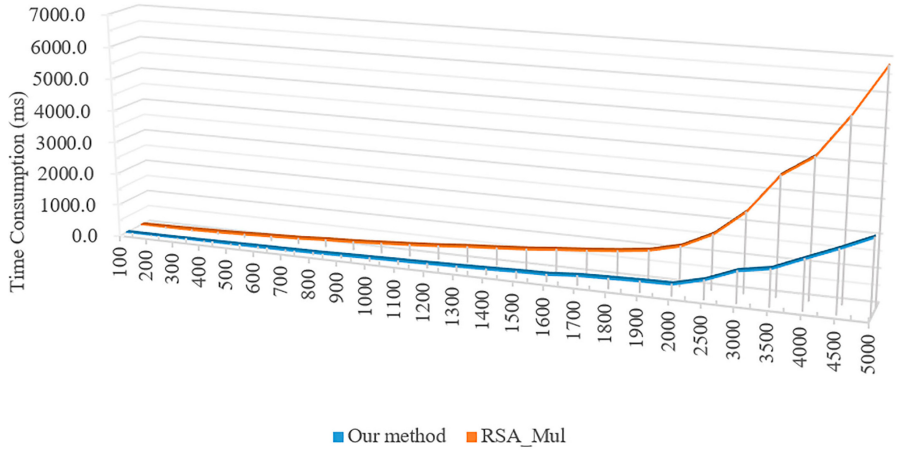

Fig. 6. Comparison of the efficiencies of this scheme and RSA multiplication homomorphic encryption.

6 Conclusion

With the continuous improvement in mobile device capabilities and the need for low-latency applications, computing tasks will increasingly be migrated locally, but this also brings issues such as device energy consumption, occupied device computing, and storage. Therefore, outsourcing data to a near local end can reduce network transmission delays and reduce pressure on mobile devices. However, the security and privacy issues arising during the outsourcing process cannot be ignored. Therefore, this paper proposes EVPP: a secure data outsourcing computing solution based on matrix operations. The outsourcing matrix adds a lightweight verification factor so that the verification process does not place excessive computing pressure on the mobile device. In terms of the trust mechanism, when the device finds a malicious service node in the system, it can verify the data generates arbitration evidence so that other nodes in the system can verify and avoid sending outsourced tasks to malicious nodes. Through a theoretical analysis and experimental comparison, it can be verified that the scheme exhibits certain improvements in efficiency, safety and correctness and can be applied to practical applications. Because the solution in this article only optimizes the training process, it has certain limitations. To better reduce the complexity of machine learning models for equipment training in edge environments, the next step will be combining with federal learning [32] to conduct a further study in the distributed system.

References

1. Petrov, C.: Big data statistics (2019). https://techjury.net/stats-about/big-data-statistics/. Accessed 22 Mar 2019

2. Zhang, X., Qiao, M., Liu, L., et al.: Collaborative cloud-edge computation for personalized driving behavior modeling. In: Proceedings of the fourth ACM/IEEE Symposium on Edge Computing (SEC). ACM/IEEE, Washington (2019)
3. Jia, K., Li, H., Liu, D., et al.: Enabling efficient and secure outsourcing of large matrix multiplications. In: 2015 IEEE Global Communications Conference (GLOBECOM), pp. 1–6. IEEE, San Diego (2015)
4. Lei, X., Liao, X., Huang, T., et al.: Achieving security, robust cheating resistance, and high-efficiency for outsourcing large matrix multiplication computation to a malicious cloud. Inf. Sci. **280**, 205–217 (2014)
5. Li, P., Li, J., Huang, Z., Gao, C.-Z., Chen, W.-B., Chen, K.: Privacy-preserving outsourced classification in cloud computing. Cluster Comput. 1–10 (2017). https://doi.org/10.1007/s10586-017-0849-9
6. Abdellatif, A.A., Mohamed, A., Chiasserini, C.F., et al.: Edge computing for smart health: context-aware approaches, opportunities, and challenges. IEEE Netw. **33**(3), 196–203 (2019)
7. Pathinarupothi, R.K., Durga, P., Rangan, E.S.: IoT-Based smart edge for global health: remote monitoring with severity detection and alerts transmission. IEEE Internet Things J. **6**(2), 2449–2462 (2018)
8. Liu, X., Deng, R.H., Choo, K.R., et al.: Privacy-preserving reinforcement learning design for patient-centric dynamic treatment regimes. IEEE Trans. Emerg. Top. Comput. 1 (2019). (Early Access)
9. Chen, S., Wen, H., Wu, J., et al.: Radio frequency fingerprint-based intelligent mobile edge computing for internet of things authentication. Sensors **19**(16), 3610 (2019)
10. Froiz-Míguez, I., Fernández-Caramés, T., Fraga-Lamas, P., et al.: Design, implementation and practical evaluation of an IoT home automation system for fog computing applications based on MQTT and ZigBee-WiFi sensor nodes. Sensors **18**(8), 2660 (2018)
11. Shi, W., Cao, J., Zhang, Q., et al.: Edge computing: vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016)
12. Li, Q., Zhu, H., Xiong, J., Mo, R., Ying, Z., Wang, H.: Fine-grained multi-authority access control in IoT-enabled mHealth. Ann. Telecommun. **74**, 389–400 (2019). https://doi.org/10.1007/s12243-018-00702-6
13. Chui, K.T., Liu, R.W., Lytras, M.D., et al.: Big data and IoT solution for patient behaviour monitoring. Behav. Inf. Technol. **38**, 1–10 (2019)
14. Liang, X., Du, X., Wang, G., et al.: A deep reinforcement learning network for traffic light cycle control. IEEE Trans. Veh. Technol. **68**(2), 1243–1253 (2019)
15. Zhou, P., Braud, T., Alhilal, A., et al.: ERL: edge based reinforcement learning for optimized urban traffic light control. In:2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pp: 849–854. IEEE, Mannheim (2019)
16. Joo, J., Park, M.C., Han, D.S., et al.: Deep learning-based channel prediction in realistic vehicular communications. IEEE Access **7**, 27846–27858 (2019)
17. Feng, B., Fu, Q., Dong, M., et al.: Multistage and elastic spam detection in mobile social networks through deep learning. IEEE Netw. **32**(4), 15–21 (2018)
18. Liu, X., Deng, R.H., Choo, K.R., et al.: Privacy-preserving outsourced support vector machine design for secure drug discovery. IEEE Trans. Cloud Comput. (2018). (Early Access)
19. Zhang, X., Wang, Y., Lu, S., et al.: OpenEI: an open framework for edge intelligence. arXiv preprint arXiv:1906.01864 (2019)

20. Sun, Y., Wen, Q., Zhang, Y., et al.: Two-cloud-servers-assisted secure outsourcing multiparty computation. Sci. World J. **2014** (2014)
21. Mohassel, P., Zhang, Y.: SecureML: a system for scalable privacy-preserving machine learning, In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 19–38. IEEE, San Jose (2017)
22. Huang, K., Liu, X., Fu, S., et al.: A lightweight privacy-preserving CNN feature extraction framework for mobile sensing. IEEE Trans. Dependable Secure Comput. 1 (2019). (Early Access)
23. Vengadapurvaja, A.M., Nisha, G., Aarthy, R., et al.: An efficient homomorphic medical image encryption algorithm for cloud storage security. Proc. Comput. Sci. **115**, 643–650 (2017)
24. Piao, C., Shi, Y., Yan, J., et al.: Privacy-preserving governmental data publishing: a fog-computing-based differential privacy approach. Future Gener. Comput. Syst. **90**, 158–174 (2019)
25. Salinas, S., Luo, C., Chen, X., et al.: Efficient secure outsourcing of large-scale sparse linear systems of equations. IEEE Trans. Big Data **4**(1), 26–39 (2018)
26. Salinas, S., Luo, C., Chen, X., et al.: Efficient secure outsourcing of large-scale linear systems of equations. In: 2015 IEEE Conference on Computer Communications (INFOCOM), pp. 1035–1043. IEEE (2015)
27. Yu, Y., Luo, Y., Wang, D., et al.: Efficient, secure and non-iterative outsourcing of large-scale systems of linear equations. In: 2016 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2016)
28. Lei, X., Liao, X., Huang, T., et al.: Cloud computing service: the caseof large matrix determinant computation. IEEE Trans. Serv. Comput. **8**(5), 688–700 (2014)
29. Chen, F., Xiang, T., Lei, X., et al.: Highly efficient linear regression outsourcing to a cloud. IEEE Trans. Cloud Comput. **2**(4), 499–508 (2014)
30. Zhou, L., Zhu, Y., Choo, K.K.R.: Efficiently and securely harnessing cloud to solve linear regression and other matrix operations. Future Gener. Comput. Syst. **81**, 404–413 (2018)
31. Hu, C., Alhothaily, A., Alrawais, A., et al.: A secure and verifiable outsourcing scheme for matrix inverse computation. In: IEEE INFOCOM 2017 IEEE Conference on Computer Communications, pp. 1–9. IEEE (2017)
32. Cheng, K., Fan, T., Jin, Y., et al.: SecureBoost: a lossless federated learning framework. arXiv preprint arXiv:1901.08755 (2019)