






Research Article

ePMLF: Efficient and Privacy-Preserving Machine Learning Framework Based on Fog Computing

Ruoli Zhao ¹, Yong Xie ¹, Hong Cheng ^{2,3}, Xingxing Jia ⁴, and Syed Hamad Shirazi ⁵

¹Department of Computer Technology and Applications, Qinghai University, Xining, China

²School of Statistics and Mathematics, Shanghai Lixin University of Accounting and Finance, Shanghai 201209, China

³Xining University No. 231, Haihu Avenue, Haihu Avenue, Chengbei District, Xining, Qinghai, China

⁴School of Mathematics and Statistics, Lanzhou University, Lanzhou, China

⁵Department of Information Technology, Hazara University, Mansehra 21300, Pakistan

Correspondence should be addressed to Yong Xie; mark.y.xie@qq.com

Received 9 September 2022; Revised 19 November 2022; Accepted 30 November 2022; Published 27 February 2023

Academic Editor: Paolo Gastaldo

Copyright © 2023 Ruoli Zhao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the continuous improvement of computation and communication capabilities, the Internet of Things (IoT) plays a vital role in many intelligent applications. Therefore, IoT devices generate a large amount of data every day, which lays a solid foundation for the success of machine learning. However, the strong privacy requirements of the IoT data make its machine learning very difficult. To protect data privacy, many privacy-preserving machine learning schemes have been proposed. At present, most schemes only aim at specific models and lack general solutions, which is not an ideal solution in engineering practice. In order to meet this challenge, we propose an efficient and privacy-preserving machine learning framework (ePMLF) in a fog computing environment. The ePMLF framework can let the software service provider (SSP) perform privacy-preserving model training with the data on the fog nodes. The security of the data on the fog nodes can be protected and the model parameters can only be obtained by SSP. The proposed secure data normalization method in the framework further improves the accuracy of the training model. Experimental analysis shows that our framework significantly reduces the computation and communication overhead compared with the existing scheme.

1. Introduction

At present, the application of the IoT can realize the real-time collection of user data. The large amount of data generated every day provides a good basis for training high-quality machine learning models. However, privacy issues hinder the application of machine learning [1, 2]. On the one hand, training data contains people's sensitive information, which is not allowed to be disclosed. On the other hand, the trained machine learning model is a valuable property of SSP and the leakage of the model will bring serious economic losses to SSP. Therefore, the training process faces a serious problem of privacy disclosure. The strong privacy requirements of data make training very difficult.

In order to solve the problem of privacy disclosure in the training process, many privacy-preserving machine learning

training schemes have been proposed. There are two main solutions: secure collaborative training and secure outsourcing training. The main application scenario of secure collaborative training is federated learning [3]. In federated learning, clients complete the training locally. Then, upload and download model parameters to the server. Therefore, frequent interaction leads to a high communication overhead of federated learning [4]. In addition, the goal of federated learning is to train a global model for each data provider, which is different from ours [5]. In this paper, the goal of our proposed framework is to train a private model for SSP by using the ciphertext data of data providers (fog nodes). Secure outsourcing training is mainly based on cloud computing. Cloud servers provide a lot of storage and computation resources. Data providers use homomorphic encryption or secret sharing technologies to convert their

data into ciphertext data and outsource them to cloud servers. In [6–8], they all adopt the dual cloud server model and assume that the two cloud servers do not collude. However, this assumption has potential security hazards [5].

Through the above analysis, the existing privacy-preserving schemes mainly have two problems. First, these schemes are only for specific machine learning algorithms and cannot meet all machine learning algorithms. The second is the lack of global normalization processing of data. Normalization without disclosing the data can make the training process converge to the optimal solution more easily. To solve these problems, Zhu et al. [5] proposed a privacy-preserving ML training framework, which contains multiple secure training protocols for the aggregation scenario and defends security under collusion situations. However, the scheme in [5] does not make a global data normalization. In addition, there are still some shortcomings to be improved, including the function of building blocks is not comprehensive and high communication and computation overhead, which makes the framework impractical.

Therefore, we propose an efficient and practical privacy-preserving machine learning training framework in a fog computing environment. Specifically, with the continuous increase of data, it brings a heavy storage burden to IoT devices with limited resources. Therefore, deploying fog nodes with higher configurations near IoT devices has become an effective solution [9]. IoT devices will store the collected data in nearby fog nodes. We assume that a SSP wants to use the data in the fog nodes to train its own private model. In the training process, the data in the fog nodes will not be leaked to SSP and other fog nodes. The model of SSP will not be leaked to fog nodes. Our contributions are as follows:

- (1) Based on the requirements of data privacy-preserving in the IoT environment and the characteristics of fog computing, we propose ePMLF. Through ePMLF, fog nodes can protect the privacy of the data and let an untrusted SSP train different machine-learning models. At the same time, ePMLF provides the function of model updating.
- (2) The ePMLF implements secure data processing. We propose two secure data normalization methods (secure z-score and secure min-max), which can normalize the data distribution of all fog nodes and form high-quality training data. Based on the OU cryptosystem, we propose a method to encrypt negative numbers. Unlike the existing scheme [10], which can only encrypt negative numbers by the party holding the private key, our proposed method can allow any party to encrypt negative numbers.
- (3) In ePMLF, we define the ciphertext as the encrypted data and its precision, which can prevent the resulting error caused by inconsistent precision in ciphertext computation. Then, we design a precision control protocol, which can avoid the plaintext overflow caused by multiple homomorphic operations. Based on the ciphertext defined by ePMLF, we propose some secure algorithms as the basic building blocks of the framework. Experimental results show that these algorithms are helpful to improve training efficiency.
- (4) We implement the proposed framework. Strict security analysis shows that our framework meets the strong privacy-preserving requirements of all participants. Through comparative evaluation, our scheme significantly reduces the communication and computation overhead.

2. Related Work

The existing privacy-preserving machine learning training scenarios are mainly divided into two categories: secure collaborative training and secure outsourcing training.

In the scenario of secure collaborative training, each participant has some computation resources and their own data. Therefore, each participant undertakes some computation and communication tasks and cooperatively trains a global machine-learning model on the joint training data. Data owners should keep their data confidential during the training [11–13]. Dani et al. [14] proposed protocols for solving the secure multi-party computation problem. Mehnaz et al. [15] proposed a secure sum protocol with strong security guarantees and used this protocol to propose two secure gradient-descent algorithms. Saha et al. [16] proposed a fog-enabled federated learning framework-FogFL to facilitate distributed learning and reduce communication latency and energy consumption of resource-constrained edge devices. Xu et al. [17] presented a secure and verifiable federated learning scheme, with which federated deep learning is achieved and the final learning results are verifiable. Wang et al. [18] proposed a privacy-preserving federated learning scheme for regression training, which is noninteractive in the whole training process. Zhao et al. [19] proposed a collaborative architecture based on orbital edge computing and low-orbit satellite network communication.

In the scenario of secure outsourcing training, data owners with limited computation resources outsource their ciphertext data to cloud servers. The cloud servers perform the privacy-preserving machine learning training process and train a private machine learning model for the training service requester. Liu et al. [6] designed a system for privacy-preserving decision tree training and evaluation in a twin-cloud architecture. Liu et al. [7] proposed a secure ML-kNN training and classification scheme. Wang et al. [8] proposed an efficient privacy-preserving outsourced SVM scheme, which protects the privacy of training data and the SVM model. Zhang et al. [20] proposed a secure deep computation model by offloading the expensive operations to the cloud. Li et al. [21] proposed an outsourced privacy-preserving C4.5 algorithm over horizontally and vertically partitioned data for multiple parties. Li et al. [22] proposed a privacy-preserving multi-party machine learning framework and the data owners do not need to participate in the training process. Liu et al. [23] proposed a privacy-preserving clinical decision support system in the outsourced cloud computing environment. Deploying machine learning services in the cloud has become a flexible training solution.

However, this approach generally relies on the assumption of noncollusion, which is a serious security risk.

3. Preliminaries

3.1. Homomorphic Encryption

3.1.1. Okamoto-Uchiyama (OU) Cryptosystem. OU cryptosystem is a public key cryptosystem with additive homomorphism [24]. We will introduce OU cryptosystem as follows:

- (i) Key Generation: choose two big primes $p, q, |p| = |q| = l, n = p^2 \cdot q, L(x) = (x - 1)/p$. Generate a random number $g \in Z_n^*$. Compute $h = g^n \bmod n$. The public key is $pk = (n, g, h, l)$. The private key is $sk = (p, q)$.
- (ii) Encryption: given $m \in [0, 2^{l-1}]$. The message m will be encrypted with pk . The ciphertext is $c = g^m h^r \bmod n$, where $r \in Z_n$ is a random number.
- (iii) Decryption: given a ciphertext c . Compute $m = (L(c^{p-1} \bmod p^2) / L(g^{p-1} \bmod p^2)) \bmod p$.
- (iv) Homomorphic computation: given two ciphertexts $[m_1], [m_2]$ under the same public key pk . The homomorphic computations are defined as $[m_1 + m_2] = [m_1] \cdot [m_2], [m_1 \cdot m_2] = [m_2]^{m_1}$.

3.2. Cloud-ElGamal Cryptosystem. In this paper, we select an enhanced version of ElGamal called Cloud-ElGamal [25], which supports multiplicative homomorphism and resists confidentiality attacks.

- (i) Key Generation: choose a big prime p . Find a generator a of Z_p^* . Generate a random integer $d, 1 < d < p - 1$ and compute $y = a^d \bmod p$. The evaluation key is $ek = p$. The private key is $sk = (a, d, y)$.
- (ii) Encryption: given $m \in Z_p$. Generate a random integer $k, 1 < k < p - 1$ and compute $c_1 = a^k \bmod p, c_2 = y^k m \bmod p$. The ciphertext is $c = (c_1, c_2)$.
- (iii) Decryption: Given a ciphertext c . Compute $m = c_2 \cdot (c_1^d)^{-1} \bmod p$.
- (iv) Homomorphic computation: given two ciphertexts $[m_1], [m_2]$ under the same private key. The homomorphic computations are defined as $[m_1 \cdot m_2] = [m_1] \cdot [m_2]$.

3.3. Machine Learning Training. Machine learning training consists of data preprocessing and model training.

For data preprocessing, we focus on data normalization for continuous data (such as age and height). There are two main methods of data normalization: the min-max method and the z-score method.

The min-max method maps the values of attributes between 0 and 1 according to the maximum and minimum values of attributes in the dataset. For attribute f , the min-max method will compute as follows:

$$x' = \frac{x - \min(f)}{\max(f) - \min(f)}. \quad (1)$$

The z-score method is to standardize attributes based on the average and standard deviation of attributes in the dataset. For attribute f , we assume that the average of f is μ_f and the standard deviation of f is σ_f . The z-score method will compute as follows:

$$x' = \frac{x - \mu_f}{\sigma_f}. \quad (2)$$

Through data normalization, high-quality training data can be formed.

For model training, there are many model training algorithms, which include many linear and nonlinear operations. For example, logistic regression, SVM, and naive Bayes. To train a logistic regression model, the parameter w will be updated by computing $w = w - \text{learn_rate} \cdot x(1/(1 + e^{-wx}) - y)$. To train a SVM model, the parameter w will be updated by computing $w = w - \text{learn_rate} \cdot (z \cdot w - z \cdot yx)$. For training a naive Bayes model, we should compute the class prior probability $P(y)$ and the conditional probability $P(x|y)$.

4. System Overview

In this section, we will introduce our proposed framework ePMLF, including the system model, design goal, and threat model.

4.1. System Model. In our system model, the goal is to train a private machine-learning model for SSP. The training task is completed by fog nodes and SSP. At the same time, the privacy data of each fog node will not be leaked to SSP and other fog nodes. SSP's trained model will not be leaked to fog nodes. Therefore, our system model is designed as shown in Figure 1.

There are four participants in our system model, which are trusted authority (TA), IoT devices, fog nodes (FNs), and software service provider (SSP).

- (i) Trusted authority (TA): TA is a trusted authority and generates system parameters for all participants.
- (ii) IoT devices: IoT devices will produce a large amount of IoT data. However, their computation and storage resources are very limited.
- (iii) Fog nodes (FNs): fog nodes have strong computation and storage capacity. They collect, store and manage the data generated by IoT devices. The data owned by FN belongs to sensitive information and cannot be leaked.
- (iv) Software service provider (SSP): SSP wants to train machine learning models through the data in the FN. SSP is not trusted and will try to obtain the data of FN during the training processing.

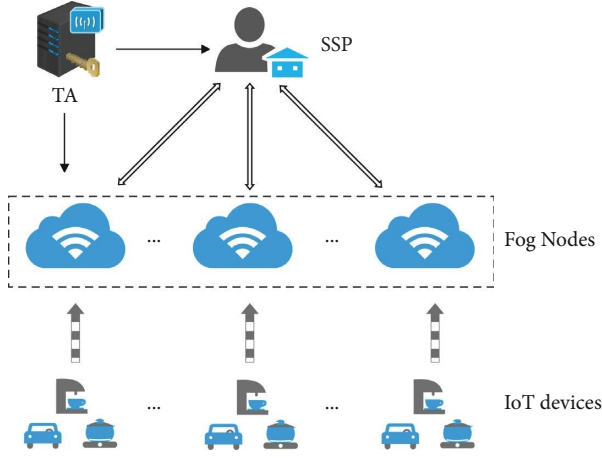


FIGURE 1: System model.

4.2. Design Goal. Based on the system model, our design goal includes privacy-preserving, training high-accuracy machine learning models, and low computation and communication overhead.

4.2.1. Privacy Preserving

- (1) The data of FNs should be preserved
- (2) The trained model of SSP should be preserved

4.2.2. Training High Accuracy Machine Learning Model. When using ciphertext data from FNs to train machine learning models, it is very important to ensure the high accuracy of the trained models. Therefore, we need to generate high-quality training data through data normalization and design a general computing framework.

4.2.3. Low Computation and Communication Overhead. For achieving privacy-preserving training, the proposed ePMLF is designed based on cryptography technology. Cryptography technology will bring significant computation and communication overhead, so we should improve the framework efficiency as much as possible.

4.3. Threat Model. In our proposed ePMLF, we assume that FNs and SSPs are honest-but-curious (TA is trusted). They will implement the protocol honestly, but they try to obtain the data of other participants by analyzing the results of the processing.

At the same time, we allow $(m - 1)$ FNs at most collude with each other to analyze the privacy of other participants

and $(m - 2)$ FNs at most collude with SSP to analyze the privacy of other participants, which is the same as [5]. To prove our proposed scheme is secure, we define an adversary. The adversary can eavesdrop and analyze data during the data transmission. The data transmission process is the interaction between participants in protocol implementation.

5. Our Proposed Framework

In this section, we describe our proposed framework in detail. The ePMLF mainly includes system initialization, data normalization, basic building blocks, privacy-preserving machine learning training, and machine learning model updating. The workflow of our proposed framework is shown in Figure 2.

In order to accurately describe our proposed scheme, we give the description of used notations in Table 1.

5.1. System Initialization. In the system initialization phase, TA generates system parameters for all participants.

Improved OU encryption.

Zhang et al. [10] achieved the negative integers encryption based on the OU cryptosystem. They divided the plaintext space into two parts as follows: $[0, p/2)$ represents positive integers and $(p/2, p - 1]$ represents negative. For a negative integer $-m$, it should be converted to $p - m$. It should be noted that p is the private key of OU cryptosystem.

Through the above analysis, we can find that the negative integers encryption method proposed by Zhang et al. can only be implemented by the participants with the private key, which is not practical. Therefore, we propose a new method to realize that any participants can encrypt negative integers based on the OU cryptosystem without disclosing the private key. Specifically, TA generates a large prime q' after generating p, q . TA computes $N = p \cdot q'$ and public N, q . When a participant without a private key needs to encrypt the negative integer $-m$, computing $N - m$ and encrypting it. In decryption, N is eliminated by modulus p . The range of encrypted integers is $(-p/2, p/2)$. We prove the correctness of our proposed method as follows.

Negative integers encryption

$$c = g^{N-m} h^r \text{ mod } n. \quad (3)$$

Negative integers decryption

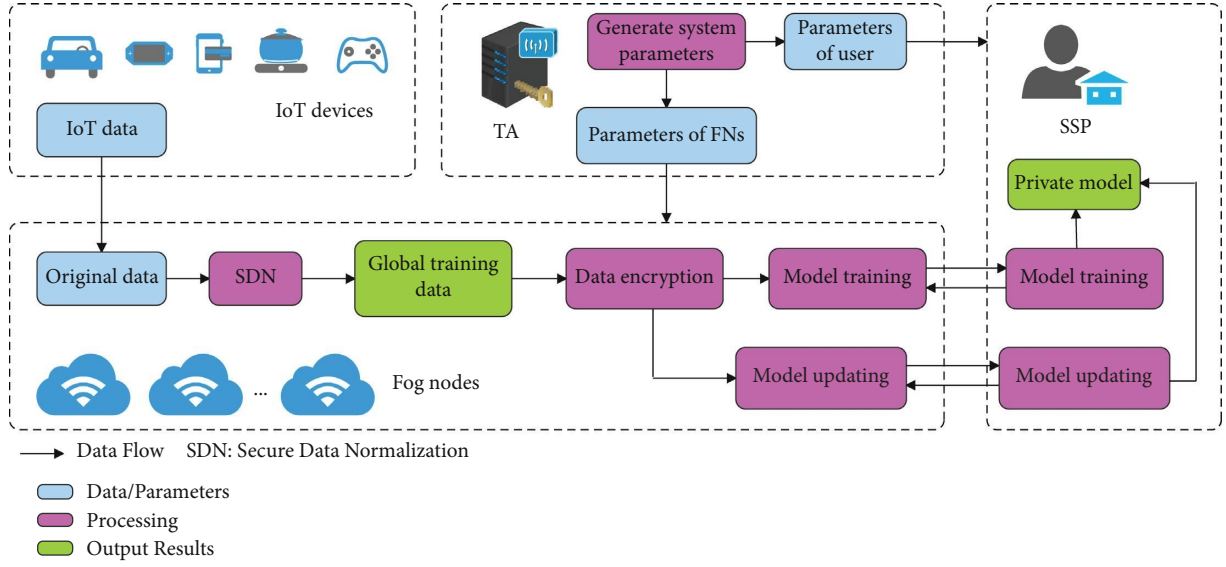


FIGURE 2: System workflow.

TABLE 1: Notation and definition.

Notation	Definition
l	The key length (bit)
$l(x)$	The key length of x
(PK_i^O, SK_i^O)	OU public-private key pair of FN_i
(PK_s^O, SK_s^O)	OU public-private key pair of SSP
(EK_i^E, SK_i^E)	Cloud-ElGamal evaluation-private key pair of FN_i
E	The precision of floating-point numbers
$[x]_{PK}$	The ciphertext of x under PK

$$\begin{aligned}
m &= \frac{L(c^{p-1} \bmod p^2)}{L(g^{p-1} \bmod p^2)} \bmod p \\
&= \frac{L((g^{p-1N-m} \cdot (g^{p-1m} \bmod p^2)) \bmod p^2)}{L(g^{p-1} \bmod p^2)} \bmod p \\
&= \frac{L((g^{p-1})^{N-m} \bmod p^2) + L((g^{p-1})^m \bmod p^2)}{L(g^{p-1} \bmod p^2)} \bmod p \\
&= \frac{L((1+kp)^{N-m} \bmod p^2) + L((1+kp)^m \bmod p^2)}{L(1+kp)} \bmod p \\
&= \frac{L((1+(N-m)kp) \bmod p^2) + L(1)}{L(1+kp)} \bmod p \\
&= \frac{L((1+(pq'-m)kp) \bmod p^2) + L(1)}{L(1+kp)} \bmod p \\
&= \frac{L((1+p^2kq' - mkp) \bmod p^2) + L(1)}{L(1+kp)} \bmod p \\
&= \frac{L(1 - mkp) + L(1)}{L(1+kp)} \bmod p.
\end{aligned} \tag{4}$$

Generate system parameters.

- (1) We assume that there are m fog nodes in our system. TA generates a OU public-private key pair $(PK_i^O = (n_i, g_i, h_i, N_i, l), SK_i^O = (p_i, q_i))$ and a Cloud-ElGamal evaluation-private key pair $(EK_i^E = p_i^e, SK_i^E = (a_i, d_i, y_i))$ for FN FN_i .
- (2) TA generates two random integers α, β , $l(\alpha) = l(\beta) < l$ and splits α, β to m random integers, $\alpha = \alpha_1 + \alpha_2 + \dots + \alpha_m$, $\beta = \beta_1 + \beta_2 + \dots + \beta_m$. Then, TA distributes α_i, β_i to FN_i and α, β to SSP by a secure communication channel.
- (3) TA generates a OU public-private key pair $(PK_s^O = (n_s, g_s, h_s, N_s, l), SK_s^O = (p_s, q_s))$ for SSP.

5.2. Privacy Preserving Data Normalization. In our proposed framework, SSP uses the encrypted data of all FNs to train a machine-learning model. Therefore, we need to normalize the data by all FNs, which can improve the quality of training data.

The data format of FN_i is $(x_1^{ij}, x_2^{ij}, \dots, x_d^{ij}, y^{ij})$, $j = 1, 2, \dots, n_i$. In the processing of data normalization, any participants cannot know the data of FN_i .

Secure z-score as follows:

- (1) For the k -th dimension data, FN_i computes $x_k^i = \sum_{j=1}^{n_i} x_k^{ij}$. Then, FN_i encrypts x_k^i with PK_s^O and sends $[x_k^i + \beta_i]_{PK_s^O}$, $i = 1, 2, \dots, m$ and n_i to SSP.
- (2) SSP computes $\bar{x}_k = (\sum_{i=1}^m x_k^i - \beta) / \sum_{i=1}^m n_i$ and sends it to each FN.
- (3) FN_i computes $\sigma = \sqrt{\sum_{j=1}^{n_i} (x_k^{ij} - \bar{x}_k)^2 / n_i}$ and $x_k^i = (x_k^i - \bar{x}_k) / \sigma$.

Secure min-max as follows:

In our proposed secure min-max, the (3) and (4) computation method is the same as [18].

- (1) Each FN computes the maximum and minimum values of each dimensional data. FN_i ($i = 1, 2, \dots, m$) can obtain $(x_1^{i-\max}, x_2^{i-\max}, \dots, x_d^{i-\max})$ and $(x_1^{i-\min}, x_2^{i-\min}, \dots, x_d^{i-\min})$.
- (2) For $x_j^{i-\max}$ ($x_j^{i-\min}$), setting $\text{pos} = 1$. Starting with $i = 2$, FN_{pos} and FN_i input $x_j^{\text{pos-max}}$ ($x_j^{\text{pos-min}}$) and $x_j^{i-\max}$ ($x_j^{i-\min}$) into the comparison protocol [26] respectively to compare. If $x_j^{\text{pos-max}} < x_j^{i-\max}$ ($x_j^{\text{pos-min}} > x_j^{i-\min}$), $\text{pos} = i$.
- (3) For $x_j^{\text{pos-max}}$ ($x_j^{\text{pos-min}}$), FN_{pos} generates a random integer R_j , $\max(R_j^{\min}, l(R_j^{\max})) = l(x_j^{\text{pos-max}} + R_j^{\max} (x_j^{\text{pos-min}} - R_j^{\min}))$ and computes $x_j^{\text{pos-max}} + R_j^{\max} (x_j^{\text{pos-min}} - R_j^{\min})$. Then, FN_{pos} public $x_j^{\text{pos-max}} + R_j^{\max} (x_j^{\text{pos-min}} - R_j^{\min})$.
- (4) After computing the global maximum and minimum values, the data will be standardized according to $(x_1^{\max} + R_1^{\max}, x_2^{\max} + R_2^{\max}, \dots, x_d^{\max})$ and $(x_1^{\min} - R_1^{\min}, x_2^{\min} - R_2^{\min}, \dots, x_d^{\min} - R_d^{\min})$. FN_i computes as follows:

$$x_k^{ij} = \frac{x_k^{ij} - (x_k^{\min} - R_k^{\min})}{(x_k^{\max} + R_k^{\max}) - (x_k^{\min} - R_k^{\min})}. \quad (5)$$

5.3. Basic Building Blocks. In order to make our framework realize the privacy-preserving machine learning training, we will modularize the proposed framework by designing some general building blocks.

5.3.1. Precision Control. In the machine learning training process, many data are floating-point numbers. Therefore, it is necessary to convert floating-point numbers to integers before encrypting. Generally, the conversion method is to multiply the floating-point number by 2^E or 10^E [26], E is the precision. This method will significantly expand the original data. However, the plaintext space of the encryption algorithm is limited. Using the expanded data for homomorphic operation will lead to the problem of plaintext overflow and the precision of the data will change. For example, $(x2^E) \cdot (y2^E) \rightarrow xy2^{2E}$, $xy2^{2E} + z2^E \rightarrow (xy2^E + z)2^E \neq (xy + z)2^E$.

To solve this problem, we propose a method called precision control. We express the encrypted data as (c, t) . The c is the ciphertext data and the t is the precision of the data. For example, $xy \cdot 2^{2E}$ is expressed as $([xy \cdot 2^{2E}], 2)$. Through the method, we can know the current precision of the data. When $E \cdot (t + 1) \geq l$, we need to reduce the precision bits of encrypted data to avoid plaintext overflow. In order to better control the precision, we require that any participant must set $t = 1$ before encrypting the data. The detailed method is as follows and is described in Algorithm 1.

- (1) SSP chooses a random integer $R, l(R) < E$. For $([x2^{t-E}]_{PK_i^O}, t)$, SSP sends $([x2^{t-E} + R2^{t-E}], t)$ to FN_i . For $([x2^{t-E}]_{SK_i^E}, t)$, SSP sends $([x2^{t-E}R]_{SK_i^E}, t)$ to FN_i .
- (2) For $([x2^{t-E} + R2^{t-E}]_{PK_i^O}, t)$, FN_i computes $x2^E + R2^E = x2^{t-E} + R2^{t-E} / 2^{(t-1)E}$. For $([x2^{t-E}R]_{SK_i^E}, t)$, FN_i computes $x2^E R = x2^{t-E}R / 2^{(t-1)E}$. Then, FN_i encrypts $x2^E + R2^E$ (or $x2^E R$) and sends $([x2^E + R2^E]_{PK_i^E}, 1)$ (or $([x2^E R]_{SK_i^E}, 1)$) to SSP.
- (3) For $([x2^E + R2^E]_{PK_i^E}, 1)$, SSP computes $([x2^E]_{PK_i^E}, 1) \leftarrow [x2^E + R2^E]_{PK_i^E} \cdot [N_i - R2^E]_{PK_i^E}$. For $([x2^E R]_{SK_i^E}, 1)$, SSP computes the inverse R^{-1} of R modulo p_i^E and obtains $([x2^E]_{SK_i^E}, 1) \leftarrow [x2^E R]_{SK_i^E} \cdot R^{-1}$.

5.3.2. Secure Addition. Given two ciphertext data encrypted with PK_i^O , $([x2^{t-E}]_{PK_i^O}, t)$ and $([y2^{t-E}]_{PK_i^O}, t)$. SSP will obtain $([(x + y)2^{t-E}]_{PK_i^O}, t) = ([x2^{t-E}]_{PK_i^O} \cdot [y2^{t-E}]_{PK_i^O}, t)$.

5.3.3. Secure Subtraction. Given two ciphertext data encrypted with PK_i^O , $([x2^{t-E}]_{PK_i^O}, t)$ and $([y2^{t-E}]_{PK_i^O}, t)$. SSP will obtain $([(x - y)2^{t-E}]_{PK_i^O}, t) = ([x2^{t-E}]_{PK_i^O} \cdot [y2^{t-E}]_{PK_i^O}^{N_i-1}, t)$.

Input: $([x2^{t-E}]_{PK_i^O}, t)$ or $([x2^{t-E}]_{SK_i^E}, t)$,
 $i \in [1, m]$
Output: $([x2^E]_{PK_i^O}, 1)$ or $([x2^E]_{SK_i^E}, 1)$
SSP:
(1) Send $([x2^{t-E} + R2^{t-E}]_{PK_i^O}, t)$
or $([x2^{t-E}R]_{SK_i^E}, t)$ to FN_i
 FN_i :
(2) Decrypt $([x2^{t-E} + R2^{t-E}]_{PK_i^O}, t)$
or $([x2^{t-E}R]_{SK_i^E}, t)$
(3) Compute $x2^E + R2^E = x2^{t-E} + R2^{t-E}/2^{(t-1)E}$
or $x2^E R = x2^{t-E}R/2^{(t-1)E}$ and encrypt $x2^E$
(4) Send $([x2^E + R2^E]_{PK_i^O}, 1)$ or $([x2^E R]_{SK_i^E}, 1)$ to SSP
SSP:
(5) Compute
 $([x2^E]_{PK_i^O}, 1) \leftarrow [x2^E + R2^E]_{PK_i^O} \cdot [N_i - R2^E]_{PK_i^E}$
or $([x2^E]_{SK_i^E}, 1) \leftarrow [x2^E R]_{SK_i^E} \cdot R^{-1}$

ALGORITHM 1: Precision control.

Input: $s, i = 1, 2, \dots, m$
Output: s
FNs:
for $i = 1 \rightarrow m$:
(1) FN_i computes $s'_i = s_i + \alpha_i$
(2) Encrypt s'_i with PK_s^O and send $[s'_i]_{PK_s^O}$ to SSP
end for
SSP:
(3) Compute $[s']_{PK_s^O} = \prod_{i=1}^m [s'_i]_{PK_s^O}$
(4) Decrypt $[s']_{PK_s^O}$ and compute $s = s' - \alpha$

ALGORITHM 2: Secure summation.

5.3.4. *Secure Multiplication (OU)*. Given a ciphertext data encrypted with PK_i^O , $([x2^{t-E}]_{PK_i^O}, t)$ and a plaintext data of SSP, $y \cdot 2^E$. SSP will obtain $([xy2^{(t+1)E}]_{PK_i^O}, t+1) = ([x2^{t-E}]_{PK_i^O}^{y \cdot 2^E}, t+1)$.

5.3.5. *Secure Multiplication (Cloud-ElGamal)*:. Given two ciphertext data encrypted with SK_i^E , $([x2^{t_1-E}]_{SK_i^E}, t_1)$ and $([y2^{t_2-E}]_{SK_i^E}, t_2)$. SSP will obtain $([xy2^{(t_1+t_2)E}]_{SK_i^E}, t_1+t_2) = ([x2^{t_1-E}]_{SK_i^E} \cdot [y2^{t_2-E}]_{SK_i^E}, t_1+t_2)$.

5.3.6. *Secure Division*:. The algorithm is inspired by [27]. Given two ciphertext data encrypted with PK_i^O , $([x2^{t_1-E}]_{PK_i^O}, t_1)$ and $([y2^{t_2-E}]_{PK_i^O}, t_2)$. SSP will obtain $([x/y2^{2E}]_{PK_i^O}, 2)$. The computation process is as follows:

- (1) SSP chooses two random integers $R_1, R_2 \in (0, 2^{l/2})$ and computes

$$\begin{aligned} [x2^{t_1-E}R_1]_{PK_i^O} &= [x2^{t_1-E}]_{PK_i^O}^{R_1} \\ [y2^{t_2-E}R_2]_{PK_i^O} &= [y2^{t_2-E}]_{PK_i^O}^{R_2} \end{aligned} \quad (6)$$

Then, SSP sends them to FN_i

- (2) FN_i decrypts $[x2^{t_1-E}R_1]_{PK_i^O}$, $[y2^{t_2-E}R_2]_{PK_i^O}$ and obtains xR_1/yR_2 . Sending $([xR_1/yR_22^E]_{PK_i^O}, 1)$ to SSP
(3) SSP computes $([x/y2^{2E}]_{PK_i^O}, 2) \leftarrow [xR_1/yR_22^E]_{PK_i^O}^{R_2/R_12^E}$
Secure power computation

Given a ciphertext data encrypted with SK_i^E , $([x2^{t-E}]_{SK_i^E}, t)$ and a plaintext data of SSP, $y \cdot 2^E$. SSP will obtain $([x^y2^E], 1)$. The computation process is as follows:

- (1) SSP computes

$$[x^{y2^E}2^{y2^E t-E}]_{SK_i^E} = (a_i^k)^{y2^E} \bmod p_i^e, \left(y_i^k x2^{t-E} y2^E \bmod p_i^e \right). \quad (7)$$

Choosing a random integer $R_1 \in (0, 2^{l/3})$, $R_2, R_2' = R_2^E \in (0, 2^{l/3})$ and computes $[x^{y2^E}2^{y2^E t-E}R_1]_{SK_i^E} = [x^{y2^E}2^{y2^E t-E}]_{SK_i^E} \cdot R_1$, $2^{y2^E t-E}R_1R_2'$. Then, sending $[x^{y2^E}2^{y2^E t-E}R_1]_{SK_i^E}$ and $[2^{y2^E t-E}R_1R_2']_{SK_i^E}$ to FN_i .

- (2) FN_i decrypts and computes

$$\frac{x^y}{R_2} = \sqrt{[2^E]} \frac{x^{y2^E}2^{y2^E t-E}R_1}{2^{y2^E t-E}R_1R_2'}. \quad (8)$$

Then, sending $([x^y/R_22^E]_{SK_i^E}, 1)$ to SSP

- (3) SSP computes $([x^y2^E]_{SK_i^E}, 1) \leftarrow [x^y/R_22^E]_{SK_i^E} \cdot R_2$

5.3.7. *Secure Inner Product*. Given an encrypted vector of FN_i , $[x]_{PK_i^O} = \{([x_12^{t-E}]_{PK_i^O}, t), ([x_22^{t-E}]_{PK_i^O}, t), \dots, ([x_d2^{t-E}]_{PK_i^O}, t)\}$ and a plaintext vector of SSP, $y = (y_12^E, y_22^E, \dots, y_d2^E)$. SSP will obtain

$$([xy2^{(t+1)E}]_{PK_i^O}, t+1) = \left(\prod_{j=1}^d [x_j2^{t-E}]_{PK_i^O}^{y_j2^E}, t+1 \right). \quad (9)$$

5.3.8. *Secure Summation.* SSP wants to obtain the summation of $\sum_{j=1}^m s_j$ and s_i belongs to FN_i . The process is described in Algorithm 2.

- (1) FN_i encrypts $s'_i = s_i + \alpha_i$ with PK_s^O and sends $[s'_i]_{PK_s^O}$ to SSP
- (2) SSP computes $[s']_{PK_s^O} = \prod_{i=1}^m [s'_i]_{PK_s^O}$ and decrypts $[s']_{PK_s^O}$. Then, SSP can obtain $s = s' - \alpha$

$$\begin{aligned} \left[e^{w_j x_j 2^E} 2^{w_j 2^E} \right]_{SK_i^E} &= \left((a_i^{k w_j 2^E} \bmod p_i^e, (y_i^k e^{x_j} 2^{E w_j 2^E} \bmod p_i^e) 2^{w_j 2^E} = (2^E)^{w_j 2^E} \right), \\ \left[e^{w_x 2^E} 2^{w_s 2^E} \right]_{SK_i^E} &= \prod_{j=1}^d \left[e^{w_j x_j 2^E} 2^{w_j 2^E} \right]_{SK_i^E} \prod_{j=1}^d 2^{w_j 2^E}, \end{aligned} \quad (10)$$

where $w_s = w_1 + w_2 + \dots + w_d$. Then, SSP chooses two random integers $R_1, R_2, R_1', R_2' = R_2^{2^E} \in (0, 2^{l/4})$ and sends $[e^{w_x 2^E} 2^{w_s 2^E} R_1]_{SK_i^E}, 2^{w_s 2^E} R_1 R_2'$ to FN_i

- (3) FN_i decrypts and computes

$$\frac{e^{w_x}}{R_2} = \sqrt{2^E} e^{w_x 2^E} \frac{2^{w_s 2^E} R_1}{2^{w_s 2^E} R_1 R_2'}. \quad (11)$$

Then, sending $[(e^{w_x}/R_2) 2^E]_{PK_i^O}$ to SSP

- (4) SSP chooses two random integers $R_3, R_4 \in (0, 2^{l/4})$ computes as follows:

$$\begin{aligned} [e^{w_x 2^E} R_3]_{PK_i^O} &= \left[\left(\frac{e^{w_x}}{R_2} \right) 2^E \right]_{PK_i^O}^{R_2 R_3}, \\ [(e^{w_x} + 1) 2^E R_3 R_4]_{PK_i^O} &= \left([e^{w_x 2^E}]_{PK_i^O} \cdot [2^E]_{PK_i^O} \right)^{R_3 R_4}. \end{aligned} \quad (12)$$

Then, SSP sends $[e^{w_x 2^E} R_3]_{PK_i^O}$ and $[(e^{w_x} + 1) 2^E R_3 R_4]_{PK_i^O}$ to FN_i

- (5) FN_i decrypts $[e^{w_x 2^E} R_3]_{PK_i^O}, [(e^{w_x} + 1) 2^E R_3 R_4]_{PK_i^O}$ and obtains

$$\frac{x_j e^{w_x}}{(e^{w_x} + 1) R_4} = \frac{x_j e^{w_x} R_3}{(e^{w_x} + 1) R_3 R_4}. \quad (13)$$

Then, SSP sends $[x_j e^{w_x} / (e^{w_x} + 1) R_4 2^E]_{PK_i^O}$ to SSP

- (6) SSP computes

$$\left(\left[\frac{x_j e^{w_x}}{(e^{w_x} + 1) R_4} 2^E \right]_{PK_i^O}, 1 \right) \leftarrow \left[\frac{x_j e^{w_x}}{(e^{w_x} + 1) R_4} 2^E \right]_{PK_i^O}^{R_4}. \quad (14)$$

5.3.10. *Secure Sign Computation.* Given encrypted data that is computed by SSP, $([x 2^{t^E}]_{PK_i^O}, t) ([x 2^{t^E}]_{SK_i^E}, t)$. FN_i needs to know if $x 2^{t^E} < 0$ but cannot know the value of $x 2^{t^E}$.

SSP flips a coin s and chooses a random integer $R \in (0, 2^{l/2-t^E})$. If $s = 1$, SSP computes $[tmp]_{PK_i^O} = [x 2^{t^E} R]_{PK_i^O} ([tmp]_{SK_i^E} = [x 2^{t^E} R]_{SK_i^E})$. If $s = 0$,

5.3.9. *Secure Sigmoid Function.* To complete the nonlinear computation of the sigmoid function, we propose an algorithm called Secure Sigmoid Function. The process is described in Algorithm 3.

- (1) FN_i has $e^x = (e^{x_1}, e^{x_2}, \dots, e^{x_d})$. FN_i encrypts $e^{x_j} 2^E$ with SK_i^E and sends it to SSP
- (2) SSP has $w = (w_1, w_2, \dots, w_d)$. SSP computes

SSP computes $[tmp]_{PK_i^O} = [-x 2^{t^E} R]_{PK_i^O} ([tmp]_{SK_i^E} = [-x 2^{t^E} R]_{SK_i^E})$. Then, SSP sends the computed data to FN_i . FN_i decrypts and obtains tmp . Let $judge = 1$ if $tmp < 0$ else $judge = 0$. FN_i sends $[judge]_{PK_i^O}$. If $s = 1$ and $judge = 1$, SSP will know $x 2^{t^E} < 0$. If $s = 1$ and $judge = 0$, $x 2^{t^E} \geq 0$. If $s = 0$ and $judge = 1$, $x 2^{t^E} \geq 0$. If $s = 0$ and $judge = 0$, $x 2^{t^E} < 0$.

Converting Cloud-ElGamal to OU

For this building block, we cite building block 7 of [5].

Converting OU:

For this building block, we cite building block 8 of [5].

5.4. *Privacy Preserving Machine Learning Training.* In this section, we achieve four training protocols based on our proposed building blocks, which are popular machine learning models. We assume that the training data has been normalized before training.

5.5. *Secure Logistic Regression (LR) Training.* We use the stochastic gradient descent (SGD) algorithm to train a Logistic Regression model. SSP randomly selects T data of FNs (all data of FNs have been numbered). The process is described in Algorithm 4.

5.6. *Secure SVM Training.* For training a SVM model, we also use the SGD algorithm. The process is described in Algorithm 5.

5.7. *Secure Naive Bayes (NB) Training.* To train a naive Bayes model, the user needs to aggregate the class prior probability $P(y)$ and the conditional probability $P(x|y)$ from FNs. We assume that it is a binary classification problem. The process is described in Algorithm 6.

5.8. *Secure Deep Neural Network Training.* The training process of deep neural networks contains nonlinear computations such as ReLU. To compute a nonlinear activation function, we will make an approximation method. The approximation method is proposed in [28] and the nonlinear


```

Input:  $\mathbf{e}^x = (e^{x_1}, e^{x_2}, \dots, e^{x_d})$ ,  $\mathbf{w} = (w_1, w_2, \dots, w_d)$ 
Output:  $e^{w^x}$ 
FNi:
for  $j = 1 \rightarrow d$ :
(1) Send  $[e^{x_j} 2^{E}]_{SK_i^E}$  to SSP
    end for
SSP:
(2)  $[e^{w_j x_j} 2^{E} 2^{w_j} 2^{E}]_{SK_i^E}$ 
     $= ((a_i^{k w_j} 2^{E} \bmod p_i^E, (y_i^k e^{x_j} 2^{E w_j} 2^{E} \bmod p_i^E))$ 
(3)  $2^{w_j} 2^{E} = (2^E)^{w_j} 2^{E}$ 
(4)  $[e^{w^x} 2^{E} 2^{w_s} 2^{E}]_{SK_i^E} = \prod_{j=1}^d [e^{w_j x_j} 2^{E} 2^{w_j} 2^{E}]_{SK_i^E}$ 
(5)  $2^{w_s} 2^{E} = \prod_{j=1}^d 2^{w_j} 2^{E}$ 
(6) Choose two random integers  $R_1, R_2, R_1, R_2' = R_2^{2^E} \in (0, 2^{l/4})$ 
(7) Send  $[e^{w^x} 2^{E} 2^{w_s} 2^{E} R_1]_{SK_i^E}, 2^{w_s} 2^{E} R_1 R_2'$  to FNi
    FNi:
(8) Decrypt and compute
     $e^{w^x} / R_2 = \sqrt{[2^E] e^{w^x} 2^{E} 2^{w_s} 2^{E} R_1 / 2^{w_s} 2^{E} R_1 R_2'}$ 
(9) Send  $[(e^{w^x} / R_2) 2^E]_{PK_i^O}$  to SSP
    SSP:
(10) Choose two random integers  $R_3, R_4 \in (0, 2^{l/4})$ 
(11)  $[e^{w^x} 2^E R_3]_{PK_i^O} = [(e^{w^x} / R_2) 2^E]_{PK_i^O}^{R_3 R_4}$ 
(12)  $[(e^{w^x} + 1) 2^E R_3 R_4]_{PK_i^O} = ([e^{w^x} 2^E]_{PK_i^O})^{R_3 R_4} \cdot [2^E]_{PK_i^O}^{R_3 R_4}$ 
(13) Send  $[e^{w^x} 2^E R_3]_{PK_i^O}$  and  $[(e^{w^x} + 1) 2^E R_3 R_4]_{PK_i^O}$  to FNi
    FNi:
(14) Decrypt  $[e^{w^x} 2^E R_3]_{PK_i^O}, [(e^{w^x} + 1) 2^E R_3 R_4]_{PK_i^O}$ 
(15)  $x_j e^{w^x} / (e^{w^x} + 1) R_4 = x_j e^{w^x} R_3 / (e^{w^x} + 1) R_3 R_4$ 
(16) Send  $[x_j e^{w^x} / (e^{w^x} + 1) R_4 2^E]_{PK_i^O}$  to SSP
    SSP:
(17)  $([x_j e^{w^x} / e^{w^x} + 12^E]_{PK_i^O}, 1) \leftarrow [x_j e^{w^x} / (e^{w^x} + 1) R_4 2^E]_{PK_i^O}^{R_4}$ 

```

ALGORITHM 3: Secure sigmoid function.

activation functions will be converted to polynomial functions. Then, the training process can be completed with our proposed building blocks. In addition, we use the training method proposed in [29] to train the deep neural network models.

5.9. Privacy Preserving Machine Learning Model Updating. With the continuous increase of fog node data in the system or the addition of new fog nodes in the system, the quantity and quality of the overall data will be significantly improved. Therefore, it is very important to update the SSP's trained model. For different types of machine learning models, we propose different updating methods. Specifically, we divide the machine learning model into the model trained by the gradient-descent method and the model trained by a nongradient-descent method. In order to prevent a differential attack, the model trained by the nongradient-descent method can be updated only when the data of the original fog nodes increases to more than 50%.

A model trained by the gradient-descent method:

For the model trained based on the gradient-descent method (e.g., LR and SVM), the updating process of the model is as follows:

- (1) TA determines the fog nodes and SSP participating in the updating process and redistributes the system parameters for them (in the way of 5.1).
- (2) FNs and SSP train a new model on the added data. SSP will obtain the new model \mathbf{w}' .
- (3) Based on the \mathbf{w}' and the original model \mathbf{w} , SSP will compute the updated model $\mathbf{w}_i = \mathbf{n}/\mathbf{n} + \mathbf{n}'\mathbf{w}_i + \mathbf{n}'/\mathbf{n} + \mathbf{n}'\mathbf{w}'_i, i = 1, 2, \dots, d$ (\mathbf{n} is the number of original data and \mathbf{n}' is the number of added data).

5.10. Model Trained by Nongradient-Descent Method. For the model trained by the nongradient-descent method (for example, naive Bayes), the updating process of the model is as follows:

- (1) TA redistributes system parameters for all fog nodes and SSP to update the model (in the way of 5.1)
- (2) FNs and SSPs use all the data to retrain a model. SSP takes the model as the updated model

6. Security Analysis

In this section, we make a security analysis of our proposed framework with the real and ideal paradigm and composition

```

Input: the selected data of FNs, iterations T, learning rate L
Output: model parameters  $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_d)$ 
FNs:
for  $i = 1 \rightarrow m$ :
(1)  $\text{FN}_i$  encrypts the selected data with  $\text{PK}_i^O$ 
    Such as  $\mathbf{x}_j$  ( $j \in [1, d]$ ), send  $([\mathbf{x}_j \mathbf{y} 2^E]_{\text{PK}_i^O}, 1)$  to SSP
end for
SSP:
for  $it = 1 \rightarrow T$ :
(2)  $([\mathbf{x}_j e^{w_x} / e^{w_x + 1} 2^E]_{\text{PK}_i^O}, 1) \leftarrow \text{Secure Sigmoid Function}$ 
for  $j = 1 \rightarrow d$ :
(3) SSP performs
    Secure Addition and Secure Multiplication (OU)
(4)  $([\mathbf{w}_j 2^{2E}]_{\text{PK}_i^O}, 2) \leftarrow [\mathbf{w}_j 2^{2E}]_{\text{PK}_i^O}$ 
     $[\mathbf{x}_j e^{w_x} / e^{w_x + 1} 2^E]_{\text{PK}_i^O}^{N_i - L 2^E} \cdot [\mathbf{x}_j \mathbf{y} 2^E]_{\text{PK}_i^O}^{L 2^E}$ 
(5) Converting OU:  $[\mathbf{w}_j 2^{2E}]_{\text{PK}_i^O} \leftarrow [\mathbf{w}_j 2^{2E}]_{\text{PK}_i^O}$ 
(6)  $\mathbf{w}_j 2^E = \mathbf{w}_j 2^{2E} / 2^E$ 
end for
end for

```

ALGORITHM 4: Secure logistic regression training.

Theorem, like [27]. We use a simulator in the ideal world to simulate the view of an adversary (honest-but-curious) in the real world. We consider the adversaries are FNs and SSP, $\mathcal{A} = (\mathcal{A}_{\text{FN}_1}, \mathcal{A}_{\text{FN}_2}, \dots, \mathcal{A}_{\text{FN}_m}, \mathcal{A}_{\text{SSP}})$. We will prove the security of our proposed basic building blocks.

Theorem 1. *The precision control is secure against honest-but-curious adversaries \mathcal{A} .*

Proof. We construct two simulators based on FN_i and SSP, ($\text{Sim}_{\text{FN}_i}, \text{Sim}_{\text{SSP}}$). The Sim_{SSP} computes $([\mathbf{x} 2^{tE} + \mathbf{R} 2^{tE}], \mathbf{t})$ (or $([\mathbf{x} 2^{tE} \mathbf{R}]_{\text{SK}^E}, \mathbf{t})$) and \mathcal{A}_{SSP} cannot distinguish the real execution from the ideal simulation due to the semantic security of the OU cryptosystem (or Cloud-ElGamal cryptosystem). For Sim_{FN_i} , the view of $\mathcal{A}_{\text{FN}_i}$ is $\mathbf{x} 2^{tE} + \mathbf{R} 2^{tE}$ (or $\mathbf{x} 2^{tE} \mathbf{R}$). Because the statistics are distinguishable, $\mathcal{A}_{\text{FN}_i}$ cannot distinguish the real execution from the ideal simulation. \square

Theorem 2. *The secure addition, secure subtraction, secure multiplication (OU), secure multiplication (Cloud-ElGamal), and secure inner product are secure against honest-but-curious adversaries \mathcal{A} .*

Proof. It should be noted that these basic building blocks are similar, so we will make a security analysis for secure addition as an example.

For secure addition, the simulator Sim_{SSP} is the same as Theorem 1. Based on the semantic security of the OU cryptosystem, \mathcal{A}_{SSP} cannot distinguish $[\mathbf{x} 2^{tE}]_{\text{PK}_i^O}$, $[\mathbf{y} 2^{tE}]_{\text{PK}_i^O}$ and $[(\mathbf{x} + \mathbf{y}) 2^{tE}]_{\text{PK}_i^O}$. The view of \mathcal{A}_{SSP} is indistinguishable in the real and ideal world. \square

Theorem 3. *The secure division is secure against honest-but-curious adversaries \mathcal{A} .*

Proof. Because $\mathbf{R}_1, \mathbf{R}_2$ are random integers, $\mathbf{x} 2^{tE} \mathbf{R}_1$ and $\mathbf{y} 2^{tE} \mathbf{R}_2$ are statistics distinguishable for $\mathcal{A}_{\text{FN}_i}$. The simulator Sim_{SSP} can only obtain $[\mathbf{x} / \mathbf{y} 2^{2E}]_{\text{PK}_i^O}$. \square

Theorem 4. *The secure power computation is secure against honest-but-curious adversaries \mathcal{A} .*

Proof. Because $\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}'_2$ are random integers, $\mathbf{x}^{y 2^E} 2^{y 2^{tE}} \mathbf{R}_1$, $2^{y 2^{tE} E \mathbf{R}_1 \mathbf{R}'_2}$ and $\mathbf{x}^y / \mathbf{R}_2$ are statistics distinguishable for $\mathcal{A}_{\text{FN}_i}$. The simulator Sim_{SSP} can only obtain $[\mathbf{x}^y 2^E]_{\text{SK}_i^E}$. \square

Theorem 5. *The secure summation is secure against honest-but-curious adversaries \mathcal{A} .*

Proof. The \mathbf{s}'_i of FN_i is computed through $\mathbf{s}_i + \alpha_i$, so the \mathcal{A}_{SSP} cannot obtain the value of \mathbf{s}_i based on the statistics distinguishable. At the same time, \mathbf{s}'_i is encrypted with PK_i^O . For $\{[\mathbf{s}'_1]_{\text{PK}_i^O}, [\mathbf{s}'_2]_{\text{PK}_i^O}, \dots, [\mathbf{s}'_m]_{\text{PK}_i^O}\}$, the adversaries cannot distinguish the real execution from the ideal simulation due to the semantic security of the OU cryptosystem. \square

Theorem 6. *The secure sigmoid function is secure against honest-but-curious adversaries \mathcal{A} .*

Proof. The Sim_{SSP} performs computation on $[e^{x_j} 2^E]_{\text{SK}_i^E}$ and \mathbf{w}_j ($j = 1, 2, \dots, d$), which means that \mathcal{A}_{SSP} cannot obtain the value of $[e^{w_x} 2^{w_x} 2^E]_{\text{SK}_i^E}$. Because the statistics are distinguishable, Sim_{FN_i} cannot obtain the value of e^{w_x} with e^{w_x} / \mathbf{R}_2 and $\mathbf{x}_j e^{w_x} / (e^{w_x} + 1) \mathbf{R}_4$. The Sim_{SSP} can only obtain $[\mathbf{x}_j e^{w_x} / e^{w_x} + 1 2^E]_{\text{PK}_i^O}$. \square

Theorem 7. *The secure sign computation is secure against honest-but-curious adversaries \mathcal{A} .*

```

Input: the selected data of FNs,
iterations  $T$ , learning rate  $L$ , regularization parameter  $z$ 
Output: model parameters  $\mathbf{W} = (w_1, w_2, \dots, w_d)$ 
FNs:
for  $i = 1 \rightarrow m$ :
(1)  $\text{FN}_i$  encrypts the selected data with  $\text{PK}_i^O$ 
    Such as  $x_j$  ( $j \in [1, d]$ ),
    send  $([x_j 2^{2E}]_{\text{PK}_i^O}, 1)$ ,  $([x_j y 2^{2E}]_{\text{PK}_i^O}, 1)$  and  $[y]$  to SSP
    end for
SSP:
for  $it = 1 \rightarrow T$ :
(2)  $([wx 2^{2E}]_{\text{PK}_i^O}, 2) \leftarrow \text{Secure Inner Product}([x]_{\text{PK}_i^O}, w)$ 
(3)  $[t_1]_{\text{PK}_i^O} = [wx 2^{2E}]_{\text{PK}_i^O} \cdot [N_i - 1]_{\text{PK}_i^O}$  ( $y = 1$ )
     $[t_2]_{\text{PK}_i^O} = [wx 2^{2E}]_{\text{PK}_i^O}^{N_i-1} \cdot [N_i - 1]_{\text{PK}_i^O}$  ( $y = -1$ )
(4) SSP and  $\text{FN}_i$  perform Secure Sign Computation
    for  $[t_1]_{\text{PK}_i^O}$ ,  $[t_2]_{\text{PK}_i^O}$ 
     $\text{FN}_i$ :
(5) In the process of Secure Sign Computation:
    if  $y = 1$ :  $\text{FN}_i$  will judge  $[t_1]_{\text{PK}_i^O}$ 
    else:  $\text{FN}_i$  will judge  $[t_2]_{\text{PK}_i^O}$ 
    SSP:
(7) To compute  $([w_j 2^{2E}]_{\text{PK}_i^O}, 2)$ , SSP performs
Secure Addition and Secure Multiplication (OU)
if  $s = 1$ , judge = 1 or  $s = 0$ , judge = 0:
for  $j = 1 \rightarrow d$ :
     $([w_j 2^{2E}]_{\text{PK}_i^O}, 2) \leftarrow [w_j 2^{2E}]_{\text{PK}_i^O}$ 
     $\cdot [w_j 2^{2E}]_{\text{PK}_i^O}^{N_i-L-z2^E} \cdot [x_j y 2^{2E}]_{\text{PK}_i^O}^{L-z2^E}$ 
    Converting OU:  $[w_j 2^{2E}]_{\text{PK}_i^O} \leftarrow [w_j 2^{2E}]_{\text{PK}_i^O}$ 
     $w_j 2^E = w_j^{2E}/2^E$ 
end for
else:
for  $j = 1 \rightarrow d$ :
     $([w_j 2^{2E}]_{\text{PK}_i^O}, 2) \leftarrow [w_j 2^{2E}]_{\text{PK}_i^O} \cdot [w_j 2^{2E}]_{\text{PK}_i^O}^{N_i-L-z2^E}$ 
    Converting OU:  $[w_j 2^{2E}]_{\text{PK}_i^O} \leftarrow [w_j 2^{2E}]_{\text{PK}_i^O}$ 
     $w_j 2^E = w_j^{2E}/2^E$ 
end for
end for

```

ALGORITHM 5: Secure SVM training.

Proof. The Sim_{FN_i} can obtain $x 2^{tE} \mathbf{R}$ or $-x 2^{tE} \mathbf{R}$ through the building blocks. Based on the statistics distinguishable, $\mathcal{A}_{\text{FN}_i}$ cannot obtain the value of x . \square

Theorem 8. *The secure machine learning training protocols are secure against honest-but-curious adversaries \mathcal{A} .*

Proof. We construct the machine learning training protocol through our designed building blocks in a modular way. The simulators are $\{\text{Sim}_{\text{FN}_1}, \text{Sim}_{\text{FN}_2}, \dots, \text{Sim}_{\text{FN}_m}, \text{Sim}_{\text{SSP}}\}$, which is the same as Theorems 1–7. We have proved the security of the proposed building blocks, so the view of adversaries cannot distinguish the real execution from the ideal simulation. \square

7. Performance Evaluation

In this section, we evaluate our proposed ePMLF and compare it with Zhu et al. [5]. Zhu et al. [5] proposed a

privacy-preserving ML training framework for the aggregation scenario. However, there are still some shortcomings to be improved, including the function of building blocks is not comprehensive and high communication and computation overhead, which makes the framework impractical. Our proposed framework effectively solves these problems. Our experimental environment is shown in Table 2.

To train LR, SVM, and NB, we implement our framework on three datasets of the UCI machine learning library, as shown in Table 3.

For the deep neural network, we will use the MNIST dataset to train a LeNet model [30]. The MNIST dataset contains 60,000 training samples and 10,000 testing samples.

7.1. Performance of Our Proposed Framework

7.1.1. Building Blocks Evaluation. To test our proposed building blocks, we test three main building blocks, secure

```

Input: the data of FNs
Output: model parameters  $\mathbf{P}(y = 1), \mathbf{P}(y = 0), \mathbf{P}(x_i = 0|y = 0),$ 
 $\mathbf{P}(x_i = 0|y = 1), \mathbf{P}(x_i = 1|y = 0), \mathbf{P}(x_i = 0|y = 1)$ 
FNs:
for  $i = 1 \rightarrow m$ :
(1)  $\text{FN}_i$  computes  $\mathbf{S}_{(y=1)}^i + \alpha_i = (\sum_{j=1}^{n_i} y_j) + \alpha_i,$ 
 $\mathbf{S}_{(x_k=0|y=0)}^i + \alpha_i, \mathbf{S}_{(x_k=0|y=1)}^i + \alpha_i, \mathbf{k} = 1, 2, \dots, \mathbf{d}$ 
Encrypt them with  $\mathbf{PK}_s^O$  and send them to SSP
end for
SSP:
(2) Compute  $[\mathbf{S}_{(y=1)} + \alpha]_{\mathbf{PK}_s^O} = \prod_{i=1}^m [\mathbf{S}_{y=1}^i + \alpha_i]_{\mathbf{PK}_s^O},$ 
 $[\mathbf{S}_{(x_k=0|y=0)} + \alpha]_{\mathbf{PK}_s^O} = \prod_{i=1}^m [\mathbf{S}_{(x_k=0|y=0)}^i + \alpha_i]_{\mathbf{PK}_s^O} = \prod_{i=1}^m [\mathbf{S}_{(x_k=0|y=0)}^i + \alpha_i]_{\mathbf{PK}_s^O}$ 
 $[\mathbf{S}_{(x_k=0|y=1)} + \alpha]_{\mathbf{PK}_s^O} = \prod_{i=1}^m [\mathbf{S}_{(x_k=0|y=1)}^i + \alpha_i]_{\mathbf{PK}_s^O}$ 
(3) Decrypt them with  $\mathbf{SK}_s^O$ 
for  $\mathbf{k} = 1 \rightarrow \mathbf{d}$ :
(4)  $\mathbf{P}(x_k = 0|y = 0) = \mathbf{S}_{(x_k=0|y=0)}/(\mathbf{n} - \mathbf{S}_{(y=1)})$ 
 $\mathbf{P}(x_k = 0|y = 1) = \mathbf{S}_{(x_k=0|y=1)}/\mathbf{S}_{(y=1)}$ 
 $\mathbf{P}(x_k = 1|y = 0) = 1 - \mathbf{P}(x_k = 0|y = 0)$ 
 $\mathbf{P}(x_k = 1|y = 1) = 1 - \mathbf{P}(x_k = 0|y = 1)$ 
end for
(5)  $\mathbf{P}(y = 1) = \mathbf{S}_{(y=1)}/\mathbf{n}, \mathbf{P}(y = 0) = 1 - \mathbf{P}(y = 1)$ 

```

ALGORITHM 6: Secure naive Bayes training.

inner product, secure summation, and secure power computation. We set the different key lengths (256 bits, 512 bits, 1024 bits, and 2048 bits). The computation time results are shown in Figure 3. It can be seen that the increase in key length will make the computation time longer. To balance security and efficiency, the key length is usually set to 1024 bits or 2048 bits.

Then, we evaluate the impact of \mathbf{d} on computation time for secure inner product and secure sigmoid function. Based on the above results, the key length will be set to 1024 bits. The results are shown in Figure 4. It can be seen that with the increase in \mathbf{d} , the computation time of secure inner product and secure sigmoid function will increase.

7.1.2. Secure Machine Learning Training Analysis. In order to test the quality of the model trained using our framework, we test the accuracy of the trained model (LR, SVM, NB, and LeNet). The results are shown in Table 4. From Table 4, it can be seen that the accuracy of the training model is high.

7.2. Comparative Analysis. In this section, we analyze the computation and communication overhead of our proposed framework. Then, we make a comparison with Zhu et al. [5]. According to [31], we know that the computation cost of an exponentiation operation is equal to $1.5\mathbf{l}$ multiplication operations, where \mathbf{l} is the length of ciphertexts. We assume that $\mathbf{l}_o, \mathbf{l}_e, \mathbf{l}_p$ and \mathbf{l}_r denote the ciphertext length of OU cryptosystem, Cloud-ElGamal cryptosystem, Paillier cryptosystem, and Cloud-RSA cryptosystem, respectively.

7.2.1. Computation Complexity Analysis. In our proposed framework, we focus on the exponentiation operation and multiplication operation. The computation complexity of

secure addition is \mathbf{l}_o . The secure subtraction costs $2.5\mathbf{l}_o$. For secure multiplication (OU) and secure multiplication (Cloud-ElGamal), they cost $1.5\mathbf{l}_o$ and $2\mathbf{l}_e$ respectively. For secure division, the computation complexity is $9\mathbf{l}_o$. For secure power computation, the computation complexity is $6.5\mathbf{l}_e$. For a secure inner product, the computation complexity is $2.5\mathbf{d} \cdot \mathbf{l}_o$. For secure summation, the computation complexity is $\mathbf{m} \cdot \mathbf{l}_o$. The secure sigmoid function costs $5.5\mathbf{d} \cdot \mathbf{l}_e + 1.5\mathbf{l}_e + 10.5\mathbf{l}_o$. The secure sign computation costs $1.5\mathbf{l}_o$ (or \mathbf{l}_e). The comparison results are shown in Table 5. In our framework, the OU cryptosystem and Cloud-ElGamal cryptosystem are used to encrypt data. In [5], the Paillier cryptosystem and Cloud-RSA cryptosystem are used to encrypt data. It should be noted that $\mathbf{l}_o < \mathbf{l}_p, \mathbf{l}_e = \mathbf{l}_r$ and $\mathbf{b}_i \gg \mathbf{d}$. From Table 5, it can be seen that our proposed scheme has lower computation costs.

7.2.2. Computation Overhead Analysis. We test the computation time of our framework and [5]. The key length of the cryptosystem is set to 1024 bits. The results are shown in Table 6. From Table 6, it can be seen that our framework is more efficient. Then, we compare the computation overhead of each participant with [5]. We assume that the SSP is the model owner and the FNs are the data owners. The comparison results are shown in Table 7 ($m = 2, d = 2$). From Table 7, the computation overhead is lower than [5] for SSP and FNs. We have obvious advantages to perform secure machine learning training.

Example of secure sigmoid function: The computation process of secure sigmoid function will cause the loss of accuracy. Therefore, it is very important to ensure accuracy. To prove the accuracy of the secure sigmoid function, we perform the building block by setting a case. We set $e^x = (1.1, 1.22)$ and $w = (1.31, 2.42)$, which is the same as [5]. In the plaintext,

TABLE 2: Experimental environment.

Operating system	Windows 10
CPU	Intel(R) Core(TM)i7-10510U, 1.80 GHz, 2.30 GHz
Memory	8 G
Program language	Python

TABLE 3: Experiments datasets.

Dataset	Records	Attributes	Classifications
Dermatology	366	34	6
Heart disease	303	13	5
Breast cancer	699	9	2

TABLE 4: The accuracy of machine learning models.

Machine learning model	Dataset	Accuracy (%)
LR	Dermatology	94.44
	Heart disease	93.33
	Breast cancer	94.16
SVM	Dermatology	98.61
	Heart disease	93.33
	Breast cancer	94.89
NB	Breast cancer	97.81
LeNet	MNIST	97.69

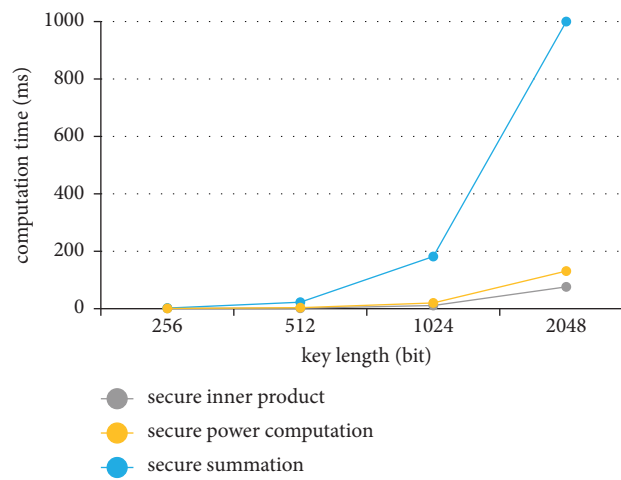


FIGURE 3: Computation time under different key lengths.

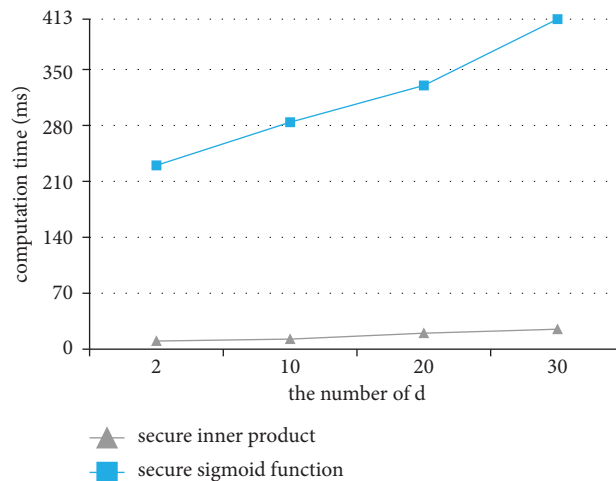


FIGURE 4: Computation time under different d.

TABLE 5: Comparison of computation complexity.

Building blocks	Ours, $k = 1024$ bit	[5], $k = 1024$ bit
Secure addition	l_o	l_p
Secure subtraction	$2.5l_o$	$2.5l_p$
Secure division	$9l_o$	Not support
Secure inner product	$2.5d \cdot l_o$	$2.5d \cdot l_p$
Secure summation	$m \cdot l_o$	$2.5m \cdot l_p$
Secure sigmoid function	$5.5d \cdot l_e + 1.5l_e + 10.5l_o$	$(\sum_{i=1}^d b_i + d)l_r + 2.5l_r + 9l_p$

TABLE 6: Comparison of computation time.

Building blocks	Ours, $k = 1024$ bits	[5], $k = 1024$ bits
Secure addition	0.019 ms	0.08 ms
Secure subtraction	42.6 ms	80.5 ms
Secure division	98.8 ms	Not support
Secure inner product	10.1 ms	79.6 ms
Secure summation	181.8 ms	1666.9 ms
Secure sigmoid function	232.4 ms	445.7 ms

TABLE 7: Comparison of computation overhead: model owner and data owners.

Building blocks	Model owner (ms)		Data owners (ms)	
	Ours	[5]	Ours	[5]
Secure addition	0.019	0.08	0	0
Secure subtraction	42.6	80.5	0	0
Secure inner product	10.1	79.6	0	0
Secure summation	11.93	1410.23	133.64	306.15
Secure sigmoid function	139.45	231.65	92.96	190.67

TABLE 8: Comparison of communication overhead.

Building blocks	Ours	[5]
Secure addition	0	0
Secure subtraction	0	0
Secure division	$3l_o$	Not support
Secure inner product	0	0
Secure summation	$m \cdot l_o$	$3m \cdot l_p$
Secure sigmoid function	$(d + 1) \cdot l_e + 4l_o$	$(d + 1) \cdot l_r + 3l_p$

the value e^{wx} is 1.83. Our result is 1.81 and the result of [5] is 1.6588. It can be seen that our result is more accurate.

7.2.3. Communication Analysis. We analyze the communication overhead and interactions of our building blocks and compare them with [5]. The secure addition, secure subtraction, secure multiplication (OU), secure multiplication (Cloud-ElGamal), and secure inner product are computed by the user, so the interactions of these building blocks are 0. For the secure division, the communication overhead is $3l_o$. For the secure summation, the communication overhead is $m \cdot l_o$ and the interactions are m . For the secure sigmoid function, the communication overhead is $(d + 1) \cdot l_e + 4l_o$. The comparison results are shown in

Table 8. It can be seen that the communication overhead of our proposed secure summation is much lower than [5].

8. Conclusion

In this paper, we propose a privacy-preserving machine learning framework, including secure training data normalization, model training, and model updating. Based on our proposed framework, SSP can train different machine learning models. The trained models have high accuracy. Compared with the existing scheme, our proposed framework significantly reduces the computation and communication overhead. In future research, we will focus on the efficiency and robustness of the

privacy-preserving machine learning training framework.

Data Availability

In our experiments, we implement our framework on four datasets, dermatology, heart disease, breast cancer, and MNIST.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The work was supported in part by the National Natural Science Foundation of China (61862052) and the Science and Technology Foundation of Qinghai Province (2019-ZJ-7065).

References

- [1] W. Tang, J. Ren, K. Deng, and Y. Zhang, "Secure data aggregation of lightweight e-healthcare iot devices with fair incentives," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8714–8726, 2019.
- [2] L. Yang, Q. Zheng, and X. Fan, "Rsp: a reliable, searchable and privacy-preserving e-healthcare system for cloud-assisted body area networks," in *Proceedings of the IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pp. 1–9, IEEE, Atlanta, GA, USA, May 2017.
- [3] S. Liu, J. Yu, X. Deng, and S. Wan, "Fedcpf: an efficient-communication federated learning approach for vehicular edge computing in 6g communication networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 1616–1629, 2022.
- [4] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: a client level perspective," 2017, <https://arxiv.org/abs/1712.07557>.
- [5] L. Zhu, X. Tang, M. Shen, F. Gao, J. Zhang, and X. Du, "Privacy-preserving machine learning training in iot aggregation scenarios," *IEEE Internet of Things Journal*, vol. 8, no. 15, Article ID 12106, 2021.
- [6] L. Liu, R. Chen, X. Liu, J. Su, and L. Qiao, "Towards practical privacy-preserving decision tree training and evaluation in the cloud," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2914–2929, 2020.
- [7] Y. Liu, Y. Luo, Y. Zhu, Y. Liu, and X. Li, "Secure multi-label data classification in cloud by additionally homomorphic encryption," *Information Sciences*, vol. 468, pp. 89–102, 2018.
- [8] J. Wang, L. Wu, H. Wang, K.-K. R. Choo, and D. He, "An efficient and privacy-preserving outsourced support vector machine training for internet of medical things," *IEEE Internet of Things Journal*, vol. 8, no. 1, pp. 458–473, 2021.
- [9] C. Zhou, A. Fu, S. Yu, W. Yang, H. Wang, and Y. Zhang, "Privacy-preserving federated learning in fog computing," *IEEE Internet of Things Journal*, vol. 7, no. 11, pp. 10782–10793, 2020.
- [10] M. Zhang, W. Song, and J. Zhang, "A secure clinical diagnosis with privacy-preserving multiclass support vector machine in clouds," *IEEE Systems Journal*, vol. 16, no. 1, pp. 67–78, 2022.
- [11] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in iot," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5986–5994, 2020.
- [12] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proceedings of the ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, Dublin, Ireland, June 2020.
- [13] X. Wu, Y. Zhang, M. Shi, P. Li, R. Li, and N. N. Xiong, "An adaptive federated learning scheme with differential privacy preserving," *Future Generation Computer Systems*, vol. 127, pp. 362–372, 2022.
- [14] V. Dani, V. King, M. Movahedi, J. Saia, and M. Zamani, "Secure multi-party computation in large networks," *Distributed Computing*, vol. 30, no. 3, pp. 193–229, 2017.
- [15] S. Mehnaz, G. Bellala, and E. Bertino, "A secure sum protocol and its application to privacy-preserving multi-party analytics," in *Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies*, pp. 219–230, New York, NY, USA, June 2017.
- [16] R. Saha, S. Misra, and P. K. Deb, "Fogfl: fog-assisted federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 8456–8463, 2021.
- [17] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2020.
- [18] F. Wang, H. Zhu, R. Lu, Y. Zheng, and H. Li, "A privacy-preserving and non-interactive federated learning scheme for regression training with gradient descent," *Information Sciences*, vol. 552, pp. 183–200, 2021.
- [19] M. Zhao, C. Chen, L. Liu, D. Lan, and S. Wan, "Orbital collaborative learning in 6g space-air-ground integrated networks," *Neurocomputing*, vol. 497, pp. 94–109, 2022.
- [20] Q. Zhang, L. T. Yang, and Z. Chen, "Privacy preserving deep computation model on cloud for big data feature learning," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1351–1362, 2016.
- [21] Y. Li, Z. L. Jiang, L. Yao, X. Wang, S. M. Yiu, and Z. Huang, "Outsourced privacy-preserving c4.5 decision tree algorithm over horizontally and vertically partitioned dataset among multiple parties," *Cluster Computing*, vol. 22, no. S1, pp. 1581–1593, 2019.
- [22] T. Li, J. Li, X. Chen, Z. Liu, W. Lou, and T. Hou, "Npmml: a framework for non-interactive privacy-preserving multi-party machine learning," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 6, pp. 1–2982, 2020.
- [23] X. Liu, R. H. Deng, K.-K. R. Choo, and Y. Yang, "Privacy-preserving outsourced clinical decision support system in the cloud," *IEEE Transactions on Services Computing*, vol. 14, no. 1, pp. 222–234, 2017.
- [24] T. Okamoto and S. Uchiyama, "A new public-key cryptosystem as secure as factoring," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 308–318, Springer, Berlin, Germany, 1998.
- [25] K. El Makkaoui, A. Beni-Hssane, and A. Ezzati, "Cloud-elgamal: an efficient homomorphic encryption scheme," in *Proceedings of the 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pp. 63–66, IEEE, Fez, Morocco, October 2016.
- [26] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, *Machine learning classification over encrypted data*, Cryptology ePrint Archive, 2014.
- [27] J. Zhang, Z. Ljiang, P. Li, and S. M. Yiu, "Privacy-preserving multikey computing framework for encrypted data in the cloud," *Information Sciences*, vol. 575, pp. 217–230, 2021.

- [28] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: applying neural networks to encrypted data with high throughput and accuracy," in *Proceedings of the International Conference on Machine Learning*, pp. 201–210, PMLR, New York, NY, USA, January 2016.
- [29] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *Journal of Network and Computer Applications*, vol. 116, pp. 1–8, 2018.
- [30] Y. LeCun, B. Boser, J. S. Denker et al., "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.
- [31] B. Xie, T. Xiang, X. Liao, and J. Wu, "Achieving privacy-preserving online diagnosis with outsourced svm in internet of medical things environment," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 6, pp. 4113–4126, 2022.