Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 8825623, 16 pages
https://doi.org/10.1155/2020/8825623

*Research Article*

# A Secure and Verifiable Outsourcing Scheme for Assisting Mobile Device Training Machine Learning Model

**Cheng Li, Li Yang, and Jianfeng Ma**

*Xidian University, Xi'an, Shaanxi 710071, China*

Correspondence should be addressed to Li Yang; yangli@xidian.edu.cn

Received 26 June 2020; Revised 15 October 2020; Accepted 6 November 2020; Published 17 November 2020

Academic Editor: Ding Wang



In smart applications such as smart medical equipment, more data needs to be processed and trained locally and near the local end to prevent privacy leaks. 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), which is a computing outsourcing scheme used in the training process of machine learning models. The edge nodes outsource the complex computing process to the edge service node. First, we conducted a certain amount of testing to confirm the parts that need to be outsourced. 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, the system can generate verifiable evidence that can be generated to build a trust mechanism when a malicious service node is found. At the same time, this paper also discusses the application of the scheme in other algorithms in order to be better applied. 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.

## 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, concurrently, a large amount of data has also been generated that is processed by different companies and 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.

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 great pressure to the network.

The traditional centralized computing architecture based on a cloud center [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 [8], and smart homes [9], 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.

Research on data outsourcing has received widespread attention, whether it is data outsourcing to cloud servers or other nodes with greater computing power. Similarly, more and more algorithms need to be calculated on the user side with the development of machine learning, Internet of

Things devices (especially the wide application of wearable devices), network technology, and so on. Therefore, it is a good choice to outsource some calculations in machine learning algorithms. Neto et al. [10] used mobile devices for health monitoring and outsourced data to improve the practical application problems caused by the limited resources of mobile devices. However, there is usually a long distance between the user and the cloud (This not only includes the distance in space; in fact, the user also needs to pass through many nodes in the network to the cloud. These nodes may not be able to be transmitted in time, or there are unstable factors such as jitter in the network.), so it will cause time consumption and some unstable factors, which are often not what users want to see. Therefore, to avoid the problem that a large amount of data cannot be processed in time, Neto et al. used near-user service nodes to perform data outsourcing calculations in the study and achieved certain results.

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, 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.

Data in medical, health, and other applications is exploding. Therefore, outsourcing data to the cloud has become the choice of many users, but it also brings security and privacy risks. Therefore, Li et al. [12] proposed an anonymous authorization scheme to ensure the confidentiality and authenticity of the data. Ding et al. [13] studied the data access control scheme under ciphertext, which can also perform effective calculations while ensuring flexible access control. Similarly, when applying edge computing to model training for local devices and nodes, data security and privacy issues cannot be ignored [12, 14, 15]. 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.

Based on the above issues, as shown in Figure 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 high availability of data and 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 which is 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 ensuring 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. At the same time, we also conducted theoretical analysis and experiments to explain which calculations need to be outsourced and how to reasonably perform outsourcing calculations, so as to better perform outsourcing calculations and optimization in practical applications. This paper also adds a trust mechanism to further increase the security of the system.

The contributions of this paper 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), the 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: Section 1 briefly introduces the related research, Section 2 will further describe the problems and challenges studied in this paper, and Section 3 will discuss the scheme and its algorithms in detail. Section 4 will focus on the security and performance proofs of the proposed goals in this paper. In Section 5, we will discuss the application of the scheme in other machine learning algorithms and related issues in related fields. In the last section, the scheme will be summarized, and future research directions will be discussed.

## 2. Related Work

Smarter healthcare [6, 7, 16], urban transportation [17, 18], connected cars [19], social networks [20], and other scenarios

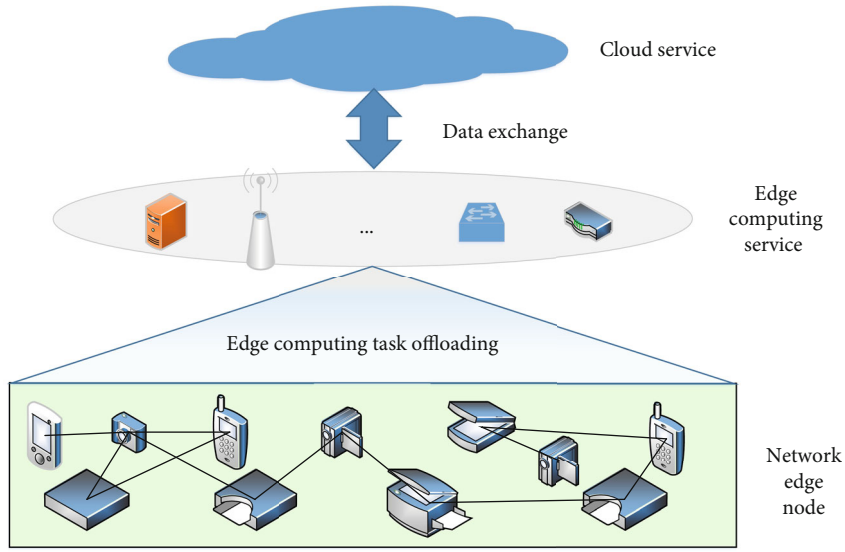



Figure 1: Schematic diagram of data processing and calculation using edge computing.

are increasingly applying machine learning algorithms for prediction and analysis. In these application scenarios, the data are outsourced to the cloud, which has strong computing power, so that resource-constrained devices can use the cloud center to complete various complex computing tasks and better serve users [21, 22].

However, with the advent of the Internet of Everything, the edge devices of the network longer generate abundant data, which is not suitable for processing in the cloud center 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. [23] 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 [24–26], including homomorphic encryption [27], differential privacy [28], and attribute-based encryption [29].

Rahulamathavan et al. [30] proposed an SVM classification scheme for outsourcing using Pailler homomorphic encryption technology. In 2018, Li et al. [31] found that the research of Rahulamathavan et al. had problems in terms of soundness and security, so they proposed a more secure solution. In 2015, Liu et al. [32, 33] published two articles on machine learning algorithm outsourcing. They conducted research on the security and privacy issues faced after the data was outsourced to the cloud server and conducted research and experimental analysis on the SVM and gradient descent algorithm, respectively. Li et al. [34] solved the outsourcing security challenge under the malicious model and used homomorphic encryption and security obfuscation circuit technology to implement the secure outsourcing calculation of the ID3 decision tree algorithm. Under the semihonest model, Zhang et al. [35] designed a privacy-protected single-layer perceptron training scheme. This scheme uses a secure two-party calculation to implement a lightweight algorithm to ensure that the participants cannot know the input data and the safety of the calculation results. At the same time, Li et al. [5] also noted that the confidentiality of the classifier is also very important. They proposed the POCC solution to solve the confidentiality of the data and classifier in the cloud. Liu et al. [36] designed a complete and secure outsourcing solution for the KNN algorithm to ensure the safety of data in the process of publishing, storing, applying, requesting, and returning results.

Most of the above studies are for cloud computing. In order to reduce the computational and storage costs of network edge devices and return the results required by users in a timely and effective manner, relevant studies have begun to consider applying edge computing to solve this problem. Wu et al. [37] designed a safe and lightweight query scheme (LPEQ) using edge computing technology. This scheme satisfies the requirements of security under the semihonest model. Tao et al. [38] proposed the outlet solution to the difficulties in the ability of wearable devices to process complex data, which uses nearby mobile edge resources to solve context dependence, scattered resources, and resource dynamics, to ensure that the Internet of Things, especially wearable devices, is timely and effective in terms of data. But these solutions are not friendly to edge devices with low computing and storage capabilities. In related research, technologies like homomorphic encryption are mostly used. These encryption schemes may also bring certain computing pressure to some mobile devices and wearable devices.

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 [29, 39–41]. Salinas et al. [29] proposed a large-scale deterministic secure outsourcing computing solution. The client can effectively verify the correctness of the calculation results of the outsourced data. Chen et al. [42] proposed a scheme for scrambling the original matrix data using diagonal matrix multiplication to ensure the security of the data; subsequently, it was improved in Zhou et al. [43], and it must be further improved in terms of security and result verification. Hu et al. [44] proposed a matrix outsourcing inversion matrix scheme that can be applied to cloud computing and other scenarios, which effectively reduces the computational complexity of the client. Therefore, these solutions are worthy of reference for solving the safety and efficiency problems of equipment in outsourcing calculation.

## 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.

(1) *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.

(2) *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.

(3) *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.

(4) *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 Section 3.1, this paper designs an outsourcing model training scheme based on edge computing, as shown in Figure 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, smartphones 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, calculation, and storage tasks of sensor devices. Concurrently, 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, and (2) it may not complete the computing task as agreed, causing the computing task of the edge nodes to fail.

As shown in Figure 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. What these two edge servers know are intermediate values, so the security of the system is guaranteed. The specific proof is proved in Hypotheses 7 and 8.

(1) The edge nodes send a request to the edge service node to assist it in calculating the inverse matrix of the matrix for subsequent steps

(2) The edge service node returns the value of the invertible matrix of the edge nodes

(3) The edge nodes send the data that needs to outsource calculation to the edge service node

(4) The edge service node returns the calculated result to the edge nodes. It is worth noting that in this process, the transmitted data is all ciphertext

It is worth noting that in the system, users should be guaranteed to be legal; secondly, the credibility (honesty) data of each node is stored and queried, which requires a trusted third party or a public verification system (for example, government, authority, and blockchain). The preparation work for system construction and operation can be done with reference to [45–47] to ensure the safety and reliability of the system.

*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

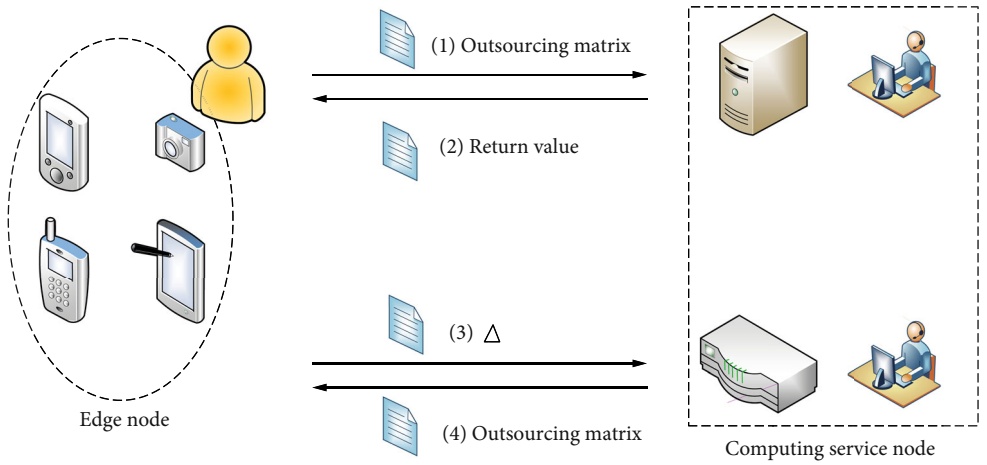



Figure 2: Training model of the outsourcing model based on edge computing.

number of multiplication operations is higher than that of addition operations. At the same time, the computational complexity of multiplication is higher than that of 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, the main ideas of the scheme are described.

Given the $n$ sample set $(X, Y)$ where the $i$th sample $X_i$ contains $d$ features, that is, $X_i = (x_1, x_2, \cdots, x_d)$, adjust the objective function $h(X_i) = (x_1 \bullet w_1 + x_2 \bullet w_2 + \cdots + x_d \bullet w_d) = X_i \bullet w^T$, where $w = (w_1, w_2, \cdots, w_d)$. Adjust the parameters through the training process to yield the appropriate to make $J_i(w) = ((1/2n)\sum_{i=1}^{n}(h(X_i - Y_i)^2))$ 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 $w_j = w_j - \alpha(\partial J_i(w)/\partial w_i) = w_j - \alpha(1/n)\sum_{i=1}^{n}(f(X_i) - Y_i)X_i^j$, and by representing vectors as a matrix, $w$ can be expressed as $w := w - \alpha X^T \times (X \times w - Y)$ [25].

Among them, $\alpha$ 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 $w := w - \alpha(1/|B|)X_B^T \times (X_B \times w - Y_B)$ [25].

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

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

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 Figure 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.

*Step 2* (outsourcing data calculation algorithm). The edge service layer node outputs the calculation result $\Delta^*$ 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 $\Delta^*$ sent back by the edge server layer node and performs a recovery operation (Step 4) based on the information held locally to obtain the calculation result $\Delta$. 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, and the scrambling operation is performed and returns to Step 2 to continue training; $w_t > w_{t-1}$ indicates that the

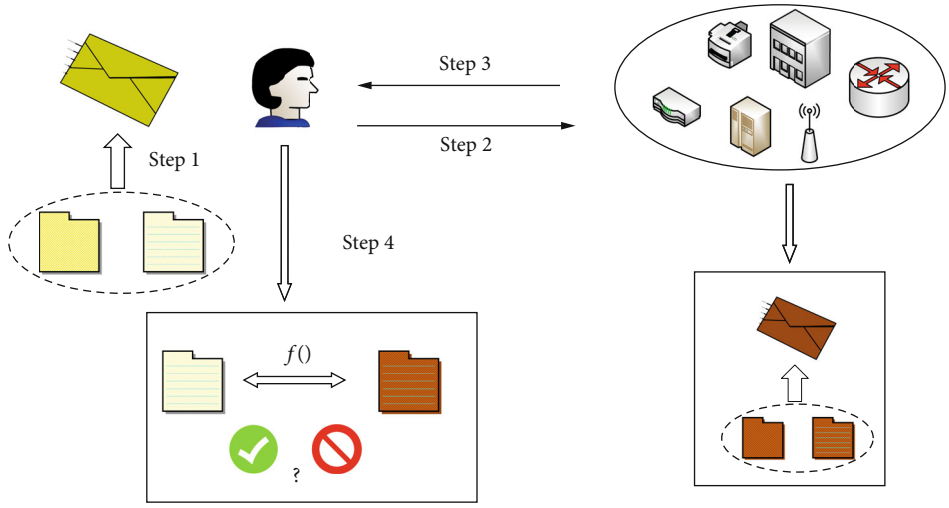



Figure 3: The main design ideas of the scheme.

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 $\Delta^*$ returned by the edge server layer node, extracts the verification matrix block $V^*$ from it, and compares it with $V \stackrel{?}{=} V^*$. 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 Figure 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 Formulas (1) and (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 nodes encrypt $m \times n$ data $X$, $m \times l$ data, and $n \times l$ data $w$ to ensure data security and then operate 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:

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

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

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

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

The construction process of the invertible matrix $D$ will be described in detail [43].

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 $P = (p_1, p_2, \cdots, p_m)$ and $Q = (1, q_1, q_2, \cdots, q_{m-1})$. $D$ 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,

$$D = \begin{bmatrix} p_1 & q_1 p_1 & q_2 p_1 & \cdots & q_{m-1} p_1 \\ p_2 & q_1 p_2 + d_1 & q_2 p_2 & \cdots & q_{m-1} p_2 \\ p_3 & q_1 p_3 & q_2 p_1 + d_3 & \cdots & q_{m-1} p_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_m & q_1 p_m & q_2 p_m & \cdots & q_{m-1} p_m + d_m \end{bmatrix}. \tag{7}$$

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

$$D = \begin{bmatrix} p_1 & q_1 p_1 & q_2 p_1 & \cdots & q_{m-1} p_1 \\ 0 & d_1 & 0 & \cdots & 0 \\ 0 & 0 & d_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & d_m \end{bmatrix}. \tag{8}$$

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 [43].

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 nodes 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, Algorithms 1 and 2 will be described according to the algorithm described in Section 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 $\Delta$ of Section 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 $D^T$, the tag set $Y$, and the initial $w$ confusion matrixes $C_2$, $C_1$, $C_3$, and $C_4$ through Algorithm 1. The constructed calculation rules and confusion matrix are sent to the edge service node for outsourced calculation.

*Step 2* (outsourcing data calculation algorithm). The edge service layer node outputs the calculation result $\Delta^*$ 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 sends the calculation result to the client.

*Step 3* (training result generation algorithm). The client receives the calculation result $\Delta^*$ 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.

It is worth noting that in Algorithm 3, Algorithm 4 is not necessarily executed every time for verification, because Algorithm 4 will bring the calculation overhead of the edge device. At the same time, the frequency of Algorithm 4 execution is related to the probability of finding dishonest service nodes. When Algorithm4 is executed every time, the scheme in this paper can be applied to the malicious model.

*Step 4* (data verification algorithm). After the client receives the calculation result $\Delta^*$ 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 valid. When the verification node records the verification result, a trust record is built locally.

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 $C_2$ , $C_1$, $C_3$, and $C_4$.
1: Calculate $D = D_1 + D_2$;
2: Go to 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 1: Outsourced data generation algorithm.

Input: Key $k$, Matrix$C_2$ , $C_1$, $C_3$, $C_4$, and Calculation Rule $f()$.
Output: Calculation result $\Delta^*$.
1: Calculate $\Delta^* = C_2 C_1 C_3 + C_2(-C_4)$;
2: Send the calculated result $\Delta^*$ to the edge computing node.

Algorithm 2: Outsourcing data calculation algorithm.

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

Algorithm 3: Training result generation algorithm.

Input: Key $k$, Calculation result $\Delta^*$.
Output: Validation result $V \stackrel{?}{=} V^*$.
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: Go to Step 5 and Step 6;
7: Find new computing nodes and perform model training tasks;
8: end if

Algorithm 4: Data verification algorithm.

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 5: End the calculation task.

Input: Key $k$, Validation matrixes $V_2$ , $V_1$, $V_3$, $V_4$, and Calculation Rule $f()$.
Output: Evidence $E$.
1: Generate evidence's signature $S \leftarrow H(V_1 \mid\mid V_2 \mid\mid V_3 \mid\mid V_4 \mid\mid V^* \| H_u(ID) \| H_s(ID))$ ;
2: Generate evidence $E_{u \to s} \leftarrow (V_1 \mid\mid V_2 \mid\mid V_3 \mid\mid V_4 \mid\mid V^* \mid\mid f() \mid\mid S)$,

Algorithm 6: Evidence generation and adjudication algorithm.

## 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.

*5.1.1. 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.

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 $(1/k)I$ 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.

*5.1.2. 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, for $\Delta = X_B^T \times (X_B \times w) + X_B^T \times (-Y_B)$ , 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. There is a transformation of $X^T D^{-1} DX$, which guarantees data security and recovery. Therefore, the service edge nodes 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.

In the training process, the learning rate (step size) is determined by the edge nodes, and the initialized parameters are random. Except for the calculations necessary for the outsourcing calculation, the other is done locally by the edge nodes. 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, thus ensuring the safety training model.

The scheme ensures the security of data during the exchange process. 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.

[44] has carried out detailed proof of the security analysis and will not repeat them here.

*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. This is similar to Hypothesis 8, 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, the adversary needs 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 nodes detect 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 nodes consider labelling the edge service node as malicious. The edge nodes verify the information, including the returned result $v$, the raw data used to generate the verification results ($V_2$, $V_1$, $V_3$, and $V_4$), and the relevant identity information $H(\mathrm{ID}_u) \mid H(\mathrm{ID}_s)$ of the edge nodes and the service node, and then generate evidence and publish 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 Hypotheses 7 and 8 that it can guarantee the security of the data. However, long-term use of the same set of matrixes (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 matrixes (for example, $C_1$ and $C_2$). In turn, it 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 (Algorithms 1 and 2). The computational cost of the initialization phase will be discussed in the experimental part.

*Hypothesis 6.* Discussion on scheme security model.

When introducing Algorithm 3, we have already mentioned that there is a certain probability of detecting dishonest edge service nodes. Since Algorithm 4 requires the edge device to run locally, it will cause additional computational costs. Assuming that the malicious edge service node is strong enough, it can know when the edge nodes perform the verification of outsourcing results. But this will increase the computational cost of the edge nodes. Although in the scheme in this paper the edge nodes can choose a smaller dimension of the verification matrix, it still has computational costs.

At the same time, we designed evidence generation and verification algorithms in the scheme. Suppose that the rational edge service node in the system is detected to be evil, and the evidence of evil will be retained in the system. Other edge nodes can verify the evidence in the system. If the evidence is true, the malicious edge service node will lose the trust of the edge nodes, so that the edge nodes will not choose to submit outsourced computing tasks to it.

Therefore, once the edge service node loses the outsourced task of the edge nodes, it will not receive the corresponding reward. So, for the rational node, it will still abide by the computing protocol. However, this also increases the computational cost of edge nodes. So, our purpose is to effectively reduce the computational cost of edge nodes, so our scheme is more used for semihonest models.

*5.2. Analysis of Computational Complexity and Time Cost.* In this section, we will discuss the computational overhead of matrix addition and multiplication operations commonly used in machine learning. Of course, matrix computing is used not only in the field of machine learning but also in many fields of scientific computing. Therefore, the final conclusion is generally applicable. For the convenience of description, it is assumed that two $n * n$ matrixes $A$, $B$ need to be added and multiplied (subtraction and division operations can be transformed into addition or multiplication operations).

(i) *Addition*. During the addition operation, we first need to traverse the elements in $A$ by row and the elements in $B$ by column. Since both $A$ and $B$ are $n * n$ matrixes, it is not difficult to see that the time complexity of addition is $O(n^2)$.

(ii) *Multiply*. During the multiplication operation, each row of $A$ needs to be multiplied with each column of $B$, and then, the results are added. So, the time complexity of multiplication is $O(n^3)$.

Next, we also conducted corresponding experiments in Python and Java.

In Figure 4, we use fixed calculation rounds to calculate the time cost of matrix addition and multiplication in different dimensions. By observing the experimental results, it is not difficult to find that although the execution efficiency is different in different operating environments, the trend of computing costs is the same. This is also consistent with our theoretical analysis results.

In Figure 5, we fix the dimension of the matrix to 50 and increase the number of calculation execution rounds each time (here is the process of iterative calculation during the training process of the simulated machine learning model). It is not difficult to find that with the increase of execution rounds, the cost overhead of multiplication increases greatly, while the trend of increase in the cost overhead of addition is relatively gentle, because, with the increase of calculation rounds, the gap of single calculation overhead is enlarged.

Therefore, through the above theoretical and experimental results, we can know that the matrix multiplication calculation cost is higher than the addition. In the field of machine learning, the size of the matrix (here refers to a small data set, because mobile devices and wearable devices cannot handle, store, and calculate large data sets) and the number of iterations are often relatively large. Therefore, it is necessary to determine which calculation overheads are needed to ensure that the device effectively offloads the calculation amount.

*5.3. System Performance Analysis.* For the performance of the system, this paper uses a real data set for comparison experiments. In the calculation process, the scheme uses a block matrix for calculation, which ensures that the calculation result is the same as the original calculation result. According to the characteristics of the block matrix calculation, it can be seen that the accuracy of the training model is not affected. Therefore, 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 Figure 6: the method proposed in this paper takes 24.76 ms, and the general training 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. Afterward, 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 Figure 7.

It can be seen from Figure 7 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.

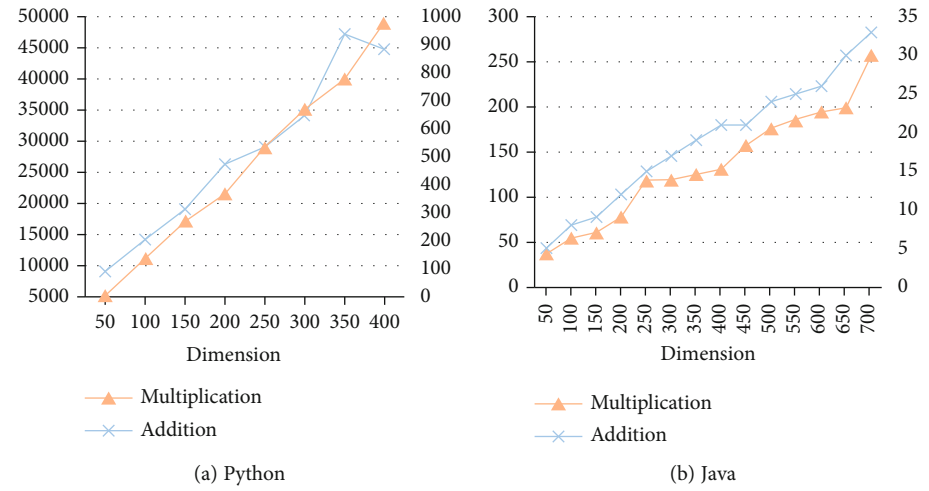
From the experimental results, the scheme in this paper saves time by approximately 20%, when compared with the unoptimized scheme. The machine learning algorithm used in this paper is relatively basic and has a small number of calculations. The results have a certain advantage in terms of time consumption. 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 paper is suitable for application scenarios where the edge device cannot execute or is difficult to handle.

At the same time, as shown in Figure 8, this paper 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 it is because the addition homomorphic encryption time based on Paillier [48] 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 at this stage.

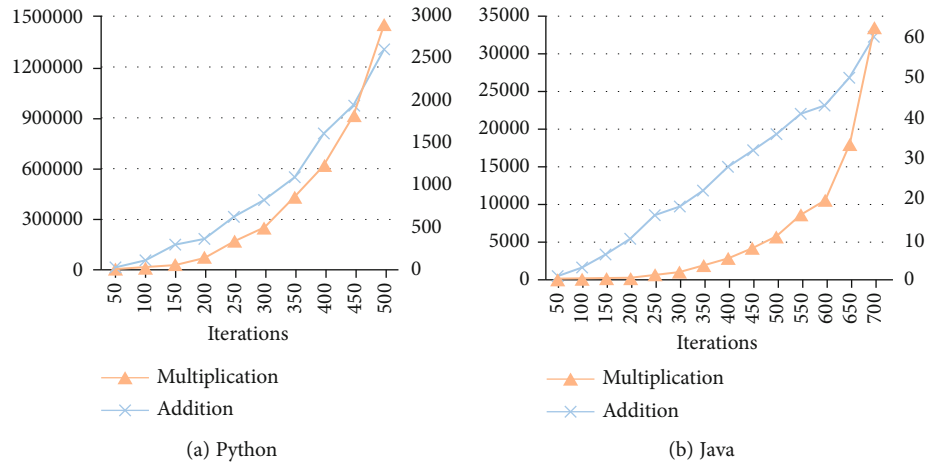
Figure 9 is a comparison of the data processing time required at the edge nodes. In the scheme in this paper, Algorithm 1 requires edge node calculation and generation, and the matrix inversion process can also be outsourced. Therefore, we calculate the computational cost of Algorithm 1. At the same time, we also compared the computational cost of the additive homomorphic encryption scheme in the encryption process (because this process is consistent with the purpose of Algorithm 1 in this paper). It is not difficult to find from Figure 9 that the program execution efficiency in this paper is higher and the trend is more gradual.

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




(a) Python (b) Java

Figure 4: Fixed matrix size (take 10 as an example).




(a) Python (b) Java

Figure 5: Fixed number of iterations (take the matrix with a dimension of 50 as an example).

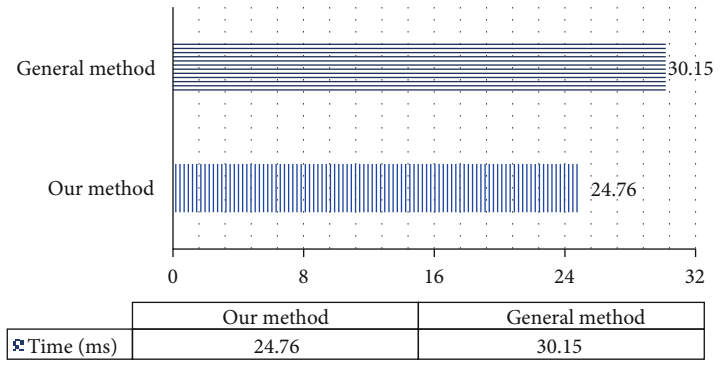



Figure 6: Comparison of time consumptions for the Boston house price prediction data set.

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.

As shown in Table 1, we also compared the performance of various aspects. First, we will divide it into five levels from low to high. Because encryption technology will increase the system's computational overhead, it also has an advantage in security. On the contrary, in the case of not using

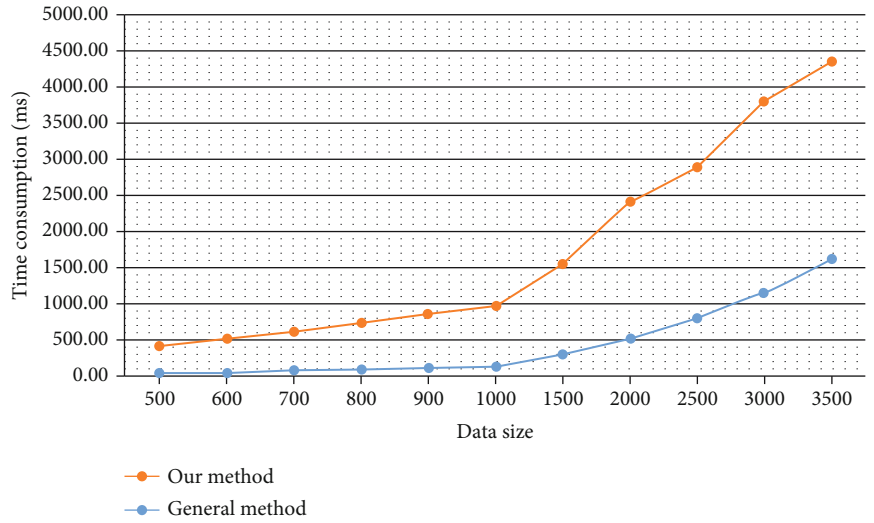



Figure 7: Comparison of the efficiencies of the proposed scheme and the general method.

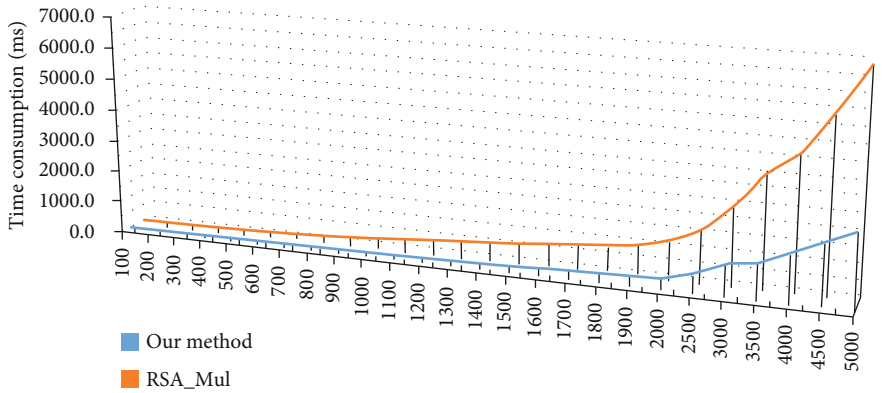



Figure 8: Comparison of the efficiencies of this scheme and RSA multiplication homomorphic encryption.

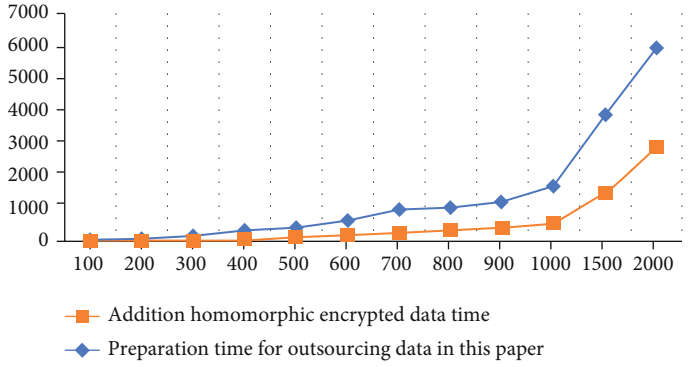



Figure 9: Comparison of data processing time required on edge nodes.

Table 1: Comparison of various performance indicators of the scheme.

| | Computational complexity | Required computing power | Security | Data fidelity |
|---|---|---|---|---|
| General outsourcing training program | Low | Low | Low | High |
| Differential privacy | Medium | Medium | Higher | Low |
| Based on homomorphic encryption [49] | High | High | High | High |
| EVPP (our scheme) | Lower | Lower | Higher | High |

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 $C_2$ , $C_1$, $C_3$, and $C_4$.
1: Calculate $D = D_1 + D_2$;
2: Go to 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) $w' \leftarrow w^T D$ and tag set $Y' \leftarrow \alpha YD$
5: Design calculation rules $f() = C_4 - C_2/1 + e^{-C_3 C_4}$;
6: According to the calculation rules, solve the verification factor det $V$;
7: Construct the sample set $X_1' \leftarrow D^{-1}X$ and its transpose matrix $X_2' \leftarrow \alpha X$
8: return $C_2 \leftarrow (X_2' \| M_2 \| V_2)$, $C_1 \leftarrow (X_1' \| M_1 \| V_1)$, $C_3 \leftarrow (w' \| M_3 \| V_3)$, $C_4 \leftarrow (Y' \| M_4 \| V_4)$

Algorithm 7: Outsourcing matrix generation algorithm (logistic regression).

cryptographic schemes or using data perturbation schemes, performance will be more prominent, but it will bring a series of problems such as security and fidelity. Therefore, we evaluated and compared the schemes in the table.

Although the outsourcing scheme using homomorphic encryption has advantages in security and fidelity schemes, its complexity is higher. The scheme of using differential privacy will bring some loss in data accuracy. The solution proposed in this paper is slightly weaker in terms of security, but it ensures that the accuracy of the data is not lost. That can effectively protect user privacy and save power on mobile devices.

## 6. Discussion on the Application of the Scheme in Other Algorithms

*6.1. Analysis of Application Background.* In the field of scientific computing, matrix operations are often used. In the field of machine learning, the optimal value cannot be obtained due to the results of one calculation. Therefore, it is necessary for the user (device) to go back to the calculation so that the calculation result can better approach the optimal value. This is often referred to as the model training process.

In general equipment, such as personal computers and notebooks, training machine learning models are generally acceptable in terms of computing costs, power consumption, data storage, and so on. With the development of Internet of Things technology, more and more smartphones, wearable devices, etc., have been applied. However, the common problems of these devices are (1) the power storage capacity is weak, and they need to work continuously for a longer time; (2) the computing power of these devices is weak; and (3) the storage space of the device is limited. Therefore, this limits the data processing and computing capabilities of the device.

Next, we will introduce the application scenarios of our scheme. Taking the smart bracelet as an example, users often use it to monitor the heart rate. If the user needs to monitor the heart rate status frequently or continuously, the power consumption of the device will be increased first, and secondly, the amount of data generated will increase as the monitoring frequency increases. For users with abnormal heart function, they may need to monitor heart rate changes in real time and be able to prompt the user in case of an abnormal heart rate. In this case, we need to get results as soon as possible while protecting user data.

Therefore, our scheme can be applied to training user models, which includes initial training of user models and later adjustment of models. The solution can reduce the computing pressure of the device and improve efficiency, while protecting user data privacy as much as possible.

*6.2. Application in Other Machine Learning Algorithms.* Next, we will take the logistic regression algorithm as an example to briefly introduce how our scheme is applied to other algorithms.

First, we give the expression of the logistic regression [50] model:

$$y = \frac{1}{1 + e^{-w^T x}}. \tag{12}$$

Because, in the early stage of model training or application process, the value of $w$ is not optimal, we need to train $w^*$ to find the value that is most similar to $w$. We still use a gradient descent algorithm for optimization. Among them, $N$ is the data or sample size. We only need to adjust Algorithm 1, and the implementation details will be given in Algorithm 13. Here, we only need to formulate the calculation rules according to Formula (13). The other algorithms are the same:

$$w_{t-1} = w_t + \alpha \sum_{n=1}^{N} \left( y_n - \frac{1}{1 + e^{-w_t^T x_n}} \right) x_n. \tag{13}$$

## 7. 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 proposed EVPP: a secure data outsourcing computing solution based on matrix

operations. In order to ensure that complex computing tasks can be effectively offloaded, we conducted theoretical and experimental analysis. Through the analysis of the results, we determined which calculation operations should be outsourced, effectively reducing the computing pressure of edge nodes. 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 that the data generates arbitration evidence so that other nodes in the system can verify and avoid sending outsourced tasks to malicious nodes. Through 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 paper only optimizes the training process, it has certain limitations. To better reduce the complexity of machine learning models for equipment training in edge environments and, at the same time, due to the insufficient data volume of a single mobile device, how to build a distributed algorithm is also a problem worth considering. Therefore, the next step will be to combine federal learning [51, 52] to study the problem of collaboration between multiple nodes in a distributed system.

## Data Availability

The data sets used in this paper are all open source network data sets.

## Disclosure

This paper is an expanded version of the SPNCE2020 conference.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This work is supported by the National Key Research and Development Project (2017YFB0801805) and the National Natural Science Foundation of China (61671360 and 62072359).

## References

[1] C. Petrov, "Big data statistics," 2019, 2019 03 https://techjury.net/stats-about/big-datastatistics/.

[2] X. Zhang, M. Qiao, L. Liu, Y. Xu, and W. Shi, "Collaborative cloud-edge computation for personalized driving behavior modeling," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pp. 209–221, Washington DC, USA, 2019.

[3] K. Jia, H. Li, D. Liu, and S. Yu, "Enabling efficient and secure outsourcing of large matrix multiplications," in *2015 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, San Diego, CA, USA, 2015.

[4] X. Lei, X. Liao, T. Huang, and F. Heriniaina, "Achieving security, robust cheating resistance, and high-efficiency for outsourcing large matrix multiplication computation to a malicious cloud," *Information Sciences*, vol. 280, pp. 205–217, 2014.

[5] P. Li, J. Li, Z. Huang, C. Z. Gao, W. B. Chen, and K. Chen, "Privacy-preserving outsourced classification in cloud computing," *Cluster Computing*, vol. 21, no. 1, pp. 277–286, 2018.

[6] A. A. Abdellatif, A. Mohamed, C. F. Chiasserini, M. Tlili, and A. Erbad, "Edge computing for smart health: context-aware approaches, opportunities, and challenges," *IEEE Network*, vol. 33, no. 3, pp. 196–203, 2019.

[7] R. K. Pathinarupothi, P. Durga, and E. S. Rangan, "IoT-based smart edge for global health: remote monitoring with severity detection and alerts transmission," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2449–2462, 2018.

[8] S. Chen, H. Wen, J. Wu et al., "Radio frequency fingerprint-based intelligent mobile edge computing for internet of things authentication," *Sensors*, vol. 19, no. 16, p. 3610, 2019.

[9] I. Froiz-Míguez, T. Fernández-Caramés, P. Fraga-Lamas, and L. Castedo, "Design, implementation and practical evaluation of an IoT home automation system for fog computing applications based on MQTT and ZigBee-WiFi sensor nodes," *Sensors*, vol. 18, no. 8, p. 2660, 2018.

[10] J. R. Torres Neto, G. P. Rocha Filho, L. Y. Mano, L. A. Villas, and J. Ueyama, "Exploiting offloading in IoT-based microfog: experiments with face recognition and fall detection," *Wireless Communications and Mobile Computing*, vol. 2019, Article ID 2786837, 13 pages, 2019.

[11] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[12] W. Li, S. Zhang, Q. Su, Q. Wen, and Y. Chen, "An anonymous authentication protocol based on cloud for telemedical systems," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 8131367, 12 pages, 2018.

[13] W. Ding, R. Hu, Z. Yan et al., "An extended framework of privacy-preserving computation with flexible access control," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 918–930, 2020.

[14] Q. Li, H. Zhu, J. Xiong, R. Mo, Z. Ying, and H. Wang, "Fine-grained multi-authority access control in IoT-enabled mhealth," *Annals of Telecommunications*, vol. 74, no. 7-8, pp. 389–400, 2019.

[15] K. T. Chui, R. W. Liu, M. D. Lytras, and M. Zhao, "Big data and IoT solution for patient behaviour monitoring," *Behaviour & Information Technology*, vol. 38, no. 9, pp. 940–949, 2019.

[16] X. Liu, R. H. Deng, K. R. Choo, and Y. Yang, "Privacy-preserving reinforcement learning design for patient-centric dynamic treatment regimes," *IEEE Transactions on Emerging Topics in Computing*, 2019.

[17] X. Liang, X. Du, G. Wang, and Z. Han, "A deep reinforcement learning network for traffic light cycle control," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1243–1253, 2019.

[18] P. Zhou, T. Braud, A. Alhilal, P. Hui, and J. Kangasharju, "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, Kyoto, Japan, 2019.

[19] J. Joo, M. C. Park, D. S. Han, and V. Pejovic, "Deep Learning-Based Channel Prediction in Realistic Vehicular Communications," *IEEE Access*, vol. 7, pp. 27846–27858, 2019.

[20] B. Feng, Q. Fu, M. Dong, D. Guo, and Q. Li, "Multistage and elastic spam detection in mobile social networks through deep learning," *IEEE Network*, vol. 32, no. 4, pp. 15–21, 2018.

[21] Y. Yang, X. Huang, X. Liu et al., "A comprehensive survey on secure outsourced computation and its applications," *IEEE Access*, vol. 7, pp. 159426–159465, 2019.

[22] X. Liu, R. H. Deng, K. R. Choo, and Y. Yang, "Privacy-preserving outsourced support vector machine design for secure drug discovery," *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 610–622, 2020.

[23] X. Zhang, Y. Wang, S. Lu, L. Liu, L. Xu, and W. Shi, "OpenEI: an open framework for edge intelligence," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1840–1851, Dallas, TX, USA, 2019.

[24] Y. Sun, Q. Wen, Y. Zhang, H. Zhang, Z. Jin, and W. Li, "Two-cloud-servers-assisted secure outsourcing multiparty computation," *The Scientific World Journal*, vol. 2014, Article ID 413265, 7 pages, 2014.

[25] P. Mohassel and Y. Zhang, "SecureML: a system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 19–38, San Jose, CA, USA, 2017.

[26] K. Huang, X. Liu, S. Fu, D. Guo, and M. Xu, "A lightweight privacy-preserving CNN feature extraction framework for mobile Sensing," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2019.

[27] A. M. Vengadapurvaja, G. Nisha, R. Aarthy, and N. Sasikaladevi, "An efficient homomorphic medical image encryption algorithm for cloud storage security," *Procedia Computer Science*, vol. 115, pp. 643–650, 2017.

[28] C. Piao, Y. Shi, J. Yan, C. Zhang, and L. Liu, "Privacy-preserving governmental data publishing: a fog-computing-based differential privacy approach," *Future Generation Computer Systems*, vol. 90, pp. 158–174, 2019.

[29] S. Salinas, C. Luo, X. Chen, W. Liao, and P. Li, "Efficient secure outsourcing of large-scale sparse linear systems of equations," *IEEE Transactions on Big Data*, vol. 4, no. 1, pp. 26–39, 2018.

[30] Y. Rahulamathavan, R. C. Phan, S. Veluru, K. Cumanan, and M. Rajarajan, "Privacy-preserving multi-class support vector machine for outsourcing the data classification in cloud," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 5, pp. 467–479, 2014.

[31] X. Li, Y. Zhu, J. Wang, Z. Liu, Y. Liu, and M. Zhang, "On the soundness and security of privacy-preserving SVM for outsourcing data classification," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 906–912, 2018.

[32] F. Liu, W. K. Ng, and W. Zhang, "Encrypted SVM for outsourced data mining," in *2015 IEEE 8th International Conference on Cloud Computing*, pp. 1085–1092, New York, NY, USA, 2015.

[33] F. Liu, W. K. Ng, and W. Zhang, "Encrypted gradient descent protocol for outsourced data mining," in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, pp. 339–346, Gwangiu, South Korea, 2015.

[34] Y. Li, Z. L. Jiang, X. Wang, J. Fang, E. Zhang, and X. Wang, "Securely outsourcing ID3 decision tree in cloud computing," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 2385150, 10 pages, 2018.

[35] X. Zhang, X. Chen, J. Wang, Z. Zhan, and J. Li, "Verifiable privacy-preserving single-layer perceptron training scheme in cloud computing," *Soft Computing*, vol. 22, no. 23, pp. 7719–7732, 2018.

[36] L. Liu, J. Su, X. Liu et al., "Toward highly secure yet efficient KNN classification scheme on outsourced cloud data," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9841–9852, 2019.

[37] Q. Wu, F. Zhou, J. Xu, D. Feng, and B. Li, "Lightweight privacy-preserving equality query in edge computing," *IEEE Access*, vol. 7, pp. 182588–182599, 2019.

[38] L. Tao, Z. Li, and L. Wu, "*Outlet*: outsourcing wearable computing to the ambient mobile computing edge," *IEEE Access*, vol. 6, pp. 18408–18419, 2018.

[39] S. Salinas, C. Luo, X. Chen, and P. Li, "Efficient secure outsourcing of large-scale linear systems of equations," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 1035–1043, Kowloon, Hong Kong, 2015.

[40] Y. Yu, Y. Luo, D. Wang, S. Fu, and M. Xu, "Efficient, secure and non-iterative outsourcing of large-scale systems of linear equations," in *2016 IEEE International Conference on Communications (ICC)*, pp. 1–6, Kuala Lumpur, Malaysia, 2016.

[41] X. Lei, X. Liao, T. Huang, and H. Li, "Cloud computing service: the caseof large matrix determinant computation," *IEEE Transactions on Services Computing*, vol. 8, no. 5, pp. 688–700, 2015.

[42] F. Chen, T. Xiang, X. Lei, and J. Chen, "Highly efficient linear regression outsourcing to a cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 4, pp. 499–508, 2014.

[43] L. Zhou, Y. Zhu, and K. K. R. Choo, "Efficiently and securely harnessing cloud to solve linear regression and other matrix operations," *Future Generation Computer Systems*, vol. 81, pp. 404–413, 2018.

[44] C. Hu, A. Alhothaily, A. Alrawais, X. Cheng, C. Sturtivant, and H. Liu, "A secure and verifiable outsourcing scheme for matrix inverse computation," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pp. 1–9, Atlanta, GA, USA, 2017.

[45] D. He, D. Wang, Q. Xie, and K. Chen, "Anonymous handover authentication protocol for mobile wireless networks with conditional privacy preservation," *Science China Information Sciences*, vol. 60, no. 5, 2017.

[46] S. Qiu, D. Wang, G. Xu, and S. Kumari, "Practical and provably secure three-factor authentication protocol based on extended chaotic-maps for mobile lightweight devices," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2020.

[47] Z. Guan, X. Liu, L. Wu et al., "Cross-lingual multi-keyword rank search with semantic extension over encrypted data," *Information Sciences*, vol. 514, pp. 523–540, 2020.

[48] "Python-Paillier," December 2019 https://github.com/data61/python-paillier.

[49] F. Bergamaschi, S. Halevi, T. T. Halevi, and H. Hunt, "Homomorphic Training of 30,000 Logistic Regression Models," in *International Conference on Applied Cryptography and Network Security*, pp. 592–611, Springer, 2019.

[50] E. Alpaydin, *Introduction to Machine Learning (3nd. ed.)*, The MIT Press, 2014.

[51] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, and Q. Yang, "SecureBoost: a lossless federated learning framework," 2019, https://arxiv.org/pdf/1901.08755.pdf.

[52] J. Zhang, Y. Zhao, J. Wang, and B. Chen, "FedMEC: improving efficiency of differentially private federated learning via mobile edge computing," *Mobile Networks and Applications*, vol. 3, 2020.