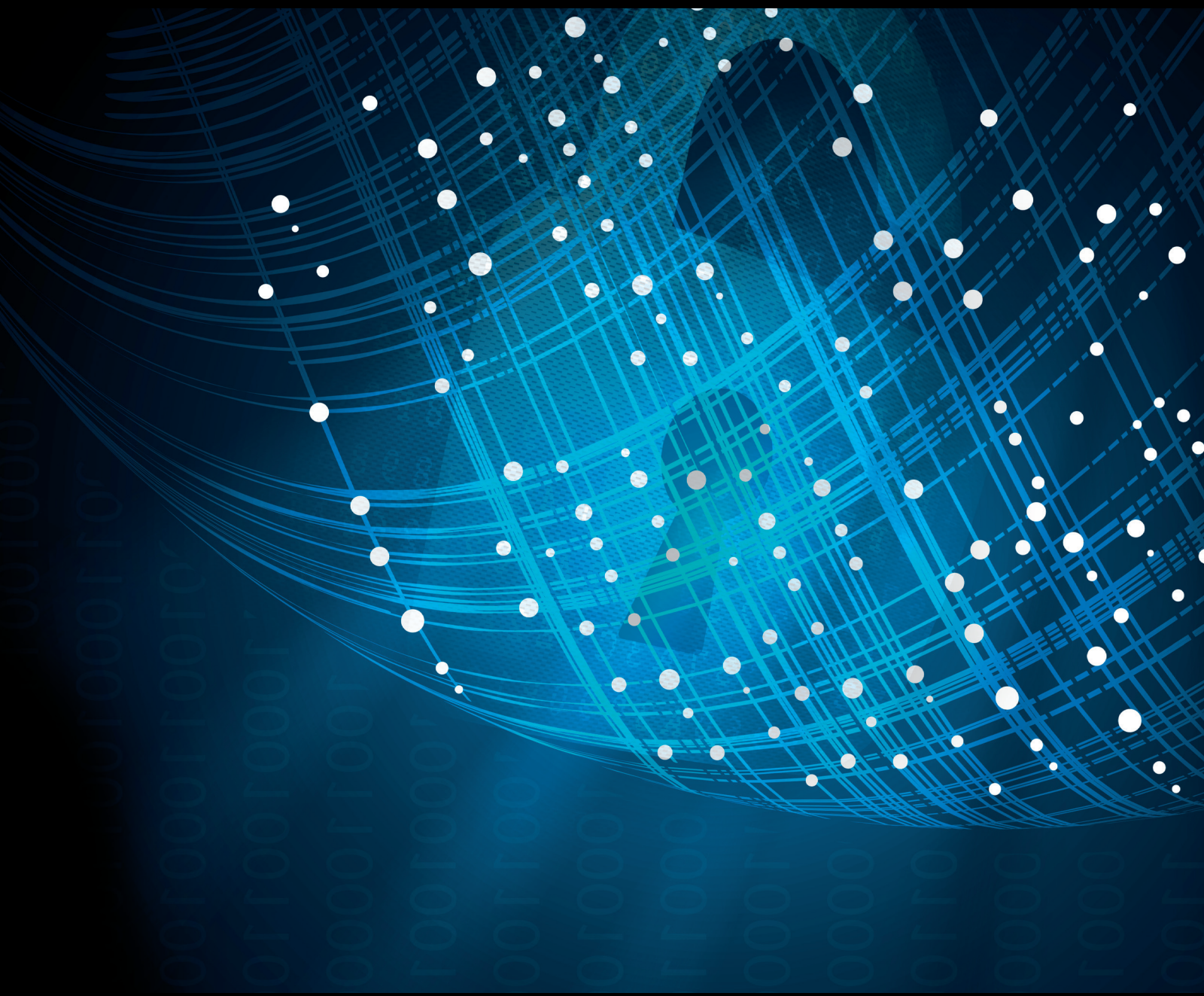
Security and Communication Networks

Cryptographic Schemes and Protocols for Artificial Intelligence

Lead Guest Editor: Debiao He
Guest Editors: Kim-Kwang Raymond Choo, Wenbo Shi, and Jong-Hyouk Lee



Cryptographic Schemes and Protocols for Artificial Intelligence

Security and Communication Networks

Cryptographic Schemes and Protocols for Artificial Intelligence

Lead Guest Editor: Debiao He
Guest Editors: Kim-Kwang Raymond Choo, Wenbo Shi, and Jong-Hyouk Lee

Contents

Hindawi
Security and Communication Networks
Volume 2023, Article ID 9308909, 12 pages
https://doi.org/10.1155/2023/9308909

Research Article

DeepDefense: A Steganalysis-Based Backdoor Detecting and Mitigating Protocol in Deep Neural Networks for AI Security

Lei Zhang,[1] **Ya Peng**,[1] **Lifei Wei**,[2] **Congcong Chen**,[1] **and Xiaoyu Zhang**[3]

[1]*College of Information Technology, Shanghai Ocean University, Shanghai 201306, China*
[2]*College of Information Engineering, Shanghai Maritime University, Shanghai 201306, China*
[3]*State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an 710071, Shaanxi, China*

Correspondence should be addressed to Lifei Wei; lfwei@shmtu.edu.cn

Received 8 September 2022; Revised 13 October 2022; Accepted 24 November 2022; Published 9 May 2023

Academic Editor: Debiao He

Backdoor attacks have been recognized as a major AI security threat in deep neural networks (DNNs) recently. The attackers inject backdoors into DNNs during the model training such as federated learning. The infected model behaves normally on the clean samples in AI applications while the backdoors are only activated by the predefined triggers and resulted in the specified results. Most of the existing defensing approaches assume that the trigger settings on different poisoned samples are visible and identical just like a white square in the corner of the image. Besides, the sample-specific triggers are always invisible and difficult to detect in DNNs, which also becomes a great challenge against the existing defensing protocols. In this paper, to address the above problems, we propose a backdoor detecting and mitigating protocol based on a wider separate-then-reunion network (WISERNet) equipped with a cryptographic deep steganalyzer for color images, which detects the backdoors hiding behind the poisoned samples even if the embedding algorithm is unknown and further feeds the poisoned samples into the infected model for backdoor unlearning and mitigation. The experimental results show that our work performs better in the backdoor defensing effect compared to state-of-the-art backdoor defensing methods such as fine-pruning and ABL against three typical backdoor attacks. Our protocol reduces the attack success rate close to 0% on the test data and slightly decreases the classification accuracy on the clean samples within 3%.

1. Introduction

Deep neural networks (DNNs) have a wide range of the current applications in the artificial intelligence applications such as image recognition, speech recognition, and natural language processing [1–3], in which security and privacy protection are considerable issues [4]. The massive amount of data and growing computing power have facilitated the development of DNNs, but the DNN models are still very expensive in training. Users often choose to train DNN models on the third-party platforms (e.g., Amazon EC2) or even use third-party trained models directly to reduce training costs. However, it is vulnerable to backdoor attacks, which can misclassify any input using attacker predefined triggers (pattern patches) and replace the corresponding label with a predefined target label. Those models with backdoors behave normally just like the clean peer-to-peer models for clean samples without triggers, which are equivalent to highly stealthy viruses that disguise themselves as normal and perform great damage [5].

The backdoor attack greatly threatens DNNs in practical applications for reducing the trustworthiness of the DNN models and even leading to safety-critical areas. The separation of data and model training in deep learning allows attackers to often gain and modify the training samples to mislead DNNs by adding some invisible perturbations to a small proportion of datasets, such as the local patches or the steganographic data in the lower right corner of an image, and even setting weights that affect the model during training [6–10]. The ability of infected DNN models to correctly classify clean samples makes it difficult for users to detect the presence of backdoors. In addition, the hidden

nature of triggers makes it difficult for users to identify them. Thus, the invisibility and stealthiness of triggers make detecting backdoor attacks a considerable challenge [11–13].

Most of the existing backdoor defensing methods are divided into two types: model-based defense and data-based defense. The former detects whether the model is infected by a backdoor, and the latter considers whether the data contain a trigger. Recently, Li et al. [14] reveal that existing backdoor attacks were easily mitigated by current defenses [15–17] mostly because their backdoor triggers are sample-agnostic, i.e., different poisoned samples contain the same trigger no matter what trigger pattern is adopted. Thus, they propose an attack method, called as *sample-specific backdoor attack* (SSBA), which makes it more difficult to detect and remove the backdoors since most of the current defensing protocols reconstruct and detect backdoor triggers according to the same behavior on different poisoned samples [15–17]. SSBA is an invisible backdoor attack that generates invisible sample-specific triggers by the pretrained encoder-decoder network. The reason why current mainstream defensing methods have difficulty in detecting sample-specific triggers is that their success based on the assumption that the triggers are sample-agnostic based types. For example, pruning-based defenses assume that the neurons associated with the backdoor are different from those activated by the clean samples. The defender can remove the hidden backdoor by pruning out the potential neurons. However, the non-overlap between the two neurons is that the sample-agnostic trigger pattern is simple, and the DNNs only need a few independent neurons to encode this trigger. This assumption might be easily broken when the trigger is sample-specific.

Inspired by image steganalysis technique [18], we find that the intensity values of the images at the same position of different color channels have a strong correlation for the poisoned images regardless of whether the triggers are sample-specific or invisible; that is, the triggers in the poisoned images belong an additional perturbation with a weak correlation among those color channels. In addition, since the poisoned samples of the backdoor attack are bounded to the target label, the correlation between the trigger pattern and the target label can be effectively broken by randomizing the class target.

We propose a new backdoor detecting and removing protocol, which can detect backdoors regardless of whether the triggers are specific to poisoned samples or not. Specifically, it detects whether a color image contains a trigger by the feature that the additional perturbation can be retained in the wider separate-then-reunion network (WISERNet). To address the weakness that poisoned samples in backdoor attacks are always bounded to the target label, our protocol breaks the correlation between the trigger pattern and the target label by backdoor unlearning and leads to model purification. In summary, our contributions are as follows:

(i) A backdoor defensing method based on secure image steganalysis is proposed. The poisoned image contains a trigger that can be considered as an additional perturbation, and the intensity value at the same location has a strong correlation between different color channels, while the trigger has a weak correlation between its channels. The protocol is proved valid whether the trigger is visible or invisible.

(ii) A secure backdoor detecting and removing protocol is designed. We design a novel protocol to achieve the goal by detecting the poisoned images in the training dataset based on the wider separate-then-reunion network regardless of whether the trigger is specific to the poisoned samples and by retraining the model for backdoor unlearning with the detected poisoned images.

(iii) Extensive experiments are conducted in the proposed protocol. We empirically show that our protocol is robust against three state-of-the-art backdoor attacks. Compared with the state-of-the-art backdoor defensing protocols, fine-pruning [15] and ABL [19], our protocol reduces the success rate of backdoor attacks to nearly 0% on both target classification and face recognition tasks and retains the accuracy after removing the backdoors.

2. Related Work

2.1. Backdoor Attacks. A common method for implementing backdoor attacks is data poisoning. When the model is training, the poisoned samples are injected into the training dataset. After that, the model is influenced by the poisoned samples, deviates from the desired training effect of the original training data, and changes "slightly" in the desired direction according to the feature of the poisoned samples, which allows the attacker to modify the model and implant a backdoor [20]. According to the visibility of trigger, backdoor attacks based on data poisoning can be classified into two categories: visible backdoor attack and invisible backdoor attack.

2.1.1. Visible Backdoor Attack. Gu et al. [21] first proposed the backdoor attack BadNets to inject backdoors by modifying part of the training data, whose triggers can be of arbitrary shapes, such as squares. Chen et al. [22] first demonstrated that data poisoning attacks can create physically implemented backdoors. Liu et al. [23] proposed a Trojan attack to design triggers based on the values of internal neurons in DNNs, which strengthens the connection between the trigger and the internal neurons, enabling the effect of implant backdoors with fewer poisoned samples. Chen et al. [24] improve the steganography of the trigger by combining generative adversarial network techniques to implant the trigger as a watermark into clean samples and reducing the variability between the trigger features and the clean sample features. There are many other works [25, 26] implemented in optimizing triggers, and although all of these attack methods have high success rates, the triggers are visible and can be easily detected by people.

2.1.2. Invisible Backdoor Attack. Zeng et al. [27] proposed that poisoned samples can be identified by frequency information and constructed frequency invisible poisoned

samples, thus achieving the invisibility of triggers. Li et al. [14] proposed to generate sample-specific triggers by the pretrained encoder-decoder network. Considering the steganography perspective, Li et al. [28] proposed an optimized framework to constrain the generation of triggers by regularization and embed the triggers in the bit space using image steganography to make the triggers invisible.

2.2. Backdoor Defenses. Due to the great potential damage of backdoor attacks to artificial intelligence applications, an increasing number of backdoor defensing protocols are proposed to mitigate such security threats. The existing defensing approaches include model-based defense and data-based defense.

2.2.1. Model-Based Defenses. Model-based defense is to detect whether a model is infected with backdoors. Liu et al. [15] found that neurons associated with backdoors are usually dormant during inference of benign samples and therefore proposed to prune the associated backdoor neurons to eliminate backdoors in the model. Zhao et al. [29] proposed to repair infected models using quantitative clean samples by pattern connectivity techniques [30]. Liu et al. [31] proposed a neural network-based artificial intelligence scanning technique inspired by EBS [32] to determine whether a model has a backdoor; however, it is effective for single-trigger attacks and ineffective for multitrigger attacks. Wang et al. [17] proposed a defense method called neural cleanse (NC) by synthesizing each class's triggers and comparing the triggers' size. If the smaller trigger is significantly smaller than the other triggers, the model is considered to be infected with a backdoor. Recently, Li et al. [19] proposed the concept of antibackdoor and designed a generic antibackdoor learning protocol ABL, which can automatically prevent backdoor attacks during model training.

2.2.2. Data-Based Defenses. Data-based defense is to detect whether a sample contains a trigger. Gao et al. [16] proposed a method, known as the STRIP, to filter malicious samples by overlaying various images onto the images of training samples and observing the randomness of their classification results. Bao et al. [33] proposed an image preprocessing method to identify the trigger region using GardCAM [34] technique, remove it, and replace it with a neutral-colored box because the region where the triggers in the poisoned samples are located has a high impact on the model inference stage. Udeshi et al. [35] proposed to make a trigger interceptor using the dominant color of the image for locating and removing backdoor triggers in poisoned samples. Han et al. [36] proposed an evaluation framework to preprocess the input samples using data enhancement techniques to disrupt the connection between the backdoor and the trigger in the poisoned sample, making the triggers invalid during inference, and fine-tuning the infection model using another data enhancement technique to eliminate the effect of backdoors.

Liu et al. [15] proposed the approach, named as *fine-pruning* (short for FP), which has a degraded defense performance for different models and datasets. Li et al. [19] proposed a more complex implementation of antibackdoor learning, which divides the model training stages into two stages: backdoor isolation and backdoor unlearning, and the choice of a turn-period from its backdoor isolation process to backdoor unlearning progress is more critical. For different attack methods and data sets, the choice of the turn-period also has different effects on the performance of the model. Our protocol performs well for different datasets, models, and attack methods.

3. Overview

In this section, we define our attack model, give the assumptions and goals of defensing protocols, and, finally, provide an intuitive overview of our approach for identifying and mitigating backdoor attacks.

3.1. Attack Models and Defense Assumption. In our attack model, the user trains a DNN model on the training dataset, denoted as D_{train}, that can be obtained from a third party, or even the training process of the DNN can be outsourced to an untrustworthy third party. An attacker may poison part of the training data, set the size and position of the triggers at will, and adjust the training stage of the model, but not access the validation dataset and manipulate the inference stage of the model. The attacker's goal is to return to the user a trained infected backdoor model that behaves like the uninfected model in terms of the output on the clean samples but classifies into the target label specified by the attacker when the samples contain the triggers.

The attacker assumed in our work is more powerful. The attacker proposed by Li et al. [14] can only access the training dataset and cannot manipulate the training stage of the model. The attacker proposed by Liu et al. [23] cannot access the training data and can only modify the trained model. The attacker defended in our work not only has access to the training dataset but also can manipulate the training stage of the model. It is reasonable for the attacker to consider an attacker with limited capabilities. However, the attacker should be assumed to be more powerful since advances in technology and defense methods.

We also assumed that the defender has access to the trained DNN model and can use a clean set of samples to test the performance of the model.

3.2. Design Goals. Our defensing protocol includes two specific goals:

(i) Backdoor detecting: After the training stage of the DNN model, a backdoor detector constructed by WISERNet can successfully detect whether a sample image contains a trigger, i.e., whether it is a poisoned image.

(ii) Backdoor mitigating: Since there is a strong correlation between triggers and target labels in backdoor

attacks, this weakness is exploited to reinput the poisoned samples into the infected model and retrain the model to achieve backdoor unlearning.

3.3. Design Intuition. We describe our high-level intuition for detecting triggers in poisoned samples and overview our defense.

3.3.1. Key Intuition. The invisibility of the trigger and the low poisoning rate make it difficult for the defender to detect whether the sample is poisoned or not. We derive the intuition behind our technique from the basic properties of a backdoor trigger, namely that whether the trigger is invisible or not, it can be regarded as additional noise, and this noise can be a special pattern or a string representing the target label. For a poisoned image, the intensity values of the three bands at the same position exhibit a strong correlation, and their expectations are similar from the perspective of statistics. On the contrary, the additional noise added in the poisoned sample has a weaker correlation between the bands and may not even correlate.

To verify the above statement, we analyzed 10,000 poisoned images generated in BadNets [21], Blend Attack [22], and SSBA [14]. Given $X = \{0, 1, \ldots, 255\}^{(C\times W\times H)}$, a poisoned image of the size of $W \times H$, it comprises three bands, namely the red, the green, and the blue band. The correlation between the different bands of the poisoned image is defined as follows:

$$\mathrm{Corr}_{j,k} = \frac{\sum_{i=1}^{K}(M_i - \overline{M})(N_i - \overline{N})}{\sqrt{\sum_{i=1}^{K}(M_i - \overline{M})^2}\sqrt{\sum_{i=1}^{K}(N_i - \overline{N})^2}}, \quad (1)$$

where $j, k \in \{R, G, B\}$, $K = W * H$, M and N indicate band map matrix vector of poisoned image, and $\overline{M}$ and $\overline{N}$ are the mean of the elements in the vector. In the experiment, Table 1 reveals the correlation between the intensity values and the corresponding color bands, and they all show strong correlation. The triggers generated in the three backdoor attacks have no effect on the correlation of the intensity values among bands. On the other hand, for BadNets, the added triggers show almost zero correlation between bands. Even for Blend Attack and SSBA, they exhibit weak correlation.

We note that it is difficult to detect whether an image is a poisoned one based on the weak correlation of the trigger among different bands. In the pipeline of our defensing method as shown in Figure 1, the backdoor target label is a frog, and the trigger is the invisible additive noises, which are embedded into the clean picture by pretrained encoder. In the training stage, we adopt the poisoned samples and clean samples to train DNNs and then get the backdoored DNN which classifies poisoned samples to the target label, while performing perfect on clean samples. The pretrained detector detects the training set and adds the sample to the detection set if it is predicted to be poisoned. Then, the detection set was re-entered into the backdoored DNNs for backdoor unlearning, which gets clean DNNs. In the inference stage, the clean DNNs will behave normally on the test samples, and the poisoned samples will not be classified into the target label.

4. Our Protocol Design

We will describe the details of the approach to detecting triggers and backdoor unlearning in this section, as outlined in Algorithm 1. Table 2 describes the symbols used in Algorithm 1.

4.1. Backdoor Detection Design. Let $D_{\text{train}} = \{x_i, y_i\}_{i=1}^{n}$ indicates the training set containing n samples, where $x_i \in X$ and $y_i \in Y = \{1, 2, \ldots, K\}$. The DNN model learns a function f_w: $x_i \longrightarrow y_i$ with parameters w, and y_i denotes the label. D_{poison} indicates the poisoned training set, and D_{clean} represents the clean training set. Specifically, D_{train} consists of D_{poison} and D_{clean}, i.e.,

$$D_{\text{train}} = D_{\text{poison}} \cup D_{\text{clean}}, \quad (2)$$

where $D_{\text{poison}} \subset D_{\text{train}}$, $\gamma = |D_{\text{poison}}|/|D_{\text{train}}|$ indicates the poisoning rate, $D_{\text{clean}} = \{(x_i, y_i)((x_i, y_i) \subset D_{\text{train}}/D_{\text{poison}}\}$. Specifically, D_{detect} indicates the set consisting of poisoned samples detected by the detector, where $D_{\text{detect}} \subset D_{\text{poison}}$. Since it is difficult to detect all the poisoned samples in the training set, some of the clean samples are also included in D_{detect}. The more clean samples are included in D_{detect}, the lower the classification accuracy of the model on the clean samples will be after it performs backdoor unlearning. Define the detection rate = $|D_{\text{train}}|/|D_{\text{train}}|$, and ρ plays a key role in the final model performance.

4.1.1. Observation. The trigger generation in most backdoor attack methods is similar to the steganography algorithm applied to images, in which additional noise is embedded in the image. For example, for the attack proposed in [22], $G(x) = \alpha \cdot t + (1 - \alpha) \cdot x, \forall x \in X$, where $G(x)$ generates poisoned sample, and t indicates the backdoor triggers. The trigger generation in SSBA is also motivated by the DNN-based image steganography [37].

Based on the observation and the key intuition, we can detect whether the image is poisoned based on steganalysis. Convolutional neural network structure is widely used in gray-scale image steganalysis. For color image, the summation normal convolution reserves strongly correlated patterns but compromises uncorrelated noise or weak correlated noise. In the process of training the detector, it is necessary to preserve the characteristics of the trigger as much as possible. The wider separate-then-reunion network (WISERNet) [18] chooses a channel-wise convolution in the bottom convolution layer, which can well preserve the features of extra added noise in the image. In addition, WISERNet initializes the convolution kernel using the high-pass filter of the null domain rich model [38] to better extract noise (trigger) features.

4.1.2. How to Build the Detector. We use the WISERNet [18] as a core for the backdoor detector. Since the image

Table 1: The correlation between the intensity of different color bands and those of corresponding triggers.

Attack	Types	Red vs. green	Red vs. blue	Blue vs. green
BadNets [21]	Intensity	0.9512	0.8950	0.9737
	Trigger	0.1781	0.1970	0.2710
Blend Attack [22]	Intensity	0.9596	0.9121	0.9744
	Trigger	0.6672	0.5545	0.8046
SSBA [14]	Intensity	0.9542	0.9005	0.9695
	Trigger	0.6424	0.5960	0.6798

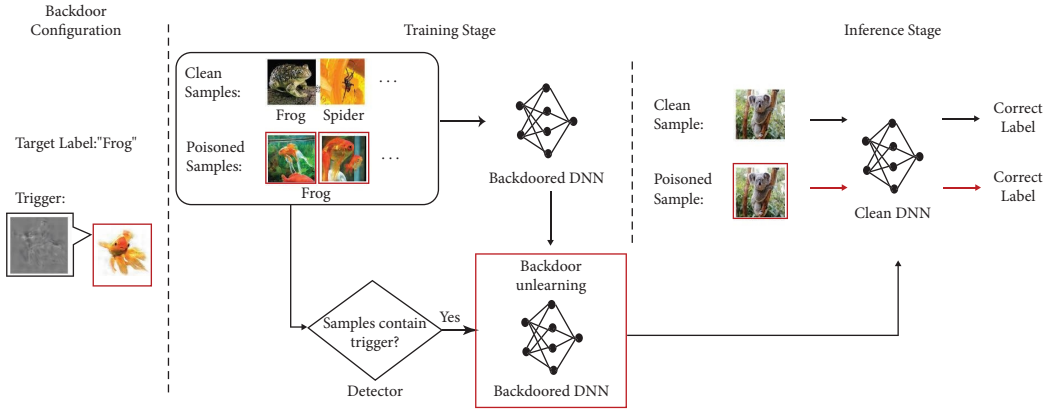


Figure 1: The pipeline of our defensing method.

Table 2: List of symbols.

Symbol	Description
$X_c = \{X_i\}_{i=1}^n$	The clean samples set
A	The backdoored DNN model
B	The clean DNN model
D	The detector
$\varnothing$	The empty set
$G(x)$	The function to generate poisoned sample
θ	The model parameters
∇	Gradient operator
y	Sample label. The sample is clean if $y = 1$ (poisoned if $y = 0$)

convolution operation affects the additional noise [18], the sum in the convolution layer retains the strong correlation pattern but damages the irrelevant noise. Therefore, WISENet uses the normal convolution summation operation in the upper convolution layer rather than using the sum operation in the bottom convolution layer. WISERNet can be divided into three parts in turn: separation, reunion, and prediction. The separated part is composed of channel convolution layer. The main purpose of convolution in the bottom convolution layer is to suppress the relevant image content. WISERNet gives up the sum in the bottom convolution layer and selects the channel volume to reduce the weakening of the network to the irrelevant noise. The reunion part is composed of three wide and relatively shallow normal convolution layers that retain summation. The number of kernels in each convolution layer will gradually increase to augment the capacity of WISERNet. The typical practical method of deep learning network is to design it deeper. However, the deeper the network is, the more output is involved in the summation, and as a result, the more severely the weakly correlated signal is damaged. Therefore, WISERNet designs the upper convolution layer wider to improve its detection performance. The prediction part is composed of four layers of fully connected neural networks to make the final prediction.

As shown in Figure 2, the image is input during the detection process dividing it into red, green, and blue bands, and then, convolution at the channel level is applied separately. The initialization of the convolution kernel weights in each channel is then performed using 30 high-pass filters in the null domain rich model, and as a result, 30 channel feature maps are generated. Finally, the three independent channels are joined together to form a 90 channel output, which is used as the input to the second convolution layer.

Input: A clean sample $X_c = \{X_i\}_{i=1}^{n}$, a training set D_{trian}, a backdoored DNN model A.
Output: A clean DNN model B.

(1) Initialize $D_{\text{detect}} = \varnothing$, $\rho = 0$, and detector D.
(2) //**step 1: generate poisoned-clean pair samples.**
(3) set $X_p = G(x), \forall x \in X_c$, where $G(x)$ generate poisoned sample;
(4) set $\chi = X_c + X_p$; $y = 1; x \in X_c$; $y = 0; x \in X_p$;
(5) //**step 2: Train detector** D**.**
(6) set $\delta \longleftarrow 0$, learning rate $\eta = 0.01$;
(7) **for** epoch $= 1, 2, \ldots, m$ **do**
(8) **for** minibatch $B \subset \chi$ **do**
(9) Update θ of detector D with stochastic gradient descent;
(10) $g_\theta = \mathbb{E}_{(x,y)\subset B}[\nabla_\theta \mathscr{L}(x + \delta, y, \theta)]$;
(11) $\theta = \theta - \eta g_\theta$;
(12) //**step 3: detect poisoned samples in training set.**
(13) set $D_{\text{detect}} = \varnothing, \rho = 0$;
(14) **for** $i = 0; i++; i \le |D_{\text{train}}|$ **do**
(15) $//D(\cdot)$ indicates the inference result of detector D
(16) **while** $\rho \le 0.04$ **do**
(17) **if** $D(x_i) = 0$ **then**
(18) $D_{\text{detect}} = D_{\text{detect}}.\text{append}(x_i)$, where $x_i \in D_{\text{train}}$;
(19) $\rho = |D_{\text{detect}}|/|D_{\text{train}}|$;
(20) break;
(21) //**step 4: Backdoor unlearning.**
(22) input D_{detect} into A and update model by using equation (5);
(23) return the clean model B.

Algorithm 1: Backdoor detection and removal.

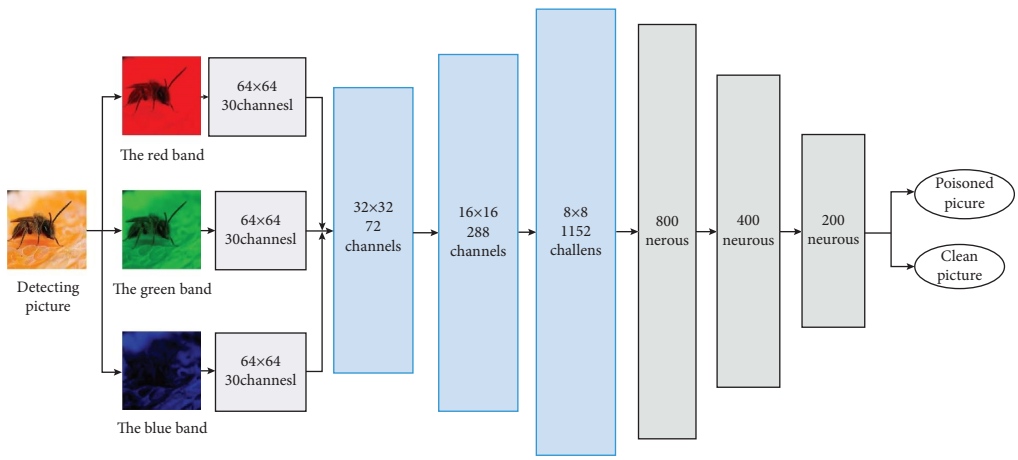


Figure 2: The architecture of wider separate-then-reunion networks [18].

From the second convolutional layer forwards, a standard convolutional approach is used, with the structure of the convolutional operation layer, the batch normalization layer, the activation function layer, and the average pooling layer in order. Since the complexity of the convolutional layers affects the feature extraction and processing, the number of convolutional kernels in each convolutional layer is correspondingly quadrupled to maintain the complexity of the convolutional layers for better noise feature extraction and processing. After the normal convolutional layer, the output feature maps are then combined as variables in 32 steps and input to the fully connected layer. The fully connected layers contain 800, 400, 200, and 2 neurons, respectively, and the three hidden layers use the ReLU activation function. The last fully connected layer performs the final classification prediction result, and if the prediction result is a poisoned sample, the backdoor is buried in the model.

4.2. Backdoor Mitigation Design. Despite the detection of poisoned samples in the training set by the detector, the backdoor in the model still exists. Let $(X, Y) = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ be the training samples, and the training of the model in the backdoor

attack can be achieved by minimizing the following empirical error:

$$\min L = \frac{1}{n}\sum_{1}^{n} D_{\text{clean}}[\ell(f_\theta(x_i), y_i)] + \frac{1}{m}\sum_{1}^{m} D_{\text{poison}}[\ell(f_\theta(x_i), y_i)], \tag{3}$$

where n and m are the number of clean samples and poisoned samples in the training set, respectively. ℓ indicates the loss function such as the cross-entropy loss commonly used in DNN training.

Equation (3) shows that the backdoor injection process can be considered an instance of multitask learning. The main task is the training on the clean samples, whereas the other task is the training on the poisoned samples, that is, the backdoor task. To prevent the model from learning the backdoor task and thus achieving the goal of backdoor unlearning, it can be achieved by minimizing the following empirical error:

$$\min L = \frac{1}{n}\sum_{1}^{n} D_{\text{clean}}[\ell(f_\theta(x_i), y_i)] - \frac{1}{m}\sum_{1}^{m} D_{\text{poison}}[\ell(f_\theta(x_i), y_i)]. \tag{4}$$

Equation (4) maximizes the backdoor task compared to (3).

Since it is difficult to detect all the D_{poison} in the training set, and the training set of detected poisoned samples is also containing some clean samples, it makes the classification accuracy of the model on clean samples drop significantly. Therefore, we use the detection dataset D_{detect} instead and achieve the effect of backdoor unlearning by minimizing the following empirical error:

$$\min L = \frac{1}{n}\sum_{1}^{n} D_{\text{clean}}[\ell(f_\theta(x_i), y_i)] - \frac{1}{m'}\sum_{1}^{m'} D_{\text{detect}}[\ell(f_\theta(x_i), y_i)]. \tag{5}$$

5. Experiments

In this section, we implement our protocol based on the datasets of CIFAR10 [39] and VGGFACE2 [40]. We experimentally test the trigger performance and analyze the effects of trigger location, trigger size, and the string representing the target label by the attack SSBA on the performance of the detector. In addition, we experimentally analyze the effect of the size of the detection rate ρ on the performance of the model and arrive at the value of ρ for which the defensing protocol achieves better results when targeting a variety of backdoor attacks. Finally, the effectiveness of this protocol is compared with existing typical backdoor defensing protocols to analyze the effectiveness of our protocol.

5.1. Experiment Setup. The implementation of the detector is based on the Caffe toolbox [41]. The network is trained using small batch stochastic gradient descent with an initial learning rate of 0.001, a learning rate adjustment strategy set to inv, and a fixed momentum of 0.9. The maximum number of training iterations is set to 20,000, and the batch size is 16 during training. All training and testing procedures are performed on a server with the hardware of NVIDIA GeForce RTX 2080 GPU and 10 GB of RAM. The software used for the server is Linux (3.2.x) operating system and Python 3.6.3. To evaluate the defensing approach, we consider two classical image classification tasks: object classification and face recognition. The detailed information about each task and the associated dataset are described in Table 3.

Object Classification (CIFAR10 [39]): This task is commonly used to evaluate attacks against DNNs and was chosen to train the model PreActResNet [42] using the CIFAR10 dataset. The original dataset contains 10 classes, which contains 50,000 training datasets and 10,000 test datasets.

Face Recognition (VGGFace2 [40]): This task recognizes the faces of 200 people by training the model ResNet [43]. The original dataset contains 3.31 million images. We randomly select 200 categories which contain 400 images for training and another 50 images for testing.

According to the backdoor attacks, we use three already infected object classification models and face recognition models by BadNets [21], Blend Attack [22], and SSBA [14].

The poisoning rate $\gamma = 10\%$ and the target label are set to 0. Figure 3 shows the poisoned samples generated by the three attacks. The backdoor trigger is set to a white square located in the lower right corner of the image, which only accounts for 1% area of the image for BadNets and Blend Attack, and the blending rate (trigger transparency) is set to 0.2 for the Blend Attack. For SSBA, the trigger is generated by the encoder that is a U-Net [44] style DNN trained on the clean samples, which achieves the invisibility and sample-specific of the trigger.

We adopt three effective performance metrics: attack success rate (ASR), which is the classification accuracy on the poisoned test set, clean accuracy (CA), which is the classification accuracy on the clean test set, and detection success rate (DSR), which is the success rate of detecting poisoned samples on the training set. Table 4 shows ASR and CA of the three backdoor attacks on the two classification tasks.

5.2. The Effect of Backdoor Detection. The success rate of detecting poisoned samples by the detector is the key factor to judge the effectiveness of our protocol. For the above three attacks, the data are poisoned accordingly and then detected by the detector. In each experiment, first 10,000 images in the training set are randomly selected to add triggers, and the way of adding triggers is kept the same as in the experimental setup. Then, 6000 pairs of clean-poisoned images are randomly selected and input into the WISERNet for training, while the remaining 4000 pairs are used for testing. Table 5 shows the detection success rates for the three attacks under the two tasks, respectively. 99% of the poisoned images can be detected for both the BadNets and Blend Attack on given datasets. For SSBA, above 94 % of the poisoned images can be detected on the CIFAR10 dataset and 99% on the VGGFACE2 dataset.

Considering the effects of changing the shape and position of the trigger and the different strings representing the target labels in SSBA on the detection success rate, we discuss the effects on the detection success rate by modifying the shape and position of the trigger and the strings and then feed them into the already trained WISERNet.

Figure 4 shows the effect of different trigger shapes and positions in BadNets on the detection success rate and the effect of different representative strings in SSBA on the detection success rate, both experiments on the VGGFACE2 dataset. The model (model1) is the detector trained with the poisoned samples generated by the BadNets, and the triggers are 9×9 white squares in the lower right corner of the image. The other model (model2) is the detector trained with the poisoned samples generated by SSBA method, and the string embedded in the image is 0. In Figure 4(a), the trigger shapes are set to white blocks with circles, ovals, and triangles and then input into model1 to get the detection results. In Figure 4(b), the position of the trigger is set at the four corners of the image, respectively, and then input into *model*1 for detection. In Figure 4(c), the strings embedded into the images are set to 0, 1, 2, and 3, respectively, and then input into model2 to get its classification results. Figure 4 shows that the content of the representative string in SSBA does not affect the efficiency of the detector, and it can achieve more than 96 % detection success rate for poisoned images. When the size of the trigger does not cover the entire picture, it changes its position and shape that can affect the efficiency of the detector.

The position and shape of the triggers affect the detection success rate, but the content of the representative string in SSBA does not affect the detection success rate. Since the way of adding the trigger in SSBA makes the trigger and the features of clean samples fused, its feature position also overlaps with the position of the main features of those clean samples. Thus, the trigger position and shape are not critical factors in the training process of WISERNet. Furthermore, the trigger features in BadNets differ from the main features, and the position and shape have some influence on the results.

5.3. The Effect of Backdoor Mitigation. The performance of the model after backdoor unlearning can be optimal in equation (4) if all poisoned samples in the training set are detected and no clean samples are mistakenly detected as poisoned samples. However, it is hard to arrive that the detection method does not detect 100 % of the poisoned samples. In addition, it may be affected by the dataset, such as the trigger set in BadNets attack is the white square in the bottom right corner of the image, yet some of the images in the CIFAR10 dataset are also white in the bottom right corner, which will lead to the wrong detection. Therefore, there will be a small number of clean samples included in D_{detect}. Usually, the larger the value of $|D_{\text{detect}}|$, the lower the success rate of the attack after the backdoor unlearning. However, if a number of the clean samples are included in D_{detect}, w will make the classification accuracy on the clean samples drop significantly. Therefore, we experimentally investigate the correlation between the value of ρ and the performance of our protocol.

In the CIFAR10 dataset, the poisoning rate is set to 10%, and thus, there are 5,000 poisoned images in the training set. Set ρ values at 0.02, 0.04, 0.06, 0.08, and 0.1. The optimal range of ρ values is experimentally derived, which maintains the classification accuracy on benign samples while reducing ASR. Figure 5 shows the implementation on the CIFAR10 dataset with different ρ values for different backdoor attacks. It can be found that our protocol is effective against all three attacks at different ρ. The backdoor attack rate can drop to very close to 0% while the classification accuracy of the model on clean samples maintains at a high level. We also find that the best performance of our protocol is achieved when $\rho \leq 0.04$.

5.4. Comparison with the Existing Defensing Protocols. To further evaluate the effectiveness of our protocol, we consider three state-of-the-art backdoor attacks and compare with two typical backdoor defensing techniques. Table 6 demonstrates our proposed method on the CIFAR-10 dataset and the VGFACE2 subset dataset. FP [15] and ABL [19] are following the configurations specified in their

Figure 3: Poisoning samples generated by different backdoor attacks: BadNets, Blend Attack, and SSBA. (a) CIFAR 10. (b) VGG face2.

Table 3: Details of datasets and model architectures.

Task	Dataset	# of labels	Input size	# of training images	Model architecture
Object classification	CIFAR10	10	$32 \times 32 \times 3$	50000	PreActResNet
Face recognition	VGGFace2	200	$64 \times 64 \times 3$	80000	ResNet

Table 4: Attack success rate (ASR) and clean accuracy (CA) of various backdoor attacks on classification tasks.

Task	Backdoored model (ASR %/CA %)			Clean model (CA %)
	BadNets	Blend Attack	SSBA	
CIFAR10	99.64/93.02	100/93.67	99.91/93.08	92.40
VGGFace2	99.40/87.80	99.98/87.84	99.67/88.59	91.31

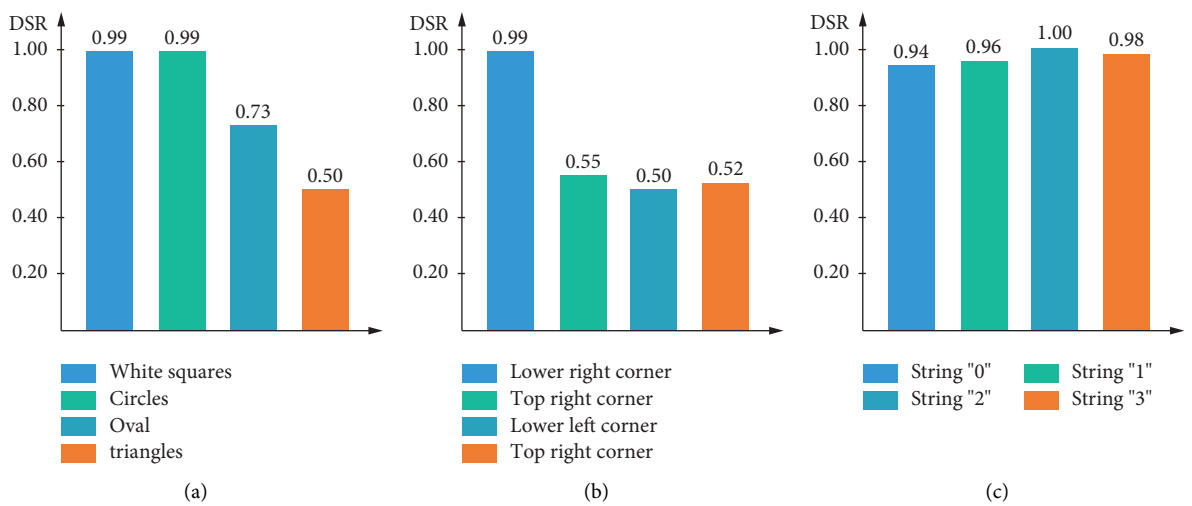


Figure 4: Detection success rate related to various trigger shapes, positions, and representative strings: (a) Effect of trigger shape on detection success rate, (b) effect of trigger location on detection success rate, and (c) effect of different string on detection success rate.

original papers. In addition, the last convolutional layer of the neural network in FP is pruned, and ASR of the model significantly decreases when 60 % of the neurons are pruned.

Let epoch $T = 107$ and turn-period $T_{te} = 25$ be set in the training of the CIFAR10 dataset, and epoch $T = 46$ and turn-period $T_{te} = 25$ in the training of the VGGFACE2 subset

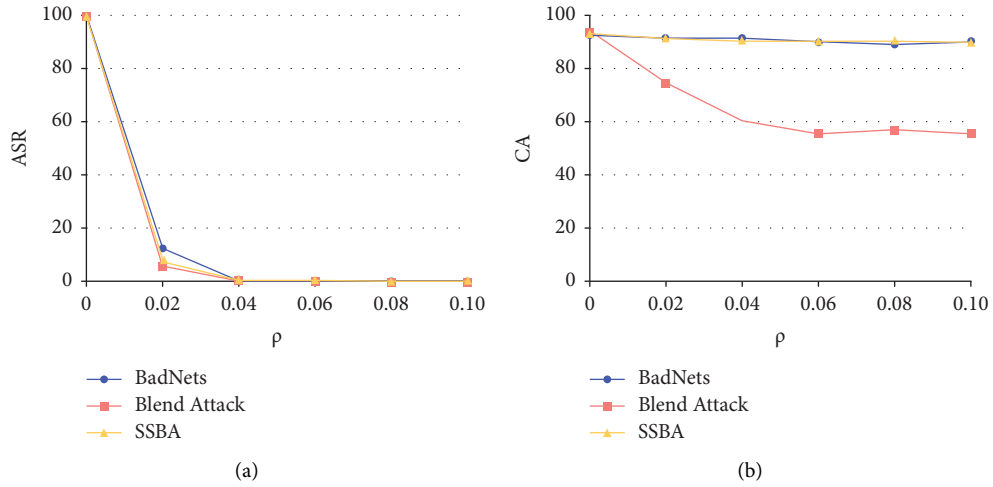


Figure 5: The effect performance on different detection rate ρ.

Table 5: Detection success rate against three typical attacks.

Datasets	Backdoored model (DSR %)		
	BadNets	Blend Attack	SSBA
CIFAR10	99.85	100.00	94.92
VGGFace2	99.78	100.00	99.68

Table 6: Effectiveness performance comparison of defensing protocols under different backdoor attacks.

Dataset	Attack type	FP [15]		ABL [19]		Ours	
		ASR %	CA %	ASR %	CA %	ASR %	CA %
CIFAR10	None Attack	0.00	91.88	0.00	92.75	0.00	**93.79**
	BadNets	99.81	90.37	0.42	**93.14**	**0.21**	90.54
	Blend Attack	100.00	**93.43**	0.48	76.56	**0.15**	60.43
	SSBA	99.90	93.09	0.50	**93.17**	**0.43**	90.81
VGGFACE2 subset	None Attack	0.00	72.62	0.00	82.96	0.00	**86.73**
	BadNets	11.79	77.26	0.00	14.90	**0.32**	**83.36**
	Blend Attack	14.89	71.46	0.00	9.72	**0.46**	**78.67**
	SSBA	11.47	72.23	0.00	7.97	**0.17**	**84.27**

For different attacks, bold values represents the best defense effect among the three defense schemes.

dataset in the defensing protocol ABL. In both datasets, our protocol is set to $\rho = 0.04$. None Attack in Table 6 means that the training data are completely clean.

In the CIFAR10 dataset, ABL can achieve better results in the classification accuracy of clean samples compared to our protocol, but our protocol can achieve the best decrease in the reduction of the attack success rate. In the subset of VGGFACE2 dataset, FP can reduce the attack success rate of the three attack methods to less than 15%, but at the same time, the classification accuracy of the clean samples also decreases to less than 75%. ABL reduces the attack success rate of the three attack methods to 0, but the performance of the clean samples of the model is poor; thus, we can assume that ABL has no defensive effect. Our protocol has better performance in both attack success rate and classification accuracy on the clean samples. In Table 6, it can be seen that Blend Attack, both ABL and our protocol, decreases in attack success rate and classification accuracy compared to other attack methods, which is because the dataset images are blurred, and the trigger pattern mixed with poisoned images produces the effect of natural artifacts, which makes it difficult to detect poisoned images. Maintaining the classification accuracy of the model on clean samples is as important as reducing the success rate of the attack. Table 6 shows that our protocol is better to maintain the classification accuracy of the model on clean samples while reducing the success rate of the attack compared with FP and ABL.

6. Conclusion

In this work, we propose a backdoor detecting and removing protocol for deep neural networks based on image steganalysis. Our protocol detects the poisoned training samples using a deep steganalyzer constructed by WISERNet and

retrains the model for backdoor unlearning by the detected poisoned samples. Compared with the SOTA backdoor defensing protocols, our protocol achieves to reduce the backdoor attack success rate while maintaining a high classification accuracy on the clean samples. In the future work, we will further study the backdoor detection and unlearning methods to obtain higher clean sample classification accuracy and lower backdoor attack success rate for different attack methods and design universal and efficient backdoor defensing protocols.

Data Availability

The data supporting the current study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 61972241 and 62102300, Natural Science Foundation of Shanghai under Grant 22ZR1427100 and 18ZR1417300, Soft Science Foundation of Shanghai under Grant 23692106700, Fishery Engineering and Equipment Innovation Team of Shanghai High-Level Local University, and Luo-Zhaorao College Student Science and Technology Innovation Foundation of Shanghai Ocean University.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei, "Imagenet Large Scale Visual Recognition Competition," 2012, https://arxiv.org/abs/1409.0575.

[3] A. Graves, M. Abdel-rahman, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649, Ieee, Vancouver, Canada, May 2013.

[4] X. Li, J. He, P. Vijayakumar, X. Zhang, and V. Chang, "A verifiable privacy-preserving machine learning prediction scheme for edge-enhanced hcpss," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 8, pp. 5494–5503, 2022.

[5] C. Zhou, D. Chen, S. Wang, A. Fu, and Y. Gao, "Research and challenge of distributed deep learning privacy and security attack," *Journal of Computer Research and Development*, vol. 58, no. 5, pp. 927–943, 2021.

[6] Y. Liu, A. Mondal, A. Chakraborty et al., "A survey on neural trojans," in *Proceedings of the 2020 21st International Symposium on Quality Electronic Design (ISQED)*, IEEE, Santa Clara, CA, USA, March 2020.

[7] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor Learning: A Survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 2022, 2022.

[8] Y. Gao, G. Bao, Z. Zhang et al., "Backdoor Attacks and Countermeasures on Deep Learning: A Comprehensive Review," 2020, https://arxiv.org/abs/2007.10760.

[9] M. Goldblum, D. Tsipras, C. Xie et al., "Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, 2022.

[10] Z. Tian, L. Cui, J. Liang, and S. Yu, "A Comprehensive Survey on Poisoning Attacks and Countermeasures in Machine Learning," *ACM Computing Surveys (CSUR)*, vol. 55, 2022.

[11] K. Doan, Y. Lao, and P. Li, "Backdoor attack with imperceptible input and latent modification," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18944–18957, 2021.

[12] K. Doan, Y. Lao, W. Zhao, and P. Li, "Lira: learnable, imperceptible and robust backdoor attacks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11966–11976, Montreal, Canada, October 2021.

[13] Q. Zhang, Y. Ding, Y. Tian, J. Guo, M. Yuan, and Y. Jiang, "Advdoor: adversarial backdoor attack of deep learning system," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, Virtual, Denmark, July 2021.

[14] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu, "Invisible backdoor attack with sample-specific triggers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16463–16472, Montreal, Canada, October 2021.

[15] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: defending against backdooring attacks on deep neural networks," in *International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 273–294, Springer, Berlin, Germany, 2018.

[16] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: a defence against trojan attacks on deep neural networks," in *Proceedings of the 35th Annual Computer Security Applications Conference*, New York, NY, USA, December 2019.

[17] B. Wang, Y. Yao, S. Shan et al., "Neural cleanse: identifying and mitigating backdoor attacks in neural networks," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*, pp. 707–723, IEEE, Francisco, CA, USA, May 2019.

[18] J. Zeng, S. Tan, G. Liu, B. Li, and J. Huang, "Wisernet: wider separate-then-reunion network for steganalysis of color images," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 10, pp. 2735–2748, 2019.

[19] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Anti-backdoor learning: training clean models on poisoned data," *Advances in Neural Information Processing Systems*, vol. 34, pp. 14900–14912, 2021.

[20] Q. Tan, Y. Zeng, Y. Han, Y. Liu, and Z. Liu, "Survey on backdoor attacks targeted on the neural network," *Chinese Journal of Network and Information Security*, vol. 7, no. 3, pp. 46–58, 2021.

[21] T. Gu, B. Dolan-Gavitt, and S. G. Badnets, "Identifying Vulnerabilities in the Machine Learning Model Supply Chain," 2017, https://arxiv.org/abs/1708.06733.

[22] X. Chen, L. Chang, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017, https://arxiv.org/abs/1712.05526.

[23] Y. Liu, S. Ma, Y. Aafer et al., "Trojaning attack on neural networks," *25th Annual Network and Distributed System Security Symposium (NDSS)*, Purdue University, West Lafayette, IN, USA, 2018.

[24] D. Chen, A. Fu, C. Zhou, and Z. Cheng, "Federated learning backdoor attack protocol based on generative adversarial

network," *Journal of Computer Research and Development*, vol. 58, no. 11, pp. 2364–2373, 2021.

[25] S. Garg, A. Kumar, V. Goel, and Y. Liang, "Can adversarial weight perturbations inject neural backdoors," in *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, pp. 2029–2032, New York, NY, USA, December 2020.

[26] E. Bagdasaryan and V. Shmatikov, "Blind backdoors in deep learning models," *30th USENIX Security Symposium USENIX Security*, vol. 21, pp. 1505–1521, 2021.

[27] Y. Zeng, W. Park, Z. M. Mao, and R. Jia, "Rethinking the backdoor attacks' triggers: a frequency perspective," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16473–16481, New York, NY, USA, December 2021.

[28] S. Li, M. Xue, B. Zhao, H. Zhu, and X. Zhang, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 1–2105, 2020.

[29] P. Zhao, P. Chen, P. Das, Karthikeyan Natesan Ramamurthy, and X. Lin, "Bridging mode connectivity in loss landscapes and adversarial robustness," 2020, https://arxiv.org/abs/2005.00060.

[30] T. Garipov, Pavel Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson, "Loss surfaces, mode connectivity, and fast ensembling of dnns," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[31] Y. Liu, Wen-Chuan Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: scanning neural networks for back-doors by artificial brain stimulation," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1265–1282, New York, NY, USA, November 2019.

[32] Wikipediapedia, "Electrical Brain Stimulation," 2022, https://en.wikipedia.org/wiki/Electricalbrainstimulation.

[33] G. Bao, E. Abbasnejad, and D. C. Ranasinghe, "Februus: input purification defense against trojan attacks on deep neural network systems," in *Proceedings of the Annual Computer Security Applications Conference*, pp. 897–912, Austin, CA, USA, December 2020.

[34] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, Cambridge, MA, USA, June 2017.

[35] S. Udeshi, S. Peng, G. Woo, L. Loh, L. Rawshan, and S. Chattopadhyay, "Model Agnostic Defence against Backdoor Attacks in Machine Learning," *IEEE Transactions on Reliability*, vol. 71, 2022.

[36] Q. Han, Y. Zeng, S. Guo, T. Zhang, M. Qiu, and B. Thuraisingham, "Deepsweep: an evaluation framework for mitigating dnn backdoor attacks using data augmentation," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pp. 363–377, New York, NY, USA, September 2021.

[37] M. Tancik, Ben Mildenhall, and N. Ren, "Stegastamp: invisible hyperlinks in physical photographs," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2117–2126, New Orleans, LA, USA, January 2020.

[38] J. Fridrich and J. Kodovsky, "Rich models for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 868–882, 2012.

[39] A. Krizhevsky and G. Hinton, *Learning Multiple Layers of Features from Tiny Images (2009)*, University of Toronto, Toronto, Canada, 2009.

[40] Q. Cao, L. Shen, W. Xie, M. Omkar, and A. Zisserman, "Vggface2: a dataset for recognising faces across pose and age," in *Proceedings of the 2018 13th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2018)*, pp. 67–74, IEEE, Vancouver, Canada, January 2018.

[41] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, and R. Girshick, "Sergio guadarrama, and trevor darrell. Caffe: convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM International Conference on Multimedia*, pp. 675–678, New Yor, NY, USA, September 2014.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European Conference on Computer Vision*, pp. 630–645, Springer, Berlin, Germany, 2016.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, New Orleans, LA, USA, May 2016.

[44] O. Ronneberger, P. Fischer, and T. Brox, "U-net: convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241, Springer, Berlin, Germany, 2015.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 7751845, 17 pages
https://doi.org/10.1155/2022/7751845

Research Article

Practical Privacy Preserving-Aided Disease Diagnosis with Multiclass SVM in an Outsourced Environment

Ruoli Zhao,[1] **Yong Xie**,[1] **Xingxing Jia**,[2] **Hongyuan Wang**,[3] **and Neeraj Kumar**[4,5]

[1]*Department of Computer Technology and Applications, Qinghai University, Xining, China*
[2]*School of Mathematics and Statistics, Lanzhou University, Lanzhou, China*
[3]*Qinghai Province Yindajihuang Project Construction and Operation Bureau, Xining, China*
[4]*School of Computer Science, University of Petroleum and Energy Studies, Dehradun, Uttarakhand, India*
[5]*Department of Computer Science and Information Engineering, Asia University, Taichung, Taiwan*

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

Received 20 April 2022; Accepted 20 September 2022; Published 12 October 2022

Academic Editor: Ch. Aswani Kumar

With the rapid development of cloud computing and machine learning, using outsourced stored data and machine learning model for training and online-aided disease diagnosis has a great application prospect. However, training and diagnosis in an outsourced environment will cause serious challenges to the privacy of data. At present, many scholars have proposed privacy preserving machine learning schemes and made a lot of progress, but there are still great challenges in security and low client load. In this paper, we propose a complete privacy preserving outsourced multiclass SVM training and aided disease diagnosis scheme. We design some efficient basic operation algorithms for encrypted data. Then, we design an efficient and privacy preserving SVM model training protocol using the basic operation algorithms. We propose a secure maximum finding algorithm and secure comparison algorithm. Then, we design an efficient online-aided disease diagnosis scheme based on the BFV cryptosystem and blinding technique. Detailed security analysis proves that our scheme can protect the privacy of each participant. The experimental results illustrate that our proposed scheme significantly reduces the computation overhead compared with the previous similar works. Our proposed scheme completes most of the operations of aided disease diagnosis by the cloud servers and the client only needs to complete a small amount of encryption and decryption operations. The overall computation overhead is 0.175 s, and the efficiency of online aided disease diagnosis is improved by 85.4%. At the same time, our proposed scheme provides multiclass diagnosis results, which can better assist doctors in their treatment.

1. Introduction

Machine learning (ML) uses the computer system to build mathematical models on sample data with statistical methods and makes predictions or decisions without being explicitly programmed. Now, ML has shown significant advantages in the field of disease diagnosis and brings more and more convenience to the prevention and treatment of diseases.

With the rapid development of cloud computing technology, cloud service providers (CSP) have high-quality computation and huge storage space, which can provide data processing, model training, diagnosis services and deployment, and other intelligent solutions based on machine learning. In this context, the local clients will outsource their medical data and machine learning models to CSP without having to build their own large-scale infrastructure and computing resources. The cloud can train a machine learning model and provide aided disease diagnosis service by using the outsourced medical data and machine learning models, which can help improve doctors' diagnosis, treatment decisions and provide patients an online disease diagnosis service. A typical cloud platform machine learning system architecture is shown in Figure 1.

However, the security and privacy of outsourced data will be threatened by various threats, making people afraid to

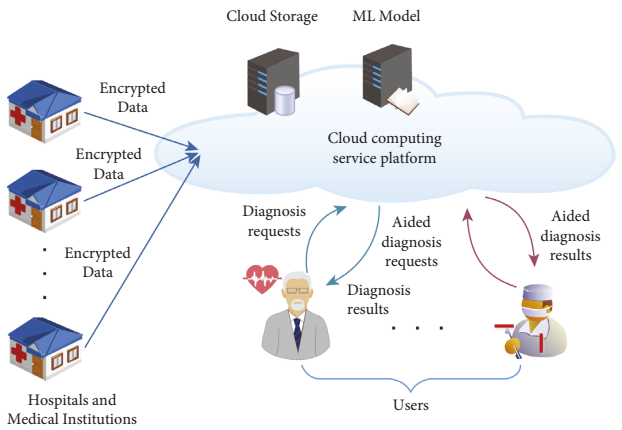


Figure 1: A typical cloud platform machine learning system architecture.

use the service of CSP. The security and privacy threats are mainly reflected in the leakage of the data, the machine learning model of the model owners, the users' request, and diagnosis results. As we all know, the leakage of medical information may cause irreversible losses or become a major event. Therefore, the security and privacy preserving of model training and diagnosis based on cloud computing have become a major challenge.

To address the abovementioned challenges, many scholars have proposed various schemes, such as a secure outsourced classification based on logistic regression model [1], an electronic medical disease risk prediction scheme based on naive Bayes model [2], and other secure disease prediction schemes based on machine learning technology [3–5]. As a machine learning algorithm with high computational efficiency and nice predictive accuracy, the support vector machine (SVM) has achieved high classification accuracy and efficiency in the medical field [6, 7]. However, the existing privacy preserving SVM schemes mainly implement secure prediction [8–11], and there are few privacy preserving SVM schemes for secure training. Most of the existing privacy preserving SVM schemes are designed for binary classification, which can only determine whether the patient has the disease [12], but cannot deal with the multiclass of the disease. In addition, multiclass SVM requires more computation, which will reduce the efficiency [13].

To solve the abovementioned problems, we propose an efficient and privacy preserving online disease diagnosis scheme based on the SVM algorithm. In our scheme, we can achieve multi-class SVM training on the encrypted outsourced data from multiple data owners and provide users with privacy preserving disease diagnosis. In summary, our contributions are as follows:

(1) Efficient and secure basic operation algorithms: Based on the Paillier cryptosystem, we design several basic operation algorithms to realize the secure outsourced data storage and computation, including secure aggregation algorithm, secure multiplication algorithm, and so on. These secure computation algorithms are the building blocks for our proposed training protocol.

(2) Completing machine learning process under privacy preserving: Aiming at the general machine learning process and the goal of privacy preserving, we propose a privacy preserving outsourced multiclass SVM model training and online-aided disease diagnosis scheme. Different from the existing privacy preserving schemes that only support training or diagnosis, our proposed scheme extends the function of privacy preserving machine learning system.

(3) Efficient and secure online aided disease diagnosis: Based on the BFV cryptosystem, we design a secure maximum finding algorithm and secure comparison algorithm. We provide an efficient and privacy preserving aided disease diagnosis scheme. Experimental results illustrate that our proposed scheme significantly reduces the computation cost than the existing similar schemes, which is suitable for practical application scenarios where a large number of users request diagnosis at the same time.

(4) Low overhead for local client: For a local client, the client only needs to perform encryption and decryption operations in our proposed scheme, which reduces the storage and computation overhead of the local client to the greatest extent and makes full use of the computation power of the cloud servers.

The remainder of this paper is organized as follows. In Section 2, we review some related works. In Section 3, we review the Paillier cryptosystem, BFV cryptosystem, and SVM algorithm as preliminaries. In Section 4, we make a system overview. Then, we propose our scheme in Section 5. In Section 6, we analyze the security of our proposed scheme. In Section 7, we make a performance evaluation. Finally, we conclude this paper in Section 8.

2. Related Work

In this section, we summarize the privacy preserving machine learning schemes in recent years.

With the development of big data era, machine learning has been widely used in many fields. Among them, the application of machine learning in the field of intelligent disease diagnosis has developed rapidly. Disease diagnosis schemes based on various machine learning classification algorithms have been proposed [14–17]. However, at the same time, the problem of privacy disclosure in the machine learning process is becoming more and more serious. So, many scholars have carried out the research studies on privacy preserving machine learning.

Triastcyn and Faltings [18] proposed the Bayesian differential privacy, considered the distribution of data and provided a more practical privacy guarantee. Laur et al. [19] proposed a privacy preserving scheme of support vector machine based on secure multiparty computation. For each training or testing phase, their scheme involves multiple parties holding encrypted data and secret sharing obtained

during training. Based on additive homomorphic encryption, Mandal and Gong [20] designed a privacy preserving scheme that performs gradient descent on data owners and cloud server. They achieved secure linear and logistic regression model training. Shen et al. [21] used blockchain technology to establish a secure and reliable data sharing platform among multiple data providers and constructed a privacy preserving support vector machine training scheme based on the Paillier cryptosystem. However, in their scheme, the data provider needs to interact with the cloud server to complete the computation. The computation cost of the data provider is large. Liu et al. [22] proposed a privacy preserving clinical decision support system using the naive Bayes (NB) classifier. The BGV homomorphic encryption system significantly improved the performance. In work [23], a framework for securely and efficiently outsourcing decision tree inference was proposed. Tan et al. [24] proposed a system for privacy-preserving machine learning that implements all operations on the GPU, which makes full use of the computing power of GPU. Zheng et al. [25] combined random permutation and arithmetic secret sharing by the compute-after-permutation technique and built a privacy-preserving machine learning framework. Li et al. [26] proposed a verifiable privacy-preserving machine learning prediction scheme for the edge-enhanced HCPSs, which outputs the verifiable prediction results for users without privacy leakage. Ma et al. [27] designed a lightweight privacy-preserving medical diagnosis mechanism on edge called LPME.

Among them, the SVM algorithm is a research hotspot and has been widely used in different data mining and machine learning schemes. Most of the existing privacy preserving SVM schemes are based on three main privacy preserving technologies: differential privacy (DP), secure multi-party computation (SMC), and homomorphic encryption (HE). DP can significantly improve the calculation and communication efficiency, but the cost is to sacrifice the accuracy of the model by adding random noise [28, 29]. Zhang et al. [30] proposed a general differential privacy model fitting method based on the genetic algorithm, but it reduces the decision accuracy of the model. SMC alleviates the limitation of computing but requires more interaction between participants. This leads to expensive communication overhead [31, 32]. Yu et al. [33] first proposed a privacy preserving SVM classification method based on vertically segmented data. They use SMC technology to obtain the global model, so as to protect the local privacy data and hide the classification model. However, this method requires at least three parties to participate in the calculation, which is complex and inefficient. HE can directly calculate the encrypted data, but it also requires a lot of computing costs [34, 35]. Bajard et al. [36] uses HE technology to protect the decision model and medical data, but it needs high computational load. Therefore, it is necessary to design an efficient and secure SVM scheme for cloud online disease diagnosis service. Wang et al. [37] proposed an efficient privacy preserving outsourced SVM scheme for Internet of medical things deployment, which protected training data privacy and guaranteed the security of the trained SVM model.

In this paper, we propose a new privacy preserving scheme for training and disease diagnosis of the multiclass SVM algorithm. We make a comparison analysis with the schemes in [38–40]. The experimental results demonstrate that our scheme has more practical application values.

3. Preliminaries

In this section, we describe some techniques as the basis of our scheme, including the Paillier cryptosystem, BFV cryptosystem, and SVM algorithm.

3.1. Paillier Cryptosystem. In the training phase, the data are encrypted by the Paillier cryptosystem [41]. The Paillier cryptosystem is a public key cryptosystem with additive homomorphic operation. We will introduce the Paillier cryptosystem as follows.

(i) Key generation: Set the security parameter k. Choose two big primes p, q, $|p| = |q| = k$, $n = p \cdot q$, $\lambda = \mathrm{lcm}(p-1, q-1)$, λ is the Carmichael function of n. Choose a random number $g \in Z_{n^2}^*$, and $\gcd(L(g^\lambda \bmod n^2), n) = 1$, $L(x) = (x-1)/n$. The public key is $pk = (n, g)$. The private key is $sk = \lambda$.

(ii) Encryption: Given $m \in Z_n$. The message m will be encrypted with pk. The ciphertext is expressed as $c = E_{pk}(m) = g^m r^n \bmod n^2$, where $r \in Z_n^*$ is a random number.

(iii) Decryption: According to the key generation stage and Carmichael's theorem, $g^\lambda \equiv 1 \bmod n$. So $g^\lambda = kn + 1$. Then, $m = D_{sk}(c) = (L(c^\lambda \bmod n^2) / L(g^\lambda \bmod n^2)) \bmod n$.

(iv) Homomorphic computation: Given two ciphertexts $E_{pk}(m_1), E_{pk}(m_2)$ under the same public key pk. The homomorphic computations are defined as $E_{pk}(m_1 + m_2) = E_{pk}(m_1) \cdot E_{pk}(m2)$, $E_{pk}(m_1 \cdot m_2) = E_{pk}(m_1)^{m_2}$.

3.2. BFV Cryptosystem. In the prediction phase, the data are encrypted by the BFV cryptosystem [34]. BFV cryptosystem is a leveled-FHE public key cryptosystem based on RLWE, which can support unlimited times additive homomorphic operation and limited times multiplicative homomorphic operation.

(i) Key generation: Generate a polynomial $s = Z[x]/(x^d + 1)$. The private key is defined as $sk = s$. Then, generate a polynomial from ciphertext polynomial space (polynomial s), $a = Z_q[x]/(x^d + 1)$. The polynomial a is used to generate public key. Define a noise polynomial $e \leftarrow \chi$. The notation χ expresses the Gaussian distribution. The public key is $pk = ([-(a \cdot s + e)]_q, a)$.

(ii) Encryption: The message $m \in R_t$. Define $p_0 = pk[0]$, $p_1 = pk[1]$, $u \leftarrow \chi$, $e_1 \leftarrow \chi$, $e_2 \leftarrow \chi$. The ciphertext c is computed as $c = (p_0 \cdot u + t \cdot e_2 + m, p_1 \cdot u + t \cdot e_1)$.

(iii) Decryption: To decrypt the ciphertext c, define $c_0 = p_0 \cdot u + t \cdot e_2 + m, c_1 = p_1 \cdot u + t \cdot e_1$. The message m is computed as $m = (c_0 + c_1 \cdot s) \bmod t$.

(iv) Homomorphic computation: BFV cryptosystem supports ciphertext batch processing. Define two z-dimensional vectors encrypted under public key pk, $E_{pk}(x_1, x_2, \ldots, x_z), E_{pk}(y_1, y_2, \ldots, y_z)$. The homomorphic computations are defined as follows:

$$\begin{aligned} &E_{pk}(x_1 + y_1, x_2 + y_2, \ldots, x_z + y_z) = E_{pk}(x_1, x_2, \ldots, x_z) \\ &\quad + E_{pk}(y_1, y_2, \ldots, y_z), E_pk(x_1 + \cdot y_1, x_2 \cdot y_2, \ldots, x_z \cdot y_z) \\ &= E_{pk}(x_1, x_2, \ldots, x_z) \cdot E_{pk}(y_1, y_2, \ldots, y_z). \end{aligned} \tag{1}$$

3.3. SVM Algorithm. SVM is a classical supervised learning algorithm to solve two kinds of classification problems. The SVM algorithm will find the best hyperplane. The classifier is a decision function $f(X) = \langle W \cdot X \rangle + b$, $f(X) \geq 0$ expresses positive class and $f(X) < 0$ expresses negative class.

There are two training methods for the SVM model: one is based on the SMO algorithm and the other is based on the gradient descent algorithm. Because the operation steps of the SMO algorithm are more complex, which makes a lot computation costs when using encrypted data. Therefore, we choose gradient descent to realize the privacy preserving SVM model training. In the SVM model training process based on the gradient descent method, the objective function $L(X) = (1/2)|W|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\langle W \cdot X \rangle + b)) = (1/2)|W|^2 + C\sum_{i=1}^{n}$ loss needs to be minimized. When $y_i(\langle W \cdot X \rangle + b) \geq 1$, it means that the classification is correct. The $loss = 0$ and the parameters do not need to be updated. When $y_i(\langle W \cdot X \rangle + b) < 1$, it means that the classification is incorrect. The loss $= 1 - y_i(\langle W \cdot X \rangle + b)$ and the parameters need to be updated.

4. System Overview

In this section, we will introduce our system model, security goals, and threat model.

4.1. System Model. Our system model should achieve the privacy preserving training and online disease diagnosis process. Therefore, our system model is designed as shown in Figure 2.

There are six participants in our system model, which are trusted authority (TA), medical centers (MCs), cloud storage server (CSS), cloud computation server (CCS), diagnosis service provider (DSP), and users.

(i) Trusted authority (TA): TA is the fully trusted party of the whole system, which is used to generate and distribute keys for other participants in the system. After initialization, TA will stay offline.

(ii) Medical centers (MCs): Each MC has its own local medical data. To reduce the local storage cost, MCs will outsource the medical data to CSS for storage.

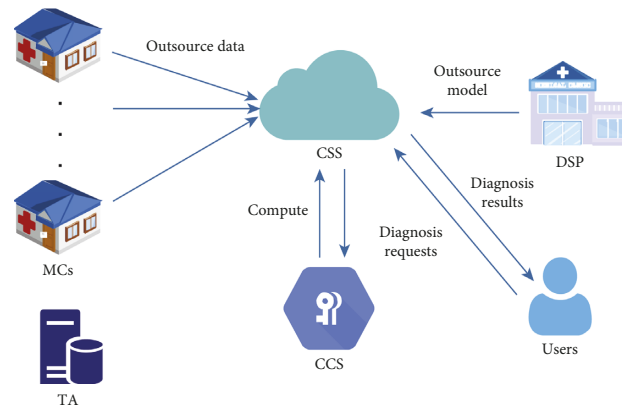


Figure 2: System model.

(iii) Cloud storage server (CSS): CSS has the ability to store and manage outsourced data. CSS can perform privacy preserving computation with its powerful computation power.

(iv) Cloud computation server (CCS): CCS assists CSS to complete privacy preserving computation.

(v) Diagnosis service provider (DSP): DSP wants to train a machine learning model on the outsourced data from MCs and provides online aided disease diagnosis for users. Due to the limited computation and communication ability, DSP will outsource the training and diagnosis to CSS.

(vi) Users: Users are patients or doctors who have unlabeled samples and want to get the diagnosis results. The users will send encrypted diagnosis requests to CSS and obtain the encrypted results. The users can decrypt the results with own private key.

4.2. Security Goals. In order to meet the security requirements of outsourced training and diagnosis, our scheme will achieve the following security goals.

(i) Medical data privacy: The outsourced data of MCs will not be leaked to other participants in the whole machine learning process.

(ii) Model privacy: Other participants cannot learn any useful information about the model of DSP.

(iii) Users privacy: The diagnosis requests and results of users will not be acquired by other participants.

(iv) Intermediate results privacy: In the execution of protocols, any participant will not infer other participants' sensitive information through the intermediate results.

In our scheme, the training and diagnosis processes are completed by CSS and CCS. All participants are semi-honest (or honest-but-curious). Specifically, they will honestly implement the secure computation protocols, but they will try to analyze the sensitive data and intermediate results to infer the useful information of other participants. Like the

previous works, we assume that CSS and CCS will not collude. Because CSS and CCS belong to different commercial companies, they will not collude with each other for their own reputation.

4.3. Threat Model. In this paper, we will define three attacks in our system model.

(i) Eavesdropping attack: This attack means that an adversary can eavesdrop and analyze data during the data transmission. The data transmission includes outsourcing process and the interaction between participants in protocol implementation.

(ii) Honest-but-curious attack: All participants will implement the protocol honestly, but they will infer the useful information during the execution of protocols.

(iii) Client-collusion attack: In the training and diagnosis process, some clients may collude to analyze the useful information of other participants.

5. Proposed Scheme

In this section, we describe the proposed scheme in detail. Our scheme mainly includes system initialization, privacy preserving machine learning training, and online disease diagnosis.

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 and distributes the parameters for MCs, CSS, CCS, and DSP, respectively. TA sends the parameters through the secure communication channel. Then, TA will stay offline. We assume that there are m MCs in our system. Because the Paillier cryptosystem and BFV cryptosystem can only encrypt integers, the floating point numbers and negative numbers should be converted into integers. Therefore, all participants should make data conversion before encrypting their sensitive information.

5.1.1. Generate System Parameters

(1) Generate a public-private key pair $(PK_P = (N_P, g), SK_P)$ of the Paillier cryptosystem and a public-private key pair (PK_B, SK_B) of the BFV cryptosystem. The BFV plaintext space is N_B. The public keys are public and the private keys are sent to the CCS.

(2) Generate a public-private key pair (PK_P^C, SK_P^C) of the Paillier cryptosystem and a public-private key pair (PK_B^C, SK_B^C) of the BFV cryptosystem for CSS. The BFV plaintext space is N_B^C. The public keys are public and the private keys are sent to CSS.

(3) Generate a public-private key pair (PK_P^D, SK_P^D) of the Paillier cryptosystem for DSP. The public key is public and the private key is sent to DSP.

Table 1: Notation and definition.

Notation	Definition
$l(x)$	The key length of x
(PK_P, SK_P)	Paillier public-private key pair of CCS
(PK_B, SK_B)	BFV public-private key pair of CCS
(PK_P^C, SK_P^C)	Paillier public-private key pair of CSS
(PK_B^C, SK_B^C)	BFV public-private key pair of CSS
(PK_P^D, SK_P^D)	Paillier public-private key pair of DSP
id_i	The authentication of MC_i
E	The precision of floating point numbers
a_i	The private key of MC_i
$[x]_{PK}$	The ciphertext of x under PK
L	Classification numbers
d	The degree of polynomial

(4) Generate a random integer $\omega \in N_P$. TA randomly splits ω to m integers, satisfying $\omega_1 + \omega_2 + \cdots + \omega_m = \omega$ and sends ω_i to MC_i. Then, generate two lists H and H'. Each list has m random integers, $H = (n_1, n_2, \ldots, n_m)$, $n_i \in N_P$, $H' = (n_1', n_2', \ldots, n_m')$, $n_i' \in N_P$. Each element in H and H' represents the ID of each MC. When MC_i sends authentication id_i to CSS, MC_i will hide g^{a_i} and ω_i with n_i and n_i', respectively. The (n_i, n_i') is sent to MC_i. H and H' are sent to CSS.

5.1.2. Data Conversion. In the machine learning application scenario, data and model parameters contain floating point numbers and negative numbers.

For a floating point number x, we enlarge x to $x \cdot 2^E$ (E is the precision of floating point numbers). For example, given a floating point number $x = 3.61$ and the precision $E = 20$, we can convert x into an integer $x' = 3785359$. For a negative number y, we divide the plaintext space N (N is expressed the plaintext space of the Paillier or BFV cryptosystem) into two parts because all variables and intermediate results in the process of training and prediction are much smaller than $N/2$. An integer in $[0, N/2)$ represents a positive integer and $(N/2, N-1]$ represents a negative integer. When encrypting the negative integer y, it is converted to encrypt $N - y$. If y is both a floating point number and a negative number, y is first converted into a negative integer.

5.2. Privacy Preserving Machine Learning Training. The privacy preserving machine learning training process is completed by CSS and CCS. We assume that the amount of outsourced data is n.

5.2.1. Local Data Outsourcing. To protect the privacy of MCs' local data, MCs will encrypt the data before outsourcing. The outsourcing process of MC_i $(i = 1, \ldots, m)$ is as follows.

(1) MC_i generates a random integer $a_i \in Z_{N_P}$. Computing $pk_i = g^{a_i} \bmod N_P^2$ as public key and the private key is $sk_i = a_i$.

(2) Computing $h_i = g^{a_i+\omega} \bmod N_P^2$.

(3) For each plaintext data, such as x, MC_i will make a data conversion as mentioned in Section 5.1. Then, compute $[x]_{MC_i} = g^x r^{N_P} + h_i$ to encrypt and outsource the encrypted data to CSS for storage.

5.2.2. Secure Basic Building Blocks for Training. To complete the privacy preserving outsourced training, we construct some algorithms as basic building blocks based on the Paillier cryptosystem: secure data aggregation (Block_1), secure multiplication algorithm (Block_2), secure inner product algorithm (Block_3), secure scalar multiplication of vector algorithm (Block_4), and secure symbol judgment algorithm (Block_5). The algorithms will be executed with CSS and CCS.

(1) Secure data aggregation algorithm (Block_1). CSS needs to aggregate MCs' outsourced data before starting machine learning training. The algorithm works as follows and is described in Algorithm 1.

(1) CSS sends a training request to MC_i, $i = 1, 2, \ldots, m$.

(2) After receiving the training request, MC_i computes $id_i = (pk_i + n_i, \omega_i + n_i')$ as authentication (The id_i indicates that CSS is allowed to use the outsourced data of MC_i for training) and sends to CSS.

(3) CSS obtains the pk_i, ω_i of MC_i through the id_i and computes $\omega = \omega_1 + \omega_2 + \cdots + \omega_m$. It should be noted that ω can be obtained only after all MCs have sent their authentication. Then, CSS computes $h_i = g^{a_i+\omega} \bmod N_P^2$ and completes the aggregation.

(2) Secure multiplication algorithm (Block_2). Given two encrypted integers $[x]_{PK_P}$ and $[y]_{PK_P}$, the algorithm needs to compute $[x \cdot y]_{PK_P}$. The algorithm works as follows and is described in Algorithm 2.

(1) CSS generates two random integers R_1, R_2 and $R_1, R_2 \in Z_{N_P}$. Then, it computes by applying the additive homomorphism, obtaining the following results.

$$\begin{aligned} [x + R_1]_{PK_P} &= [x]_{PK_P} \cdot g^{R_1}, \\ [y + R_2]_{PK_P} &= [y]_{PK_P} \cdot g^{R_2}, \end{aligned} \tag{2}$$

Then, sending them to CCS.

(2) CCS generates a random integer $T, T \in Z_{N_P}$. It decrypts $[y + R_2]_{PK_P}$ by using SK_P. Then, it encrypts $(y + R_2 + T) \bmod N_P^C$ with PK_P^C to get $[y + R_2 + T]_{PK_P^C}$. Computing $[xT + R_1T]_{PK_P} = [x + R_1]_{PK_P}^T$ and encrypting T with PK_P. Sending $[y + R_2 + T]_{PK_P^C}$, $[xT + R_1T]_{PK_P}$ and $[T]_{PK_P}$ to CSS.

(3) CSS decrypts $[y + R_2 + T]_{PK_P^C}$ with SK_P^C and computes $y + T$. Then, computing by applying the additive homomorphism, obtaining the following results.

$$\begin{aligned} [xy + xT]_{PK_P} &= [x]_{PK_P}^{y+T}, \\ [xy - R_1T]_{PK_P} &= [xy + xT]_{PK_P} \cdot [xT + R_1T]_{PK_P}^{-1}, \\ [R_1T]_{PK_P} &= [T]_{PK_P}^{R_1}. \end{aligned} \tag{3}$$

Computing the result,

$$[xy]_{PK_P} = [xy - R_1T]_{PK_P} \cdot [R_1T]_{PK_P}. \tag{4}$$

(3) Secure inner product algorithm (Block_3). Given two encrypted vectors $[X]_{PK_P}$, $[Y]_{PK_P}$. The algorithm will compute $[X \cdot Y]_{PK_P}$ and is described in Algorithm 3.

(4) Secure scalar multiplication of vector algorithm (Block_4). Given a encrypted vector $[X]_{PK_P}$ and a encrypted integer $[y]_{PK_P}$, the algorithm will compute $[y \cdot X]_{PK_P}$ and is described in Algorithm 4.

(5) Secure symbol judgment algorithm (Block_5). Given an encrypted integer $[x]_{PK_P}$, the algorithm will compute the sign of $[x]_{PK_P}$. Let judge = 1 if $x \geq 0$ else judge = 0. The algorithm works as follows and is described in Algorithm 5.

(1) CSS chooses a random integer r, $l(r) < l(N_P)/2$. Then, it computes $[x \cdot r]_{PK_P} = [x]_{PK_P}^r$ by applying the additive homomorphism and sends $[x \cdot r]_{PK_P}$ to CCS.

(2) CCS decrypts $[x \cdot r]_{PK_P}$. Let judge = 1 if $x \cdot r \geq 0$ else judge = 0. Then, it sends $[\text{judge}]_{PK_P^C}$ to CSS.

(3) CSS decrypts and obtains the symbol judge.

5.2.3. Privacy Preserving Outsourced Training with Multiclass SVM. In this section, we construct a privacy preserving outsourced training protocol to train a multiclass SVM model using the proposed building blocks. DSP outsources the training task to CSS and CSS completes the aggregation of outsourced data. Then, CSS and CCS complete the model training. After finishing the training, CSS transforms $([W_1]_{PK_P}, \ldots [W_L]_{PK_P})$ into $([W_1]_{PK_P^D}, \ldots [W_L]_{PK_P^D})$. To achieve the transformation, we use the algorithm proposed in reference [38].

For multiclass SVM training, there are two methods: one to rest (ovr) and one to one (ovo). In order to improve the efficiency and reduce the number of iterations, we choose the ovr method for training. We need to construct L binary SVM classifiers, each of which corresponds to one classification. The process is described in Algorithm 6.

5.3. Privacy Preserving Online-Aided Disease Diagnosis. In this section, our proposed scheme consists of four steps: diagnosis outsourcing, secret diagnosis request generation, diagnosis values computation, and diagnosis result generation. The privacy preserving online-aided disease diagnosis is completed by CSS and CCS.

5.3.1. Diagnosis Outsourcing. To reduce the computation and communication overhead, DSP outsources the SVM

Input: the authentication and outsourced data of MC_i.
Output: the training data.
CSS:
(1) Send a training request to MC_i, $i = 1, 2, \ldots, m$.
MCs:
(2) for $i = 1 \longrightarrow m$:
MC_i sends $id_i = (pk_i + n_i, \omega_i + n_i')$ to CSS
end for CCS:
(3) CSS obtains (pk_i, ω_i), $i = 1, 2, \ldots, m$
(4) Compute $\omega = \omega_1 + \omega_2 + \cdots + \omega_m$
(5) for $i = 1 \longrightarrow m$:
Compute $h_i = g^{a_i+\omega} \bmod N_P^2$
For each outsourced data of MC_i, such as $[x]_{MC_i}$,
Compute $[x]_{PK_P} = [x]_{MC_i} - h_i$ to complete aggregate
end for

Algorithm 1: Secure data aggregation (Block_1).

Input: $[x]_{PK_P}, [y]_{PK_P}$
Output: $[xy]_{PK_P}$
CSS:
(1) $R_1, \quad R_2 \in Z_{N_P}$
(2) $[x + R_1]_{PK_P} = [x]_{PK_1} \cdot g^{R_1}$
$[y + R_2]_{PK_P} = [y]_{PK_P} \cdot g^{R_2}$
(3) Send $[x + R_1]_{PK_P}, [y + R_2]_{PK_P}$ to CCS.
CCS:
(4) $T \in Z_{N_P}$
(5) Decrypt $[y + R_2]_{PK_P}$
(6) $[xT + R_1T]_{PK_P} = [x + R_1]_{PK_P}^T$
(7) Encrypt $(y + R_2 + T) \bmod N_P^C$ with PK_P^C
(8) Encrypt T with PK_P
(9) Send $[xT + R_1T]_{PK_P}, [y + R_2 + T]_{PK_P^C}$ and $[T]_{PK_P}$ to CSS.
CSS:
(10) Decrypt $[y + R_2 + T]_{PK_P^C}$ with SK_P^C and Compute $y + T$
(11) $[xy + xT]_{PK_P} = [x]_{PK_P}^{y+T}$
(12) $[xy - R_1T]_{PK_P} = [xy + xT]_{PK_P} \cdot [xT + R_1T]_{PK_P}^{-1}$
(13) $[R_1T]_{PK_P} = [T]_{PK_P}^{R_1}$
(14) $[xy]_{PK_P} = [xy - R_1T]_{PK_P} \cdot [R_1T]_{PK_P}$

Algorithm 2: Secure multiplication (Block_2).

Input: $[X]_{PK_P}, [Y]_{PK_P}$
Output: $[X \cdot Y]_{PK_P}$
CSS:
(1) Define $[X \cdot Y]_{PK_P} = [1]_{PK_P}$.
(2) for $i = 1 \longrightarrow X$.length:
$[X \cdot Y]_{PK_P} \cdot = \text{Block\_1}([x_i]_{PK_P}, [y_i]_{PK_P})$
end for

Algorithm 3: Secure inner product (Block_3).

model parameters to CSS and authorizes CSS to provide diagnosis service for users.

The SVM parameters of DSP are expressed as $(W^1, W^2, \ldots, W^L)$ (There are L classifiers),

$$W^i = \left(w_1^i, w_2^i, \ldots, w_{t+1}^i\right), \quad (i = 1, 2, \ldots, L). \tag{5}$$

Input: $[X]_{PK_P}$, $[y]_{PK_P}$
Output: $[y \cdot X]_{PK_P}$
CSS:
(1) Define $[y \cdot X]_{PK_P} = [1, 1, \ldots, 1]_{PK_P}$.
(2) for $i = 1 \longrightarrow X$.length:
$[yx_i]_{PK_P} = \text{Block\_1}([y]_{PK_P}, [x_i]_{PK_P})$
end for

Algorithm 4: Secure scalar multiplication of vector (Block_4).

Input: $[x]_{PK_P}$
Output: judge
CSS:
(1) Choose a random integer r, $l(r) < l(N_P)/2$
(2) $[x \cdot r]_{PK_P} = [x]^r_{PK_P}$
(3) Send $[x \cdot r]_{PK_P}$ to CCS
CCS:
(4) Decrypt $[x \cdot r]_{PK_P}$
(5) if $x \cdot r \geq 0$: judge = 1, else: judge = 0
(6) Encrypt judge with PK_P^C
(7) Send $[\text{judge}]_{PK_P^C}$ to CSS

Algorithm 5: Secure symbol judgment (Block_5).

Input: outsourced data of MCs
$([X_1]_{PK_P}, [y_1]_{PK_P}), \ldots, ([X_n]_{PK_P}, [y_n]_{PK_P})$,
iterations T, learning rate learnrate,
regularization parameter z
Output: L encrypted binary SVM classifiers parameters
$([W_1]_{PK_P}, \ldots [W_L]_{PK_P})$
(1) for $k = 1 \longrightarrow L$:
for it $= 1 \longrightarrow T$:
$[\text{grad}]_{PK_P} = [W_k]_{PK_P}$
for $i = 1 \longrightarrow n$:
(2) $[W_k \cdot X_i]_{PK_P} = \text{Block\_3}([W_k]_{PK_P}, [X_i]_{PK_P})$
(3) $\text{tmp} = \text{Block\_2}([y_i]_{PK_P}, [W_k \cdot X_i]_{PK_P}) \cdot [1]^{-1}_{PK_P}$
(4) if Block_5(tmp) == 0:
$[y_i \cdot X_i]_{PK_P} = \text{Block\_4}(y_i, X_i)$
$[\text{grad}]_{PK_P} \cdot = [W_j] \cdot [y_i \cdot X_i]^{z \cdot (N_P - 1)}_{PK_P}$
end for
(5) $[W]_{PK_P} \cdot = [\text{grad}]^{\text{learnrate} \cdot (N_P - 1)}_{PK_P}$
end for
end for
(6) return $([W_1]_{PK_P}, \ldots, [W_L]_{PK_P})$

Algorithm 6: Secure multiclass SVM training.

For W^i and the corresponding class result class^i, DSP generates a $t+1$-dimensional random integer vector $R^i = (R^i_1, R^i_2, \ldots, R^i_{t+1})$ and a random integer r^i, $l(R^i_j) = l(r^i) < l(N^C_B)/2$, $l(R^i_j) < l(N_B)/2$. Then, DSP computes $W^i + R^i$ and class$^i + r^i$ to hide the parameters class results.

According to the combination of subtraction of L random integer vectors, DSP constructs a combination table. The combination table has C^2_L values, as shown in Table 2. The values in combination table are used to eliminate the blinding factors in subsequent computation.

Table 2: Combination table.

$(1,2)$: $index = 1 \cdot L + 2$	...	$(L-1, L)$: $index = (L-1) \cdot L + L$
$R^1 - R^2$	...	$R^{L-1} - R^L$

DSP encrypts $W^i + R^i$ with PK_B, $\text{class}^i + r^i$ with PK_B^C, r^i with PK_P and all values of combination table with PK_B^C. Then, DSP sends them as the outsourced parameters to CSS. After receiving the outsourced parameters, CSS decrypts $[\text{class}^i + r^i]_{PK_B^C}$ and the combination table with SK_B^C. CSS computes as follows:

$$\left[\text{class}^i\right]_{PK_P} = \left[\text{class}^i + r^i_{PK_P}\right] \cdot \left[r^i\right]^{-1}_{PK_P}, \quad i = 1, 2, \ldots, L. \tag{6}$$

5.3.2. Secret Diagnosis Request Generation. For user_i, the symptom is expressed as $X^i = (x_1^i, x_2^i, \ldots, x_t^i, 1)$ (The last 1 is added to facilitate the computation of vector inner product). The user_i generates a $t+1$-dimensional random integer vector $T^i = (T_1^i, T_2^i, \ldots, T_j^i, \ldots, T_{t+1}^i)$ and $l(T_j^i) < l(N_B^C)/2$, $l(T_j^i) < l(N_B)/2$. Then, user_i hides plaintext symptoms $X^i + T^i$.

The user_i encrypts symptom $X^i + T^i$ with PK_B and encrypts T^i with PK_B^C. Let S as the secret prediction request of user_i.

$$S = \left(\left[X^i + T^i\right]_{PK_B}, \left[T^i\right]_{PK_B^C}\right). \tag{7}$$

Then, the user_i sends S to CSS.

5.3.3. Diagnosis Value Computation. In our proposed diagnosis scheme, it is a multiclassification problem, so it is necessary to compute the diagnosis value of each classification. After receiving the secret prediction request S, CSS decrypts $[T^i]_{PK_B^C}$ with SK_B^C. Then, it computes $[X^i + T^i]_{PK_B} - T^i$ by the homomorphic operation of the BFV cryptosystem.

According to the decision function $f(X) = W \cdot X + b$ of the SVM algorithm, a diagnosis value needs to be computed by one multiplication homomorphic operation and one addition homomorphic operation. Because the BFV encryption algorithm supports ciphertext packaging, batch operation can be realized and the computation efficiency is significantly improved. The process is described in Algorithm 7.

5.3.4. Diagnosis Result Generation. After computing the diagnosis values, CSS obtains L encrypted diagnosis values and each value corresponds to a class result. Then, CSS needs to select the classification corresponding to the maximum value from the L encrypted values as the diagnosis result.

Therefore, we design a secure maximum find protocol and a secure comparison algorithm. In this process, CSS and CCS jointly execute the protocol.

(1) Secure maximum finding. CSS sets an initial maximum position pos = 1. Then, CSS executes L cycles and each cycle executes a secure comparison algorithm to continuously update the pos value.

After L cycles, CSS obtains the final diagnosis result $[\text{class}^{\text{pos}}]_{PK_P}$ and converts $[\text{class}^{\text{pos}}]_{PK_P}$ into $[\text{class}^{\text{pos}}]_{PK_{\text{user}_i}}$ under the public key PK_{user_i} of user_i. To achieve the transformation, we use the algorithm proposed in literature [38]. Then, CSS sends $[\text{class}^{\text{pos}}]_{PK_{\text{user}_i}}$ to user_i. The user_i decrypts the encrypted result with SK_{user_i}. The process is described in Algorithm 8.

(2) Secure comparison (SC). For the i-th cycle, CSS computes $[\Delta_{pos-i}] = [(W^{\text{pos}} + R^{\text{pos}})X^i]_{PK_B} - [(W^j + R^j)X^i]_{PK_B}$. Then, according to pos and j, computing index = $\text{pos} \cdot L + j$. The index corresponds to the value $(R^{\text{pos}} - R^j)_{\text{index}}$ in the combination table and computing as follows:

$$\begin{aligned} \left[X^i\left(R^{\text{pos}} - R^j\right)_{\text{index}}\right]_{PK_B} &= \left[X^i\right]_{PK_B} \cdot \left(R^{\text{pos}} - R^j\right)_{\text{index}}, \\ \left[\Delta_{\text{pos}-j}'\right]_{PK_B} &= \left[\Delta_{\text{pos}-j}\right]_{PK_B} - \left[X^i\left(R^{\text{pos}} - R^j\right)_{\text{index}}\right]_{PK_B}. \end{aligned} \tag{8}$$

At this time, $[\Delta_{pos-j}']_{PK_B}$ has eliminated $(R^{\text{pos}} - R^j) \cdot X^i$ in $[\Delta_{\text{pos}-j}]_{PK_B}$.

CSS chooses $t+1$ equal random integers r', $R' = (r', \ldots, r')$ and $l(r') < l(N_B)/2$. Computing $[\Delta_{\text{pos}-j}']_{PK_B} = [\Delta_{\text{pos}-j}']_{PK_B} \cdot R'$. Then, CSS chooses $t+1$ different random integers, $R'' = (r_1'', \ldots, r_{t+1}'')$, $l(r_1'') = \cdots = l(r_{t+1}'') < l(N_B)/2$ and computing $[\Delta_{pos-j}']_{PK_B} = [\Delta_{pos-j}']_{PK_B} + R''$. Summing all elements in R'' to get R_{css} and encrypting it with PK_P. CSS sends $[\Delta_{\text{pos}-j}']_{PK_B}$, $[R_{\text{css}}]_{PK_P}$ to CCS.

CCS decrypts $[\Delta_{\text{pos}-j}']_{PK_B}$ with SK_B and $[R_{\text{css}}]_{PK_P}$ with SK_P, $((w_1^{pos}x_1^i - w_1^j x_1^i)r' + r_1'', \ldots, (w_{t+1}^{pos}\ x_{t+1}^i - w_{t+1}^j x_{t+1}^i)r' + r_{t+1}'')$. Then, summing each dimension, $s = \text{sum}((w_1^{\text{pos}}x_1^i - w_1^j x_1^i)r' + r_1'', \ldots, (w_{t+1}^{\text{pos}}x_{t+1}^i - w_{t+1}^j x_{t+1}^i)r' + r_{t+1}'')$.

CCS removes R_{css} from s by computing $(s - R_{\text{css}}) \bmod N_B$. Let judge = 1 if $s > N_B/2$, else judge = 0.

CCS encrypts judge with PK_B^C and sends it to CSS. CSS decrypts it and if judge = 1, updates the value of pos.

The process is described in Algorithm 9.

6. Security Analysis

In this section, we analyze the security of the proposed scheme. The focus is on the outsourced data of MCs, the SVM model parameters of DSP, the symptoms, and diagnosis results of users.

6.1. Security Analysis of Training. In the training phase, the outsourced data of MCs and the SVM model parameters of DSP need privacy preserving. The training protocol is composed of building blocks designed in Section 5.2.2, which are completed by CSS and CCS. According to the

Input: $[X^i]_{PK_B}$, $[W^j + R^j]_{PK_B}$, $j = 1, 2, \ldots, L$
Output: L encrypted diagnosis values
$[(W^1 + R^1)X^i]_{PK_B}, \ldots, [(W^L + R^L)X^i]_{PK_B}$
CSS:
(1) for $j = 1 \longrightarrow L$:
$[(W^j + R^j)X^i]_{PK_B} = [W^j + R^j]_{PK_B} \cdot [X^i]_{PK_B}$
end for

Algorithm 7: Diagnosis value computation.

Input: L diagnosis values and corresponding class results
$[(W^j + R^j)X^i]_{PK_B}, \text{class}^j, \quad j = 1, 2, \ldots, L;$
initial pos = 1
Output: $[\text{class}^{\text{pos}}]_{PK_{\text{user}_i}}$
CSS:
for $j = 2 \longrightarrow L$:
(1) judge = $SC([(W^{\text{pos}} + R^{\text{pos}})X^i]_{PK_B}, [(W^j + R^j)X^i]_{PK_B})$
(2) if judge = 1:
pos = i
end for
(3) Transform $[\text{class}^{\text{pos}}]_{PK_p}$ into $[\text{class}^{\text{pos}}]_{PK_{\text{user}_i}}$ with CCS
(4) Send $[\text{class}^{\text{pos}}]_{PK_{\text{user}_i}}$ to user_i.
user_i:
(5) user_i decrypts $[\text{class}^{\text{pos}}]_{PK_{\text{user}_i}}$ with SK_{user_i}.

Algorithm 8: Secure maximum finding.

Input: $[(W^{\text{pos}} + R^{\text{pos}})X^i]_{PK_B}$, $[(W^j + R^j)X^i]_{PK_B}$
Output: judge
CSS:
(1) $[\Delta_{\text{pos}-j}]_{PK_B} =$
$[(W^{\text{pos}} + R^{\text{pos}})X^i]_{PK_B} - [(W^j + R^j)X^i]_{PK_B}$
(2) index = pos $\cdot L + j$
(3) $[X^i(R^{\text{pos}} - R^j)_{\text{index}}]_{PK_B} = [X^i]_{PK_B} \cdot (R^{\text{pos}} - R^j)_{\text{index}}$
(4) $[\Delta_{\text{pos}-j}']_{PK_B} = [\Delta_{\text{pos}-j}]_{PK_B} - [X^i(R^{\text{pos}} - R^j)_{\text{index}}]_{PK_B}$
(5) Generate $R\prime$.
(6) $[\Delta_{\text{pos}-j}']_{PK_B} = [\Delta_{\text{pos}-j}']_{PK_B} \cdot R\prime$
(7) Generate $R\prime\prime = (r_1'', r_2'', \ldots, r_{t+1}'')$.
(8) $[\Delta_{\text{pos}-j}']_{PK_B} = [\Delta_{\text{pos}-j}']_{PK_B} + R\prime\prime$
(9) $R_{\text{css}} = r_1' + r_2' + \cdots + r_{t+1}'$
(10) Send $[\Delta_{\text{pos}-i}']_{PK_B}, [R_{\text{css}}]_{PK_p}$ to CCS.
CCS:
(11) Decrypt $[\Delta_{\text{pos}-i}']_{PK_B}, [R_{\text{css}}]_{PK_p}$.
(12) $s = \text{sum}((w_1^{\text{pos}} x_1^i - w_1^j x_1^i)r\prime + r_1'', \ldots,$
$(w_{t+1}^{\text{pos}} x_{t+1}^i - w_{t+1}^j x_{t+1}^i)r\prime + r_{t+1}'')$
(13) $s = (s - R_{\text{css}}) \bmod N_B$
(14) if $s > N_B/2$: judge = 1
else: judge = 0
(15) Encrypt judge. Send it to CSS
CSS:
(16) Decrypt $[\text{judge}]_{PK_B^C}$

Algorithm 9: Secure comparison (SC).

threat models proposed in Section 4.3, we analyze the security of the training protocol.

6.1.1. Eavesdropping Attack. The data transmission process in the training phase includes that MCs outsources the encrypted data to CSS and the interactions of training protocol between CSS and CCS.

In the outsourcing process, the data of MC_i have been encrypted. MC_i combines the system public key PK_P, parameter ω, and its own public key g^{a_i} to ensure that the data are hidden while encrypting. Suppose an adversary obtains the private key SK_P and eavesdrops when MCs outsource their data to CSS. Because the data of MC_i have been encrypted, such as $[x]_{MC_i} = g^x r^{N_P} + h_i$, the adversary cannot obtain any useful information. Similarly, the authentications are also hidden by random numbers. In the training protocol execution process, CSS and CCS will interact and the transformed data have been encrypted and hidden the real values with random numbers. The adversary also cannot obtain any useful information.

6.1.2. Honest-But-Curious Attack. During the training phase, CSS and CCS will get some intermediate results from the proposed building blocks in Section 5.2.2.

In the Block_2, CSS hides x, y with R_1, R_2 by homomorphic operation before sending them to CCS. Then, CCS sends $[xT + R_1T]_{PK_P}$, $[y + R_2 + T]_{PK_P^C}$ and $[T]_{PK_P}$ to CSS after computing. Therefore, both CSS and CCS cannot learn any useful information about x, y. Because the Block_3 and Block_4 are designed based on the Block_2, we will not analyze them. In the Block_5, CSS hides x with r and sending $[x \cdot r]_{PK_P}$ to CCS. CCS can only know the symbol of x, but cannot obtains the real value of x. CCS only returns the result judge (0 or 1) to CSS. Through the abovementioned analysis, CSS and CCS cannot learn any useful information in the training process.

6.1.3. Client-Collusion Attack. For MCs, each MC_i only know its own ω_i. Therefore, if $(m-1)$ MCs collude with each other to steal the privacy of another MC, they cannot learn any useful information.

6.2. Security Analysis of Disease Diagnosis. In the diagnosis phase, the SVM parameters of DSP, the symptom $\mathbf{X}^i$ and the diagnosis result $\text{class}^{\text{pos}}$ of user_i need privacy preserving. The diagnosis process consists of diagnosis outsourcing, secret diagnosis request generation, diagnosis value computation, and diagnosis result generation. Therefore, we conduct security analysis on the main steps by the threat model.

6.2.1. Eavesdropping Attack. The data transmission process includes that DSP outsources $[W^i + R^i]_{PK_B}$, $[\text{class}^i + r^i]_{PK_B^C}$, $[r^i]_{PK_P}$, $i = 1, 2, \ldots, L$ and $[R^1 - R^2]_{PK_B^C}, \ldots, [R^{L-1} - R^L]_{PK_B^C}$ to CSS, user_i sends request S to CSS and the interaction of diagnosis process between CSS and CCS.

Through the encrypted data of outsourcing process, it can be seen that the adversary(CCS) can only decrypt $[W^i + R^i]_{PK_B}$ and $[r^i]_{PK_P}$ with SK_B and SK_P. However, the adversary cannot learn W^i because of the R^i and the r^i do not contain any useful information. When user_i sends S to CSS, the symptom X^i may be eavesdropped and decrypted by the adversary, but X^i is hidden by random numbers. In the interaction of SC algorithm between CSS and CCS, all transmitted data are hidden by random numbers and ciphertext state, so the adversary cannot learn any useful information.

6.2.2. Honest-But-Curious Attack. In the diagnosis value computation process, CSS can only obtain the L encrypted diagnosis values under PK_B and does not know the corresponding classification meaning. The whole process is executed in the ciphertext state, so CSS cannot learn any useful information. The process of diagnosis result generation consists of secure maximum finding protocol and secure comparison algorithm. When CSS and CCS execute the secure comparison algorithm, CSS computes the difference between the two encrypted vectors to be compared. The obtained difference vector can confuse the positive and negative of the two numbers on each dimension of the original two vectors. At the same time, random integers are used to hide the difference vector. After decrypting the difference vector, CCS can eliminate the random number only after summing. During this process, CSS and CCS cannot obtain any useful information.

After CSS and CCS execute secure maximum finding protocol, CSS obtains the diagnosis result $[\text{class}^{\text{pos}}]_{PK_P}$. When performing key conversion on $[\text{class}^{\text{pos}}]_{PK_P}$, CSP hides $\text{class}^{\text{pos}}$ with a random integer R. Then, sending $[\text{class}^{\text{pos}} + R]_{PK_P}$ to CCS. CCS can decrypt it. However, because there is a random integer hidden, CCS cannot obtain $\text{class}^{\text{pos}}$.

6.2.3. Client-Collusion Attack. For all users, they can only get the diagnosis results and cannot get any other information. Therefore, our proposed scheme can resist the client-collusion attack.

7. Performance Evaluation

In this section, we implemented our scheme and evaluated the performance of training and diagnosis.

Our experimental environment is shown in Table 3.

In our experiments, we evaluated our proposed scheme with a real dataset from UCI machine learning library called dermatology. The dermatology dataset is a multiclassification dataset with 6 categories and 34 symptoms.

7.1. Privacy Preserving Machine Learning Training Evaluation

7.1.1. Effect of Key Length on Computation Overhead. The key length in cryptosystem has a great impact on efficiency and security. Therefore, we tested the data encryption time and main building blocks time (Block_1 and Block_3), which have high computation overhead. The test results are shown in Table 4.

Table 3: Experimental environment.

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

Table 4: Computation overhead under different key length.

Key length (bit)	Data encryption (s)	Block_1 (s)	Block_3 (s)
$l = 256$	$4.15e\,10 - 4$	$2.08\,e10 - 5$	0.094
$l = 512$	$2.52e\,10 - 3$	$5.0e\,10 - 5$	0.417
$l = 1024$	0.0153	$2.1e\,10 - 4$	2.52
$l = 2048$	0.103	$9.9e\,10 - 4$	17.2

From Table 4, it can be seen that the increase of key length has a great impact on the computation overhead. Based on the experimental results and security considerations, the key length of the Paillier cryptosystem is set to 1024 bit in the training phase.

7.1.2. Privacy Preserving Multiclass SVM Training Analysis. In order to meet the requirements of data encryption, we convert all floating-point numbers to integers. The conversion accuracy E of floating-point numbers has a great impact on the accuracy of the SVM model. We tested the accuracy of the SVM model under different E values; the results are shown in Figure 3.

Through the abovementioned experimental analysis, it can be seen that the larger the E, the higher the accuracy of the model. With the increase of E, the accuracy of the model tends to be stable. When $E = 20$, the accuracy of the model is the highest. At the same time, we also used the gradient descent method to train the SVM model in the plaintext state. We compared the accuracy with the model trained in ciphertext state and the results are shown in Table 5.

Through the abovementioned experimental analysis, it can be seen that the accuracy of our proposed scheme is the same as the plaintext state (98.61%). Therefore, it is verified that our proposed scheme is correct and available.

7.2. Privacy Preserving Online-Aided Disease Diagnosis Evaluation. We implemented our proposed scheme by using SEAL library in the diagnosis phase.

7.2.1. Noise Effect of BFV Cryptosystem. When using the BFV cryptosystem for homomorphic operation, the influence of noise needs to be considered. The noise of ciphertext will be increased when the multiplication homomorphic operation is carried out. If the noise is too large after computation, the correct result cannot be obtained after decryption.

Therefore, the BFV cryptosystem in SEAL will set the noise budget during initialization. If the noise budget is greater than 0 after the computation, it can be decrypted correctly. The value of noise budget is related to the setting of parameters. We evaluated the influence of poly module degree (d) on the encryption time, the change of noise budget after homomorphic operation, the computation time and whether the decryption result is correct. The results are shown in Table 6. It can be seen that the noise consumption of the BFV cryptosystem is relatively large when performing multiplication homomorphism, so the BFV cryptosystem can only perform multiplication homomorphism for a limited number of times. When computing the diagnosis values, only one inner product operation and one addition operation are required. Therefore, it is completely feasible to use the BFV cryptosystem.

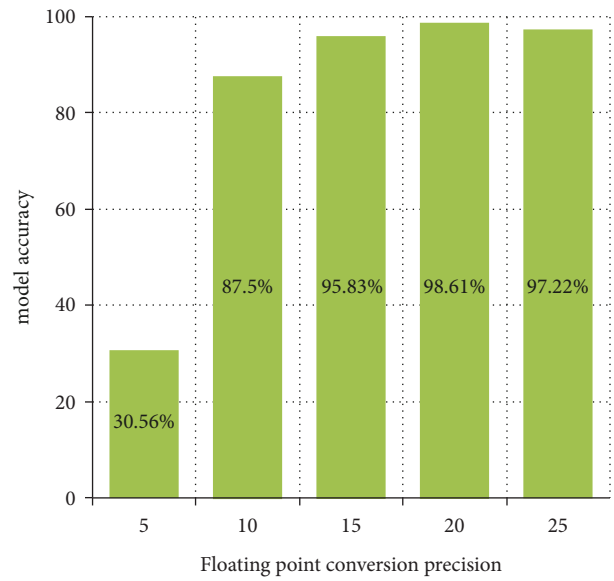


Figure 3: The influence of precision.

We comprehensively consider the encryption time and computation time and ensure that the computation results can be decrypted correctly. The parameter we set is poly module degree (d) = 8192.

7.2.2. Influence of Different Classification Numbers on Computation Overhead. When using the BFV cryptosystem to encrypt data, multiple plaintext data can be packaged and encrypted into a ciphertext. The number of classifications is L.

Table 5: Comparison analysis of model accuracy.

Dataset	Plaintext state	Our proposed scheme
Dermatology	98.61%	98.61%

Table 6: The influence of poly modulus degree (d) on noise budget.

d	Encryption time (s)	Initial noise budget (bit)	Noise budget (bit) after operation	Computation time (s)	Decryption result (correct or wrong)
2048	0.013	2	0	0.012	×
4096	0.023	9	0	0.031	×
8192	0.69	110	64	0.123	✓
32768	1.178	761	713	2.268	✓

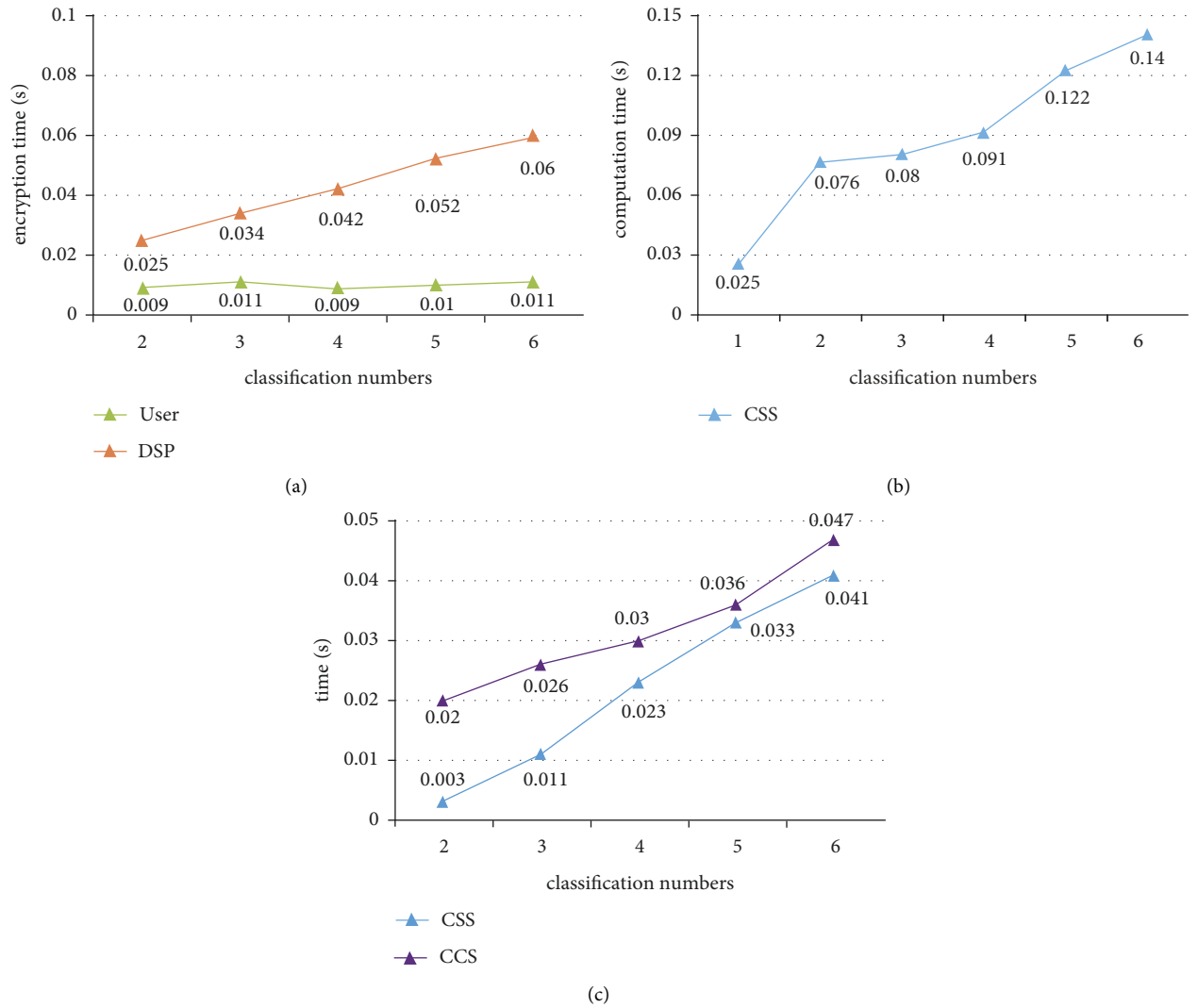


Figure 4: Influence of different classification numbers on computation overhead. (a) Data encryption, (b) diagnosis values computation, and (c) diagnosis result generation.

We tested the impact of different L on user$_i$ and DSP. The results are shown in Figure 4(a). With the increase of L, the encryption time of DSP is gradually increasing, and the encryption time of user$_i$ can be considered as unchanged. We also tested the impact of different L on the diagnosis values computation of CSS. The results are shown in Figure 4(b). With the continuous increase of L, the computation time for CSS is also increasing. The process of

Table 7: Comparison analysis.

Schemes	Data encryption (s)	Diagnosis values computation (s)	Total time (s)
Reference [38], $l = 512$	0.658	0.005	0.663
Reference [39], $l = 512$	0.57	0.552	1.122
Reference [40], $l = 512$	0.004	1.092	1.096
Ours	0.008	0.096	0.104

Table 8: Comparison analysis of the cloud server.

Schemes	Data encryption (s)	Diagnosis value computation (s)	Total time (s)
Reference [38], $l = 512$	0	0	0
Reference [39], $l = 512$	0.57	0.001	0.571
Reference [40], $l = 512$	0	1.092	1.092
Ours	0	0.096	0.096

Table 9: Comparison analysis of the client.

Schemes	Data encryption (s)	Diagnosis value computation (s)	Total time (s)
Reference [38], $l = 512$	0.658	0.005	0.663
Reference [39], $l = 512$	0	0.551	0.551
Reference [40], $l = 512$	0.004	0	0.004
Ours	0.008	0	0.008

Table 10: Comparison analysis of diagnosis result generation.

Schemes	Cloud server computation (s)	Client computation (s)	Diagnosis result generation (s)
Reference [38], $l = 512$	1.584	0.097	1.681
Reference [39], $l = 512$	0.007	0.041	0.048
Reference [40], $l = 512$	0.101	0	0.101
Ours	0.071	0	0.071

generating diagnosis result is jointly completed by CSS and CCS. We tested the effect of different L on the diagnosis result generation. The results are shown in Figure 4(c). With the continuous increase of L, the time for CSS and CCS is also increasing.

7.2.3. Comparison Analysis of Secret Diagnosis Request Generation and Diagnosis Values Computation. In our proposed scheme, secret diagnosis request generation can be regarded as data encryption of user_i and diagnosis value computation can be regarded as homomorphic operation. We compared with the other three privacy preserving schemes. The results are shown in Table 7.

Through the comparison analysis, it can be seen that the time of data encryption in our proposed scheme is significantly reduced compared with [38, 39]. In the computation of decision function, our scheme has significantly reduced the computational cost compared with the scheme in [39, 40]. At the same time, it can be seen from the total time that our proposed scheme is significantly lower than the other three schemes.

Next, we make further analysis. The names of participants may be slightly different in different schemes. In order to facilitate analysis, we divided participants into cloud server and client. We compared the computation overhead of cloud server and client, respectively. The results are shown in Tables 8 and 9.

In our proposed scheme, the client only needs to encrypt the data and can be offline after uploading the data to the cloud server. The cloud server only needs to compute the decision function. This model reduces the computation overhead of the client to the greatest extent and performs privacy preserving computation through the powerful computing power of the cloud server. In scheme [38], the cloud server does not participate in the whole process, so it brings heavy computation overhead to the client. In scheme [39], the computation of the diagnosis values needs to be completed by the cloud server and the client. Therefore, it not only brings heavy computation overhead to the client but also requires the client to always stay online in this process.

7.2.4. Comparison Analysis of Diagnosis Result Generation. In our proposed scheme, after CSS completes the diagnosis values computation, it will jointly execute the secure protocol with CCS to generate the diagnosis result. We continued to make comparison analysis with schemes in [38–40]. The results are shown in Table 10.

Through the comparison analysis in Table 10, it can be seen that the computation time of our proposed scheme is

Table 11: Comprehensive comparison analysis.

	Data encryption (s)	Diagnosis value computation (s)	Diagnosis result generation (s)	Total time (s)
Reference [38], $l = 512$	0.658	0.005	1.681	2.389
Reference [39], $l = 512$	0.57	0.552	0.048	1.17
Reference [40], $l = 512$	0.004	1.092	0.101	1.197
Ours	0.008	0.096	0.071	0.175

Table 12: Scheme summary.

Schemes	Multidata owners	Training	Multiclassification	Low client overhead
Reference [38]	×	×	✓	×
Reference [39]	×	×	✓	×
Reference [40]	×	×	✓	✓
Reference [37]	✓	✓	×	✓
Ours	✓	✓	✓	✓

significantly lower than the scheme in references [38, 40]. In our proposed scheme, the client does not need to participate in the process of diagnosis result generation. The schemes in references [38, 39] require the participation of the client, which brings heavy computation overhead to the client.

7.2.5. Comprehensive Comparison Analysis. We made a comparison analysis of the whole privacy preserving online disease diagnosis process. It is divided into the secret diagnosis request generation (data encryption), diagnosis value computation, and diagnosis result generation. The results are shown in Table 11.

Through the comparison analysis in Table 11, the total time of our proposed scheme is significantly lower than the schemes in references [38–40]. Considering that in the actual application scenario, a large number of users will constantly initiate secret diagnosis requests. It is very important to be able to quickly respond to the diagnosis results for users. Therefore, our scheme has more practical application value. Then, we made a summary as shown in Table 12.

8. Conclusion

In this paper, we propose an efficient and privacy preserving outsourced multiclass SVM training and online-aided disease diagnosis scheme. We design some secure basic operation algorithms for machine learning training over the outsourced data from multiple data owners. We achieve a privacy preserving multiclass SVM training based on the basic operation algorithms. In the diagnosis phase, we achieve a privacy preserving multiclass diagnosis through our proposed the secure maximum find algorithm and secure comparison algorithm. Security analysis proves that our proposed scheme ensures that outsourced data, model parameters, users' symptoms, and diagnosis results will not be leaked. Experimental evaluation illustrates that our proposed scheme significantly reduces the computation overhead. In the future, we will study more efficient and privacy preserving machine learning schemes.

Data Availability

The data supporting the results of this study can be obtained from the corresponding author.

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] M. Kim, Y. Song, S. Wang, Y. Xia, and X. Jiang, "Secure logistic regression based on homomorphic encryption: design and evaluation," *JMIR medical informatics*, vol. 6, no. 2, p. e19, 2018.

[2] X. Yang, R. Lu, J. Shao, X. Tang, and H. Yang, "An efficient and privacy-preserving disease risk prediction scheme for e-healthcare," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3284–3297, 2019.

[3] E. Ayday, J. L. Raisaro, P. J. McLaren, J. Fellay, and J.-P. Hubaux, "Privacy-preserving computation of disease risk by using genomic, clinical, and environmental data," in *Proceedings of the 2013 { USENIX} Workshop on Health Information Technologies (HealthTech 13)*, Washington, DC, USA, August 2013.

[4] J. Zhang, L. Zhang, M. He, and S.-M. Yiu, "Privacy-preserving disease risk test based on bloom filters," in *Proceedings of the International Conference on Information and Communications Security*, pp. 472–486, Singapore, December 2017.

[5] Z. Ma, J. Ma, Y. Miao, and X. Liu, "Privacy-preserving and high-accurate outsourced disease predictor on random forest," *Information Sciences*, vol. 496, pp. 225–241, 2019.

[6] N. Bouguila, "Hybrid generative/discriminative approaches for proportional data modeling and classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 12, pp. 2184–2202, 2012.

[7] M. Ćwiklińska-Jurkowska, "Performance of the support vector machines for medical classification problems,"

Biocybernetics and Biomedical Engineering, vol. 29, no. 4, pp. 63–81, 2009.

[8] S. G. Teo, S. Han, and V. C. Lee, "Privacy preserving support vector machine using non-linear kernels on hadoop mahout," in *Proceedings of the 2013 IEEE 16th International Conference on Computational Science and Engineering*, pp. 941–948, IEEE, Sydney, Australia, December 2013.

[9] M. Z. Omer, H. Gao, and F. Sayed, "Privacy preserving in distributed svm data mining on vertical partitioned data," in *Proceedings of the 2016 3rd International Conference on Soft Computing & Machine Intelligence (ISCMI)*, pp. 84–89, IEEE, Dubai, UAE, November 2016.

[10] L. Wang, J. J. Shi, C. Chen, and S. Zhong, "Privacy-preserving face detection based on linear and nonlinear kernels," *Multimedia Tools and Applications*, vol. 77, no. 6, pp. 7261–7281, 2018.

[11] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: a hybrid secure computation framework for machine learning applications," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pp. 707–721, Incheon, Korea, June 2018.

[12] H. Zhu, X. Liu, R. Lu, and H. Li, "Efficient and privacy-preserving online medical prediagnosis framework using nonlinear svm," *IEEE journal of biomedical and health informatics*, vol. 21, no. 3, pp. 838–850, 2017.

[13] Y. Rahulamathavan, R. C.-W. 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.

[14] S. Perveen, M. Shahbaz, K. Keshavjee, and A. Guergachi, "A systematic machine learning based approach for the diagnosis of non-alcoholic fatty liver disease risk and progression," *Scientific Reports*, vol. 8, no. 1, pp. 2112–12, 2018.

[15] L. Jena, S. Nayak, and R. Swain, "Chronic disease risk (cdr) prediction in biomedical data using machine learning approach," in *Advances in Intelligent Computing and Communication*, pp. 232–239, Springer, Cham, Switzerland, 2020.

[16] X. Liu, R. Lu, J. Ma, L. Chen, and B. Qin, "Privacy-preserving patient-centric clinical decision support system on naive bayesian classification," *IEEE journal of biomedical and health informatics*, vol. 20, no. 2, pp. 655–668, 2016.

[17] J. Ramírez, J. Górriz, F. Segovia et al., "Computer aided diagnosis system for the alzheimer's disease based on partial least squares and random forest spect image classification," *Neuroscience Letters*, vol. 472, no. 2, pp. 99–103, 2010.

[18] A. Triastcyn and B. Faltings, "Bayesian differential privacy for machine learning," in *Proceedings of the International Conference on Machine Learning*, pp. 9583–9592, PMLR, Shenzhen, China, February 2020.

[19] S. Laur, H. Lipmaa, and T. Mielikäinen, "Cryptographically private support vector machines," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 618–624, Philadelphia, PA, USA, August 2006.

[20] K. Mandal and G. Gong, "Privfl: practical privacy-preserving federated regressions on high-dimensional data over mobile networks," in *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, pp. 57–68, London, UK, November 2019.

[21] M. Shen, X. Tang, L. Zhu, X. Du, and M. Guizani, "Privacy-preserving support vector machine training over blockchain-based encrypted iot data in smart cities," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7702–7712, 2019.

[22] 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.

[23] Y. Zheng, H. Duan, C. Wang, R. Wang, and S. Nepal, "Securely and efficiently outsourcing decision tree inference," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 3, pp. 1841–1855, 2022.

[24] S. Tan, B. Knott, Y. Tian, and D. J. Wu, "Cryptgpu: fast privacy-preserving machine learning on the gpu," in *Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP)*, pp. 1021–1038, IEEE, San Francisco, CA, USA, May 2021.

[25] F. Zheng, C. Chen, and X. Zheng, "Towards secure and practical machine learning via secret sharing and random permutation," 2021, https://arxiv.org/abs/2108.07463.

[26] X. Li, J. He, P. Vijayakumar, X. Zhang, and V. Chang, "A verifiable privacy-preserving machine learning prediction scheme for edge-enhanced hcpss," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 8, pp. 5494–5503, 2022.

[27] Z. Ma, J. Ma, Y. Miao et al., "Lightweight privacy-preserving medical diagnosis in edge computing," *IEEE Transactions on Services Computing*, vol. 15, no. 3, pp. 1606–1618, 2022.

[28] L. Sun, W.-S. Mu, B. Qi, and Z.-J. Zhou, "A new privacy-preserving proximal support vector machine for classification of vertically partitioned data," *International Journal of Machine Learning and Cybernetics*, vol. 6, no. 1, pp. 109–118, 2015.

[29] R. Chen, Q. Xiao, Y. Zhang, and J. Xu, "Differentially private high-dimensional data publication via sampling-based inference," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 129–138, Sydney Australia, August 2015.

[30] J. Zhang, X. Xiao, Y. Yang, Z. Zhang, and M. Winslett, "Privgene: differentially private model fitting using genetic algorithms," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 665–676, New York, NY, USA, June 2013.

[31] O. Ohrimenko, F. Schuster, C. Fournet et al., "Oblivious multi-party machine learning on trusted processors," in *Proceedings of the 25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 619–636, Austin TX USA, August 2016.

[32] K. A. Jagadeesh, D. J. Wu, J. A. Birgmeier, D. Boneh, and G. Bejerano, "Deriving genomic diagnoses without revealing patient genomes," *Science*, vol. 357, no. 6352, pp. 692–695, 2017.

[33] H. Yu, J. Vaidya, and X. Jiang, "Privacy-preserving svm classification on vertically partitioned data," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 647–656, Springer, Cham, Switzerland, 2006.

[34] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-lwe and security for key dependent messages," in *Annual Cryptology Conference*, pp. 505–524, Springer, 2011.

[35] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *Proceedings of the Internet Society Network and Distributed System Security Symposium*, pp. 1–4, San Diego, CA, USA, February 2015.

[36] J.-C. Bajard, P. Martins, L. Sousa, and V. Zucca, "Improving the efficiency of svm classification with fhe," *IEEE*

Transactions on Information Forensics and Security, vol. 15, pp. 1709–1722, 2020.

[37] 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.

[38] 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.

[39] T. Li, Z. Huang, P. Li, Z. Liu, and C. Jia, "Outsourced privacy-preserving classification service over encrypted data," *Journal of Network and Computer Applications*, vol. 106, pp. 100–110, 2018.

[40] 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*, p. 1, 2021.

[41] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of the International conference on the theory and applications of cryptographic techniques*, pp. 223–238, Springer, Prague, Czech Republic, May 1999.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 3670107, 16 pages
https://doi.org/10.1155/2022/3670107

Research Article

A User-Centered Medical Data Sharing Scheme for Privacy-Preserving Machine Learning

Lianhai Wang, Lingyun Meng, Fengkai Liu, Wei Shao, Kunlun Fu, Shujiang Xu, and Shuhui Zhang

Qilu University of Technology (Shandong Academy of Sciences), Shandong Computer Science Center (National Supercomputer Center in Jinan), Shandong Provincial Key Laboratory of Computer Networks, Jinan 250014, China

Correspondence should be addressed to Lianhai Wang; wanglh@sdas.org

Received 25 June 2022; Accepted 24 August 2022; Published 30 September 2022

Academic Editor: Wenbo Shi

With the rapid development and application of artificial intelligence technology, medical data play an increasingly important role in the medical field. However, there are privacy protection and data ownership issues in the process of data sharing, which brings difficulties to machine learning and data mining. On the one hand, for fear that they may risk being held accountable by users or even breaking the law due to these issues, healthcare providers are reluctant to share medical data. On the other hand, users are also reluctant to share medical data due to the possibility of privacy disclosure in the data sharing process. To improve the security and privacy of shared medical data, we propose a user-centered medical data sharing scheme for privacy-preserving machine learning. Our solution combines blockchain and a trusted execution environment to ensure that adversaries cannot steal the ownership and control of user data during sharing. A blockchain-based noninteractive key sharing scheme is proposed that allows only the users and the TEE to decrypt the shared data. At the same time, we design an auditing mechanism to facilitate users to audit the sharing process. The security analysis shows that the scheme ensures the privacy and security of user data during storage and sharing. We have completed simulation experiments to demonstrate the effectiveness and efficiency of our scheme.

1. Introduction

In the era of the digital economy, data have become a new factor of production and an important basic strategic resource. Data support the future development and drive the progress in business or scientific fields. In particular, with the development of IoT technology in health care, the healthcare ecosystem generates many medical data, such as electronic medical records, monitoring data, imaging data, and smart wearable device data. These data contain a huge amount of information [1], which can assist physicians in clinical decision-making and play an important role in drug development, intelligent diagnosis, medical image recognition, and precision medicine [2]. Therefore, how to handle and utilize the growing amount of healthcare data has become an unavoidable problem.

With the rapid development and application of artificial intelligence (AI) technology, scholars have established several medical artificial intelligence models for intelligent analysis and decision-making using of medical data, especially for specialized medical record data. They have achieved fruitful research results [1]. The AI-based medical analysis requires medical data from multiple medical institutions, pharmaceutical companies, or individuals for extensive sample annotation and training [3].

For hospitals, due to the characteristics of medical data, such as privacy and sensitivity, there may be data security and privacy leakage risks in the sharing process. Once the patient's private data are leaked, the hospital will face medical disputes and even legal liability. On the other hand, the issue of ownership and access control of medical data are also controversial. If the hospital shares or uses the patient's medical data without authorization, it may be held accountable by the patient. Therefore, hospitals are always reluctant to share these data.

For patients, sharing data also exposes them to the risk of privacy breaches. Moreover, most patients' medical data are recorded in hospitals or healthcare facilities. Even if these

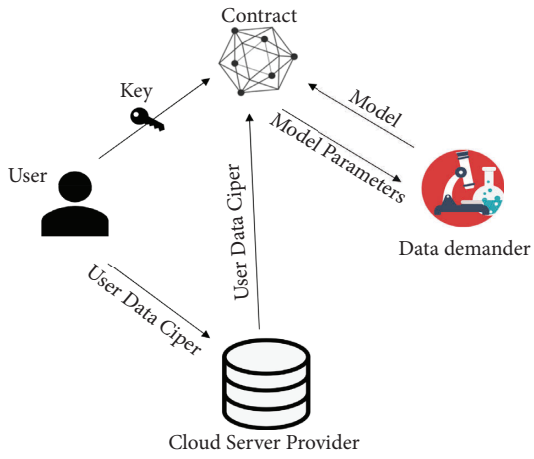


Figure 1: Medical data sharing framework.

data belong to the patients [4], it is difficult for patients to access their medical data. In addition, because of the reproducible nature of data, patients risk losing ownership of their data once they share them. So, there is also a reluctance to share their medical data with patients.

With the development of blockchain technology, its broad application prospects and technical features have attracted the attention of scholars related to various industries. Its distributed storage, peer-to-peer transmission, consensus mechanism, and confidentiality algorithm provide numerous novel solutions for data storage and sharing. Some researchers have used blockchain to implement on-chain data storage and smart contract-based access control to data [5]. Access rights are controlled by techniques such as identity-based [6] and attribute-based access controls [7] to ensure the privacy and security of shared data. However, under existing strategies, the efficiency in performing data sharing is lower due to the blockchain's storage space and computational performance. Moreover, once data are acquired by data demanders, they can access or use the data without any control. Therefore, this approach cannot prevent individuals or organizations from sharing data illegally, and it is even more difficult to ensure illegal analysis and misuse of data.

Some schemes combine blockchain with proxy re-encryption [8] to ensure the security and privacy of medical data during the sharing process. However, proxy re-encryption requires many computational resources, and the system is relatively inefficient. Moreover, these schemes do not consider the issue of control and ownership of data [9].

To address the above issues, we propose a user-centric medical data sharing scheme oriented to privacy-preserving machine learning to achieve efficient medical data sharing that protects data privacy. The model framework is shown in Figure 1. To ensure control and ownership of user data during storage and sharing, we design a new sharing model by combining blockchain with a trusted execution environment (TEE), although the combination of TEE and blockchain will encounter difficulties in data exchange and trust. But we propose a way of combining on-chain and off-chain to solve the problem of data exchange. In addition, we solve the trust problem between the two through signature authentication. We also design a new key sharing scheme for authorization management. In summary, this paper contributes the following:

(i) We propose a blockchain-based system for data sharing and privacy protection. A TEE obtains and deploys machine learning models from the blockchain, where data are decrypted for model training. Then, the training results are verified on the chain and shared with the data demander.

(ii) We have designed a new key sharing scheme to share and manage keys through smart contracts, which allows users to authorize their data off-chain.

(iii) We build a user audit mechanism. The record of the sharing process of users' medical data is stored on the blockchain, in which users can query at any time for auditing the sharing process.

(iv) We implement a prototype of our model and validate its effectiveness. The experiment shows that our scheme can protect the privacy and security of user medical data and ownership without incurring significant additional time overhead compared to existing solutions.

The remaining part of the paper is organized as follows. We begin by introducing some related works in Section 2. Section 3 is concerned with some preliminaries used in this paper. In Section 4, we describe the system model and design goals. In Section 5, the proposed system operational details are presented. In Section 6, we did a security and functional analysis. Program design and evaluation are presented in Section 7. Finally, this paper is concluded in Section 8.

2. Related Work

In this section, we review some research solutions for secure data sharing. To solve the problems of inefficiency and poor scalability of traditional medical data sharing systems, some schemes [10–12] are proposed using blockchain combined with cloud storage to solve the data security problem of data sharing. The data are stored encrypted in the cloud, and then, the storage index and data hash are uploaded to the blockchain for secure data storage and sharing. However, this data sharing method has no access control mechanism, and data security is difficult to guarantee. Meanwhile, when the data are shared, the data owner loses control and ownership of the data, which also poses a threat to the privacy and security of the data owner.

To protect the privacy and security of user data, some solutions propose controlled access [13] to manage healthcare data sharing. The literature [14] uses attribute-based sharing techniques, but when the access policy is modified, attribute revocation and encryption of the data are required again, which increases the computational overhead. Moreover, the tamper-proof nature of blockchain also complicates the modification of access control policies. Gu et al. proposed an efficient and simple attribute-based sharing scheme [15] that reduces computational costs and enables privacy protection. However, this scheme consumes

many computing resources when modifying the access policy. Wang and Song [16] proposed an electronic health record system built using attribute-based controlled access and blockchain technology. However, the whole system is too large, expensive to run, and inefficient to execute. Guo et al. [17] proposed an attribute-based signature scheme combined with blockchain technology that can protect user data privacy. However, the computational performance of the blockchain leads to an inefficient system. Moreover, when a malicious user obtains data through access control, there is a threat to user data privacy.

To ensure secure data sharing, several research proposals have proposed the use of cryptography [18] for data privacy security protection. Renpeng Zou et al. proposed SPChain [19] to enable medical data sharing for users in a privacy-preserving manner by using a proxy re-encryption scheme. However, this solution is difficult to implement, consumes too many resources, and has slow processing speed and poor portability. Chen et al. [20] proposed an anonymous medical data sharing scheme based on a cloud server and proxy re-encryption algorithm to improve the security of private medical data sharing. However, the original data in this scheme will still be accessed, and data privacy may be compromised.

Some scholars have used cryptography-based schemes to address these issues to protect the privacy and security of medical data during data analysis and after requesters access the data. Kosba et al. proposed a blockchain-based platform for contract development [21]. The platform uses a zero-knowledge proof-based cryptographic protocol to handle private data, rather than storing private transaction data directly on the blockchain, effectively guaranteeing private data security. However, this scheme requires many computing resources and is not scalable. The literature [22] proposes a blockchain privacy protection scheme based on homomorphic encryption, but it requires many computational resources, and its practicality and applicability are greatly limited.

To ensure that data in shared data do not leave the authorization system and thus do not disclose sensitive information, the literature in [23] proposes a combined blockchain and federation learning scheme for sharing privacy-preserving IoT data among distributed multiple parties. Another framework for sharing vehicle data based on blockchain and federated learning for edge computing is proposed in the literature [24] for the internet of vehicles (IoV). However, this scheme suffers from data poisoning that affects the global model and backdoor attacks. Zhou et al. proposed a health insurance storage system [25], which utilizes secret sharing techniques and secure multiparty computing that allows for the sharing of patient data between hospitals and insurance companies. However, this scheme does not guarantee that all servers are fully trusted, and data requesters have to receive responses from multiple nodes before accessing the data. Shamir's secret sharing and secure multiparty computation (MPC) were applied in Shrier et al. [26] to achieve data sharing while satisfying user privacy. Yue et al. [27] proposed a medical data gateway (HDG) to analyze medical data using secure multiparty computing while ensuring user control privileges. The literature [28, 29] combines multiparty secure computation and differential privacy to guarantee the accuracy of the output results without losing data privacy at the user's end. However, in complex computational tasks, the results can significantly differ from the noise-free results, making the results unusable.

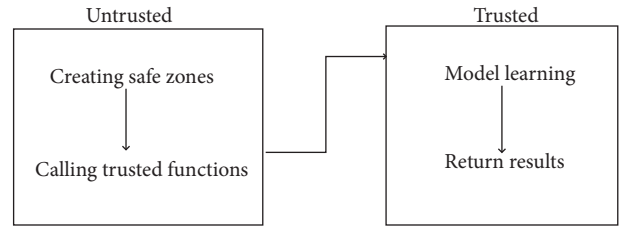


Figure 2: SGX schematic.

3. Preliminaries

3.1. Trusted Execution Environment. The trusted execution environment (TEE) is a secure zone in the computing platform, using a combination of trusted computing and virtualization isolation techniques. The TEE provides a trusted execution environment for "security-sensitive applications" while protecting the confidentiality and integrity of associated data. ARM's TrustZone technology implements hardware isolation mechanisms, mainly for embedded mobile terminal processors, to create separation between the secure and nonsecure worlds. In addition to TrustZone, based on the ARM architecture, Intel has also released a trusted execution environment based on its processor architecture: Intel SGX. SGX is a set of instructions that enhance the security of application code and data, providing them with more robust protection against disclosure or modification. Calling a program in the trusted zone requires defining the eCall interface and declaring the structure and size of the data to be passed. Intel SGX provides good integrity and confidentiality protection for its applications due to its hardware-level implementation. Since its release, it has been sought after by academia and industry and is used in scenarios such as outsourced cloud computing and sensitive data aggregation. Microsoft has proposed a database architecture EnclaveDB [30], based on SGX that runs within a secure zone. The data within the trusted database are implemented so that even when hackers compromise the server operating system; the data are still not accessible to the hackers. In addition to database applications, Kunkel et al. [31] ported machine learning [32] to the SGX secure zone, allowing the machine learning training and prediction process to take place within the secure zone, thereby protecting the privacy of the raw data. The SGX processing is shown in Figure 2.

3.2. Smart Contracts. Ethereum is a common public blockchain open-source platform whose main and most characteristic feature is integrating smart contract function. Ethereum provides a decentralized Ethereum virtual

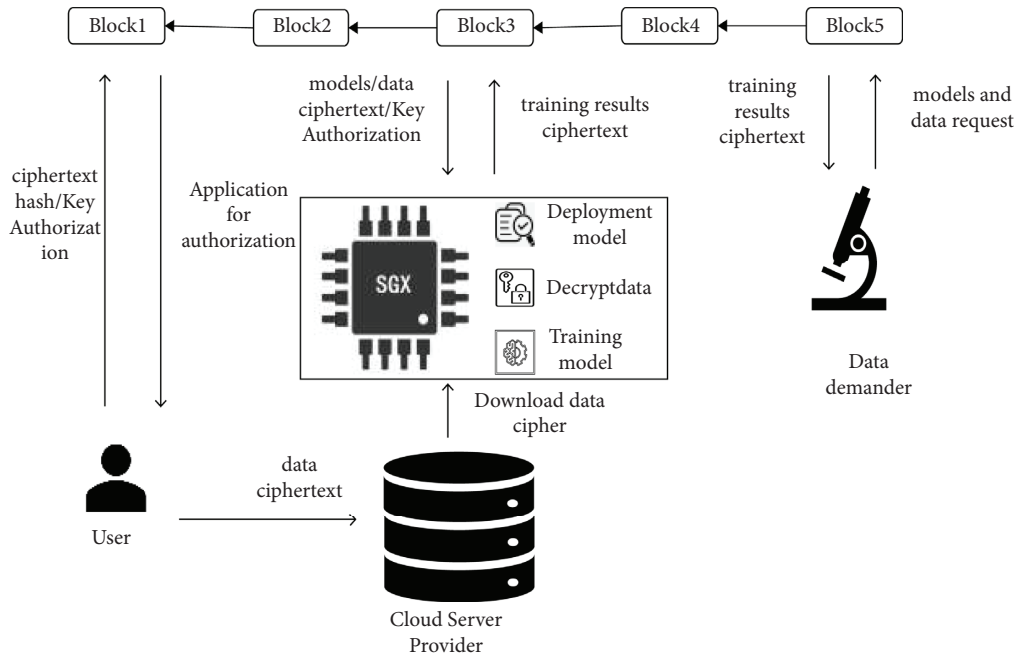


Figure 3: System model.

machine to users who join the Ether Place through its cryptocurrency, Ether (ETH), which provides peer-to-peer smart contract computing to all users. Ethereum is a typical representative of Blockchain 2.0, which increases the scalability and flexibility of the protocol based on Blockchain 1.0. By providing users with various modules, users can flexibly build smart contracts that suit their needs and deploy them into the Ethereum network by consuming Ether. Since the Ethereum virtual machine is Turing-complete, the business provided by Ethereum smart contracts is almost endless. In theory, any computer business can be deployed into Ethereum in the form of smart contracts to realize decentralized business operation and ensure the proper operation of the business as much as possible. Each command executed by the smart contract requires a certain amount of consumption, which uses gas as the unit. Also, different commands require different gases. Each transaction must first set a value called gasLimit, the maximum consumption value and miners have the right to choose which transaction to pack first. Generally, the larger the value of gasLimit, the more attractive miners are to pack. Ethereum is a highly integrated blockchain system in practical use, and users/developers mainly use smart contracts to publish some transactions and functional modules in Ethereum.

4. System Model

4.1. High-Level Overview. The flow of users sharing personal medical data in this solution is shown in Figure 3. The user encrypts and uploads the data to the cloud server for storage. Then, the user uploads the data cryptographic hash and data storage index to the blockchain. The data demander uploads the model and data request to the blockchain and then deploys the model into a TEE via a smart contract. The smart contract sends the data request to the user, who performs the key authorization.

The TEE downloads the data cipher hash and storage index on the blockchain and uses the storage index to download the data cipher from the cloud server. Then, TEE gets the key authorization from the smart contract and decrypts the data cipher to get the original data. The model is then trained using the user's medical data. Finally, the model training results are encrypted using the public key of the data demander and uploaded to the blockchain. The data demander then retrieves the training result ciphertext from the blockchain, makes its private key to decrypt it, and obtains the model training result.

4.2. System Architecture. The architecture of the proposed system is shown in Figure 3. It consists of six entities: user, medical research institute, TEE, blockchain, smart contract, and storage server. More details of the system are shown below.

User. The user stores medical data encrypted in the storage container and stores information such as the returned storage index and ciphertext hash in the blockchain. At the same time, users also authorize access requests to medical research institutions and encrypt the encryption key using TEE's public key, which is then stored on the blockchain. After the shared data are finished, the user can also call the chain code to query the data generated during the sharing process, thus playing a supervisory role.

Data Demander. Medical research institutions make the required models and submit them to the model review smart contract for review, then upload the machine learning models to the blockchain, and finally get the trained models from the blockchain.

TEE. A trusted, isolated, and independent execution environment exists independently of an untrusted operating system, providing a secure and confidential space for private data and sensitive computations in an untrusted environment, whose security is typically guaranteed through hardware-related mechanisms. In this scheme, we perform operations such as decrypting the user's data cipher, verifying data integrity, training model, and uploading training model parameters to the blockchain.

Storage Device. It is used in this solution to store user-encrypted medical data.

Smart Contracts. They are used to deploy machine learning algorithms, invoke medical data for sharing, and data transfer. They rely on smart contracts to achieve data scheduling and processing of data generated by the process of sharing data to achieve a regulatory mechanism.

Blockchain. The Ethernet public chain is used here. In this scheme, the blockchain is used to store data hashes, model, data integrity verification results, and model learning results.

4.3. Threat Model. To better describe the working process of the system model, we rely on the following assumptions:

(1) Smart contract records are reliable and readily available. This is because it is difficult for an attacker to tamper with the records posted on the blockchain, which is essentially a distributed ledger that runs all the time.

(2) This solution uses an off-chain storage system, mainly responsible for storing user medical data ciphertext. When the data are stored, a storage index is generated and used to store it in the blockchain.

(3) In this scheme, the computing process in the TEE cannot be accessed by the outside world in any way.

4.4. Design Goals. In this solution, we aim to achieve secure sharing of user medical data, for which we propose the following design objectives:

(1) *Secure Storage and Sharing.* Users' medical data should be securely stored, and no entity should be able to tamper with this information. In addition, it is guaranteed that no entity may view and tamper with the user's medical information during the sharing process.

(2) *Shared Data Can Be Supervised.* Users want to supervise the operations that their personal medical data undergo during the sharing process to prevent illegal use and analysis of personal medical data.

(3) *Efficiency.* A large amount of user medical data need to be stored and shared on time, so it should have high storage and sharing efficiency.

(4) *Data Dedicated Exclusive Use.* The user's medical data will only be used and analyzed for legitimate dedicated use, and any illegal manipulation of the user's data is not feasible.

(5) *User's Data Ownership.* This is done first to prevent malicious accounts from changing the user's account and tampering with the user's data ownership. Second, it ensures that after data sharing, the ownership of the data remains with the user.

(6) *Security of Computing Environment.* It ensures the privacy and security of users' medical data processing and does not disclose any private information when computing and analyzing data.

(7) *Ability to Resist Other Attacks.* To enhance security further, the protocol should provide resilience to other common attacks, such as replay attacks.

5. Our Proposed Protocol

The controlled medical data sharing for machine learning proposed in this solution can be specifically divided into the following phases: system initialization, medical data storage, machine learning model deployment, training model, and data demander to obtain training results. In Table 1, we illustrate some of the notations in the scheme.

5.1. System Initialization. Before the system starts to execute, we complete the initialization work. The specific steps are as follows:

(1) *Basic Initialization.* A cyclic additive group G with generating element g and prime order Q is chosen on an elliptic curve $E(Fp)$ over a finite field Fp and a one-way hash function $H1$: $\{0,1\}^* -> Z_q$. Then, the symmetric key encryption function Encrypt and decryption function Decrypt, Encrypt_*ECC* asymmetric key encryption algorithm and decryption algorithm Decrypt_*ECC*, and RSA signature function Sig_*RSA* and verification function Verify_*RSA* are selected.

(2) *Blockchain Initialization.* We create the file genesis.json containing the configuration parameters to build the Ethereum blockchain. Each node generates a public-private key pair $\{PK, SK\}$. One set of predesignated nodes is responsible for mining. The rest of the network collectively trusts these nodes to validate transactions and create new blocks. In our case, the trusted institutions consisted of medical research institutions, insurance companies, and regulatory agencies. Trusted institutions perform various functions, including adding data to decentralized file systems, uploading corresponding transactions to the blockchain, and validating various transactions received from external users, such as permission requests and permission grants.

(3) *Smart Contract Deployment.* In this scheme, there are four types of contract components: registration contract, data contract, authorization contract, model contract, and audit contract:

(a) *Registration Contract.* All nodes are registered anonymously on the registration contract to prevent users from providing false data or data

Table 1: Table notations.

Notation	Description
SK	Private key
PK	Public key
KE	Symmetric key
UDATA_i	Raw data of user
Model_j	Model of demander
UDT	Data type

demanders from providing illegal models. The registration information includes the public key and the role of the node.

(b) *Data Contract*. The data contract stores a list of data records that indicate the mapping relationship between the data and the user. Each data in the list consist of the user's public key, the data cipher hash, the off-chain original data storage index, and the user's signature. The ability to add, modify, and delete data is also provided in this contract. In addition, the TEE and users can retrieve and download data through this contract.

(c) *Authorized Contract*. The authorization contract assists the user in encrypting the data encryption key and authorizing the key to the data demander.

(d) *Model Contract*. The model contract stores a list of models and a list of model training results. Data demanders can upload models and download model training results through this contract. TEE can also store model training results through model contracts.

(e) *Audit Contract*. The audit list is set up, and the list data include information about the data owner, the data demander, the data integrity verification results, and the models trained on the data. Users can audit the process of sharing personal medical data through an audit contract.

5.2. Medical Data Storage. Due to the limitation of SGX memory, the user's medical data need to be preprocessed before uploading. We sort and label data by system requirements prior to data storage before storing it. In order to store the user medical data information, the structure $\{\mathrm{PK_i}, \mathrm{UDT}, \mathrm{DA_i}, \mathrm{US_i}, \mathrm{HDS_i}, DT_i, CM_i\}$ of the data storage transaction TD_i is designed. In this, TD_i is a public transaction and any node can access the data in TD_i. TD_i contains the user's public key PK_i, timestamp DT_i, hash of data ciphertext $H\ DS_i$, user signature US_i, data storage index DA_i, medical data typology $U\ DT$, and encryption key ciphertext CM_i. The following description illustrates the process of storing user medical data.

5.2.1. Data Preprocessing

(a) The user cuts and divides the data according to the system requirements, and then labels the divided data.

5.2.2. Data Upload

(a) *Encrypt the Raw Data*. User i calls Encrypt function to generate a new symmetric key KEY_i using private key SK_i and random number n_i. Then, we encrypt the medical data $U\mathrm{DATA}_i$ with key KEY_i to generate ciphertext DS_i.

$$DS_i = \mathrm{Encrypt}(\mathrm{KEY}_i, U\mathrm{DATA}_i). \tag{1}$$

(b) *Store ciphertext to Cloud Server*. User i stores medical data ciphertext to the cloud server and then gets the storage index DA_i.

5.2.3. Data on the Chain

(a) *Generate Hash Index*. User i uses hash function to generate hash value $H\ DS_i$ for data ciphertext DS_i.

$$HDS_i = H1(DS_i). \tag{2}$$

(b) *User Signature*. User i uses the signature function Sig_RSA to sign the data ciphertext hash to get the signature US_i.

$$US_i = \mathrm{Sig}_R SA(SK_i, H\ DS_i). \tag{3}$$

(c) *Encrypting the Symmetric Key*. User i invokes the authorized contract to obtain the public key PK_T of the TEE, the public key $PK_C$of the contract. Then, the user generates random numbers r_U and r_C, encrypts the symmetric key, and generates the ciphertext CM_i.

$$CM_i = \mathrm{KEY}_i + r_C PK_C + r_U PK_T. \tag{4}$$

(d) *Publish Stored Data Transactions*. User i invokes the data contract to store TD_i into the blockchain.

$$TD_i = \{PK_i, UDT\ , DA_i, US_i, HDS_i, DT_i, CM_i\}. \tag{5}$$

(e) *User Authorization*. The user invokes the authorization contract to upload the $r_U g$ and $r_C PK_C$.

Algorithm 1 shows the process of storing medical data from user i to the cloud server and blockchain.

5.3. Machine Learning Model Deployment

5.3.1. Model Storage. In order to store the machine learning model on the blockchain, the data structure $\{\mathrm{MID}_j, PK_j, MT_j, \mathrm{model}_j, H\mathrm{model}_j, SM_j, RM_j\}$ of the model storage transaction IM_j is designed. It contains the data demander node identifier MID_j, public key PK_j, timestamp MT_j, model model_j, hash of model $H\mathrm{model}_j$, signature SM_j of the medical institution, and data demand RM_j. The storage process of the user medical data is described as follows:

Input: Raw Data $UDATA_i$; Public Key PK_i; Data Type UDT; Random Number n_i; Private Key SK_i;
Output: Storage Index DA_i; Blockchain Transaction TD_i;
Stage 1: Upload data to the cloud server:
(1) Generate symmetric key KEY_i by PK_i
(2) Encrypted raw data $DS_i = \text{Encrypt}(KEY_i, UDATA_i)$
(3) Send DS_i to the cloud server and get DA_i
Stage 2: Upload TD_i to Blockchain
(4) Generate timestamp DT_i
(5) Generate ciphertext hash $HDS_i = H1(DS_i)$
(6) User Signature $US_i = \text{Sig\_RSA}(SK_i, HDS_i)$
(7) Call the authorization contract to get $\{r_C, PK_C, PK_T\}$
(8) Generate random number r_U
(9) Encryption symmetric key KEY_i: $CM_i = KEY_i + r_C PK_C + r_U PK_T$
(10) $TD_i = \{PK_i, UDT, DA_i, US_i, HDS_i, DT_i, CM_i\}$
(11) Call the data contract to upload data TD_i to the blockchain
(12) Call the Authorized contract to upload data TD_i, $r_U g$ and $r_C PK_C$
(13) return (DA_i, TD_i);

Algorithm 1: Medical data storage.

(1) We upload models to the blockchain.

(a) *Generating Models*. The data demander j produces and generates machine learning models and data requirements.

(b) *Obtaining the Model Hash*. The data demander j uses the hash function to calculate the hash value of the model.

$$H\text{model}_j = H1(\text{model}_j). \tag{6}$$

(c) *Data Demander Signature*. Data demander j uses the signature function Sig_*RSA* to sign the data ciphertext hash to get the signature SM_j.

$$SM_j = \text{Sig\_}RSA(SK_j, H\text{model}_j). \tag{7}$$

(d) *Posting Stored Data Transactions*. Data demander j invokes a data contract to store IM_j into the blockchain.

$$IM_j = \{\text{MID}_j, PK_j, MT_j, \text{model}_j, H\text{model}_j, SM_j, RM_j\}. \tag{8}$$

Algorithm 2 shows the process of storing the machine learning model and data requirements to the blockchain by the data demander j.

5.3.2. Model Deployment. The TEE retrieves the blockchain to get the model after it is stored on the blockchain. To ensure the authenticity of the model, the model needs to be validated. For this purpose, the data structure $\{ST_j, \text{HRT}_j, VT_j\}$ for model validation is designed, where ST_j is the signature of TEE, HRT_j is the model integrity verification result, and VT_j is the signature verification result. The process of model deployment is illustrated as described in the following:

(1) Model deployment

(a) *Download the Model*. After the model is stored to the blockchain, TEE retrieves the blockchain through the model contract and gets IM_j.

(b) *Verify Integrity*. TEE first retrieves the model hash $H\text{model}_j$ from IM_j, takes the model hash, and gets the hash $H\text{model}_j'$. TEE compares whether $H\text{model}_j$ and $H\text{model}_j'$ are equal and gets the result HRT_j. Then, it verifies the signature and gets the verification result VT_j.

$$\begin{aligned} H\text{model}_j' &= H1(\text{model}_j), \\ \text{HRT}_j &= \text{if}(H\text{model}_j == H\text{model}_j'), \\ VT_j &= \text{Verify\_}RSA(PK_j, SM_j). \end{aligned} \tag{9}$$

(c) *Upload Integrity Result*. TEE uses the private key SK_T to sign $\{\text{HRT}_j, VT_j\}$ to get the signature ST_j. Then, it invokes the authorization contract to upload the integrity verification result and store $\{ST_j, \text{HRT}_j, VT_j\}$ into the blockchain.

$$ST_j = \text{Sig\_RSA}(SK_T, \{\text{HRT}_j, VT_j\}). \tag{10}$$

(d) *Deploy Model*. The TEE deploys the model after verifying its integrity.

Algorithm 3 shows the process of model deployment to TEE.

5.4. Model Training

5.4.1. Data Verification. After the successful deployment of the model, TEE uses the authorization contract to retrieve the blockchain and obtain the user data $\{PK_i, UDT, DA_i, US_i, HDS_i, DT_i, CM_i\}$ that match the data requirements. Then, we download the data ciphertext DS_i from the cloud server according to the label type of the data required by the model. To ensure the authenticity and

Input: **model**: Data demander node identification MID_j; Public Key PK_j; Private Key SK_j;
Output: Blockchain Transaction IM_j;
(1) Generate models and data requirements
(2) Generate timestamp MT_j
(3) Generate model hash $H\mathrm{model}_j = H1(\mathrm{model}_j)$
(4) Signature $SM_j = \mathrm{Sig\_}RSA(SK_j, H\mathrm{model}_j)$
(5) $IM_j = \{\mathrm{MID}_j, PK_j, MT_j, \mathrm{model}_j, H\mathrm{model}_j, SM_j, RM_j\}$
(6) Call the model contract to upload data IM_j to the blockchain

Algorithm 2: Model storage.

Input: Private Key SK_T;
Output: Verify the result of the hash value HRT_j; The result of verifying the signature VT_j; Signature ST_j;
(1) Call the model contract to download model $\{\mathrm{MID}_j, PK_j, MT_j, \mathrm{model}_j, H\mathrm{model}_j, SM_j, RM_j\}$
(2) Generate model hash $H\mathrm{model}_j\prime = H1(\mathrm{model}_j)$
(3) $\mathrm{HRT}_j = \mathrm{if}(H\mathrm{model}_j == H\mathrm{model}_j\prime)$
(4) Verify signature $VT_j = \mathrm{Verify\_RSA}(PK_j, SM_j)$
(5) Sign off on the validation results $ST_j = \mathrm{Sig\_}RSA(SK_T, \{\mathrm{HRT}_j, VT_j\})$
(6) Deploy model
(7) Call the model contract to upload $\{ST_j, \mathrm{HRT}_j, VT_j\}$ to the blockchain
(8) return $\{ST_j, \mathrm{HRT}_j, VT_j\}$

Algorithm 3: Model deployment.

integrity of the data, the data need to be verified. For this purpose, the data structure $\{ST_i, \mathrm{HRT}_i, VT_i\}$ for data validation is designed, where ST_i is the signature of TEE, HRT_i is the model integrity verification result, and VT_i is the signature verification result. As described below, the process of model deployment is illustrated:

(a) *DownloadTD_i*. TEE invokes the data contract to retrieve the blockchain and download the TD_i that matches the data requirements.

(b) *Download Data CipherDS_i*. We retrieve the cloud server to download the corresponding data cipher DS_i according to the data type required by the model.

(c) TEE first retrieves the data cipher hash HDS_i from TD_i and then hashes the model to get the hash HDS_i'.TEE compares whether HDS_i' is equal to HDS_i and gets the result HRT_i. Then, it verifies the signature and gets the verification result VT_i.

$$\begin{aligned} \mathrm{HDS}_i' &= H1(DS_i), \\ \mathrm{HRT}_i &= \mathrm{if}(\mathrm{HDS}_i == \mathrm{HDS}_i'), \\ VT_i &= \mathrm{Verify\_RSA}(PK_i, US_i). \end{aligned} \tag{11}$$

(d) *Upload Integrity Result*. TEE uses private key SK_T to sign $\{\mathrm{HRT}_i, VT_i\}$ to get signature ST_i, and then invoke authorization contract to upload integrity verification result and store $\{ST_i, \mathrm{HRT}_i, VT_i\}$ into blockchain.

$$ST_i = \mathrm{Sig\_}RSA(SK_T, \{\mathrm{HRT}_i, VT_i\}). \tag{12}$$

Algorithm 4 shows the process of data downloading.

5.4.2. Data Acquisition and Model Training. After obtaining the user data cipher and verifying the data integrity, TEE invokes the authorization contract to request a key. TEE uploads the data integrity verification result and model integrity verification result to the authorization contract. The authorization contract will receive the $r_U g$ and $r_C PK_C$ and send to TEE. TEE receives the $r_U g$ and $r_C PK_C$ to calculate the decryption key. The process is illustrated as described in the following:

(1) Key authorization

(a) *Contract Authorization*. The authorization contract sends $r_C PK_C$ to TEE along with $r_U g$.

(2) Key acquisition

(a) *Retrieve Key*. TEE retrieves the key cipher CM_i from TD_i.

(b) *Decrypt Key*. TEE uses $r_C PK_C$ and $r_U g$ to decrypt the encryption key.

$$\begin{aligned} \mathrm{KEY}_i &= CM_i - r_C PK_C - r_U PK_T \\ &= CM_i - r_C PK_C - r_U SK_T g \\ &= CM_i - r_C PK_C - r_U g SK_T. \end{aligned} \tag{13}$$

(3) Data decryption

(a) *Decrypt the Data Cipher*. TEE calls the decrypt function Decrypt, decrypts the data cipher $\mathrm{DS_i}$ using KEY_i, and gets the data $U\mathrm{DATA}_i$.

$$U\mathrm{DATA}_i = \mathrm{Decrypt}(\mathrm{KEY}_i, DS_i). \tag{14}$$

(4) Model training

Input: Private Key $\mathbf{SK_T}$;
Output: Verify the result of the hash value HRT_i; The result of verifying the signature VT_i; Signature ST_i;
(1) Call the data contract to download data $\{PK_i, U\ DT\ , DA_i, US_i, H\ DS_i, DT_i, CM_i\}$
(2) Download data ciphertext from cloud storage server DS_i
(3) Generate model hash $\mathrm{HDS}_i\prime = H1(DS_i)$
(4) $\mathrm{HRT}_i = \mathrm{if}(\mathrm{HDS}_i == \mathrm{HDS}_i\prime)$
(5) Verify signature $VT_i = \mathrm{Verify\_}RSA(PK_i, US_i)$
(6) Sign off on the validation results $ST_i = \mathrm{Sig\_RSA}(SK_T, \{\mathrm{HRT}_i, VT_i\})$
(7) Call the data contract to upload $\{ST_i, \mathrm{HRT}_i, VT_i\}$ to the blockchain
(8) return $\{ST_i, \mathrm{HRT}_i, VT_i\}$

Algorithm 4: Data verification.

Input: Private Key SK_T; Ciphertext of the key CM_i; Ciphertext of the data $\mathrm{DS_i}$;
Output: Encrypted keys KEY_i; raw data $U\mathrm{DATA}_i$; Training results of the model MRT_j;
(1) Call the Authorization contract to download $\{r_C PK_C, r_U g\}$
(2) Decrypted keys: $\mathrm{KEY}_i = CM_i - r_C PK_C - r_U PK_T$
$= CM_i - r_C PK_C - r_U SK_T g$
$= CM_i - r_C PK_C - r_U g SK_T$
(3) Decrypt data: $U\mathrm{DATA}_i = \mathrm{Decrypt}(\mathrm{KEY}_i, DS_i)$
(4) Training model with data
(5) return $\{\mathrm{KEY}_i, U\mathrm{DATA}_i, \mathrm{MRT}_j\}$

Algorithm 5: Data acquisition and model training.

(a) TEE uses data from multiple users to train the model on the demander of the data and obtains the training result MRT_j after the model is trained.

Algorithm 5 shows the process of decrypting the data and training the model.

5.5. Training Result Acquisition. After the model training is completed, the training results need to be encrypted and stored in the blockchain in order to protect the privacy and security of the model training results. The data demander gets the model ciphertext from the chain and then decrypts it to get the model training results. The process is illustrated as described in the following.

5.5.1. Training Result Storage

(a) *Encrypt Training Results*. TEE uses the public key of the data demander to encrypt the training results and get the ciphertext.

$$\mathrm{CMRT}_j = \mathrm{Encrypt\_}ECC\left(PK_j, \mathrm{MRT}_j\right). \quad (15)$$

(b) *Hash*. TEE takes a hash of the training result ciphertext.

$$\mathrm{HCMRT}_j = H1\left(\mathrm{CMRT}_j\right). \quad (16)$$

(c) *Generate Signature*. To ensure the authenticity of the training results, TEE signs the cryptographic hash of the training results.

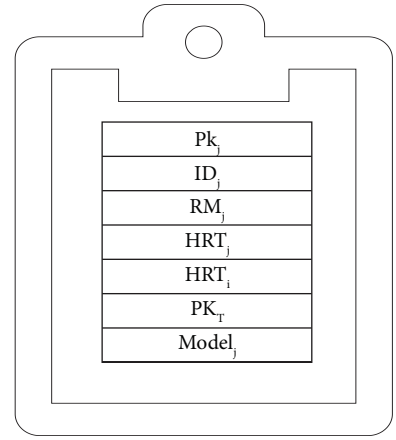


Figure 4: Structure of audit information.

$$\mathrm{SMRT}_j = \mathrm{Sig\_RSA}\left(SK_T, \mathrm{HCMRT}_j\right). \quad (17)$$

(d) *Upload to Blockchain*. TEE invokes the model contract to upload $\left\{\mathrm{CMRT}_j, \mathrm{HCMRT}_j, \mathrm{SMRT}_j\right\}$ to the blockchain.

Algorithm 6 demonstrates this process.

5.5.2. User Audit Information Storage. To ensure that users audit the sharing process of personal medical data at any time and avoid the unauthorized use of data, we design the user audit structure $\left\{\mathrm{MID}_j, RM_j, \mathrm{HRT}_j, \mathrm{HRT}_i, PK_T, PK_j, \mathrm{model}_j\right\}$ to store the audit information and then invoke

Input: Public Key PK_j; Training result of the model MRT_j; Private Key SK_T;
Output: Ciphertext of training result CMRT_j; Hash HCMRT_j; Signature SMRT_j;
(1) Encrypt training result $\mathrm{CMRT}_j = \mathrm{Encrypt\_}ECC(PK_j, \mathrm{MRT}_j)$
(2) Generate hash value $\mathrm{HCMRT}_j = H1(\mathrm{CMRT}_j)$
(3) Generate signature $\mathrm{SMRT}_j = \mathrm{Sig\_RSA}(SK_T, \mathrm{HCMRT}_j)$
(4) Call the model contract to upload $\{\mathrm{CMRT}_j, \mathrm{HCMRT}_j, \mathrm{SMRT}_j\}$ to the blockchain
(5) return $\{\mathrm{CMRT}_j, \mathrm{HCMRT}_j, \mathrm{SMRT}_j\}$

Algorithm 6: Training result storage.

Input: Hash HCMRT_j; Signature SMRT_j; ciphertext CMRT_j; Private Key SK_j;
Output: Training result of the model MRT_j;
(1) Call the Authorization contract to download $\{\mathrm{CMRT}_j, \mathrm{HCMRT}_j, \mathrm{SMRT}_j\}$
(2) Generate hash value $\mathrm{HCMRT}_j' = H1(\mathrm{CMRT}_j)$
(3) Result = If $(\mathrm{HCMRT}_j == \mathrm{HCMRT}_j')$
(4) $\mathrm{Verify\_RSA}(PK_T, \mathrm{SMRT}_j)$
(5) Decrypt result ciphertext $\mathrm{MRT}_j = \mathrm{Decrypt\_}ECC(SK_j, \mathrm{CMRT}_j)$
(6) return MRT_j

Algorithm 7: Decrypt result ciphertext.

the audit contract to store it into the blockchain for user audit. Here, MID_j is the ID of the data demander, RM_j is the data demand, HRT_i is the model integrity verification result, HRT_i is the data verification result, $\mathrm{PK_T}$ is the public key of TEE, PK_j is the public key of the data demander, and model_j is the model of the data demander. The audit information structure is shown in Figure 4.

5.5.3. Training Rezsult Acquisition

(a) *Download the Training Result Ciphertext*. The data demander invokes the model contract to obtain $\{\mathrm{CMRT}_j, \mathrm{HCMRT}_j, \mathrm{SMRT}_j\}$.

(b) *Verify Ciphertext*. We calculate the hash of the training result ciphertext, compare it with the hash downloaded from the chain, and then verify the signature. If the verification passes, the decryption process is carried out, and if the verification does not pass, the feedback is sent to the blockchain.

$$\begin{aligned}\mathrm{HCMRT}_j' &= H1\left(\mathrm{CMRT}_j\right),\\ \mathrm{Result} &= \mathrm{If}\left(\mathrm{HCMRT}_j == \mathrm{HCMRT}_j'\right),\\ &\quad \cdot \mathrm{Verify\_RSA}\left(PK_T, \mathrm{SMRT}_j\right).\end{aligned} \tag{18}$$

(c) *Decrypt Ciphertext*. The data demander decrypts the data using the private key after successful verification.

$$\mathrm{MRT}_j = \mathrm{Decrypt\_}ECC\left(SK_j, \mathrm{CMRT}_j\right). \tag{19}$$

Algorithm 7 demonstrates this process.

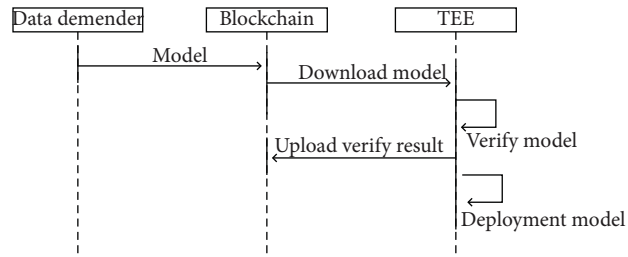


Figure 5: The logical flow of machine learning model deployment.

5.6. Interaction Process Description. The interaction process of the solution consists of ten steps as follows. First, steps 1–4 describe the process of deploying the machine learning model. The logical flowchart is shown in Figure 5. Then, steps 5–8 describe the process of user data storage, sharing, and key authorization. The logic flowchart is shown in Figure 6. The final steps 9-10 describe the process of using user data to train the machine learning model and the process of transmitting the model learning results to the data demander. The logical flowchart is shown in Figure 7:

Step 1: the data demander uploads the machine learning model and data demand to the block company.

Step 2: the data demander invokes the contract to transfer the machine learning model to the TEE.

Step 3: the TEE verifies the integrity of the machine learning model.

Step 4: the TEE deploymodel.

Step 5: the user encrypts the medical data and stores it in the cloud server. Then, the information such as data cryptographic hash, ciphertext of the key, and storage index are uploaded to the blockchain. Then, the user invokes the authorization contract to upload the key.

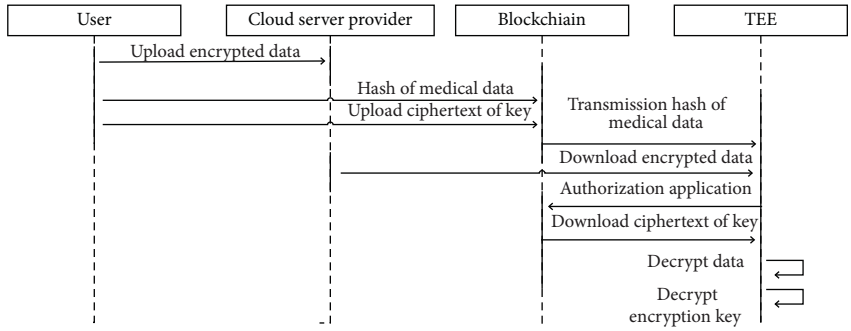


Figure 6: The logical flow of user medical data sharing and key transfer.

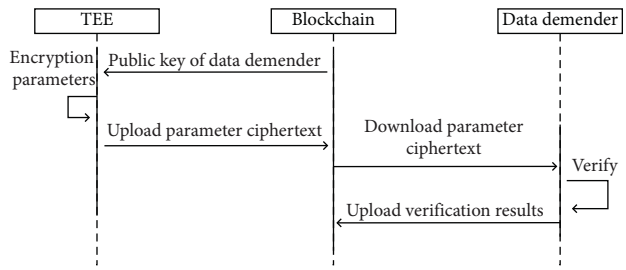


Figure 7: The logical flow of model training and return of the training results.

Step 6: the TEE downloads the user medical data and verifies the integrity of the data.

Step 7: the TEE calls the authorization contract to obtain the key.

Step 8: the TEE decrypts the key ciphertext and then encrypts the data ciphertext to obtain the user data.

Step 9: the TEE uses the user's data to train the model and then encrypts the training results.

Step 10: the TEE calls the contract to upload the encrypted training results to the blockchain.

Step 11: the data demander downloads the training result ciphertext, decrypts it, and obtains the training result.

6. Security and Functional Analysis

In this section, security analysis and functional analysis of the proposed scheme are performed in order to verify that the previously mentioned design objectives are met.

6.1. Security Analysis

6.1.1. Secure Storage and Sharing. In most cases, the storage server is trusted, but sometimes the storage server gets curious about the data and looks at the user's personal data. Therefore, in this scheme, the user data are stored in the storage server in an encrypted state, and the user data cannot be viewed without the key. At the same time, the hash value of data ciphertext is stored in the blockchain to verify the integrity of data, and this mechanism effectively ensures data storage security. During the sharing process, user data are in an encrypted state. Only TEE can use the private key decryption to obtain the encryption key. TEE ensures that the internal computation is hidden, and the internal data cannot be accessed from outside. Therefore, this solution can also ensure the security of data sharing.

6.1.2. Shared Data Can Be Regulated. In this solution, first, users can view their personal medical data at any time. Second, information such as the results of data integrity verification, the identity of the medical research institution, and the models to be trained generated during the sharing process are uploaded to the blockchain. All these information can be viewed by users at any time as a way to regulate personal data and thus prevent the illegal use of personal medical data.

6.1.3. Efficiency. First, in this scenario, we use DES symmetric key to encrypt the user's medical data and upload the data ciphertext to the storage device. The hash of the data ciphertext is then uploaded to the blockchain for storage. This reduces the storage burden of the blockchain and improves the storage efficiency of the system. Second, to achieve efficient machine learning model training on the blockchain, we introduce a combination of on-chain and off-chain approaches. On-chain contracts perform low-complexity operations such as data provisioning and integrity verification, while off-chain TEE performs high-complexity calculations such as data encryption and decryption hash calculation, signing, signature verification, and model training. In this way, the computational burden of the blockchain is significantly reduced, and the efficiency of the whole system is improved.

6.1.4. Dedicated to Medical Data. In this solution, medical data are only used to train machine learning models of medical research institutions. No entity can access the user's medical data and let alone perform other operations on the user's medical data, to ensure the exclusive use and non-misuse of the user's medical data.

6.1.5. User Data Ownership. First, the user's medical data are not really shared but used for training models, and the medical research organization does not really have access to

Table 2: The comparison of functionality and security with current solutions.

Security properties	Zou et al. [19]	Miao et al. [33]	Chen et al. [20]	This article
Safe storage and sharing	√	√	√	√
Blockchain based	√	√	√	√
User auditing	×	√	×	√
Environment security	×	×	×	√
Protecting user ownership	×	×	×	√
Minimized data usage	×	×	×	√

the user's data. Second, the user's data are trained in the TEE, and no entity can be aware of the computational process inside, and much less access the data in the TEE. In this way, the ownership of user data can be protected.

6.1.6. Computing Environment Security. This scheme uses Intel SGX for data decryption and model training. SGX aims to safeguard the confidentiality and integrity of users' critical code and data from malware by making hardware security mandatory and not to rely on firmware and software's security status. SGX's trusted computing base (TCB) contains only hardware, avoiding the pitfalls of software-based TCBs that have their own software security vulnerabilities and threats. In addition, SGX guarantees a TEE at runtime so that malicious code cannot access and tamper with the protected content of other programs at runtime, further enhancing the system's security. SGX's robust, trusted, flexible security features and hardware scalable performance guarantees provide a secure computing environment for this solution.

6.1.7. Resilience against Other Attacks

(1) Replay Attack. The scheme is effective against replay attacks because all transactions in the system contain timestamps and digital signatures. Moreover, since all transactions in the blockchain are transparent, any user can extract the time when the transaction was generated. This way, if a malicious user tries to duplicate a transaction request using a transaction written on the blockchain, then during the validation phase of the transaction, the relevant validation node will detect the time discrepancy and discard the transaction.

(2) Impersonation Attack. In this scenario, TEE provides proof for the issued data, for example, the integrity verification result of the data, the integrity verification result of the model, the public key of the TEE, and the completed model of the training. This is to prevent the illegal elements from impersonating TEE to cheat the user's data encryption key. On the other hand, the user's data will also contain the user's signature to ensure the authenticity of the user's data. This prevents illegals from using malicious data to influence the model's training.

(3) Tampering Attacks. In the process of medical data sharing, there may be cases where users tamper with blockchain information or transaction information, such as changing the owner of published medical data to their own account, thus tampering with the ownership of medical data. The solution is based on Ethernet deployment, and the authenticated nodes generate the blocks. Here, the authenticated nodes need to complete a mandatory authentication process to get the right to generate new blocks. Therefore, blocks are packed by trusted certified nodes, and malicious nodes cannot learn the private keys of trusted certified nodes, so they cannot forge the identity of certified nodes to pack blocks and thus cannot modify block information to forge signatures. Since it is difficult for malicious nodes to tamper with the data on the blockchain, and we store the cryptographic hash of medical data and share records on the chain, this ensures the accuracy and authenticity of the records.

6.2. Function Analysis. In Table 2, we compare our scheme with existing schemes. As can be seen from the table, all these schemes are based on blockchain for data sharing. Among them, Zou et al. [19], Miao et al. [33], and Chen et al. [20] designed privacy protection for the medical data sharing process. However, our solution meets the practical needs of dedicated data dedication, secure data handling, data regulation, and data ownership.

7. Program Design and Evaluation

In this section, we analyze the effectiveness of the proposed scheme through experiments. We have conducted a simulation experiment, which is divided into four parts. First of all, we build the Ethernet blockchain on the Ubuntu 20.0 virtual machine and write an intelligent contract using solidity. Then, we build Intel SGX in Intel (R) Core (TM) i7-9750H CPU @ 2.60 GHz, 16gb RAM, Microsoft Windows 10 operating system, implement trusted execution environment (TEE), and redesign encryption and decryption algorithm, hash algorithm, and signature algorithm in SGX. We realize the functions of data decryption, machine learning, hash generation, signature, and learning result encryption in the security zone. In order to compare the impact of SGX on the efficiency of the whole shared system, we also implement the above functions in a non-SGX environment and compare the computing time overhead in the two environments. Finally, by adjusting different difficulties, we test the appropriate block time and test the throughput of the system.

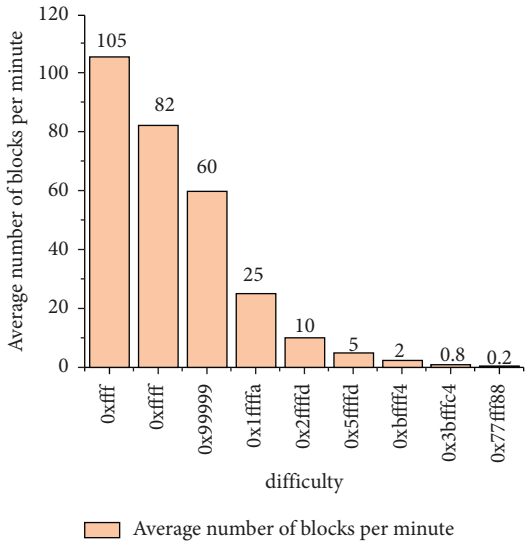


Figure 8: The average number of blocks per minute.

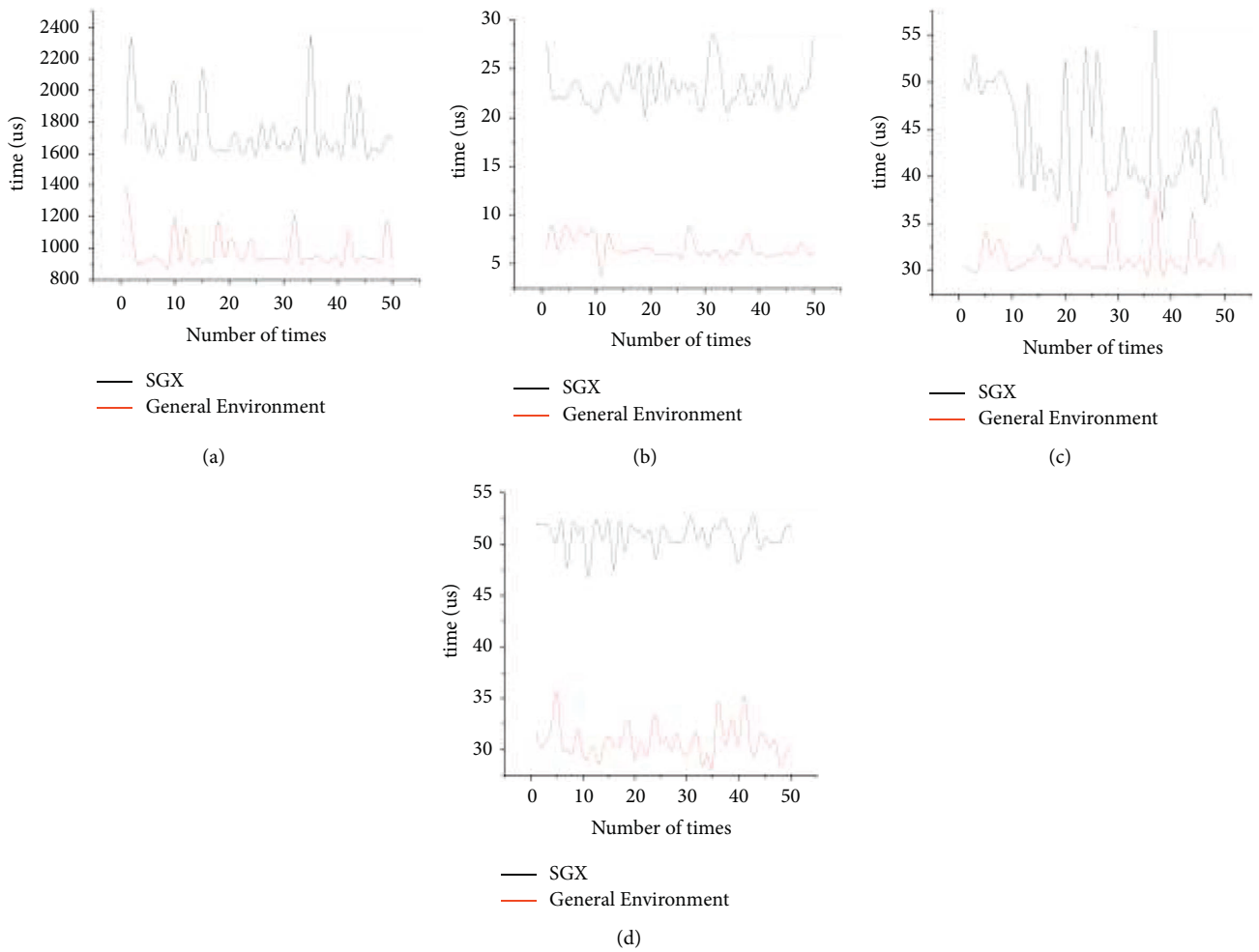


Figure 9: Comparison between SGX and general environment. (a) ECC key pair generation time comparison, (b) comparison of the time overhead of calculating hash values, (c) comparison of the time overhead of generating signatures, and (d) comparison of the time overhead of the whole system.

Table 3: System time overhead for different file sizes.

Data size (kB)	1	2	3	4	5	6
Minimum of NOSGX (us)	58216	170421	276373	460588	544606	705014
Average of NOSGX (us)	67868	179441	307853	524873	600304	725899
Maximum of NOSGX (us)	78341	194061	357842	625825	672885	795646
Minimum of SGX (us)	230948	416418	474531	858997	1028615	1238720
Average of SGX (us)	278031	441992	507291	887812	1181386	1366159
Maximum of SGX (us)	315733	478269	548383	932949	1235315	1456512

7.1. Ethereum Blockchain Building. The blockchain built in this system is based on the Geth client, version 1.9.25-stable-e7872729, which is based on Ether and uses POW as the consensus algorithm. There are 4 mining nodes built in the federated chain network, and the 4 nodes take turns to start mining. We modify the difficulty value in the genesis file to test the block generation time overhead at different difficulty levels. As shown in Figure 8, the vertical coordinate represents the difficulty and the horizontal coordinate represents the number of blocks generated per minute, and we set the scheme out of blocks to 60 blocks per minute.

7.2. Building a Trusted Execution Environment. Based on the excellent performance of Intel SGX, we use SGX as the trusted execution environment. The design of the Enclave of the security zone needs to consider both function and implementation. The analysis of the scheme shows that Enclave needs to have the following functions: (1) generating asymmetric key pairs inside Enclave; (2) exporting public keys outside Enclave; (3) decrypting key ciphertexts and data; (4) training models using data; (5) encrypting training results; and (6) signing, verifying signatures, and computing hash values.

Since SGX's internal functions are not convenient, we rebuilt the algorithm in Enclave for the above functions. We rebuilt ECC asymmetric encryption and symmetric encryption and decryption for higher security and smaller key size. We use the RSA signature for signature and verification signature. For the use of the hash function, we choose the SHA-256 algorithm.

7.3. SGX Performance Evaluation. To test the impact on the overall sharing system efficiency after using SGX, we designed SGX-based data sharing and non-SGX data sharing for comparison. This is shown in Figure 9. We use 3 kB data to compare the ECC key pair generation time comparison, data hash generation time comparison, signature time comparison, and the total time comparison of the whole sharing system between the SGX environment and the non-SGX environment. Figure 9(a) shows the key pair generation time comparison, which takes a little more time in the SGX environment than in the normal environment. However, this time overhead is not significant, and on average, the SGX environment takes 757.307 us more time overhead. We also tested the time overhead of generating data hashes, signing, and verifying signatures in both environments. The additional time overhead for generating hashes, signatures, and verifying signatures in the SGX environment is

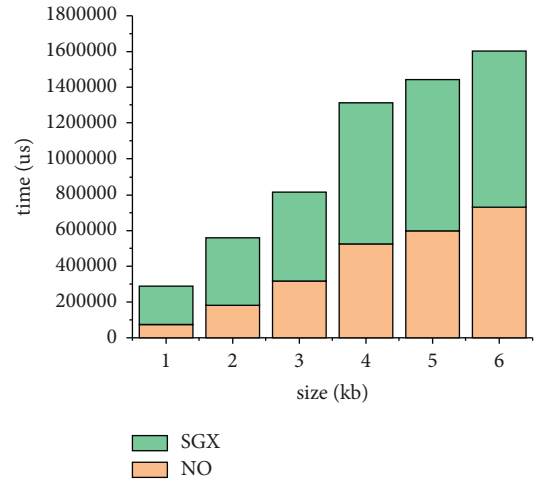


Figure 10: Time comparison for different data sizes.

16.566 us, 12.464 us, and 6.916 us, respectively. There is a small additional time overhead for the above calculations when using SGX, but it is still a microsecond overhead. It does not have a significant impact on the overall system. Figure 9(d) shows the time overhead of the whole system in the SGX environment compared to the non-SGX environment. Overall, the SGX environment still sacrifices some efficiency, but in terms of average time, the SGX environment has an additional 19.9443 ms overhead.

To further test the impact of SGX on the system, we performed test simulations using 1 to 6 kB of data, as shown in Table 3. Since the excess time loss of SGX is due to the data going in and out of Enclave, the time increase with SGX is lower than the time increase without SGX as the data size increases, as shown in Figure 10.

7.4. Blockchain Throughput. In this section, we separately calculate the throughput of various transactions. Since we are using the Ethernet blockchain, the block gasLimit of Ethernet determines the number of transactions that can be packed in a block. The current block gasLimit of the Ethernet blockchain we built is 12,000,000 gas. After testing, we get the gasLimit for user-submitted personal medical data transactions to be 62,060 gas. Since we set the block-out speed to 60 minutes, the data processing per second (TPS) for personal user data is 193. Similarly, testing the gasLimit for data demander-submitted transactions is to be 82,355 and the TPS for deployment model transactions is to be 145. The gasLimit of integrity verification transaction is 38444

gas, and the TPS is 312. The gasLimit of SGX uploading machine learning model training result ciphertext and its hash value transaction is 79,928 gas, and the TPS is 150.

8. Conclusions

This paper proposes a user-centric medical data sharing scheme for privacy-preserving machine learning, which implements data encryption storage, blockchain-based data resource distribution, data authorization, and machine learning model training. We also design an auditing mechanism to assist users in auditing the data sharing process. Compared with existing schemes, our proposed scheme ensures the privacy and security of users' data and safeguards the ownership of users' data and achieves the dedicated use and nonmisuse of data. Finally, the functionality of this solution is implemented through simulation experiments, and the experimental results prove the feasibility and effectiveness of the solution. Analyzing the impact of the TEE on the overall system performance demonstrates that the privacy and security of data and the user's data ownership are guaranteed without significant performance degradation. In future work, we intend to reduce the communication overhead of users and increase the throughput of the system.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Shandong Provincial Key Research and Development Program (2021CXGC010107 and 2020CXGC010107) and the National Natural Science Foundation of China (62102209).

References

[1] T. J. Hirschauer, H. Adeli, and J. A. Buford, "Computer-aided diagnosis of Parkinson's disease using enhanced probabilistic neural network," *Plenum Press*, vol. 101, 2015.

[2] H. Jin, Y. Luo, P. Li, and J. Mathew, "A review of secure and privacy-preserving medical data sharing," *IEEE Access*, vol. 7, no. 99, Article ID 61656, 2019.

[3] D. W. Bates, S. Saria, L. Ohno-Machado, A. Shah, and G. Escobar, "Big data in health care: using analytics to identify and manage high-risk and high-cost patients," *Health Affairs*, vol. 33, no. 7, pp. 1123–1131, 2014.

[4] L. Campanile, M. Iacono, F. Marulli, and M. Mastroianni, "Designing a GDPR compliant blockchain-based IoV distributed information tracking system," *Information Processing & Management*, vol. 58, no. 3, Article ID 102511, 2021.

[5] M. Pradnya Patil, "Sangeetha, vidhyacharan bhaskar. Blockchain for IoT access control, security and privacy: a review," *Wireless Personal Communications*, vol. 35, 2020.

[6] P. C. K. Hung and Y. Zheng, "Privacy access control model for aggregated e-health services," in *Proceedings of the EDOC Conference Workshop*, Eleventh International IEEE, Annapolis, MD, USA, October 2007.

[7] Pavithran Deepa and N. Al-Karaki Jamal, "Shaalan khaled. Edge-based blockchain architecture for event-driven IoT using hierarchical identity based encryption," *Information Processing & Management*, vol. 58, no. 3, 2021.

[8] P. Shabisha, A. Braeken, A. Touhafi, and K. Steenhaut, "Elliptic curve qu-vanstone based signcryption schemes with proxy Re-encryption for secure cloud data storage," vol. 21, 2019.

[9] L. Campanile, M. Iacono, F. Marulli, and M. Mastroianni, "Designing a GDPR compliant blockchain-based IoV distributed information tracking system," *Information Processing & Management*, vol. 58, no. 3, Article ID 102511, 2021.

[10] A. Eb, A. Fc, and B. Cg, "BlockHealth: blockchain-based secure and peer-to-peer health information sharing with data protection and right to be forgotten," *ICT Express*, vol. 34, 2021.

[11] W. Shen, T. Hu, C. Zhang, and S. Ma, "Secure sharing of big digital twin data for smart manufacturing based on blockchain," *Journal of Manufacturing Systems*, vol. 61, pp. 338–350, 2021.

[12] Y. Chen, Z. Lu, H. Xiong, and W. Xu, "Privacy-preserving data aggregation protocol for fog computing-assisted vehicle-to-infrastructure scenario," *Security and Communication Networks*, vol. 2018, pp. 1–14, 2018.

[13] L. Yang, W. Zou, and J. Wang, "EdgeShare: a blockchain-based edge data-sharing framework for industrial Internet of things," vol. 43, 2021.

[14] F. Chen, J. Huang, C. Wang et al., "Data access control based on blockchain in medical cyber physical systems," *Security and Communication Networks*, vol. 34, pp. 1–14, 2021.

[15] K. Gu, W. Jia, G. Wang, and S. Wen, "Efficient and secure attribute-based signature for monotone predicates," *Acta Informatica*, vol. 54, no. 5, pp. 521–541, 2017.

[16] H.. Wang and Y.. Song, "Secure cloud-based EHR system using attribute-based cryptosystem and blockchain," *Journal of Medical Systems*, vol. 42, no. 8, pp. 152–164, 2018.

[17] R.. Guo, H.. Shi, Q.. Zhao, and D. Zheng, "Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems," *IEEE Access*, vol. 6, pp. 11676–11686, 2018.

[18] J. S. Lee, C. J. Chew, and J. Y. Liu, "Medical blockchain: data sharing and privacy preserving of EHR based on smart contract,"vol. 74.

[19] R. Zou, X. Lv, and J. Zhao, "SPChain: blockchain-based medical data sharing and privacy-preserving eHealth system," *Information Processing & Management*, vol. 58, no. 4, Article ID 102604, 2021.

[20] Z. Chen, W. Xu, B. Wang, and H. Yu, "A blockchain-based preserving and sharing system for medical data privacy," *Future Generation Computer Systems*, vol. 124, pp. 338–350, 2021.

[21] A. Kosba, A. Miller, and E. Shi, "Hawk: the blockchain model of cryptography and privacy-preserving smart contracts," in *Proceedings of the 2016 IEEE Symposium on Security and Privacy*, pp. 839–858, IEEE Press, Los Alamitos, CA, USA, 2016.

[22] P. Golle, K. Leytonbrown, and I. Mironov, "Incentives for sharing in peer-to-peer networks," *Lecture Notes in Computer Science*, vol. 49, pp. 75–87, 2001.

[23] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2020.

[24] X. Huang, Y. Lu, K. Zhang, S. Maharjan, and Y. Zhang, "Blockchain empowered asynchronous federated learning for secure data sharing in Internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4298–4311, 2020.

[25] L. Zhou, L. Wang, and Y. Sun, "MIStore: a blockchain-based medical insurance storage system," *Journal of Medical Systems*, vol. 42, no. 8, pp. 149–165, 2018.

[26] A. A. Shrier, A. Chang, and N. Diakun-Thibault, *Blockchain and Health IT: Algorithms*, Privacy, and Data, 2016.

[27] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, "Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control," *Journal of Medical Systems*, vol. 40, no. 10, p. 218, 2016.

[28] S. Truex and N. Baracaldo, "A hybrid approach to privacy-preserving federated learning," https://arxiv.org/abs/1812.03224.

[29] H. Li Hao, G. Xu, S. Liu, and H. Yang, "Towards efficient and privacy-preserving federated deep learning," in *Proceedings of the ICC 2019 - 2019 IEEE International Conference on Communications*, pp. 1–6, ICC), Shanghai, China, May 2019.

[30] C. Priebe, K. Vaswani, and M. Costa, "EnclaveDB: A Secure Database Using SGX," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP)*, IEEE, San Francisco, CA, USA, May 2018.

[31] R. Kunkel, D. L. Quoc, and F. Gregor, "TensorSCONE: a secure TensorFlow framework using intel SGX," vol. 42, 2019.

[32] M. Abadi, P. Barham, and J. Chen, "TensorFlow: a system for large-scale machine learning," *USENIX Association*, vol. 12, 2016.

[33] Q. Miao, H. Lin, J. Hu, and X. Wang, "An intelligent and privacy-enhanced data sharing strategy for blockchain-empowered Internet of Things," *Digital Communications and Networks*, vol. 88, 2022.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 9032828, 12 pages
https://doi.org/10.1155/2022/9032828

Research Article

LPS-ORAM: Perfectly Secure Oblivious RAM with Logarithmic Bandwidth Overhead

Yunping Gong,[1,2] **Fei Gao**,[1] **Wenmin Li**,[1] **Hua Zhang**,[1] **Zhengping Jin**,[1] **and Qiaoyan Wen**[1]

[1]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China*

Correspondence should be addressed to Fei Gao; gaof@bupt.edu.cn

Received 9 May 2022; Accepted 11 July 2022; Published 12 August 2022

Academic Editor: Wenbo Shi

Oblivious Random Access Machine (ORAM) is a cryptographic tool used to obfuscate the access pattern. In this paper, we focus on perfect security of ORAM. A perfectly secure ORAM is an ORAM that can resist against an adversary with unlimited computing power, and the failure probability of ORAM is zero rather than negligible. Since all existing perfectly secure single-server ORAM solutions require at least sublinear worst-case bandwidth overhead, we pose a natural and open question: *can we construct a perfectly secure single-server ORAM with logarithmic worst-case bandwidth overhead?* In this paper, we propose the first tree-based perfectly secure ORAM scheme, named LPS-ORAM. To meet the requirements of perfectly secure ORAM, two techniques are presented. One technique is dynamic remapping associated with a mutable scope, and the other is dynamically balanced eviction. Their combined effect allows the root bucket to never fill up while maintaining its statistical security in tree-based ORAM. In the worst case, our solution achieves logarithmic bandwidth overhead. Therefore, our solution answers the open question in the affirmative. In terms of overhead for temporary storage on the client side, compared with the latest perfectly secure ORAM solution, our solution is reduced from sublinear to logarithmic, and even if the server storage overhead scales lightly, it is still at the same level of quantity as the state of the art. Finally, the evaluation results show that our LPS-ORAM has a significant advantage in terms of bandwidth overhead and overhead for temporary storage on the client side.

1. Introduction

Thanks to the interconnectivity of a large number of mobile smart devices in the Internet of Things, huge amounts of data are being generated. To save money on data storage, consumers choose to store their private data on the cloud server. In order to guarantee confidentiality of the private data, consumers need to encrypt the data before uploading them to the server. But using encryption alone, the data access pattern might still be broken, and the opponent can deduce some sensitive information from this [1–3]. Oblivious Random Access Machine (ORAM) [4–6] was presented decades ago to mitigate this security issue. Nevertheless, these early ORAM solutions are not viewed favorably by most researchers due to the poor efficiency. Since then, a large number of ORAM solutions [7–26] have been put forward to make the efficiency better. Among them, Path ORAM [13] algorithm is very simple and very efficient in logarithmic bandwidth overhead, so it is excellent.

Goldreich and Ostrovsky [6] proposed the first lower bound of $O(\log N)$ bandwidth overhead, where N is the number of real blocks outsourced to the cloud server. The lower bound holds if the client storage size is $O(1)$-block and the ORAM is in a balls-and-bins model with a uniform block size of $O(\log N)$-bit. Boyle and Naor [27] further stated that this lower bound only holds for statistically secure ORAM that is in a "balls-and-bins" manner. Larsen and Nielsen [28] then stated that the lower bound of $O(\log N)$ bandwidth

overhead for computationally secure ORAM still holds even it is not in a "balls-and-bins" manner. However, for perfectly secure ORAM, it is not clear whether the lower bound holds.

The theme of a lot of works [13, 27–35] associated with ORAM is to find a lower bound of bandwidth overhead in different settings. Typically, there are two ways to measure the bandwidth overhead. One approach is worst-case overhead, which is the maximum overhead of completing a single request in a long list of requests. The other is amortized-case overhead, which is the average overhead of each request in a long list of requests. The most famous lower bound in the worst case was proposed by Stefanov et al. [13], who presented a tree-based statistically secure ORAM with $O(\log N)$ bandwidth overhead when the block size is set to $O(\log^2 N)$-bit. Thus, this only partly matches the lower bound of Goldreich and Ostrovsky [6] because the lower bound only holds when the block size is $O(\log N)$-bit. The most famous lower bound in the amortized case was proposed by Asharov et al. [31], who presented a computationally secure ORAM with $O(\log N)$ bandwidth overhead, which completely matches the lower bound of Larsen and Nielsen [28].

The first perfectly secure ORAM was designed by Damgard et al. [36], which has the amortized-case bandwidth overhead of $O(\log^3 N)$-block and the server storage overhead of $O(N * \log N)$-block. This was further improved by Chan et al. [37], who presented a perfectly secure ORAM with a lower server storage overhead of $O(N)$-block and the same amortized-case bandwidth overhead. Recently, another perfectly secure ORAM, Lookahead ORAM, was proposed by Raskin and Simkin [38], which has the worst-case bandwidth overhead of $O(\sqrt{N})$-block and the server storage overhead of $O(N)$-block. This is the first perfectly secure single-server ORAM with sublinear worst-case bandwidth overhead. Since all existing perfectly secure single-server ORAM solutions require at least sublinear worst-case bandwidth overhead, we pose a natural and open question.

Can we construct a perfectly secure single-server ORAM with logarithmic worst-case bandwidth overhead?

This is an important academic question because it facilitates the process of reaching the lower bound for perfectly secure single-server ORAM. Whether this open question can be resolved is a necessary step in the development of ORAM research. The importance of perfectly secure ORAM was elaborated by Chan et al. [37]. In addition to the three points listed, we have added another point. To the best of our knowledge, in the standard model without server computing, the lower bounds of both computationally secure ORAM and statistically secure ORAM are logarithmic, which are Goldreich–Ostrovsky lower bound [6] and Larsen and Nielsen lower bound [28], respectively. However, so far, the lower bound of perfectly secure ORAM has not yet emerged. Therefore, it is significant to keep approaching the lower bound by constructing a perfectly secure ORAM solution with better bandwidth overhead.

1.1. Our Contribution. In this paper, we propose a new perfectly secure ORAM solution, called LPS-ORAM, which is designed to resolve the above open question. The main contributions of our paper are summarized as follows:

(i) Design of LPS-ORAM construction. We propose the detailed design of the first tree-based perfectly secure ORAM construction. The proposed techniques can be applied to implement other perfectly secure tree-based ORAM solutions.

(ii) Simplicity and logarithmic worst-case bandwidth overhead. Our scheme has an extremely simple algorithm that makes it practical to implement, and it gains logarithmic bandwidth overhead in the worst case.

(iii) Small overhead for temporary storage on the client side. Our solution achieves logarithmic overhead for temporary storage on the client side, instead of the previous sublinear. Thus, our solution can be applied to small smart devices with limited client storage in the Internet of Things.

1.2. An Overview of Our Techniques. To the best of our knowledge, if only the requested block is remapped after each access in tree-based ORAM, the root bucket in the ORAM tree will be full sooner or later because if the path corresponding to the remapped leaf label and the eviction path are exactly at two branches of the binary tree, the requested block will have to be evicted into the root bucket. The detailed reason is as follows. It is assumed that the new remapped leaf label of the requested block and the eviction path at that time are exactly at two branches of the binary tree. This is the worst case. Even if the greedy strategy is applied to the eviction process, the requested blocks are continuously allocated to the root bucket, causing the root bucket to accumulate until it is full. To ensure perfect security of ORAM, we need to achieve the goal of allowing the root bucket to never fill up while maintaining its statistical security in tree-based ORAM. As a result, it is not feasible that only requested block is remapped after each access in tree-based ORAM. Thus, dynamic remapping associated with a mutable scope is proposed.

1.2.1. Dynamic Remapping Associated with a Mutable Scope. In tree-based ORAM, to ensure the obliviousness property of ORAM, the new remapped leaf label of the requested block is random and uniform after each access. However, each remaining real block retrieved from the path is remapped to a new leaf label that belongs to the scope from the corresponding leaf label of the eviction path to the original leaf label of the block. This is a dynamic remapping associated with a mutable scope. The scope is mutable because depending on the leaf label of the real block, the real block may be written back into the bucket closer to the leaf bucket, but which bucket in the path the real block is located in is a secret for the honest-but-curious server. For example, assuming that the eviction path at that time is labeled by 3, the original leaf label of a remaining real block is 5, and then the scope at that time is [3, 5]. In our ORAM solution, if the block cannot be directly located to a bucket lower than the original bucket according to the original leaf label, then the new remapped leaf label needs to ensure that it places the

bucket in which the block is located closer to the leaf bucket than the original leaf label.

1.2.2. Dynamically Balanced Eviction. In our solution, the goal is allowing the root bucket to never fill up while maintaining its statistical security in tree-based ORAM. Thus, what kind of eviction should be combined with the above dynamic remapping to implement the goal? The dynamically balanced eviction is proposed at this time. During the eviction, the requested block is first arranged to the deepest bucket of the path according to the new remapped leaf label, which is closest to the leaf bucket. Because the new remapped leaf label of the requested block is random and uniform after each access, according to dynamic remapping with a mutable scope, if one path is accessed multiple times, almost all of the real blocks in the path will be squeezed to the buckets near the leaf bucket. This information is harmful because it is inferred easily by the honest-but-curious server. Thus, the dynamically balanced eviction is utilized to avoid the above harmful information. Whether a real block in the path needs to be located in a bucket lower than the original bucket depends on whether the root bucket is empty. If the root bucket is not empty, the above dynamic remapping needs to be executed. Otherwise, it cannot be executed. Thus, the proposed eviction is dynamically balanced.

1.3. Other Related Work. A great deal of work has contributed to implementing perfectly secure ORAM. In the rest of this section, we provide only a high-level overview of solutions that are directly relevant to our work.

In order to reach the lower bound of $O(\log N)$, a large number of solutions have been proposed in the client-server environment. Mayberry et al. [39] proposed a solution with server-side computations, called Path-PIR, in order to obtain a actually very small, but still poly-logarithmic bandwidth overhead. Apon et al. [40] formally defined the primitive for verifiable oblivious storage by allowing server-side computations to be generated from the ORAM primitive and by providing a solution with constant bandwidth overhead, but it is based on fully homomorphic encryption (FHE), so it shows that $O(\log N)$ lower bound has been broken in their setting with FHE. Another solution, called Onion ORAM [41], is proposed that also breaks the $O(\log N)$ lower bound, but it relies on additively homomorphic encryption (AHE). However, in this work, we will focus on the client-server setting without server-side computations.

Demertzis et al. [42] proposed a computationally secure ORAM solution with worst-case bandwidth overhead of $O(N^{1/3})$ and perfect correctness. Perfect correctness means that the ORAM solution fails with the probability of 0. Subsequently, several works cite this and claim that their solution is perfectly secure. However, according to their definition of perfectly secure ORAM, Raskin and Simkin [38] stated that this is not correct and this claim is not made by the authors of that paper either.

1.4. Organization. The rest of this paper is organized as follows. Section 2 introduces the background knowledge including the definition of perfectly secure ORAM and an overview of Path ORAM. Section 3 provides the details of our LPS-ORAM solution. Section 4 gives the performance analysis in terms of bandwidth overhead, storage overhead, and further optimization. A detailed evaluation is introduced in Section 5. Finally, the conclusion is provided in Section 6.

2. Preliminaries

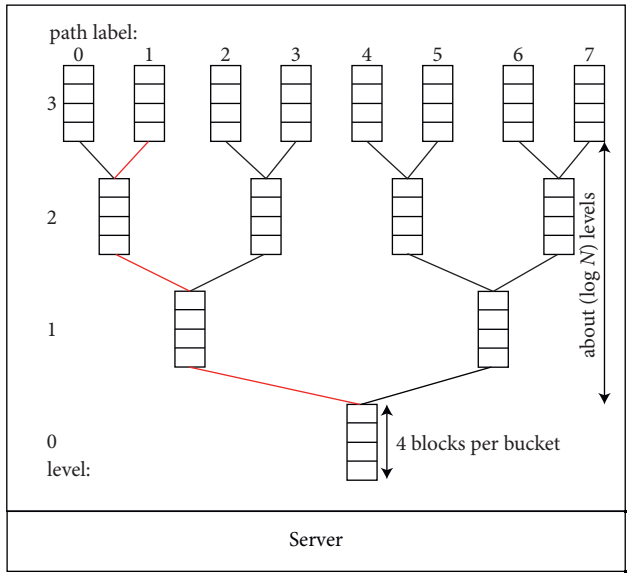
2.1. Security Model. In the security model of ORAM, it is assumed that there is an honest-but-curious server and a trusted client. It requires that for any two requests sequences with the same length, the corresponding access pattern should be indistinguishable. Note that all blocks are encrypted by the client before they are uploaded to the server. The following security definition of perfectly secure ORAM is taken from Raskin and Simkin [38].

Definition 1. (security definition of perfectly secure ORAM). Let $\overrightarrow{U} = (Y_1, Y_2, Y_3, \ldots)$ indicate a request sequence of ORAM. In $\overrightarrow{U}$, Y_i is an access of Read ($\mathrm{ID_i}$) or Write (ID_i, Data_i^*), where $\mathrm{ID_i}$ means the unique block identifier and Data_i^* refer to the new content of block ID_i to be written. It is noted that each real block has a unique identifier. Let DAP ($\overrightarrow{U}$) represent the data access pattern when $\overrightarrow{U}$ is the input of the ORAM algorithm. In reality, the data access pattern is viewed as a distribution. The ORAM solution is statistically/computationally secure for the honest-but-curious server, if and only if DAP ($\overrightarrow{U}$) and DAP ($\overrightarrow{V}$) are statistically/computationally indistinguishable for any two ORAM request sequences $\overrightarrow{U}$ and $\overrightarrow{V}$ with the same length. The ORAM solution is perfectly correct if and only if it returns on input $\overrightarrow{U}$ that is consistent with $\overrightarrow{U}$ with probability 1. We call an ORAM perfectly secure if and only if the ORAM solution can resist against an adversary with unlimited computing power and is perfectly correct at the same time.

According to the above definition, the ORAM is perfectly secure if an ORAM is statistically secure and has a failure probability of 0 at the same time.

2.2. An Overview of Path ORAM. We provide a simple overview of Path ORAM (see [13] for more details). As described in Figure 1, the Path ORAM solution consists of two parts, one is the server storage, and the other is the client storage. The server storage is a complete binary tree with about log N-level. The red line is a target path that the requested block is stored, which is from the remapped leaf label of the position map (PosMap) on the client.

In the complete binary tree, each node is a bucket that can accommodate at most Z-block where Z is a constant. Z-block contains some real blocks, and the rest of the space is populated with virtual blocks. The difference between a real block and a virtual block is that the content of the virtual block is a random string, while the content of the real block consists of real data. Each path in the complete binary tree is



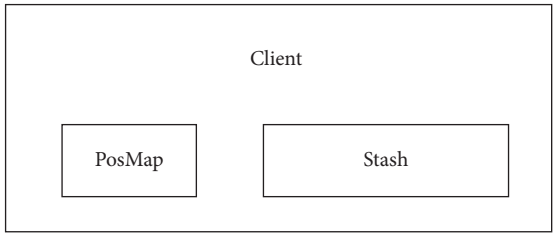


Figure 1: The structure of Path ORAM solution [13].

a set of buckets from the root bucket to a leaf bucket. After each access, every requested block is remapped to a random and uniform leaf label, which means that the requested block either resides somewhere on the path numbered by the leaf label or in stash on the client side. In Path ORAM, to execute an ORAM request, the PosMap is queried first by the client, which is a list table on the client side that tracks the path to which each real block is currently remapped, and then about $(Z * \log N)$ blocks on that path are retrieved to the local stash. Subsequently, the requested block is remapped to a new random and uniform leaf label and the PosMap is updated accordingly. Finally, the eviction procedure is executed, the same path is populated with some real blocks, and the rest of the space is populated with virtual blocks. The various symbols and their meanings are listed in Table 1.

The bandwidth overhead of Path ORAM is about $2Z * \log N$ because a path is fetched and then it is written back into the complete binary tree for each ORAM request. To make Path ORAM fail with a negligible probability in N, the value of Z must be at least 4 in reality or 5 in theory.

3. LPS-ORAM Solution

In this section, we present an extremely simple tree-based perfectly secure ORAM protocol. As far as we know, Path ORAM has an extremely simple algorithm and efficient efficiency of $O(\log N)$-block bandwidth overhead when the block size is set to $O(\log^2 N)$-bit. In our design, our perfectly secure tree-based ORAM, called LPS-ORAM, will inherit the benefits of Path ORAM. In addition, our works focus on perfect security of ORAM, that is, we are committed to achieving the goal of allowing the root bucket to never fill up while maintaining its statistical security in tree-based ORAM. The various symbols used in this solution and their meanings are also listed in Table 1.

Table 1: Symbols and meanings.

Symbol	Meaning
N	The number of real blocks in total
L	The height of the full binary tree
Z	The bucket size in blocks
B	The block size in bits
p	Path p refers to the set from the root bucket to leaf bucket labeled p
$P(j, i)$	The bucket at level i along the path labeled j

3.1. Storage Structure. In our LPS-ORAM, there are N real blocks, which are outsourced to the server storage. Each block's modality is (*identifier*, p; *data*). It represents that the block numbered as *identifier* is either on the path numbered by leaf label p or in the local stash. For each real block, it has a unique identifier, *identifier*, and the content of the block labeled as *identifier* contains *data*. For each virtual block, both p and *data* are populated with random strings. In order to obfuscate all the blocks with each other, they are set to a constant size no matter whether the block is a real block or a virtual block. To differentiate decrypted blocks retrieved from the server storage, all virtual blocks have same block identifiers. In addition, the purpose of adding virtual blocks to the server storage is to confuse all blocks so that the server cannot differentiate between any encrypted block being a real block or a virtual block. Note that all blocks are encrypted by the client before they are uploaded to the server. That is, all blocks in the server storage are in the state of encryption. Therefore, adding virtual blocks to the server storage is part of the security effort to hide the data access pattern.

3.1.1. Server Storage. On the server side, there is a complete binary tree. In it, there are $L+1$ levels in total. They are marked as 0, 1, 2, . . ., and L, respectively. Theoretically, the height of the complete binary tree is set to $L = \lceil \log N \rceil + 1$. For the sake of description, we let $L = \log N$, resulting in a full binary tree with N leaves and $N-1$ non-leaves. In the complete binary tree, the root node is at layer 0 and all the leaf nodes are at layer L.

Each node of the complete binary tree is one bucket in our LPS-ORAM solution. Z blocks at most in each bucket. As a result, each bucket is Z-block in size. In our scheme, Z is set to a constant. Each bucket can accommodate some real blocks, and dummy blocks populate the rest of the space.

The complete binary tree has about $2N$ nodes, so there are about $2Z * N$ blocks on the server side. That is, the server storage size is $O(N)$ blocks.

Algorithm: Access (a, op, $data^*$)

// The requested block a is retrieved and operated.

// The algorithm consists of the Retrieval algorithm and the Eviction algorithm.

1. $j \leftarrow$PosMap(a) // lookup the *PosMap* locally

2. **for** i= 0, 1, 2, …, H **do** //Fetch all buckets of the target path
3. Stash$\leftarrow$Stash+P (j, i) // get the requested block a
4. **end for**

5. $data\leftarrow$ Fetch block a from Stash
6. **if** op = '*write*'
7. **then** $data\leftarrow data^*$
8. **end if**

9. PosMap(a)$\leftarrow$Uniform_And_Random (0, N]
   //assuming that all identifiers of dummy blocks are set to 0.
10. get the leaf label of each fetched real block
11. get the mutable scope of each fetched real block
12. **If** the real block can be directly written back into a lower bucket than the original bucket
13. **Then** the block need not to be remapped
14. **Else**
15. **If** the root bucket is not empty
16. **Then** each fetched real block is remapped to a new leaf label from the above scope, until the real block can be written back into a lower bucket than the original bucket.
17. **end if**
18. **end if**

19. **for** i= L, L- 1, …, 0 **do** // scheduled from the corresponding leaf bucket to the root bucket.
20. Each real block is written back into the bucket as close to the leaf bucket as possible.
21. Write back $P(j)$ into the binary tree.
22. **end for**

23. **return** $data$

Figure 2: The algorithm of our LPS-ORAM solution.

3.1.2. Client Storage. There are two structures on the client side, PosMap and stash.

In our scheme, there are N real blocks, and each real block is remapped to a leaf label, so there are N leaf labels. All of the above N leaf labels are stored in the PosMap. The complete binary tree has N leaves, and each leaf is numbered by a leaf label, which results in each leaf label having a size of $\log N$ bits. Thus, the size of the PosMap is $(N * \log N)$ bits. In the client-server environment, the client can entirely store the PosMap, rather than the server storing the PosMap recursively because storing the PosMap on the client is virtually negligible when the block size is not set to very small size of O ($\log N$) bits. Moreover, if the server stores the PosMap recursively, both the average time latency and the number of interaction rounds increase significantly.

The client has a stash to store temporary blocks retrieved from the server storage. Why is it temporary storage? Because all blocks on the target path are retrieved to the local stash, and then all the blocks are written back into the path of the complete binary tree. In the previous tree-based ORAM solutions, there might have been some stranded real blocks on the stash because the root bucket might have been full. In our perfectly secure tree-based ORAM solution, the stash size is exactly the size of retrieved path because there are no stranded real blocks on the stash.

3.2. Detailed Execution. In this section, the detailed execution procedures are described as follows. There are two algorithms to implement our LPS-ORAM protocol, which are retrieval algorithm and eviction algorithm, respectively. A detailed description of the whole LPS-ORAM algorithm is shown in Figure 2.

3.2.1. Retrieval. The retrieval algorithm is to fetch all blocks on the target path corresponding to the leaf label of the requested block. All the fetched blocks are stored in the stash locally. After the retrieval, all the fetched blocks are decrypted and then the dummy blocks are discarded. That is, only the real blocks are stored in the stash on the client. Subsequently, the requested block is assigned to a new leaf label from random and uniform remapping. In order to

implement a failure probability of 0, that is, to allow the root bucket to never fill up while maintaining its statistical security in tree-based ORAM, all remaining fetched real blocks need to be remapped if necessary. However, instead of applying a random and uniform remapping to them, a new remapping is applied to them. Since the distribution of real blocks on the path in which buckets is dynamic and the server cannot know the distribution, we can take the step of infiltrating the real block down the position of a bucket to free up space of the upper bucket. Thus, the root bucket will not be full even if the remapping of the requested block is in the worst case, where both the path corresponding to the leaf label of the requested block and the eviction path at that time are two branches of the binary tree. The question is what kind of remapping would allow fetched real blocks to move down one bucket? At this point, the dynamic remapping associated with a mutable scope is proposed.

Now we illustrate our proposed dynamic remapping with a mutable scope, as shown in Figure 3. The target path is marked by read line and taken from the PosMap on the client, and all real blocks in the target path are (a, 3), (b, 2), (e, 1), (f, 2), (h, 3). It is noted that data of each real block is ignored to descript simply. For example, block (a, 3, data) is written as (a, 3). Among the fetched real blocks, block (a, 3) is the requested block at that time. Since the root bucket is not empty and block (f, 2) cannot be located in a bucket lower than the original bucket, block (f, 2) has to be remapped from a mutable scope [2, 3], which is the set from the original leaf label to the leaf label corresponding to the eviction path. In Figure 3, it is marked in yellow. Also, the rest of the fetched real blocks (b, 2), (e, 1), (h, 3) need not be remapped because they can be directly written back into a lower bucket than the original bucket. The requested block (a, 3) is remapped to a new random and uniform leaf label because of the obliviousness property of ORAM.

As shown in Figure 2, step 1 is the lookup procedure to get the leaf label of the target path, which is the leaf label of the requested block. From step 2 to step 4 is the retrieval procedure to get all buckets of the target path in the binary tree. Then, from step 5 to step 8 is the operation procedure of the requested block during each access. If the operation is "*read*," *data* of the requested block are directly returned to the client, as described in step 23. If the operation is "*write*," *data* are updated by *data* $*$. From step 9 to step 18 is the remapping procedure to get a new leaf label for each fetched real block. The procedure is divided into two cases. One case is for the requested block, and the other case is for all other fetched real blocks from the target path. Due to the obliviousness property of ORAM, after each access, the requested block needs to be remapped to a new random and uniform leaf label. Nevertheless, all other fetched real blocks can avoid the random and uniform remapping because they are dynamically distributed on each path and this distribution is secret for the server. Thus, it is secure to adjust the distribution of them to achieve some certain goal. If the root bucket is not empty, each real block in the path will be written back into a lower bucket than the original bucket through adjusting the corresponding leaf label. Note that a lower bucket is closer to the corresponding leaf bucket in the binary tree. If some real block of them can be directly written back into a lower bucket than the original bucket, the block need not be remapped. Else, the block is remapped to a new leaf label from the above scope, until the real block can be written back into a lower bucket than the original bucket, this process takes almost no time.

3.2.2. Eviction. In this section, the dynamically balanced eviction algorithm is proposed. The proposed eviction algorithm not only follows the greed strategy but also further makes use of the space of the eviction path in the binary tree. The greed strategy in the eviction algorithm is that as many fetched blocks as possible are written back from the stash locally to the eviction path in the binary tree. The above dynamic remapping associated with a mutable scope can make each real block locate into a lower bucket. Thus, our dynamically balanced eviction algorithm can be combined with the above dynamic remapping associated with a mutable scope to make better use of the space of the eviction path in the binary tree than a single greed strategy.

During the eviction algorithm, the requested block is first arranged to the deepest bucket of the path according to the new remapped leaf label, which is closest to the leaf bucket. Because the new remapped leaf label of the requested block is random and uniform after each access, according to dynamic remapping with a mutable scope, if one path is accessed multiple times, almost all of the real blocks in the path will be squeezed to the buckets near the leaf bucket. This information is harmful because it is inferred easily by the honest-but-curious server. Thus, the dynamically balanced eviction is utilized to remove the above harmful information. Whether a real block in the path needs to be located into a bucket lower than the original bucket, it depends on whether the root bucket is empty. If the root bucket is not empty, the above dynamic remapping needs to be executed. Otherwise, it cannot be executed. Thus, the proposed eviction is dynamically balanced. In our eviction algorithm, for each access, the goal is that the requested block can be written back into the path in the binary tree, rather than being stranded in the stash locally. This goal is the focus of our scheme in the setting of tree-based ORAM. In tree-based ORAM, the root bucket may be full because only the requested block needs to be remapped to a random and uniform leaf label, while other fetched real blocks do not need to be remapped. If the requested block for each access is in the worst case, with a small but non-negligible probability, the root bucket will accumulate until it is full as the number of different requested blocks increases. However, our eviction algorithm can avoid this case.

As shown in Figure 2, from step 19 to step 22 is the procedure of the eviction algorithm to write back all fetched real blocks containing the requested block into the eviction path in the binary tree. During this procedure, the path is scheduled from the corresponding leaf bucket to the root bucket. Then, each fetched real block is written back into some lower bucket than the original bucket if the root bucket is not empty. Finally, the eviction path is written back into the binary tree on the server storage.

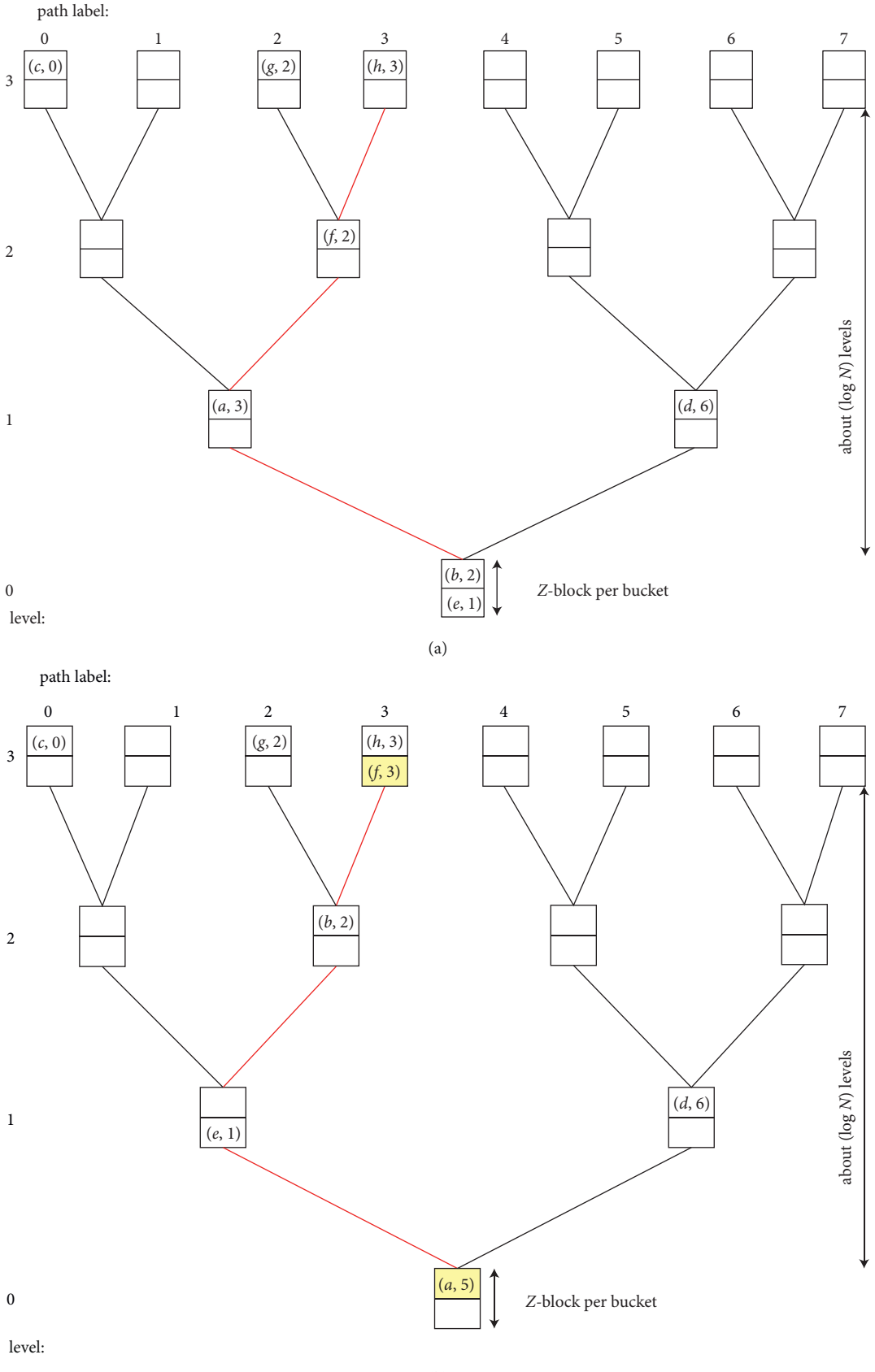


FIGURE 3: The proposed dynamic remapping. (a) The state of the binary tree before the proposed dynamic remapping. The red line is the target path. The block of bold lines is the requested block labeled a, namely, block $(a, 3)$, where *data* is ignored to descript simply. (b) The state of the binary tree after eviction. The fetched real blocks that are not marked yellow need not to be remapped, such as blocks $(b, 2)$, $(e, 1)$, $(h, 3)$, while the fetched real blocks marked yellow except the requested block need to follow the dynamic remapping with a mutable scope, for example, block $(f, 2)$ is modified to $(f, 3)$. The requested block $(a, 3)$ is remapped to a new random and uniform leaf label 5 because of the obliviousness property of ORAM.

3.3. Security Analysis. In this section, we will analyze the perfect correctness and perfect security of our LPS-ORAM. The perfect correctness of ORAM means that the ORAM scheme fails with the probability of 0, rather than a negligible probability. The perfect security of ORAM means that the ORAM scheme can resist against an adversary with unlimited computing power, and simultaneously the ORAM scheme has perfect correctness.

3.3.1. Perfect Correctness

Claim 1. Our LPS-ORAM scheme has perfect correctness.

Proof. If the block can be directly written back into a lower bucket than the original bucket, the fetched real block need not be remapped. Else, each fetched real block is remapped to a new leaf label from a mutable scope, until the real block can be written back into a lower bucket than the original bucket. In a word, each fetched real block from the target path needs to be located in a lower bucket than the original bucket. However, the requested block needs to follow the random and uniform remapping because of the oblivious property of ORAM. When the requested block is in the worst case, namely, both the path corresponding to the new leaf label of the requested block and the eviction path are two branches of the binary tree, the requested block will have to be written back into the root bucket. However, since each of all other real blocks can be written back into a lower bucket than the original bucket, the root bucket is filled with at most one real block at any epoch. As a result, as long as the size of the root bucket is larger than one block, the root bucket cannot be full. That is, our LPS-ORAM scheme can fail with the probability of 0 as long as the bucket size is longer than 1. Therefore, our LPS-ORAM scheme has perfect correctness. □

3.3.2. Perfect Security

Claim 2. Our LPS-ORAM scheme is statistically secure for the honest-but-curious server.

Proof. In our LPS-ORAM, each path fetched is random and uniform for the honest-but-curious server. That is, all blocks of each bucket fetched from the binary tree are random and uniform. As a result, for any two kinds of access, the two paths retrieved are statistically indistinguishable for the server. Therefore, our LPS-ORAM scheme is statistically secure for the honest-but-curious server. □

Theorem 1. *Our LPS-ORAM scheme is perfectly secure for the honest-but-curious server.*

Proof. According to Claim 2, our LPS-ORAM is statistically secure for the honest-but-curious server. As a result, our scheme can resist against an adversary with unlimited computing power. In addition, according to Claim 1, our LPS-ORAM scheme has perfect correctness. Therefore, according to the security definition from Definition 1, our LPS-ORAM scheme is perfectly secure for the honest-but-curious server. □

4. Performance Analysis

In this section, we will analyze the asymptotic performance, which is mainly in bandwidth overhead and storage overhead. We proposed the measures of further optimization. Our LPS-ORAM solution will be compared with all the previous perfectly secure single-server ORAM solutions, which are listed in Table 2.

4.1. Bandwidth Overhead. In our solution, for each access, only one path is fetched and then is written back into the binary tree. Thus, to fetch a requested block, the number of blocks transferred between the client and the server is O (log N) blocks. As a result, the bandwidth overhead of our solution is O (log N)-block.

4.2. Storage Overhead. In our solution, the bucket size Z is a constant and the binary tree on the server storage has O (N) buckets. As a result, the binary tree has O (N) blocks. That is, the server storage overhead of our solution is O (N)-block. The client storage consists of PosMap and stash. PosMap is practically negligible in the setting of client-server, as mentioned in S^3ORAM [26]. Thus, the stash size is the client storage overhead, which is one path size. Therefore, the client storage overhead of our solution is O (log N)-block.

4.3. Further Optimization. In our LPS-ORAM solution, if the bucket size Z is set to 1, the asymptotic performance of our solution can be further optimized. In this case, the bandwidth overhead is $(L+1)$-block, the server storage overhead is $(2^{L+1}-1)$-block, and the client storage overhead is one path size of $(L+1)$-block. So, these overheads are determined by the value of L. In theory, the value of L is set to $\lceil \log N \rceil + 1$. So, the number of buckets is about $4N$. There is enough space to percolate down for the real blocks to release the root bucket. However, to release the root bucket more likely, a larger number of buckets or a larger Z is needed. Thus, the value of L and Z is in a dynamic equilibrium to achieve a trade-off.

5. Evaluation

To give the actual performance of our LPS-ORAM solution, we implemented a prototype with a client-side position map and evaluated it based on bandwidth overhead, temporary storage overhead, and server storage overhead. Our solution will be compared with all the previous perfectly secure single-server ORAM solutions. So far, there are three such solutions. They are proposed by Damgard et al. [36], Chan et al. [37], and Raskin and Simkin [38], respectively. In addition, Path ORAM solution [13] is also compared with ours because it is a tree-based ORAM with efficient efficiency of logarithmic bandwidth overhead.

In comparison, for different values of N, we measure the bandwidth overhead, namely, the total amount of data

Table 2: Asymptotic performance comparison of all the perfectly secure single-server ORAM schemes.

Perfectly secure ORAM scheme	Structure	Amortized-case bandwidth	Worst-case bandwidth	Client storage	Server storage
Damgard et al. [36]	Layer	$O(\log^3 N)$	$O(N * \log N)$	$O(1)$	$O(N * \log N)$
Chan et al. [37]	Layer	$O(\log^3 N)$	$O(N * \log N)$	$O(1)$	$O(N)$
Raskin et al. [38]	Matrix	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(N)$
Ours	Tree	$O(\log N)$	$O(\log N)$	$O(\log N)$	$O(N)$

Note. All asymptotic overheads are represented in blocks.

transferred per access between the client and the server. In addition, both the temporary storage size on the client and the total storage size on the server are measured in our experiments. In the respective works, the values in all the compared ORAM schemes with our LPS-ORAM are calculated based on the concrete formulas and constants that are reported.

We make the following assumptions, as mentioned in the evaluation of Lookahead ORAM solution [38]. There is an additional encryption/MAC overhead of about 40 bytes in each encrypted block because the random encryption is applied to all blocks of all ORAM schemes. Within each stash slot, there is an additional state header of about 20 bytes that contains location information and the state. Also, 4-byte words are used to indicate the ORAM request types. During initialization procedure, the storage is populated with random strings and they are directly uploaded to the server. In all solutions, the block size is fixed to 1024 bytes.

We first analyze the concrete value of bandwidth overhead, temporary storage overhead, and server storage overhead in our LPS-ORAM solution. In it, $L = \lceil \log N \rceil + 1$ in theory and $Z = 1$. Thus, when N real blocks of each size B-bit are encrypted, the total server storage overhead of the server is $Z \times (2^{L+1} - 1) \times (B + 40)$-bit $= (4N - 1) \times (B + 40)$-bit. The corresponding position map is $(N \times \log N)$-bit. The stash size is $(B + 40) \times (2 + \log N)$-bit, which is the temporary storage overhead. For each access, $(B + 40) \times (2 + \log N)$ bits need to be downloaded and then $(B + 40) \times (2 + \log N)$ bits need to be uploaded, and thus the bandwidth overhead is $2 \times (B + 40) \times (2 + \log N)$ bits.

Finally, for the sake of description in the following figures, the solutions proposed by Damgard et al. [36] and Chan et al. [37] are called ORAM_1, ORAM_2, and the solution proposed by Raskin and Simkin [38] is called Lookahead ORAM.

5.1. Bandwidth Overhead. The bandwidth overhead refers to the number of blocks transferred between the client and the server to obtain a requested block. The bandwidth overheads of the above compared solutions are listed in the following.

The ORAM_1 and ORAM_2 solutions are based on a hierarchical structure, which have no position map on the client. Their concrete bandwidth overheads are $(\log^2 N) * (1 + \log N)/2$ blocks, which are self-reported. As a result, the value is $(B + 40) \times (\log^2 N) \times (1 + \log N)/2$ bits. The Lookahead ORAM is based on a matrix structure, which also has a position map on the client. The position map is also $O(N \times \log N)$-bit. However, the recursion technique is not considered to be applied to the position map. Thus, its concrete bandwidth overhead is $\{40 + (B + 40) \times (\sqrt{N} + 1)\} + \{80 + (B + 40) \times (\sqrt{N} + 1)\} = 120 + 2 \times (B + 40) \times (\sqrt{N} + 1)$ bits, which is self-reported. In addition, the bandwidth overhead of Path ORAM is also shown in Figure 4. In Path ORAM, the bandwidth overhead $= (B + 40) \times 2 \quad Z \times \log \quad N = 10 \times (B + 40) \times \log N$ bits, as shown in the evaluation of Lookahead ORAM.

Finally, the results are shown in Figure 4. As expected, our LPS-ORAM has the smallest bandwidth overhead of all the compared solutions.

5.2. Temporary Storage Overhead. The temporary storage overhead on the client side refers to the number of blocks stored on the client, which is temporary because after each access, all fetched blocks stored in the stash locally are written back into the server storage. The temporary storage overheads of the above compared solutions are listed in the following.

The ORAM_1 and ORAM_2 solutions are based on a hierarchical structure. The temporary storage overheads contain one block, which is self-reported. As a result, the value is $(B + 40)$ bits. The Lookahead ORAM is based on a matrix structure, which also has a position map on the client. Its concrete temporary storage overhead on the client is $80 + (B + 40) \times (\sqrt{N} + 1)$ bits, which is self-reported. Additionally, in Path ORAM, the temporary storage overhead on the client is about $10\,N \times (B + 40)$ bits, which is self-reported.

We observed that the temporary storage overhead is about half of the bandwidth overhead in Lookahead ORAM, Path ORAM, and our solution, while the temporary storage overheads in the ORAM_1 and ORAM_2 solutions are only considered to be a small constant. Thus, the figure of results for Lookahead ORAM, Path ORAM, and our solution is similar to that of Figure 4. As expected, our LPS-ORAM has the smallest temporary storage overhead among the above three solutions.

5.3. Server Storage Overhead. The storage overhead on the server side refers to the number of blocks stored on the server, which not only contains real blocks but also dummy blocks. The storage overheads of the above compared solutions are listed in the following.

The ORAM_1 solution is based on a hierarchical structure. Its concrete storage overhead of the server is $(2N - 1) \times \log N$ blocks, which is self-reported. As a result, the value is $(B + 40) \times (N - 1) \times \log N$ bits. The ORAM_2 solution is based on the ORAM_1 solution. Its concrete storage overhead of the server is reduced to $2N$ blocks, which is self-reported. As a result, the value is $(B + 40) \times 2N$ bits. The Lookahead ORAM

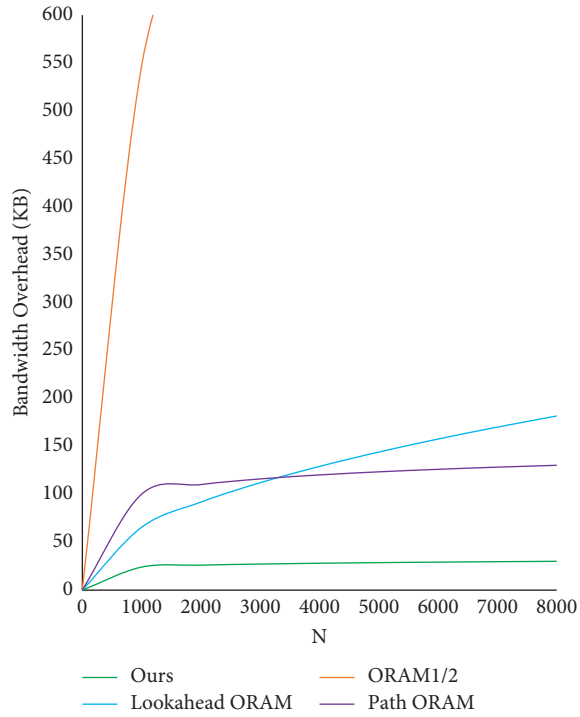


Figure 4: Comparison of the bandwidth overheads of all the perfectly secure single-server ORAM solutions. The X-axis shows different values of N. The Y-axis shows the total amount of data transferred per access in KB between the client and the server.

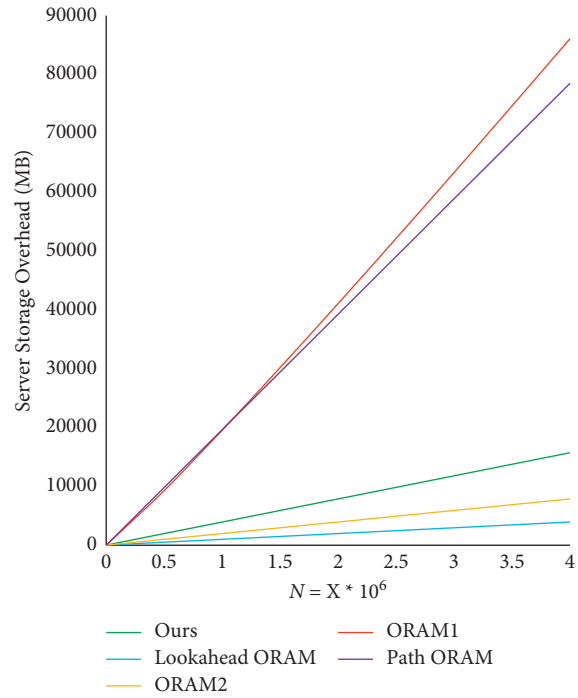


Figure 5: Comparison of the server storage overheads of all the perfectly secure single-server ORAM solutions. In the X-axis, $N = X * 10^6$. The Y-axis shows the total required storage on the server side in MB.

is based on a matrix structure, which also has a position map on the client. The position map is also $O(N * \log N)$-bit. However, the recursion technique is also not considered to be applied to the position map. Thus, its concrete storage overhead of the server is $N \times (B + 40)$ bits, which is self-reported. In addition, the serve storage overhead of Path

ORAM is also shown in Figure 5. In Path ORAM, the server storage overhead is about $20N \times (B + 40)$ bits, as shown in the evaluation of Lookahead ORAM.

Finally, the results are shown in Figure 5. As expected, our LPS-ORAM is slightly larger than Lookahead ORAM in terms of server storage overhead.

6. Conclusion

In this paper, we focus on perfect security of ORAM. Since all existing perfectly secure single-server ORAM solutions require at least sublinear worst-case bandwidth overhead, a natural and open question is posed: *can we construct a perfectly secure single-server ORAM with logarithmic worst-case bandwidth overhead*? To affirmatively answer the question, we propose the first tree-based perfectly secure ORAM scheme with logarithmic worst-case bandwidth overhead, called LPS-ORAM. To meet the requirements of perfectly secure ORAM, two techniques are used. One technique is dynamic remapping associated with a mutable scope, and the other is dynamically balanced eviction. Their combined effect allows the root bucket to never fill up while maintaining its statistical security in tree-based ORAM. In terms of overhead for temporary storage on the client side, compared with the latest perfectly secure ORAM solution, our solution is reduced from sublinear to logarithmic, even if the server storage overhead scales lightly, it is still at the same level of quantity as the state of the art. Finally, the evaluation results show that our LPS-ORAM has a significant advantage in terms of bandwidth overhead and overhead for temporary storage on the client side.

Data Availability

The data used to support the findings of this study are available from the authors upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the National Key R&D Program of China under grant no. 2020YFB1005900 and National Natural Science Foundation of China under grant no. 62072051.

References

[1] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: ramification, attack and mitigation," in *Proceedings of the 2012 Network and Distributed System Security Symposium*, pp. 1–15, San Diego, CA, USA, February 2012.

[2] J. L. Dautrich and C. V. Ravishankar, "Compromising privacy in precise query protocols," in *Proceedings of the 16th International Conference on Extending Database Technology - EDBT '13*, pp. 155–166, Genoa Italy, March 2013.

[3] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15*, pp. 668–679, Denver, CO, USA, October 2015.

[4] O. Goldreich, "Towards a theory of software protection and simulation by oblivious RAMs," in *Proceedings of the nineteenth annual ACM conference on Theory of computing - STOC '87*, pp. 182–194, New York, NY, USA, 1987.

[5] R. Ostrovsky, "Efficient computation on oblivious RAMs (extended abstract)," in *Proceedings of the STOC '90 Proceedings of the twenty-second annual ACM symposium on Theory of Computing*, pp. 514–523, Baltimore, MD, USA, May 1990.

[6] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious RAMs," *Journal of the ACM*, vol. 43, no. 3, pp. 431–473, 1996.

[7] S. Y. Chiou and Y. X. He, "Generalized proxy oblivious signature and its mobile application," *Security and Communication Networks*, vol. 2021, Article ID 5531505, 16 pages, 2021.

[8] B. Pinkas and T. Reinman, "Oblivious RAM revisited," in *Advances in Cryptology – CRYPTO 2010*, vol. 6223, pp. 502–519, Springer, Berlin, Heidelberg, 2010.

[9] E. Shi, T.-H. H. Chan, E. Stefanov, and M. Li, "Oblivious RAM with $O((\log N)^3)$ worst-case cost," in *Advances in Cryptology – ASIACRYPT 2011*, vol. 7073, pp. 197–214, Springer, Berlin, Heidelberg, 2011.

[10] Z. Li, C. Xiang, and C. Wang, "Oblivious transfer via lossy encryption from lattice-based cryptography," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 5973285, 11 pages, 2018.

[11] E. Stefanov, E. Shi, and D. Song, "Towards practical oblivious RAM," in *Proceedings of the 2012 Network and Distributed System Security Symposium*, pp. 1–40, San Diego, CA, USA, February 2012.

[12] E. Kushilevitz, S. Lu, and R. Ostrovsky, "On the (In)security of hash-based oblivious RAM and a new balancing scheme," in *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 143–156, San Francisco, CA, USA, January 2012.

[13] E. Stefanov, M. Van Dijk, E. Shi et al., "Path ORAM: an extremely simple oblivious RAM protocol," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS '13*, pp. 299–310, Berlin, Germany, November 2013.

[14] S. Zhao, X. Song, H. Jiang, M. Ma, Z. Zheng, and Q. Xu, "An efficient outsourced oblivious transfer extension protocol and its applications," *Security and Communication Networks*, vol. 2020, Article ID 8847487, 12 pages, 2020.

[15] S. Gordon, X. Huang, A. Miyaji, C. Su, K. Sumongkayothin, and K. Wipusitwarakun, "Recursive matrix oblivious RAM: an ORAM construction for constrained storage devices," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 12, pp. 3024–3038, 2017.

[16] H. Ding, H. Jiang, and Q. Xu, "Postquantum cut-and-choose oblivious transfer protocol based on LWE," *Security and Communication Networks*, vol. 2021, Article ID 9974604, 15 pages, 2021.

[17] X. Zhang, G. Sun, C. Zhang et al., "Fork path: improving efficiency of ORAM by removing redundant memory accesses," in *Proceedings of the 48th International Symposium on Microarchitecture - MICRO-48*, pp. 102–114, Waikiki, Hl, USA, December 2015.

[18] Z. Chang, D. Xie, and F. Li, "Oblivious RAM: a dissection and experimental evaluation," *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 1113–1124, 2016.

[19] B. Li, Y. Huang, Z. Liu, J. Li, Z. Tian, and S.-M. Yiu, "HybridORAM: practical oblivious cloud storage with constant bandwidth," *Information Sciences*, vol. 479, pp. 651–663, 2019.

[20] Z. Liu, Y. Huang, J. Li, X. Cheng, and C. Shen, "DivORAM: towards a practical oblivious RAM with variable block size," *Information Sciences*, vol. 447, pp. 1–11, 2018.

[21] J. Sancho, J. García, and A. Alesanco, "Oblivious inspection: on the confrontation between system security and data privacy at domain boundaries," *Security and Communication Networks*, vol. 2020, Article ID 8856379, 9 pages, 2020.

[22] X. Yu, S. K. Haider, L. Ren et al., "PrORAM: dynamic prefetcher for oblivious RAM," in *Proceedings of the 42nd Annual International Symposium on Computer Architecture - ISCA '15*, pp. 616–628, Portland, OR, USA, June 2015.

[23] H. Yang, J. Shen, J. Lu, T. Zhou, X. Xia, and S. Ji, "A privacy-preserving data transmission scheme based on oblivious transfer and blockchain technology in the smart healthcare," *Security and Communication Networks*, vol. 2021, Article ID 5781354, 12 pages, 2021.

[24] Y. Ishai, E. Kushilevitz, R. Ostrovsky, M. Prabhakaran, and A. Sahai, "Efficient non-interactive secure computation," in *Advances in Cryptology – EUROCRYPT 2011*, vol. 6632, pp. 406–425, Springer, Berlin, Heidelberg, 2011.

[25] L. Ren, C. Fletcher, A. Kwon et al., "Constants count: practical improvements to oblivious RAM," in *Proceedings of the 24th USENIX Security Symposium*, pp. 415–430, Washington, D.C, USA, August 2015.

[26] T. Hoang, C. D. Ozkaptan, A. A. Yavuz, J. Guajardo, and T. Nguyen, "S^{3}ORAM: a computation-efficient and constant client bandwidth blowup ORAM with shamir secret sharing," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security - CCS '17*, pp. 491–505, Dallas, TX, USA, November 2017.

[27] E. Boyle and M. Naor, "Is there an oblivious RAM lower bound?" in *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science - ITCS '16*, pp. 357–368, Cambridge, MA, USA, January 2016.

[28] K. G. Larsen and J. B. Nielsen, "Yes, there is an oblivious RAM lower bound," in *Advances in Cryptology – CRYPTO 2018*, vol. 10992, pp. 523–542, Springer, Cham, 2018.

[29] X. Wang, H. Chan, and E. Shi, "Circuit ORAM: on tightness of the goldreich-ostrovsky lower bound," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15*, pp. 850–861, Denver, CO, USA, October 2015.

[30] S. Patel, G. Persiano, M. Raykova, and K. Yeo, "PanORAMa: oblivious RAM with logarithmic overhead," in *Proceedings of the 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 871–882, Paris, France, October. 2018.

[31] G. Asharov, I. Komargodski, W.-K. Lin, K. Nayak, E. Peserico, and E. Shi, "OptORAMa: optimal oblivious RAM," in *Proceedings of the Advances in Cryptology – EUROCRYPT 2020*, pp. 403–432, Springer, Zagreb, Croatia, May 2020.

[32] K. G. Larsen, M. Simkin, and K. Yeo, "Lower bounds for multi-server oblivious RAMs," in *Theory of Cryptography Conference*, vol. 12550, pp. 486–503, Springer, Cham, 2020.

[33] I. Abraham, C. W. Fletcher, K. Nayak, B. Pinkas, and L. Ren, "Asymptotically tight bounds for composing ORAM with PIR," in *Public-Key Cryptography – PKC 2017*, vol. 10174, pp. 91–120, Springer, Berlin, Heidelberg, 2017.

[34] D. Cash, A. Drucker, and A. Hoover, "A lower bound for one-round oblivious RAM," in *Theory of Cryptography Conference*, vol. 12550, pp. 457–485, Springer, Cham, 2020.

[35] I. Komargodski and W.-K. Lin, "A logarithmic lower bound for oblivious RAM (for all parameters)," in *Advances in Cryptology – CRYPTO 2021*, vol. 12828, pp. 579–609, Springer, Cham, 2021.

[36] I. Damgård, S. Meldgaard, and J. B. Nielsen, "Perfectly secure oblivious RAM without random oracles," in *Theory of Cryptography Conference*, vol. 6597, pp. 144–163, Springer, Berlin, Heidelberg, 2011.

[37] T.-H. H. Chan, K. Nayak, and E. Shi, "Perfectly secure oblivious parallel RAM," in *Theory of Cryptography Conference*, vol. 11240, pp. 636–668, Springer, Cham, 2018.

[38] M. Raskin and M. Simkin, "Perfectly secure oblivious RAM with sublinear bandwidth overhead," in *Proceedings of the Advances in Cryptology – ASIACRYPT 2019*, p. 27, Kobe, Japan, December 2019.

[39] T. Mayberry, E.-O. Blass, and A. H. Chan, "Efficient private file retrieval by combining ORAM and PIR," in *Proceedings of the ISOC Network and Distributed System Security Symposium*, San Diego, CA, USA, February 2014.

[40] D. Apon, J. Katz, E. Shi, and A. Thiruvengadam, "Verifiable oblivious storage," in *Public-Key Cryptography – PKC 2014*, vol. 8383, pp. 131–148, Springer, Berlin, Heidelberg, 2014.

[41] S. Devadas, M. van Dijk, C. W. Fletcher, L. Ren, E. Shi, and D. Wichs, "Onion ORAM: a constant bandwidth blowup oblivious ram," in *Theory of Cryptography Conference*, vol. 9563, pp. 145–174, Springer, Berlin, Heidelberg, 2016.

[42] I. Demertzis, D. Papadopoulos, and C. Papamanthou, "Searchable encryption with optimal locality: achieving sublogarithmic read efficiency," in *Advances in Cryptology – CRYPTO 2018*, vol. 10991, pp. 371–406, Springer, Cham, 2018.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 7963004, 13 pages
https://doi.org/10.1155/2022/7963004

Research Article

Horizontally Partitioned Data Publication with Differential Privacy

Zhen Gu,[1,2] **Guoyin Zhang**,[1] **and Chen Yang**[1]

[1]*College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China*
[2]*The Department of Basic Education, East University of Heilongjiang, Harbin 150066, China*

Correspondence should be addressed to Guoyin Zhang; zhangguoyin@hrbeu.edu.cn

Received 13 April 2022; Revised 13 June 2022; Accepted 6 July 2022; Published 31 July 2022

Academic Editor: Debiao He

In this paper, we study the privacy-preserving data publishing problem in a distributed environment. The data contain sensitive information; hence, directly pooling and publishing the local data will lead to privacy leaks. To solve this problem, we propose a multiparty horizontally partitioned data publishing method under differential privacy (HPDP-DP). First, in order to make the noise level of the published data in the distributed scenario the same as in the centralized scenario, we use the infinite divisibility of the Laplace distribution to design a distributed noise addition scheme to perturb the locally shared data and use Paillier encryption to transmit the locally shared data to the semitrusted curator. Then, the semitrusted curator obtains the estimator of the covariance matrix of the aggregated data with Laplace noise and then obtains the principal components of the aggregated data and returns them to each data owner. Finally, the data owner utilizes the generative model of probabilistic principal component analysis to generate a synthetic data set for publication. We conducted experiments on different real data sets; the experimental results demonstrate that the synthetic data set released by the HPDP-DP method can maintain high utility.

1. Introduction

The ability of people to collect and analyze data is gradually improving with the development of the artificial intelligence. Sometimes the data are stored by different sites(data owners), and each site holds a smaller number of samples. For example, in Figure 1, there are three hospitals, the patients in each hospital are different from each other, but the data features of each patient are the same. In order to better mine the useful information behind the data, a large number of samples are needed. Pooling data in one central location enables efficient data analysis and mining, but data contain sensitive privacy; directly sharing or pooling the data will lead to privacy leakage [1, 2], which prevents people from sharing data. That is to say, data are facing serious privacy leakage risks in the process of data sharing, network transmission, and storage [3]. It is important to protect the privacy of shared data and weigh the security and availability of data [4, 5]. Therefore, it is desirable to propose an efficient distributed algorithm, which can provide the utility close to the centralized case and protect the privacy of data. In recent years, there have been some researches on privacy-preserving data publishing and sharing, for example, the *k*anonymity [6] technology, the encryption techniques, such as lattice-based cryptography [7] and quantum cryptography [8, 9]. The differential privacy [10] has been widely used for privacy-preserving data publishing; privacy-preserving data publishing based on differential privacy has become a research hot spot [11–15].

However, there are still some challenges when using the differential privacy technique to protect the privacy of the published data. One is that the data are stored by different data owners; directly pooling and publishing the data will lead to privacy leakage. When data are stored by multiple data owners, as the number of data owners increases, if differential privacy is used independently to add noise to the locally shared data, the utility of the published data will be reduced. In view of this, we propose a horizontally partitioned data publication approach with differential privacy. We make the following contributions:

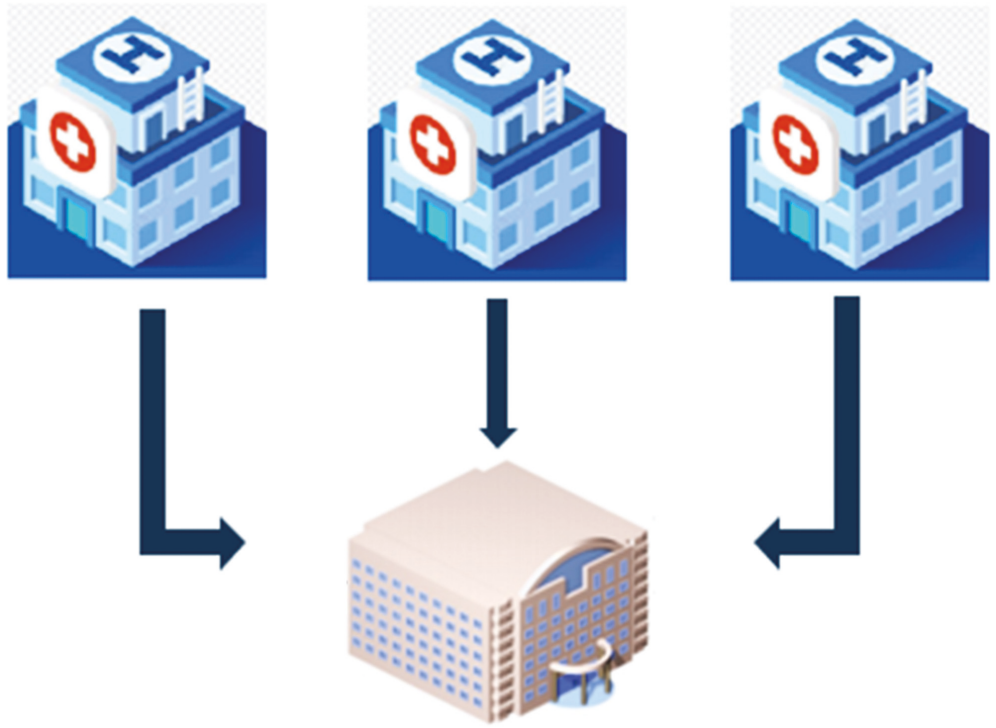


Figure 1: Aggregate data from different hospitals.

(1) We propose a method for horizontally partitioned data publication with differential privacy (HPDP-DP). In a distributed environment, data are owned by multiple parties. We use the weighted average of the noised covariance matrices of the local data to estimate the covariance matrix of the pooled data. The data owners and a semitrusted curator collaborate to get the principal components of the pooled data and generate a synthetic data set for publishing.

(2) In the distributed scenario, in order to make the noise level of the aggregated data the same as in the centralized scenario, the HPDP-DP method utilizes the infinite divisibility of the Laplace distribution and Paillier homomorphic encryption to alleviate the effects of noise and can achieve the same noise level as the centralized scenario.

(3) We evaluate the performance of HPDP-DP method through experiments on real data sets, and the experimental results show that HPDP-DP method can generate synthetic data with high efficiency.

2. Related Work

In this section, we introduce the research status of privacy-preserving data release based on differential privacy in the centralized and distributed scenarios, respectively.

2.1. Privacy-Preserving Data Publishing in Centralized Environment. In recent years, there are many researches on privacy-preserving data publishing based on differential privacy. Jiang et al. [16] proposed a method that adding Laplace noise to the covariance matrix and the projection matrix and then using the noisy projection matrix to restore and generate the synthetic data set for publishing. Zhang et al. proposed the PrivBayes method in [17]; they used the relationship between the features to build a Bayesian network. They added Laplace noise to the low-dimensional marginal distribution to make the Bayesian network satisfy differential privacy, and then they used the Bayesian network to generated a synthetic data set for publishing. Chen et al. proposed the Jtree method in [18]. First, they proposed a sampling-based testing framework that is used to explore pairwise dependencies while satisfying differential privacy. Then, they applied the connection tree algorithm to construct an inference mechanism to infer the joint data distribution. Finally, they efficiently generated a synthetic data set by using the noise margin table and inference model. Xu et al. [19] proposed DPPro scheme; they released high-dimensional data by using randomly projected. They projected the original high-dimensional data into a randomly selected low-dimensional subspace and added noise to the low-dimensional projected data. They theoretically demonstrated that the data published by the DPPro method have similar squared Euclidean distances to the original data. In order to solve the problem of dimensional disaster in high-dimensional data publishing, Zhang et al. [20] presented the PrivHD method with the junction tree. First, they used exponential mechanism to construct a Markov network; in order to reduce the candidate space, high-pass filtering technique is used in sampling. Then, they used the maximum spanning tree method to build a better joint tree. At last, a high-dimensional synthetic data set is generated for publication. Zhang et al. [21] presented the PrivMN method.

They first constructed a Markov model to express the relationship of features. Then, they used the Laplace mechanism to add noise to the marginal distribution to generate the noisy marginal distribution table. Finally, they used the noisy marginal distribution to generate a synthetic data set for publishing. Gu et al. [22] proposed the PPCA-DP method; they first used the principal component analysis to reduce the dimensionality of high-dimensional data and then added Laplace noise to the low-dimensional projection data; finally, they used the generative model of probabilistic principal component analysis to generate a synthetic data set for publishing. The above are all studies on privacy-preserving data publishing in centralized scenarios.

2.2. Privacy-Preserving Data Publishing in Distributed Environment. At present, most of the existing privacy-preserving data publishing works focus on the centralized scenario; there are fewer studies on privacy-preserving data publishing in distributed scenario. The multiparty data release scenario studied in this paper is that each data owner owns a data set and uses the differential privacy technology to protect the privacy of the local data set rather than the scenario that multiple individuals keep their data locally. The latter typically utilize the local differential privacy [23] techniques to protect the privacy of individual data [24, 25]. In the following, we will introduce the research status of privacy-preserving data release in multiparty data release, where each data owner owns a data set.

Alhadidi et al. [26] proposed the first noninteractive two-party horizontally partitioned data publication method that satisfies differential privacy and secure multiparty computation. The data set published by this method is suitable for classification tasks. Hong et al. [27] constructed the framework (CELS protocol) that enables distributed parties to securely generate outputs while satisfying differential privacy. The security and differential privacy guarantees of the protocol are proved. Ge et al. [28] presented the DPS-PCA algorithm. Data owners collaborated to compute the principal components while protecting the privacy of data. The DPS-PCA algorithm can trade off the relationship between the accuracy of estimating principal components and the degree of privacy protection, but this method only outputs a low-dimensional subspace of high-dimensional sparse data. An efficient and scalable distributed PCA protocol is proposed by Wang et al. [29] for the computation of principal components of split horizon data in a distributed environment. First, the shared data are encrypted and sent to a semitrusted third party. Second, the shared data are aggregated by a semitrusted third party, and the aggregated result is sent to the data consumer. Finally, the data consumer performed a principal component analysis and obtained the principal components of the pooled data. Cheng et al. [30] presented the DP-SUBN^3 approach; the data owners built a Bayesian network with the assistance of a semitrusted curator, and then the Bayesian network is used to generate a synthetic data set. In DP-SUBN^3 approach, the four stages of correlation quantification, structure initialization, structure update, and parameter learning all need to access the local data set, and each stage satisfies differential privacy, which in turn makes the DP-SUBN^3 approach satisfy differential privacy. For the privacy protection of data publishing in arbitrary partitions between two parties, Wang et al. [31] presented the first distributed algorithm, which generates anonymous data from two parties. In order to prevent both parties from leaking private information, the anonymization process satisfies both differential privacy and secure two-party computation. Gu et al. [32] presented the PPCA-DP-MH approach. The data owners collaborate with a semitrusted curator to reduce the dimensionality of the data, and then the data owners used the probabilistic generative model of principal component analysis to generate a published data set. In the PPCA-DP-MH method, since multiple data owners add noise to the data locally and independently, the utility of publishing data gradually decreases as the number of data owners increases. In response to this challenge, we propose the HPDP-DP method in this paper. We design the generation and addition scheme of correlated noise, so that the utility of publishing data will not decrease with the increase of data owners, and even the utility of publishing data will gradually increase with the increase of data owners.

3. Preliminaries

3.1. Probabilistic Principal Component Analysis (PPCA). Principal component analysis is one of the commonly used dimensionality reduction methods. Principal component analysis is a statistical analysis method that converts multiple variables into a few hidden variables through dimensionality reduction techniques. These fewer low-dimensional and not correlated hidden variables are also called principal components. The principal components can reflect most of the information of the original variables. Next, the main process of finding principal components is introduced. First, computing the covariance matrix Σ of the data. Then perform eigenvalue decomposition on the covariance matrix Σ, $\Sigma = U\Lambda U^T$, where Λ is a diagonal matrix and the elements on the diagonal are the eigenvalues of the matrix Σ, $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_p)$, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$. The corresponding eigenvectors are as follows: $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p$ which are called the principal components. U is an orthogonal matrix consisting of the eigenvectors. Usually, the top k principal components retained are determined by the cumulative contribution rate $c = \sum_{i=1}^{k} \lambda_i / \sum_{i=1}^{p} \lambda_i$.

However, Michael et al. [33] proposed that the principal component analysis (PCA) is a nongenerative model, they presented that the principal component analysis (PCA) also has a generative model called probabilistic principal component analysis (PPCA). The most common model to associate low-dimensional latent variables with high-dimensional observable variables is the factor analysis model, i.e. $\mathbf{x} = W\mathbf{s} + \mu + \xi$, where $\mathbf{x}$ is p -dimensional observation vector consisting of the p original variables, $\mathbf{s}$ is a k -dimensional vector consisting of k latent variables, $\xi \sim N(\mathbf{0}, \Psi)$, the matrix W associates the vector $\mathbf{x}$ with the vector $\mathbf{s}$. The vector μ allows the model to have a nonzero mean vector.

Theorem 1 [33]. *From Figure 2 and the latent variable model* $\mathbf{x} = W\mathbf{s} + \mu + \xi$, *when* $\xi \sim N(\mathbf{0}, \sigma^2 I)$, $\mathbf{s} \sim N(\mathbf{0}, I_k)$, *then* $\mathbf{x}|\mathbf{s} \sim N(W\mathbf{s} + \mu, \sigma^2 I_p), \sigma > 0, W \in R^{p\times k}$, *where the maximum likelihood estimation of* μ, σ^2 , *and* W *are*

$$\begin{aligned} \widehat{\boldsymbol{\mu}} &= \tilde{\boldsymbol{\mu}}, \\ \widehat{\sigma}^2 &= \frac{1}{p-k}\sum_{i=k+1}^{p}\lambda_i, \\ \widehat{W} &= U_k\left(\Lambda_k - \widehat{\sigma}^2 I\right)\frac{1}{2}, \end{aligned} \tag{1}$$

where $\tilde{\mu}$ *is the mean vector, the column vectors in* U_k *is the eigenvectors corresponding to the top* k *eigenvalues of the covariance matrix.*

3.2. Differential Privacy. Differential privacy is a strong privacy protection model independent of background knowledge. If the output of a privacy-preserving algorithm is insensitive to small changes in the input, the algorithm satisfies differential privacy. The essence of differential privacy is to randomly perturb the query results, so that people cannot infer the original input information based on the query results.

Definition 1 (Differential Privacy) [10]. A random algorithm $\mathcal{M}$ satisfies ϵ differential privacy, if for any two neighboring data sets $D, \widehat{D}$ (only one record differs between the two data sets) and for any $S(S \in Rang(M))$ there is

$$\left|\ln\frac{P_r\{\mathcal{M}(D) \in S\}}{P_r\left\{\mathcal{M}(\widehat{D}) \in S\right\}}\right| \leq \varepsilon, \tag{2}$$

ε is a small positive real number, which is also called privacy budget.

In the Definition 1, ε is used for controlling the probability ratio of the random algorithm $\mathcal{M}$ to obtain the same output on the two neighboring data sets D and $\widehat{D}$; it reflects the level of privacy protection that the algorithm $\mathcal{M}$ can provide.

Definition 2 (Sensitivity). [10]. Let f be a function that maps a data set into a fixed size vector of real numbers, $f: D \longrightarrow R^d$, for any neighboring data sets D and $\widehat{D}$, the sensitivity of f is defined as follows:

$$\Delta f = \max_{D,\widehat{D}} \|f(D) - f(\widehat{D})\|_1, \tag{3}$$

where $\|\cdot\|_1$ denotes the L_1 norm.

Definition 3 (Laplace mechanism). [34]. For any function $f: D \longrightarrow R^d$, if the random algorithm $\mathcal{M}$ satisfies the equation:

$$\mathcal{M}(D) = f(D) + \left(Lap_1\left(\frac{\Delta f}{\varepsilon}\right), \ldots, Lap_d\left(\frac{\Delta f}{\varepsilon}\right)\right), \tag{4}$$

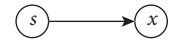


Figure 2: Graphical model for principal component analysis.

then the algorithm $\mathcal{M}$ satisfies ε differential privacy, $\text{Lap}_1(\Delta f/\varepsilon), \ldots, \text{Lap}_d(\Delta f/\varepsilon)$ are independent Laplace random variables.

Theorem 2 [35]. *Let* $Y \sim$ *Laplace* (λ), *then, the distribution of* Y *is infinitely divisible. Furthermore, for every integer* $M \geq 1$, $Y = \sum_{m=1}^{M}(Y_{1m} - Y_{2m})$, *where* Y_{1m} *and* Y_{2m} *are i.i.d. with the Gamma density* $f(x) = ((1/\lambda)^{(1/n)}/\Gamma(1/n))x^{(1/n)-1}e^{-(x/\lambda)}$, $x \geq 0$.

Theorem 3 (Sequential Composition). [34]. *Let* $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_n$ *be a series of privacy algorithms, and their privacy budgets are* $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$, *for the same data set* D, *the combined algorithm* $\mathcal{M}(\mathcal{M}_1(D), \mathcal{M}_2(D), \ldots, \mathcal{M}_n(D))$ *provides* $\sum_{i=1}^{n}\varepsilon_i$ *differential privacy.*

Theorem 4 (Parallel Composition). [34]. *Let* $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_n$ *be a series of privacy algorithms, which privacy budgets are* $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$, $D_1, D_2, \ldots, D_n$ *are disjoint data sets, the combined algorithm* $\mathcal{M}(\mathcal{M}_1(D_1), \mathcal{M}_2(D_2), \ldots, \mathcal{M}_n(D_n))$ *provides* $\max_{1\leq i\leq n}\varepsilon_i$ *differential privacy.*

3.3. Paillier Encryption and Decryption. In this paper, we use Paillier encryption scheme [36] to encrypt the local shared data before being aggregated. The Paillier encryption scheme is described as follows:

(1) Key generation: $n = pq$, where p and q are large primes, $\lambda = \text{lcm}(p-1, q-1)$. Euler function $\Phi(n) = (p-1)(q-1)$, $g \in Z_{n^2}^*$, the (n, g) is public key and λ is private key.

(2) Encryption: plaintext $m < n$, randomly select $r < n$, ciphertext $c = g^m \cdot r^n \bmod n^2$.

(3) Decryption: ciphertext $c < n^2$, plaintext $m = (L(c^\lambda \bmod n^2)/L(g^\lambda \bmod n^2)) \bmod n$, where $L(u) = (u-1)/n$.

Paillier encryption is additively homomorphic. We use $[[m]]$ to represent the encrypted ciphertext of m. Then, $\forall m_1, m_2 \in Z_n, k \in N$, $[[m_1]] \cdot [[m_2]] = [[m_1 + m_2]]$ and $[[m]]^k = [[k \cdot m]]$.

4. The HPDP-DP Method

4.1. Problem Statement. There exist $M(M \geq 2)$ data owners, the m-th data owner P_m holds a local data set denoted as $X_m = \left\{\mathbf{x}_1^m, \mathbf{x}_2^m, \ldots, \mathbf{x}_{N_m}^m\right\}$, N_m is the number of individuals owned by data owner $P_m, m = 1, \ldots, M$, $N = \sum_{m=1}^{M} N_m$. Each individual is a p-dimensional vector. The data sets $X_1, X_2, \ldots, X_M$ can be viewed as horizontally split the integrated data set $X = \cup_{m=1}^{M} X_m$ by M data owners. That is all the local data sets have the same attributes and do not intersect with each other. Our goal is to design an algorithm that can publish these horizontally partitioned data sets privately; specifically, it is that with the assistance of a

Input: Data sets X_m, $m = 1, 2, \ldots, M$. Private key λ, public key (n, g). θ_m $(m = 0, 1, 2, \ldots, M)$, where $\theta_0 \cdot \theta_1 \cdot \cdots \cdot \theta_M = 1$. Privacy budget ε and cumulative contribution rate c

Output: Synthetic data set $\widetilde{X} = \cup_{m=1}^{M} \widetilde{X}_m$

(1) **for** $m = 1$ to M **do**
(2) Data owner generates $p \times p$ noise matrices $B_{m1} = (b_{ij}^{m1})_{p\times p}$ and $B_{m2} = (b_{ij}^{m2})_{p\times p}$, let B_{m1} and B_{m2} be the symmetric matrix with the upper triangle (including the diagonal) entries are sampled from Gamma $(1/M, p + p^2/M\varepsilon)$, and set $b_{ji}^{m1} = b_{ij}^{m1}, b_{ji}^{m2} = b_{ij}^{m2}, \forall i < j$.
(3) Compute: $L_m = (l_{ij}^m)_{p\times p} = \sum_{k=1}^{N_m} (\mathbf{x}_k^m - \mu^m)(\mathbf{x}_k^m - \mu^m)^T$
(4) Compute: $\widetilde{L}_m = (\widetilde{l}_{ij}^m)_{p\times p} = (l_{ij}^m + b_{ij}^{m1} - b_{ij}^{m2})_{p\times p}$
(5) **for** $i = 1$ to p **do**
(6) **for** $j = 1$ to p **do**
(7) Compute: $\theta_m \cdot [[\widetilde{l}_{ij}^m]] \leftarrow \theta_m \cdot g^{l_{ij}^m + b_{ij}^{m1} - b_{ij}^{m2}} \cdot r^n \bmod n^2$
(8) **end for**
(9) **end for**
(10) **end for**
(11) **return** $C_m = (\theta_m \cdot [[\widetilde{l}_{ij}^m]])_{p\times p}, m = 1, 2, \ldots, M$
(12) Compute the Hadamard product: $C \leftarrow (\theta_0)_{p\times p} \circ C_1 \circ C_2 \circ \cdots \circ C_M$
(13) Decrypt C: $\widetilde{L} = (\sum_{m=1}^{M} (l_{ij}^m + b_{ij}^{m1} - b_{ij}^{m2}))_{p\times p} \leftarrow C$
(14) Compute: $\Sigma = (1/N)\widetilde{L}$
(15) Eigenvalue decomposition of matrix Σ, return eigenvalues in descending order $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$, and corresponding eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p$
(16) **for** $k = 1$ to p **do**
(17) **if** $\sum_{i=1}^{k} \lambda_i / \sum_{i=1}^{p} \lambda_i \geq c$ **then**
(18) $\Lambda_k = (\lambda_1, \lambda_2, \ldots, \lambda_k)$
(19) $U_k = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k)$
(20) **end if**
(21) **end for**
(22) **return** Λ_k, U_k
(23) **for** $m = 1$ to M **do**
(24) Compute $S_m = X_m \times U_k$
(25) Use the model defined in Theorem 1 to generate a synthetic data set $\widetilde{X}_m$
(26) **end for**
(27) **return** $\widetilde{X} = \cup_{m=1}^{M} \widetilde{X}_m$

Algorithm 1: HPDP-DP algorithm.

semitrusted curator, the M data owners and the curator collaborate to publish a synthetic data set $\widetilde{X} = \cup_{m=1}^{M} \widetilde{X}_m$, which has the same scale and statistical properties as the data set $X = \cup_{m=1}^{M} X_m$. Typically, we assume that the data owners and the curator are honest-but-curious, that is, they will follow the protocol but try to find out as much secret information as possible.

In view of the above scenario, we propose a horizontally partitioned data publishing method with differential privacy (HPDP-DP). The Algorithm 1 depicts the HPDP-DP algorithm. First, the data owner perturbs the local scatter matrix with random noise that obeys the Gamma distribution and sends it to the semitrusted curator. Then the semitrusted curator aggregates all the local scatter matrices to get the noisy estimator of the covariance matrix of the pooled data. The semitrusted curator performs eigenvalue decomposition on the covariance matrix to get the principal components and then the top k principal components are sent to each data owner. At last, each data owner uses the top k principal components and the generative model of probabilistic principal component analysis to generate a synthetic data set.

In order to reduce the impact of noise on the availability of published data, the HPDP-DP algorithm employs a distributed Laplace mechanism to add noise to the local scatter matrix. According to Theorem 2, the infinite additivity of Laplace distribution, we perturb the local scatter matrix with the noise follows a Gamma distribution, which makes the estimator of the covariance matrix of the pooled data contain the same level of noise as the centralized scene. Inspired by [37], since the step of perturbing the local scatter matrix with gamma-distributed noise does not satisfy differential privacy, we will use the Paillier encryption scheme to encrypt the perturbed scatter matrix to protect the privacy of local data. The HPDP-DP algorithm mainly consists of the following stages.

Initialization phase: in the initialization phase, the Paillier cryptographic system generates the public key (n, g) and the private key λ. The system also generates $M + 1$ factors $\theta_0, \theta_1, \ldots, \theta_M$, where $\theta_m \in Z_{n2}, m = 0, 1, 2, \ldots, M$ and $\theta_0 \cdot \theta_1 \cdot \cdots \cdot \theta_M = 1$. The factor θ_0 and the private key λ are secretly sent to the curator. The public key (n, g) and θ_m are secretly sent to the data owner P_m, $m = 1, 2, \ldots, M$.

Perturbation and encryption phase: each data owner randomly perturbs the local scatter matrix. The scatter matrix of the data owner P_m is given by

$$L_m = \left(l_{ij}^m\right)_{p\times p} = \sum_{k=1}^{N_m} \left(\mathbf{x}_k^m - \mu^m\right)\left(\mathbf{x}_k^m - \mu^m\right)^T = \sum_{k=1}^{N_m} \mathbf{x}_k^m \left(\mathbf{x}_k^m\right)^T - N_m \mu^m \left(\mu^m\right)^T. \tag{5}$$

where $\mu^m = (1/N_m)\sum_{k=1}^{N_m} \mathbf{x}_k^m$.

The data owner P_m generates two $p \times p$ symmetric random matrices $B_{m1} = (b_{ij}^{m1})_{p\times p}$ and $B_{m2} = (b_{ij}^{m2})_{p\times p}$; b_{ij}^{m1} and b_{ij}^{m2} are sampled from Gamma$((1/M), (p + p^2)/M\varepsilon)$, $1 \le i \le j \le p$. Then, the local noisy scatter matrix is $\widetilde{L}_m = (\widetilde{l}_{ij}^m)_{p\times p} = (l_{ij}^m + b_{ij}^{m1} - b_{ij}^{m2})_{p\times p}$. Using the public key (n, g) and θ_m to encrypt each element of $\widetilde{L}_m$ to get the encrypted matrix $C_m = (\theta_m[[\widetilde{l}_{ij}^m]])_{p\times p} = (\theta_m \cdot g^{l_{ij}^m + b_{ij}^{m1} - b_{ij}^{m2}} \cdot r^n \cdot \mathrm{mod} n^2)_{p\times p}$ which will be sent to the curator, $m = 1, 2, \ldots, M$.

Aggregation and decryption phase: After receiving these encrypted matrices $C_1, C_2, \ldots, C_M$, the curator performs the Hadamard product on these encrypted matrices. We use the symbol $\circ$ as the Hadamard product of matrices.

$$\begin{aligned}
(\theta_0)_{p\times p} \circ C_1 \circ C_2 \circ \cdots \circ C_M &= \left(\theta_0 \prod_{m=1}^{M} \theta_m \cdot g^{l_{ij}^m + b_{ij}^{m1} - b_{ij}^{m2}} \cdot r^n \cdot \mathrm{mod} n^2\right)_{p\times p} \\
&= \left(\prod_{m=1}^{M} \cdot g^{l_{ij}^m + b_{ij}^{m1} - b_{ij}^{m2}} \cdot r^n \cdot \mathrm{mod} n^2\right)_{p\times p} \\
&= \left(g^{\sum_{m=1} M l_{ij}^m + \sum_{m=1}^{M}\left(b_{ij}^{m1} - b_{ij}^{m2}\right)} \cdot r^{Mn} \cdot \mathrm{mod} n^2\right)_{p\times p} \\
&= \left(g^{\sum_{m=1} M l_{ij}^m + \mathrm{Lap}\left(p + p^2/\varepsilon\right)} \cdot r^{Mn} \cdot \mathrm{mod} n^2\right)_{p\times p} \\
&= \left(\left[\left[\sum_{m=1}^{M} l_{ij}^m + \mathrm{Lap}\left(\frac{p + p^2}{\varepsilon}\right)\right]\right]\right)_{p\times p},
\end{aligned} \tag{6}$$

where $\sum_{m=1}^{M}(b_{ij}^{m1} - b_{ij}^{m2}) \sim \mathrm{Lap}((p + p^2)/\varepsilon)$ holds due to Theorem 2. The curator decrypts the above results to get the sum of local scatter matrices with Laplace noise $\widetilde{L} = \sum_{m=1}^{M} \widetilde{L}_m = \sum_{m=1}^{M} L_m + (\mathrm{Lap}((p + p^2)/\varepsilon))_{p\times p}$, which is used as an estimation of the scatter matrix of the pooled data, and then the estimation of the covariance matrix of the pooled data is $\Sigma = (1/N)\widetilde{L}$.

In this stage, our idea is to use the weighted average of the local covariance matrices to estimate the covariance matrix of the pooled data. Assuming that the covariance matrix of data owner P_m is $\widetilde{\Sigma}_m$, the relationship with the scatter matrix is $\widetilde{\Sigma}_m = (\widetilde{L}_m/N_m)$, and then the estimation of the covariance matrix of the pooled data is $\Sigma = \sum_{m=1}^{M}(N_m/N)\widetilde{\Sigma}_m = (1/N)\sum_{m=1}^{M}\widetilde{L}_m = (1/N)\widetilde{L}$.

Principal component analysis phase: the curator performs eigenvalue decomposition on matrix Σ . The curator gets the eigenvectors (the top k principal components) $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k$ and then sends them to each data owner.

Generate synthetic data set phase: Each data owner uses the returned top k principal components and the generative model of probabilistic principal component analysis in Theorem 1 to generate a synthetic data set.

4.2. Analysis

4.2.1. Security Analysis

Theorem 5. *The data set owned by P_m is X_m and its corresponding scatter matrix is $L_m = (l_{ij}^m)_{p\times p}$, $m = 1, 2, \ldots, M$. Defining the query function,*

$$f(X_1, X_2, \ldots, X_M) = \sum_{m=1}^{M} L_m, \tag{7}$$

the output result intended to be protected. $B_{m1} = (b_{ij}^{m1})_{p\times p}$ and $B_{m2} = (b_{ij}^{m2})_{p\times p}$ are symmetric random matrices will be added to L_m, b_{ij}^{m1} and b_{ij}^{m2} are sampled from Gamma$((1/M), (p + p^2/M\varepsilon))$, $1 \le i \le j \le p$. If the random algorithm $\mathscr{M}$ holds

$$\mathscr{M}(X_1, X_2, \ldots, X_M) = f(X_1, X_2, \ldots, X_M) + \sum_{m=1}^{M}(B_{m1} - B_{m2}), \tag{8}$$

then the algorithm $\mathscr{M}$ satisfies ε differential privacy.

Proof. According to Theorem 2, it can be known each element of $\sum_{t=1}^{M}(B_{m1} - B_{m2})$ obeys $\mathrm{Lap}(p(1+p)/\varepsilon)$. So, next we will prove if algorithm $\mathscr{M}$ holds

$$\mathscr{M}(X_1, X_2, \ldots, X_M) = f(X_1, X_2, \ldots, X_M) + B, \tag{9}$$

$B = (b_{ij})_{p\times p}$ is a symmetric random matrix and b_{ij} is sampled from $\mathrm{Lap}(p(1+p)/\varepsilon)$, $1 \le i \le j \le p$, then the algorithm $\mathscr{M}$ satisfies ε differential privacy.

We denote the two neighboring data sets as $X = \cup_{m=1}^{M} X_m$ and $\widehat{X} = \cup_{m=1}^{M} \widehat{X}_m$; there is only one individual is different, without losing general assumption, suppose the different individuals are in X_M and $\widehat{X}_M$. We denote the only two different individuals as $\mathbf{x}_{N_M}^{M} \in X_M$ and $\widehat{\mathbf{x}}_{N_M}^{M} \in \widehat{X}_M$. Assume that all individual data have been normalized to the [0,1] interval. The estimation of the scatter matrices of X and $\widehat{X}$ are as follows:

$$L = \sum_{m=1}^{M} L_m = \sum_{m=1}^{M} \left(l_{ij}^{m}\right)_{p\times p}, \tag{10}$$

and

$$\widehat{L} = \sum_{m=1}^{M-1} L_m + \widehat{L}_M = \sum_{m=1}^{M-1} \left(l_{ij}^{m}\right)_{p\times p} + \left(\widehat{l}_{ij}^{M}\right)_{p\times p}. \tag{11}$$

Let $B = (b_{ij})_{p\times p}$ and $\widehat{B} = (\widehat{b}_{ij})_{p\times p}$ be two independent symmetric random matrices, where b_{ij} and $\widehat{b}_{ij}$ are sampled from $\mathrm{Lap}(p(1+p)/\varepsilon)$, $1 \le i \le j \le p$.

Let $S = L + B$ and $\widehat{S} = \widehat{L} + \widehat{B}$, then the log ratio of the probabilities of S and $\widehat{S}$ at a point H is given by

$$\left|\ln \frac{P\{H|X\}}{P\{H|\widehat{X}\}}\right| = \left|\ln \frac{P\{H - L|X\}}{P\{H - \widehat{L}|\widehat{X}\}}\right|. \tag{12}$$

According to the definition of differential privacy (Definition 1), we need to prove that the following inequalities holds:

$$\left|\ln \frac{P\{H|X\}}{P\{H|\widehat{X}\}}\right| = \left|\ln \frac{P\{H - L|X\}}{P\{H - \widehat{L}|\widehat{X}\}}\right| \le \varepsilon. \tag{13}$$

The mean vectors of X_M and $\widehat{X}_M$ are as follows:

$$\boldsymbol{\mu}^{M} = \frac{1}{N_M} \sum_{k=1}^{N_M} \mathbf{x}_k^{M}, \tag{14}$$

and

$$\widehat{\boldsymbol{\mu}}^{M} = \frac{1}{N_M} \left(\sum_{k=1}^{N_M - 1} \mathbf{x}_k^{M} + \widehat{\mathbf{x}}_{N_M}^{M} \right), \tag{15}$$

so $\widehat{\mu}^{M} = \mu^{M} + (1/N_M)(\widehat{x}_{N_M}^{M} - \mathbf{x}_{N_M}^{M})$. Hence, we have the following:

$$\begin{aligned}
\left|l_{ij}^{M} - \widehat{l}_{ij}^{M}\right| &= \left|\sum_{k=1}^{N_M} x_{ik}^{M} x_{jk}^{M} - N_M \mu_i^{M} \mu_j^{M} - \left(\sum_{k=1}^{N_M - 1} x_{ik}^{M} x_{jk}^{M} + \widehat{x}_{iN_M}^{M} \widehat{x}_{jN_M}^{M} - N_M \widehat{\mu}_i^{M} \widehat{\mu}_j^{M} \right)\right| \\
&= \left| x_{iN_M}^{M} x_{jN_M}^{M} - \widehat{x}_{iN_M}^{M} \widehat{x}_{jN_M}^{M} + N_M \left(\widehat{\mu}_i^{M} \widehat{\mu}_j^{M} - \mu_i^{M} \mu_j^{M}\right)\right| \\
&= \left| x_{iN_M}^{M} x_{jN_M}^{M} - \widehat{x}_{iN_M}^{M} \widehat{x}_{jN_M}^{M} + \mu_i^{M} \left(\widehat{x}_{jN_M}^{M} - x_{jN_M}^{M}\right) + \mu_j^{M} \left(\widehat{x}_{iN_M}^{M} - x_{iN_M}^{M}\right) + \frac{1}{N_M} \left(\widehat{x}_{iN_M}^{M} - x_{iN_M}^{M}\right)\left(\widehat{x}_{jN_M}^{M} - x_{jN_M}^{M}\right)\right| \\
&= \left|\left(x_{iN_M}^{M} - \widehat{x}_{iN_M}^{M}\right)\left(x_{jN_M}^{M} - \mu_j^{M}\right) + \left(x_{jN_M}^{M} - \widehat{x}_{jN_M}^{M}\right)\left(\widehat{x}_{iN_M}^{M} - \mu_i^{M}\right)\right| \\
&\le \left|\left(x_{iN_M}^{M} - \widehat{x}_{iN_M}^{M}\right)\left(x_{jN_M}^{M} - \mu_j^{M}\right)\right| + \left|\left(x_{jN_M}^{M} - \widehat{x}_{jN_M}^{M}\right)\left(\widehat{x}_{iN_M}^{M} - \mu_i^{M}\right)\right| \le 2.
\end{aligned} \tag{16}$$

Therefore, the following formula holds:

$$\begin{aligned}
\left|\ln \frac{P\{H|X\}}{P\{H|\widehat{X}\}}\right| &= \left|\ln \frac{P\{H - L|X\}}{P\{H - \widehat{L}|\widehat{X}\}}\right| \\
&= \frac{\varepsilon}{p(1+p)} \sum_{1\le i\le j\le p} \left(\left|h_{ij} - \widehat{l}_{ij}^{M}\right| - \left|h_{ij} - l_{ij}^{M}\right| \right) \\
&\le \frac{\varepsilon}{p(1+p)} \sum_{1\le i\le j\le p} \left|l_{ij}^{M} - \widehat{l}_{ij}^{M}\right| \\
&\le \frac{\varepsilon}{p(1+p)} p(1+p) = \varepsilon.
\end{aligned} \tag{17}$$

So the conclusion of Theorem 5 holds.

Security against external attacks: external attacker will eavesdrop on data sent by local data owners to the curator. According to the semantic security of Paillier encryption against plaintext attacks, external attacker unable to decrypt data $(\theta_m \cdot g^{l_{ij}^{m} + b_{ij}^{m1} - b_{ij}^{m2}} \cdot r^{n} \cdot \mathrm{mod} n^2)_{p\times p}$ without knowing private key λ and θ_m, $1 \le m \le M$. External attacker may also eavesdrop on the aggregated value of the data owners $(g^{\sum_{m=1}^{M} l_{ij}^{m} + b_{ij}^{m1} - b_{ij}^{m2}} \cdot r^{Mn} \cdot \mathrm{mod}\, n^2)_{p\times p}$, external attacker unable to decrypt data without knowing private key λ. Even though the external attacker get the sum of scatter matrices with noise $(\sum_{m=1}^{M} l_{ij}^{m} + b_{ij}^{m1} - b_{ij}^{m2})_{p\times p}$, because it contains Laplace noise, so the local data are still safe according to Theorem 5. *Security against internal attacks*: internal adversaries are data owners and the curator. The data owner P_m holds θ_m

secretly, the rest of the data owners and the curator cannot decrypt $(\theta_m \cdot g^{l_{ij}^m + b_{ij}^{m1} - b_{ij}^{m2}} \cdot r^n \cdot \mathrm{mod} n^2)_{p\times p}$ without private key λ and θ_m unless the curator colluded with the $M-1$ data owners. The curator can use private key λ and θ_0 to decrypt the aggregated value $(\prod_{m=1}^{M} (\theta_m \cdot g^{l_{ij}^m + b_{ij}^{m1} - b_{ij}^{m2}} \cdot r^n \cdot \mathrm{mod} n^2)_{p\times p}$, but the curator can only get the aggregated value with Laplace noise, so the local data are safe according to Theorem 5. □

4.2.2. Complexity Analysis. Computation time cost analysis: the total time complexity of Algorithm 1 is $O(Mp^2 + Mn)$, where M is the number of data owners, p is the number of attributes, $n = n_1 + n_2 + \cdots + n_M$, n_m is the number of samples owned by data owner P_m, $m = 1, 2, \cdot, M$. It is due to the following facts. In Algorithm 1, the major computational cost of Algorithm 1 is reflected in lines 1–11, lines 16–21, and lines 23–26. The lines $1-11$ are to perturb and encrypt the scatter matrix of the local data of the M data owners, and the time complexity is $O(Mp^2)$. The lines $16-21$ are to perform principal component analysis on the aggregated scatter matrix, and its time complexity is $O(K)$, where K is the number of retained principal components, which is proportional to p, so the complexity is $O(p)$. The lines $23-26$ are that each data owner uses Theorem 1 to generate a published data set, and the time complexity is $O(Mn_1 + Mn_2 + \cdot + Mn_M) = O(Mn)$. In summary, the time complexity of Algorithm 1 is $O(Mp^2 + p + Mn)$, which is $O(Mp^2 + Mn)$.

Communication cost analysis. There exist three stages that incur communication costs. The first stage is the M data owners send the local scatter matrix to the curator, the size of the message sent by each data owner is p^2, the total size of the message sent in this stage is Mp^2. The second stage is the curator sends the top K eigenvalues and their corresponding eigenvectors to each data owner; the total size of the message sent in this stage is MpK^2. The third stage is each data owner sends the synthetic data set to the curator; the size of the message sent by data owner P_m is $n_m p, m = 1, 2, \ldots, M$; the total size of the message sent during this stage is $np = (n_1 + n_2 + \cdots + n_M)p$.

5. Experiment

In this section, we experimentally evaluate the performance of HPDP-DP algorithm by comparing with the DP-SUBN^3 algorithm [30]. We conduct experiments on different real data sets that are NLTCS [38] and Adult [39] data sets. NLTCS data set contains 21574 individuals, each individual has 16 attributes. Adult data set contains 45222 individuals, each individual has 15 attributes. We use the method in [30] to preprocess the Adult data set. After processing, the number of attributes in the Adult data set is 52. We use SVM classification accuracy to evaluate the performance of HPDP-DP algorithm. We train multiple classifiers on published synthetic data sets. For NLTCS data set, predicting whether a person is unable to go outside and whether a person is unable to manage money. For Adult data set, predicting whether a person holds a postsecondary degree and whether a person earns more than $50K$. In each classification task, we use 20% of the individuals as the test set and 80% of the individuals as the training set. Each experiment is run five times, and the average results are reported. The number of retained principal components is determined by the cumulative contribution rate c. The cumulative contribution rate c is set to 0.8 for NLTCS data set and 0.95 for Adult data set. In order to measure the performance of the HPDP-DP algorithm more clearly, the same SVM classifier are trained on the original data set; we label the SVM classification accuracy on the original data set with "No Privacy."

5.1. The Impact of the Number of Principal Components Retained on the SVM Classification Accuracy. In this section, we train multiple classifiers to study the influence of the number of principal components retained on the SVM classification accuracy. In this set of experiments, the number of data owners is set to 3; the privacy budget ε is set to 0.5.

For the Adult data set, Figures 3(a) and 3(c) show the cumulative contribution rate and individual contribution rate of the principal components. Because there are more attributes after preprocessing the Adult data set, so we only marked the corresponding SVM classification accuracy when the number of retained principal components k are $5, 10, 15, 20, 25, 30, 35$, and 40 in Figures 3(b) and 3(d). For the NLTCS data set, it can be seen from Figures 3(e) and 3(g) that the contribution rate of only the first principal component has reached more than 30%. The cumulative contribution rate of the top seven principal components can reach 80%, and it can be seen from Figures 3(f) and 3(h) that the corresponding SVM classification accuracy can reach more than 80%.

The common conclusion is that when the cumulative contribution rate increases (the number of principal components retained increases), the SVM classification accuracy increases accordingly. This phenomenon is consistent with the principle of principal component analysis. The principal components are not correlated with each other and contain the information of the original data. The more principal components retained, the more information of the original data contained in the published data, and the better the performance of the published data set.

5.2. Performance Comparison of HPDP-DP and DP-SUBN^3 with Different Privacy Budgets. In this part of the experiments, we fixed the number of data owners to three while making the privacy budget ε take different values. Figure 4 shows the impact of privacy budgets on HPDP-DP and DP-SUBN^3 algorithms. Figures 4(a) and 4(b) show the SVM classification accuracy of the HPDP-DP and DP-SUBN^3 algorithms on Adult data set. Figures 4(c) and 4(d) show the SVM classification accuracy of the HPDP-DP and DP-SUBN^3 algorithms on NLTCS data set. From Figure 4, except for the salary classifier of the Adult data set, the performance of HPDP-DP algorithm is

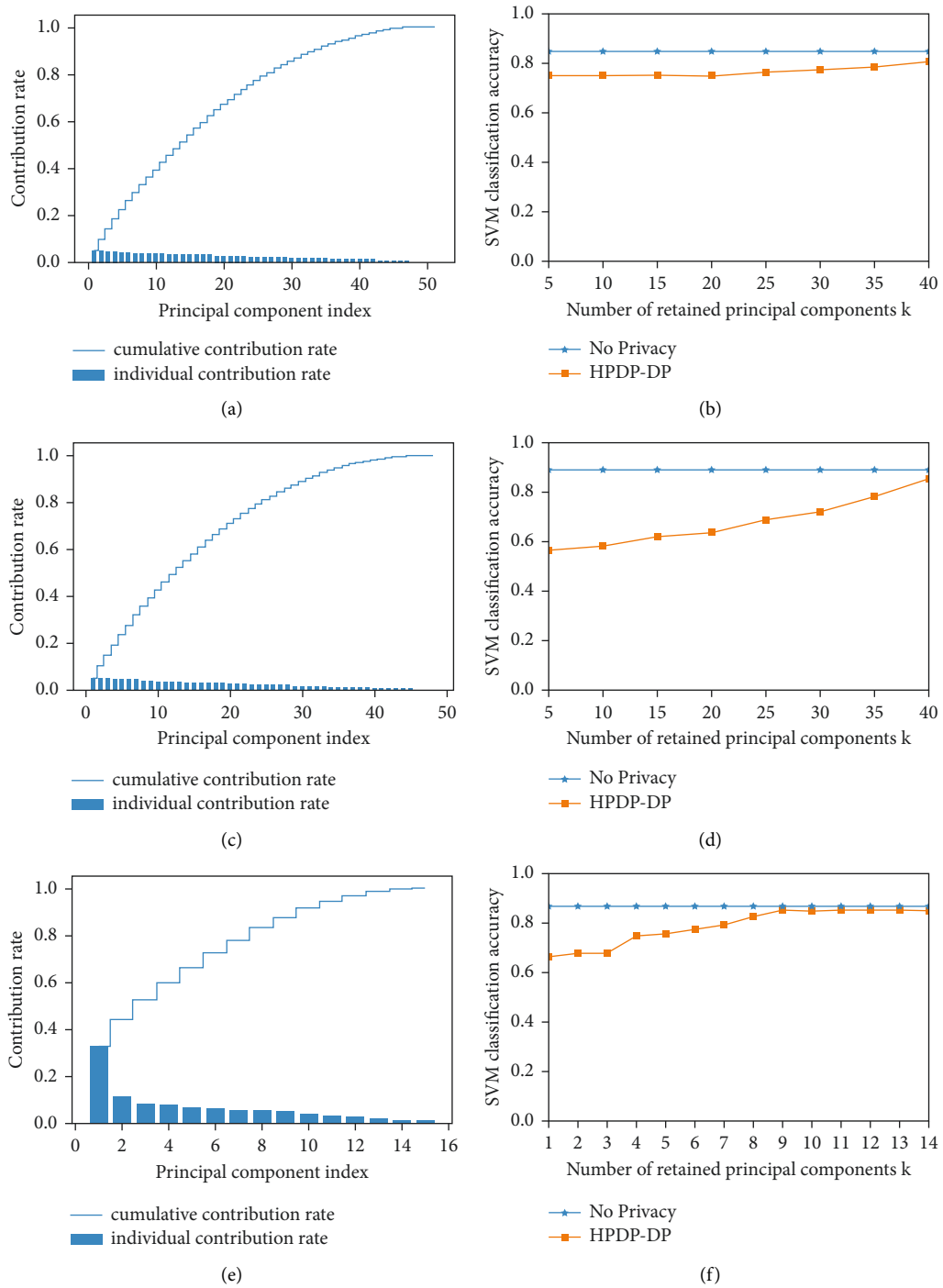


Figure 3: Continued.

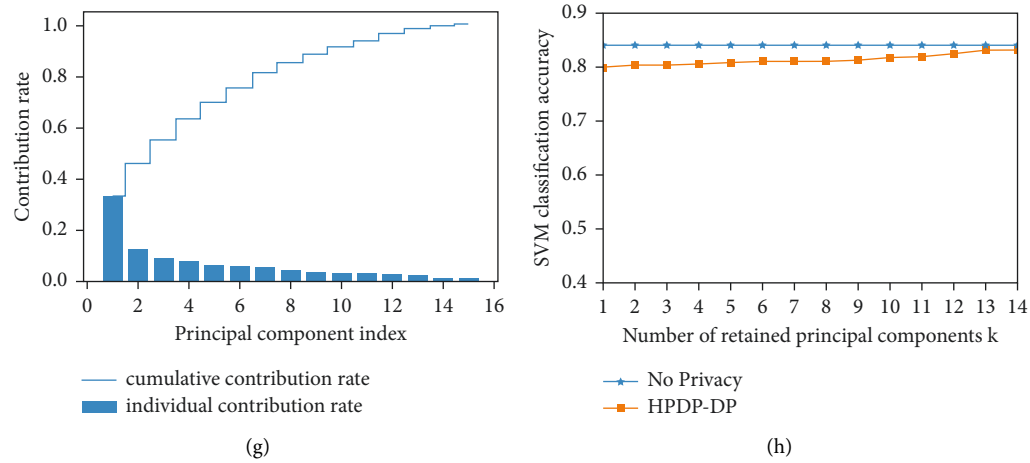


Figure 3: The impact of the number of principal components retained on the SVM classification accuracy. (a) Adult, Y = salary. (b) Adult, Y = salary. (c) Adult, Y = education. (d) Adult, Y = education. (e) NLTCS, Y = money. (f) NLTCS, Y = money. (g) NLTCS, Y = outside. (h) NLTCS, Y = outside.

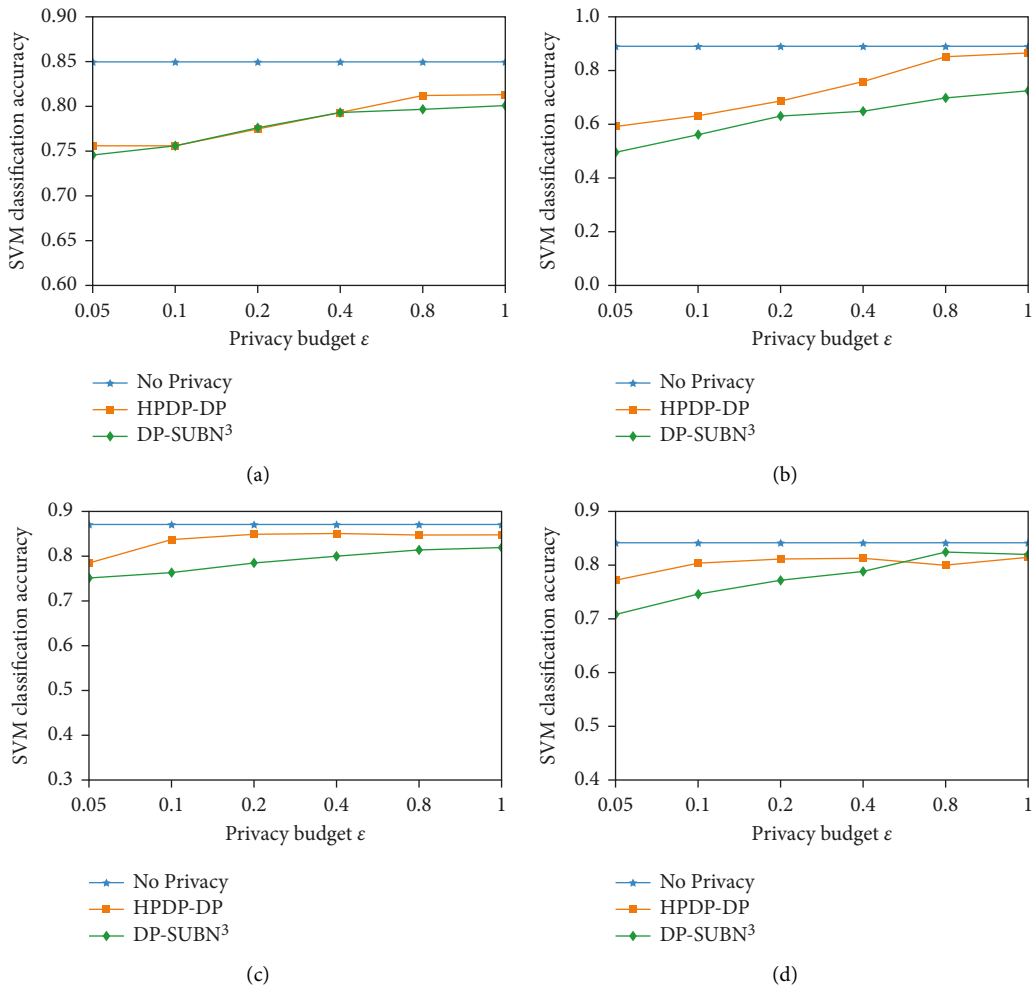


Figure 4: Performance comparison of HPDP-DP and DP-SUBN ^{3} with different privacy budgets. (a) Adult, Y = salary. (b) Adult, Y = education. (c) NLTCS, Y = money. (d) NLTCS, Y = outside.

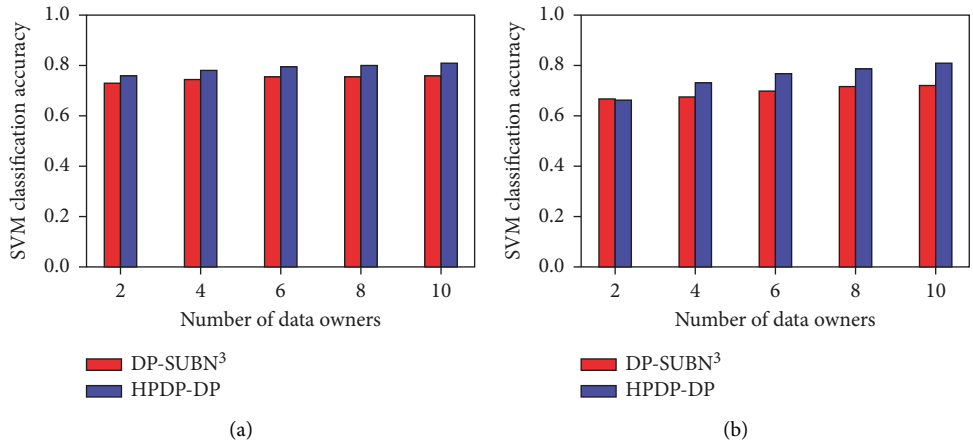


Figure 5: The impact of the number of data owners on the SVM classification accuracy. (a) Adult, Y = salary. (b) Adult, Y = education.

significantly better than DP-SUBN^3 algorithm. Even for the salary classifier of the Adult data set, the SVM classification accuracy of HPDP-DP algorithm is still not lower than DP-SUBN^3 algorithm. From Figure 4, the experimental results show that the SVM classification accuracy of both synthetic data sets released by HPDP-DP and DP-SUBN^3 algorithms increases with the increase of the privacy budget. This is because, according to the definition of differential privacy, when the privacy budget ε increases, the degree of privacy protection decreases and the availability of the released data increases.

5.3. The Impact of the Number of Data Owners on the SVM Classification Accuracy. In order to study the effect of the number of data owners on the performance of the HPDP-DP algorithm, in this section, we set the number of data owners to 2, 4, 6, 8, and 10. We fix the privacy budget ε to 0.2. The results in Figure 5 show that the performance of HPDP-DP algorithm is better than that of DP-SUBN^3 algorithm. We can observe that when the number of data owners increases, the SVM classification accuracy of the synthetic data sets released by HPDP-DP and DP-SUBN^3 algorithms increases accordingly. For DP-SUBN^3 algorithm, the reason is that when the number of data owners increases, the number of update iterations in DP-SUBN^3 algorithm increases, which helps to get better Bayesian network. For HPDP-DP algorithm, we use the weighted average of the local covariance matrices as an estimate of the covariance matrix of the pooled data, and the estimation effect will get better as the number of data owners increases. At the same time, we use the distributed Laplace mechanism to add noise to the shared data, so even when the number of data owners increases, the aggregated result still contain only one share of random noise (the same level as the centralized scene). The scale of random noise is determined only by the privacy budget and the sensitivity. Therefore, the SVM classification accuracy of the synthetic data set released by HPDP-DP algorithm increases as the number of data owners increases.

6. Conclusion

In this paper, in order to privately publish the horizontally partitioned data owned by multiple parties, we present a multiparty horizontally partitioned data publishing method with differential privacy. We use the weighted average of the covariance matrices of the local data to estimate the covariance matrix of the pooled data and then obtain the principal components of the pooled data. In order to protect the privacy of the local data and improve the utility of the published data, we exploit the infinite divisibility of the Laplace distribution to add noise to the locally shared data to improve the utility of the published data. The experimental results show that the synthetic data set released by the HPDP-DP algorithm can maintain high utility. However, this paper also has limitations. (1) The principal component analysis is only suitable for linear dimensionality reduction and not for nonlinear dimensionality reduction. (2) The HPDP-DP algorithm is only suitable for horizontally partitioned data publishing, not for vertically partitioned data publishing. We will conduct research on these aspects in the future.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] J. Kolter and T. Jaakkola, "Approximate inference in additive factorial hmms with application to energy disaggregation," *International Conference on Artificial Intelligence and Statistics*, vol. 04, pp. 1472–1482, 2012.

[2] R. Lu, X. Liang, L. Xu, X. Lin, and X. Shen, "Eppa: an efficient and privacy-preserving aggregation scheme for secure smart grid communications," *IEEE Transactions on*

Parallel and Distributed Systems, vol. 23, no. 9, pp. 1621–1631, 2012.

[3] C. Wang, D. Wang, G. Xu, and D. He, "Efficient privacy-preserving user authentication scheme with forward secrecy for industry 4.0," *Science China Information Sciences*, vol. 65, no. 1, pp. 767–784, 2020.

[4] Y. T. Tsou, "PPDCA: privacy-preserving crowdsourcing data collection and analysis with randomized response," *IEEE Access*, vol. 6, pp. 76970–76983, 2018.

[5] X. Ren, C. M. Yu, W. Yu et al., "High-dimensional crowdsourced data publication with local differential privacy," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 9, pp. 2151–2166, 2018.

[6] L. Sweeney, "k-anonymity: a model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[7] Z. Li and D. Wang, "Achieving one-round password-based authenticated key exchange over lattices," *IEEE Transactions on Services Computing*, vol. 15, no. 1, pp. 308–321, 2022.

[8] Z. Li, D. Wang, and E. Morais, "Quantum-safe round-optimal password authentication for mobile devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 3, pp. 1885–1899, 2020.

[9] Q. Wang, D. Wang, C. Cheng, and D. He, "Quantum2fa: Efficient quantum-resistant two-factor authentication scheme for mobile devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 24, p. 1, 2021.

[10] C. Dwork, F. Mcsherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," *Journal of Privacy and Confidentiality*, vol. 7, no. 3, pp. 17–51, 2017.

[11] C. Han and K. Wang, "Sensitive disclosures under differential privacy guarantees," *IEEE International Congress on Big Data*, vol. 25, pp. 110–117, 2015.

[12] Q. Wang, Y. Zhang, L. Xiao, Z. Wang, and K. Ren, "Rescuedp: real-time spatio-temporal crowd-sourced data publishing with differential privacy," in *Proceedings of the IEEE Infocom -the IEEE International Conference on Computer Communications*, 10-14 April 2016.

[13] W. Hao and Z. XuCts-dp, "Publishing correlated time-series data via differential privacy," *Knowledge-Based Systems*, vol. 122, pp. 167–179, 2017.

[14] H. Wang and H. Wang, "Correlated tuple data release via differential privacy," *Information Sciences*, vol. 560, pp. 347–369, 2021.

[15] S. Chen, A. Fu, S. Yu, H. Ke, and M. Su. Dp-qic, "A differential privacy scheme based on quasi-identifier classification for big data publication," *Soft Computing*, vol. 25, no. 3, p. 2021, 2021.

[16] X. Jiang, Z. Ji, S. Wang, N. Mohammed, S. Cheng, and L. Ohno-Machado, "Differential-private data publishing through component analysis," *Transactions on data privacy*, vol. 6, no. 1, pp. 19–34, 2013.

[17] J. Zhang, G. Cormode, C. M. Procopiuc, and D. Srivastava, "PrivBayes," *ACM Transactions on Database Systems*, vol. 42, no. 4, pp. 1–41, 2017.

[18] C. Rui, X. Qian, Z. Yu, and J. Xu, "Differentially private high-dimensional data publication via sampling-based inference," in *Proceedings of the 21th ACM SIGKDD International Conference*, 2015.

[19] C. Xu, J. Ren, Y. Zhang, Z. Qin, and K. Ren, "Dppro: differentially private high-dimensional data release via random projection," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 12, pp. 3081–3093, 2017.

[20] X. Zhang, L. Chen, K. Jin, and X. Meng, "Private high-dimensional data publication with junction tree," *Journal of Computer Research and Development*, vol. 55, no. 12, pp. 2794–2809, 2018.

[21] W. Zhang, J. Zhao, F. Wei, and Y. Chen, "Differentially private high-dimensional data publication via Markov network," *ICST Transactions on Security and Safety*, vol. 6, no. 19, p. 159626, 2019.

[22] Z. Gu, G. Zhang, C. Ma, and L. Song, "Differential privacy data publishing method based on the probabilistic principal component analysis," *Journal of Harbin Engineering University*, vol. 42, no. 8, pp. 1217–1223, 2021.

[23] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *Proceedings of the IEEE Symposium on Foundations of Computer Science*, 26-29 October 2013.

[24] T. T. Nguyên, X. Xiao, Y. Yang, S. C. Hui, H. Shin, and J. Shin, "Collecting and analyzing data from smart device users with local differential privacy," p. 11, 2016, http://arxiv.org/abs/1606.05053.

[25] Y. Sei, J. Andrew, H. Okumura, and A. Ohsuga, "Privacy-preserving collaborative data collection and analysis with many missing values," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2022.

[26] D. Alhadidi, N. Mohammed, B. Fung, and M. Debbabi, "Secure distributed framework for achieving-differential privacy," *Springer, Berlin, Heidelberg*, vol. 15, no. 4, pp. 316–333, 2012.

[27] Y. Hong, J. Vaidya, H. Lu, P. Karras, and S. Goel, "Collaborative search log sanitization: toward differential privacy and boosted utility," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 5, pp. 504–518, 2015.

[28] J. Ge, Z. Wang, M. Wang, and L. Han, "Minimax-optimal privacy-preserving sparse pca in distributed systems," in *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, pp. 1589–1598, Playa Blanca, Lanzarote, Canary Islands, April 9 - 11, 2018.

[29] S. Wang and J. M. Chang, "Differentially private principal component analysis over horizontally partitioned data," in *Proceedings of the 2018 IEEE Conference on Dependable and Secure Computing*, 10-13 December 2018.

[30] X. Cheng, P. Tang, S. Su, R. Chen, Z. Wu, and B. Zhu, "Multi-party high-dimensional data publishing under differential privacy," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 8, pp. 1557–1571, 2020.

[31] R. Wang, B. Fung, Y. Zhu, and Q. Peng, "Differentially private data publishing for arbitrarily partitioned data," *Information Sciences*, vol. 553, no. 10, pp. 247–265, 2021.

[32] Z. Gu, G. Zhang, and C. Yang, "Multi-party high-dimensional related data publishing via probabilistic principal component analysis and differential privacy," in *Security and Privacy in New Computing Environments*, W. Shi, X. Chen, and K. K. R. Choo, Eds., , pp. 117–131, Springer International Publishing, 2022.

[33] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B*, vol. 61, no. 3, pp. 611–622, 1999.

[34] R. Cynthia and A. Dwork, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2013.

[35] S. Kotz, T. Kozubowski, and K. Podgorski, *The Laplace Distribution and Generalizations*, p. 01, Birkhäuser, Boston, MA, 2001.
[36] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," *Proc. EUROCRYPT'99, Czech Republic, May*, vol. 34, pp. 223–238, 1999.
[37] T. Li, J. Li, Z. Liu, P. Li, and C. Jia, "Differentially private naive bayes learning over multiple data sources," *Information Sciences*, vol. 444, pp. 89–104, 2018.
[38] Lib, "StatLib---Datasets Archive," Available at: http://lib.stat.cmu.edu/datasets/, September 8.
[39] D. Dua and C. Graff, *Uci Machine Learning Repository*, University of california, school of information and computer science, irvine, ca, 2019, http://archive.ics.uci.edu/ml.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 1321198, 17 pages
https://doi.org/10.1155/2022/1321198

Research Article

Privacy-Preserving Outsourced Logistic Regression on Encrypted Data from Homomorphic Encryption

Xiaopeng Yu,[1] **Wei Zhao**,[1] **Yunfan Huang**,[1] **Juan Ren**,[1] **and Dianhua Tang**[1,2]

[1]*Science and Technology on Communication Security Laboratory, Chengdu 610041, China*
[2]*School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China*

Correspondence should be addressed to Dianhua Tang; tangdianhua86@163.com

Received 23 March 2022; Revised 11 May 2022; Accepted 24 May 2022; Published 21 July 2022

Academic Editor: Debiao He

Logistic regression is a data statistical technique, which is used to predict the probability that an event occurs. For some scenarios where the storage capabilities and computing resources of the data owner are limited, the data owner wants to train the logistic regression model on the cloud service provider, while the high sensitivity of training data requires effective privacy protection methods that enable efficient model training without exposing information about the training data to untrusted cloud service providers. Recently, several works have used cryptographic techniques to implement privacy-preserving logistic regression in such application scenarios. However, on large-scale training datasets, the existing works still have the problems of long model training time and poor model performance. To solve these problems, based on the homomorphic encryption (HE), we propose an efficient privacy-preserving outsourced logistic regression (P^2OLR) on encrypted training data, which enables data owners to utilize the powerful storage and computing resources of cloud service providers for logistic regression analysis without exposing data privacy. Furthermore, the proposed scheme can pack multiple messages into one ciphertext and perform the same arithmetic evaluations on multiple plaintext slots by using the batching technique and single instruction multiple data (SIMD) mechanism in HE. On three public training datasets, the experimental results show that, compared with the existing schemes, the proposed scheme has better performance in terms of the encryption and decryption time of the data owner, the storage of encrypted training data, and the training time and accuracy of the model.

1. Introduction

Logistic regression (LR) [1] is a popular classification method, which has been used in numerous practical applications including cancer diagnosis [2], credit scoring [3], genome-wide association study [4], and more. LR can not only be applied to the problem of predicting the probability of occurrence of various events, but also is competitive with other classification algorithms in terms of prediction accuracy. In some practical application setting, the data owners have the limited computing and storage resources, and thus wants to outsource some of the heavy computation in logistic regression model training, the outsourced data analysis [5] has received considerable attention recently, which enables data owners to train a LR model using the powerful storage capacity and computing resources of cloud service providers [6].

However, the high sensitivity of training data requires to perform an effective privacy protection [7–10] that enable efficient and secure logistic regression analysis without leaking information about the training data to untrusted cloud service provider. Recently, to meet such application requirements, based on the cryptographic techniques like secure multiparty computation (MPC) [11] and homomorphic encryption (HE) [12], there have been several researches on the privacy-preserving logistic regression (PPLR) [13–22], which enables data owners to employ the service providers' powerful data storage and computing resources for logistic regression model training without exposing its own data privacy. Specifically, the data owner encrypts its training data, and sends encrypted training data to the service provider. The service provider can train a logistic regression model on encrypted training data, and

returns the encrypted training result to the data owner. The data owner can decrypt the encrypted training result to obtain final training result.

Unfortunately, on large-scale training dataset, the existing PPLR schemes [13–22] still have the bottlenecks of high model training time and low model precision. To solve these problems, based on the HE cryptographic technique [23] that has the property that the operation results on ciphertexts are consistent with those on plaintexts, we design an efficient privacy-preserving outsourced logistic regression (P^2OLR). The main contributions are as follows:

(1) Firstly, we propose a method for achieving P^2OLR on encrypted data from HE. To speed up the model training, the proposed P^2OLR scheme employs the batching technique to pack multiple elements into multiple plaintext slots, encrypts them into one ciphertext, and performs the same arithmetic operations to multiple plaintext slots in the SIMD mechanism.

(2) Secondly, we evaluate the proposed P^2OLR on three public datasets [18]. Under the same experimental environment, compared with the related P^2OLR [17, 18, 22], the model training time of the proposed P^2OLR is reduced by more than 71.7%, and the proposed P^2OLR has a better model performance.

The rest of this paper is arranged as follows. We present the related works in Section 2. We review the preliminaries related to our P^2OLR in Section 3. In Section 4, ourP^2OLR is described. The performance evaluation for our P^2OLR is presented in Section 5. The security analysis of our P^2OLR is shown in Section 6. Finally, we conclude in Section 7.

2. Related Works

There have been a lot of works on achieving PPLR using cryptographic techniques. In this paper, we mainly focus on the PPLR based on HE. To outsource the LR model training to a cloud service provider in a privacy-preserving manner, based on the HE scheme (FV) [24], Charlotte et al. [13] proposed an algorithm to train a LR model on an homomorphically encrypted dataset, which is implemented based on the FV-NFLlib library [25]. However, the accuracy of model is poor due to the use of a quadratic polynomial to approximate the sigmoid function. Furthermore, the training time grows linearly in the number of training samples. Using the HE scheme (FV) [24] and 1 bit gradient descent (GD) method, Chen et al. [14] presented a method to train LR over encrypted data, which is implemented through the SEAL library [26], and allows an arbitrary number of iterations by using bootstrapping [27] in FV, but bootstrapping introduces a significant decrease in performance. Focusing on the prediction process of LR, based on the HE scheme (BGV) [28], Li and Sun [15] proposed a secure protocol to solve the data leakage problem during the LR prediction process, and implement their scheme by the HElib library [29]. Based on the Chimera framework [30] that allows switching between HE schemes TFHE [31] and CKKS [23], Carpov et al. [16] proposed a solution to achieve semi-parallel LR on encrypted genomic data, which performs the bootstrapping [27] without re-encrypting the genomic data for an arbitrary number of iterations, and is implemented by using TFHE library [32] and HEAAN library [33].

Adapting the packing and parallelization techniques of approximate HE scheme (CKKS) [23], Kim et al. [17] proposed a PPLR, which is implemented through using the HEAAN library [33], and uses least squares approximation to improve the accuracy and efficiency of LR model training. However, as the number of iterations increases, the parameters of the CKKS scheme also need to become larger, which makes the training time increase dramatically. Kim et al. [18] applied the HE scheme (CKKS) [23] to achieve PPLR. Their scheme is implemented via using the HEAAN library [33]. Moreover, they devised an encoding method to decrease the storage of encrypted training data and adapted Nesterov's accelerated GD method to reduce the number of iterations as well as the computational cost. However, their scheme requires the assumption that both the number of training samples and features are power-of-two, which makes the scheme unsuitable for practical applications. To reduce the number of iterations, Cheon et al. [19] proposed an ensemble GD method based on the HE scheme (CKKS) [23], and applied it to the PPLR, in which they approximate the sigmoid function using a polynomial of 5-degree obtained by least squares approximation. Their scheme is implemented based on the HEAAN library [34]. To run a genome-wide association study on encrypted data, using the SIMD capabilities of HE scheme (CKKS) and Nesterov's accelerated GD, Bergamaschi et al. [20] introduced a method for homomorphic training of LR model, which is implemented based on the HElib library [29]. To protect the private information of both parties, based on the HE scheme (CKKS) [23] and gradient sharing technology, Wei et al. [21] proposed a protocol to train an LR model on vertically distributed data between two parties, which does not require trusted third-party nodes and is implemented by the HElib library [29]. Based on the HE scheme (CKKS) [23], Fan et al. [22] offered a PPLR algorithm, where they approximate the sigmoid function in LR by Taylor's theorem, and use row encoding to encrypt training samples, but as the number of samples increased, this will lead to longer model training time.

3. Preliminaries

3.1. System Model. As can be seen in Figure 1, the system model of the proposed P^2OLR considers two entities, namely a data owner (DO) and a service provider (SP). For readability, the definitions of the notations in this paper are shown in Table 1. DO: It has limited computational resources, and wants to use SP's data analysis service on encrypted data to train a LR model without revealing its own training data privacy. SP: It is a semi-trusted entity with powerful data storage and computing capabilities, and can provide data analysis and statistical services on encrypted data for DO. Specifically, DO chooses poly_modulus_degree N, coeff_modulus Q, and runs key_generation algorithm to

generate the secret_key *sk*, public_key *pk*, relinearization_key *rk*, galois_key *gk*. Next, DO encrypts the training data $\mathrm{D} \in \mathbb{R}^{m\times n}$ into ciphertexts D, encrypts the initial weight $\{w_0^{(0)}, w_1^{(0)}, \cdots, w_{n-1}^{(0)}\}$ into ciphertexts $W^{(0)}$, encrypts the learning rate α into one ciphertext $\alpha/\mathbf{m}$, and sends $N, Q, \Delta, pk, rk, gk, t, \mathrm{D}, W^{(0)}, \alpha/m$ to SP. SP performs the P^{2}OLR algorithm and returns the ciphertext result $W^{(t)}$ of the t -th iteration to DO. DO decrypts the ciphertext result $W^{(t)}$ to obtain final result $\{w_0^{(t)}, w_1^{(t)}, \cdots, w_{n-1}^{(t)}\}$.

3.2. Homomorphic Encryption. Homomorphic encryption (HE) is a cryptographic technique, which allows operations on ciphertexts without decryption, and guarantees that the computation results on ciphertexts are consistent with the computation results on plaintexts. We adopt the HE scheme (CKKS) [23] based on the Ring Learning with Errors (RLWE) problem, which can encrypt multiple elements in one ciphertext and supports the single instruction multiple data (SIMD) operations. Suppose $\Phi_M(X) = X^N + 1$ denotes the M -th cyclotomic polynomial, where N is power of 2. $\mathscr{R} = \mathbb{Z}[X]/(X^N + 1)$ denotes the cyclotomic ring of polynomials. $\mathscr{R}_q = \mathscr{R}/q\mathscr{R} = \mathbb{Z}_q[X]/(X^N + 1)$ denotes the residue ring of $\mathscr{R}$ modulo q. $\mathbb{H}$ denotes a subring of complex vector $\mathbb{C}^N$ that is isomorphic to $\mathbb{C}^{N/2}$. σ: $\mathscr{R} \longrightarrow \sigma(\mathscr{R}) \subseteq \mathbb{H}$ denotes a canonical embedding that transforms a plaintext polynomial $\mathscr{R}$ into a complex vector $\mathbb{H}$. π: $\mathbb{H} \longrightarrow \mathbb{C}^{N/2}$ denotes a natural projection that transforms a complex vector $\mathbb{C}^N$ to $\mathbb{C}^{N/2}$. HE scheme (CKKS) [23] supports the operations as follows, which can be found in the Appendix. For ease of description, we define the Algorithms 1–9.

3.3. Sigmoid Approximation. Since the existing HE scheme can only effectively support polynomial arithmetic computations, the computation of sigmoid function $\sigma(x) = 1/(1 + e^{-x})$ using HE is a barrier to the realization of P^{2}OLR. To find a approximate polynomial of $\sigma(x)$, adapting the least squares method, we consider the 7° polynomial $g(x) = a_0 + a_1x + a_3x^3 + a_5x^5 + a_7x^7$ over the domain $[-8, 8]$, where $a_0 = 1/2$, $a_1 = 1.73496/8$, $a_3 = 4.19407/8^3$, $a_5 = 5.43402/8^5$, $a_7 = 2.50739/8^7$. $\sigma(x)$ and $g(x)$ can be seen in Figure 2, the maximum errors between $\sigma(x)$ and $g(x)$ are about 0.032. $g(x)$ over encrypted data $\mathbf{x}$ from HE can be achieved by the Algorithm 10.

3.4. Logistic Regression. Logistic regression (LR) is a statistical analysis method for predicting the probability of an event. We consider the case where the predicted value is a binary dependent variable. Assuming that a dataset consists of m samples of the form $\{y_i, \mathbf{x}_i\}$ with $y_i \in \{0, 1\}$ and $\mathbf{x}_i = \{x_{i,1}, x_{i,2}, \cdots, x_{i,N/2-1}\} \in \mathbb{R}^{n-1}$, the goal of LR is to find the optimal parameters $\mathbf{w} = \{w_0, w_1, \cdots, w_{n-1}\}$ that minimizes the negative log-likelihood function (loss function) $J(w) = 1/m \cdot \sum_{i=0}^{m-1} (y_i \cdot \log(\sigma(d_i \cdot w^T)) + (1 - y_i) \cdot (1 - \log(\sigma(d_i \cdot w^T))))$, where $\mathbf{d}_i = \{1, \mathbf{x}_i\}$. A common method for minimizing loss function $J(\mathbf{w})$ is a gradient descent (GD) algorithm, which finds the local extremum of a loss function by following the direction of the gradient. The gradient of $J(\mathbf{w})$ with respect to $\mathbf{w}$ is calculated by $\nabla J(w) = 1/m \cdot \sum_{i=0}^{m-1} ((\sigma(-d_i \cdot w^T) - y_i) \cdot d_i)$. Let $\mathbf{w}^k$ be the regression parameters and α^k is a learning rate in the k-th iteration of the GD algorithm, the GD algorithm can update $\mathbf{w}^{k+1}$ by $\mathbf{w}^{k+1} \leftarrow \mathbf{w}^k - \alpha^k/m \cdot \sum_{i=0}^{m-1} ((\sigma(\mathbf{d}_i \cdot \mathbf{w}^{\mathbf{T}}) - y_i) \cdot \mathbf{d}_i)$.

4. Privacy-Preserving Outsourced Logistic Regression

Based on the HE scheme, we propose a P^{2}OLR, where we employ the batching method to pack multiple elements into multiple plaintext slots, and encrypt them into one ciphertext, and then perform the same arithmetic evaluations to multiple plaintext slots through the SIMD mechanism. To reduce the parameters of HE scheme (CKKS) as well as improve the performance of P^{2}OLR, the proposed P^{2}OLR allows the interaction between DO and SP during iterative training. Specifically, SP returns the ciphertext training result to DO after certain number of iterations t. DO decrypts the ciphertext training result, and determines whether the performance of the model has met the requirements. If so, stops training. Otherwise, sends encrypted weights to SP to continue training. Let

$$D = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{m-1} \\ x_{0,1} \\ x_{1,1} \\ \vdots \\ x_{m-1,1} \\ x_{0,2} \\ x_{1,2} \\ \vdots \\ x_{m-1,2} \\ \cdots \\ \cdots \\ \ddots \\ \cdots \\ x_{0,n-1} \\ x_{1,n-1} \\ \vdots \\ x_{m-1,n-1} \end{bmatrix}. \tag{1}$$

denote the training data sets held by DO, where D consists of m samples of the form $\{y_i, x_{i,1}, x_{i,2}, \cdots, x_{i,N/2-1}\}$ with $y_i \in \{0, 1\}$ and $\{x_{i,1}, x_{i,2}, \cdots, x_{i,N/2-1}\} \in \mathbb{R}^{n-1}$. The first column of D denotes the label, other columns D denote the features. Since DO has limited computational resources, DO

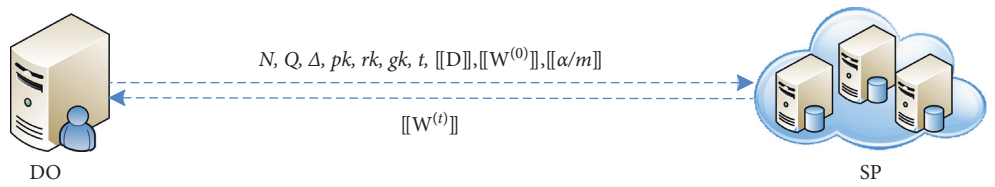


Figure 1: System model.

Table 1: The definitions of the notations.

Notations	Definitions
$\mathbf{x}$	A message vector $[x_0, x_1, \cdots, x_{N/2-1}]$
$\langle \mathbf{x} \rangle$	The plaintext of message vector $\mathbf{x}$
$\mathbf{x}$	The ciphertext of message vector $\mathbf{x}$
X	A list $\{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{n-1}\}$
X_i	The i ciphertext of ciphertext list X_i
$\mathbf{x} * \mathbf{y}$	The multiplication of x and y, namely $[x_0 \cdot y_0, x_1 \cdot y_1, \cdots, x_{N/2-1} \cdot y_{N/2-1}]$
$\mathbf{x} \cdot \mathbf{y}$	The product of x and y, namely $[x_0 \cdot y_0, x_1 \cdot y_1, \cdots, x_{N/2-1} \cdot y_{N/2-1}]$
$\mathbf{x} + \mathbf{y}$	The addition of x and y, namely $[x_0 + y_0, x_1 + y_1, \cdots, x_{N/2-1} + y_{N/2-1}]$
$\mathbf{x} - \mathbf{y}$	The subtraction of x and y, namely $[x_0 + y_0, x_1 + y_1, \cdots, x_{N/2-1} + y_{N/2-1}]$

Input: x
Output: x
(1) encode_double (x, Δ, x)
(2) encrypt ($\langle x \rangle$, x)
(3) **return**: x

Algorithm 1: $x = \mathrm{Enc}(x)$.

Input: X
Output: $\{x_0, x_1, \cdots, x_{n-1}\}$
(1) **for** ($i = 0$ to $n - 1$) do
(2) decrypt (X_i, $\langle x_i \rangle$)
(3) decode_double ($\langle x_i \rangle$, x_i)
(4) $x_i = \mathbf{x_i}$.get(0)
(5) **end for**
(6) **return**: $\{x_0, x_1, \ldots, x_{n-1}\}$

Algorithm 2: $\{x_0, x_1, \cdots, x_{n-1}\} = \mathrm{Dec}(X)$.

Input: x, y
Output: $\mathbf{x} * \mathbf{y}$
(1) mod_switch_to_inplace (y, x.parms_id())
(2) multiply (x, y, $x * y$)
(3) relinearize_inplace ($x * y$, rk)
(4) rescale_to_next_inplace ($x * y$)
(5) $x * y$.set_scale (Δ)
(6) **return**: $x * y$

Algorithm 3: $x * y = \mathrm{Mul}(x, y)$.

Input: x, y
Output: $x * y$
(1) encode_double (y, Δ, $\langle y \rangle$)
(2) mod_switch_to_inplace ($\langle y \rangle$, x.parms_id())
(3) multiply_plain (x, $\langle y \rangle$, $x * y$)
(4) rescale_to_next_inplace ($x * y$)
(5) $x * y$.set_scale (Δ)
(6) **return**: $x * y$

Algorithm 4: $x * y = \mathrm{Mul\_Plain}(x, y)$.

Input: x, y
Output: $x + y$
(1) mod_switch_to_inplace (y, x.parms_id())
(2) add (x, y, $x + y$)
(3) **return**: $x + y$

Algorithm 5: $x + y = \mathrm{Add}(x, y)$.

Input: x, y
Output: $x + y$
(1) encode_double (y, Δ, $\langle y \rangle$)
(2) mod_switch_to_inplace ($\langle y \rangle$, x.parms_id())
(3) add_plain (x, $\langle y \rangle$, $x + y$)
(4) **return**: $x + y$

Algorithm 6: $x + y = \mathrm{Add\_Plain}(x, y)$.

Input: x, y
Output: $x - y$
(1) mod_switch_to_inplace (y, x.parms_id())
(2) sub (x, y, $x - y$)
(3) **return**: $x - y$

Algorithm 7: $x - y = \mathrm{Sub}(x, y)$.

Input: x, y
Output: x
(1) mod_switch_to_inplace (x, y.parms_id())
(2) add _inplace (x, y)
(3) **return**: x

Algorithm 8: $x = \mathrm{Add\_Inplace}(x, y)$.

Input: $x = [x_0, x_1, \cdots, x_{N/2-1}]$
Output: $y = [\sum_{i=0}^{N/2-1} x_i, \sum_{i=0}^{N/2-1} x_i, \cdots, \sum_{i=0}^{N/2-1} x_i]$
(1) $y = x$
(2) **for** ($k = N/2$; $k \geq 1$; $k = k/2$) **do**
(3) rotate_vector (y, k, gk, z)
(4) add_inplace (y, z)
(5) **end for**
(6) **return**: y

Algorithm 9: $y = \mathrm{Rotate\_Sum}(x)$.

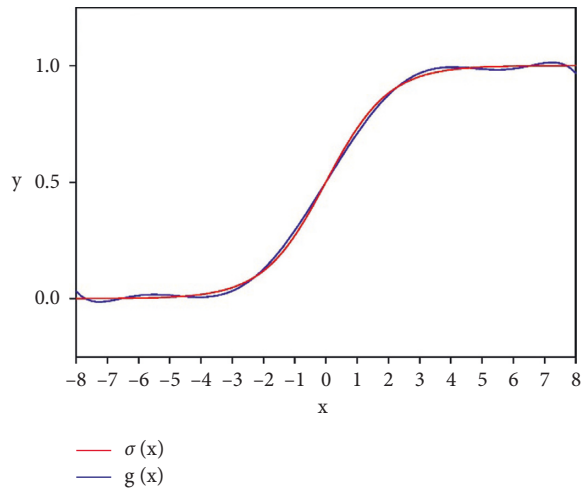


Figure 2: Sigmoid approximation.

wants to outsource to SP to train a LR model without disclosing its own training data privacy. The specific description of the proposed P^{2}OLR is as follows.

(1) DO generates $\{sk, pk, rk, gk\}$, computes $l = 2m/N$, calls the Algorithm 1 to encrypt the training data D into $l \times n$ ciphertexts

Input: x
Output: $g(x)$
(1) $x * x = \mathrm{Mul}(x, x)$
(2) $x * x * x * x * x = \mathrm{Mul}(x * x, x * x)$
(3) $x * x * x * x * x * x * x * x = \mathrm{Mul}(x * x * x * x * x, x * x)$
(4) $a_7 * x = \mathrm{Mul\_Plain}(x, a_7)$
(5) $a_7 * x * x * x * x * x * x * x * x * x = \mathrm{Mul}(x * x * x * x * x * x * x * x, a_7 * x)$
(6) $a_5 * x = \mathrm{Mul\_Plain}(x, a_5)$
(7) $a_5 * x * x * x * x * x * x * x = \mathrm{Mul}(x * x * x * x * x, a_5 * x)$
(8) $a_3 * x = \mathrm{Mul\_Plain}(x, a_3)$
(9) $a_3 * x * x * x * x = \mathrm{Mul}(x * x, a_3 * x)$
(10) $a_1 * x = \mathrm{Mul\_Plain}(x, a_1)$
(11) $a_0 + a_1 * x = \mathrm{Add\_Plain}(a_1 * x, a_0)$
(12) $a_0 + a_1 * x - a_3 * x * x * x * x = \mathrm{Sub}(a_3 * x * x * x * x, a_0 + a_1 * x)$
(13) $a_0 + a_1 * x - a_3 * x * x * x * x * x + a_5 * x * x * x * x * x * x * x = \mathrm{Add}(a_5 * x * x * x * x * x * x * x, a_0 + a_1 * x - a_3 * x * x * x * x)$
(14) $a_0 + a_1 * x - a_3 * x * x * x * x * x + a_5 * x * x * x * x * x * x * x - a_7 * x * x * x * x * x * x * x * x * x * x = \mathrm{Sub}(a_7 * x * x * x * x * x * x * x * x * x * x * x, a_0 + a_1 * x - a_3 * x * x * x * x * x + a_5 * x * x * x * x * x * x * x)$
(15) **return**: $g(x) = a_0 + a_1 * x - a_3 * x * x * x * x * x + a_5 * x * x * x * x * x * x * x * x - a_7 * x * x * x * x * x * x * x * x * x * x * x$.

Algorithm 10: $g(x) = \mathrm{Sigmoid\_Approximation}(x)$.

$$
\begin{aligned}
[\![\mathbf{y}_0^T]\!] &= \mathrm{Enc}\left(\left[y_0, y_1, \cdots, y_{N/2-1}\right]\right),\\
[\![\mathbf{y}_1^T]\!] &= \mathrm{Enc}\left[\left(y_{N/2}, y_{N/2+1}, \cdots, y_{N-1}\right)\right],\\
[\![\mathbf{y}_{l-1}^T]\!] &= \mathrm{Enc}\left(\left[y_{m-(l-1)\cdot N/2}, y_{m-(l-1)\cdot N/2+1}, \cdots, y_{m-1}\right]\right),\\
[\![\mathbf{x}_{0,1}^T]\!] &= \mathrm{Enc}\left(\left[x_{0,1}, x_{1,1}, \cdots, y_{N/2-1,1}\right]\right),\\
[\![\mathbf{x}_{0,n-1}^T]\!] &= \mathrm{Enc}\left(\left[x_{0,n-1}, x_{1,n-1}, \cdots, y_{N/2-1,n-1}\right]\right),\\
[\![\mathbf{x}_{0,2}^T]\!] &= \mathrm{Enc}\left(\left[x_{0,2}, x_{1,2}, \cdots, y_{N/2-1,2}\right]\right),\\
[\![\mathbf{x}_{1,1}^T]\!] &= \mathrm{Enc}\left(\left[x_{N/2,1}, x_{N/2+1,1}, \cdots, y_{N-1,1}\right]\right),\\
[\![\mathbf{x}_{1,2}^T]\!] &= \mathrm{Enc}\left(\left[x_{N/2,2}, x_{N/2+1,2}, \cdots, y_{N-1,2}\right]\right),\\
[\![\mathbf{x}_{1,n-1}^T]\!] &= \mathrm{Enc}\left(\left[x_{N/2,n-1}, x_{N/2+1+1,n-1}, \cdots, y_{N-1,n-1}\right]\right),\\
[\![\mathbf{x}_{l-1,1}^T]\!] &= \mathrm{Enc}\left(\left[x_{m-(l-1)\cdot N/2,1}, x_{m-(l-1)\cdot N/2+1,1}, \cdots, y_{m-1,1}\right]\right),\\
[\![\mathbf{x}_{l-1,2}^T]\!] &= \mathrm{Enc}\left(\left[x_{m-(l-1)\cdot N/2,2}, x_{m-(l-1)\cdot N/2+1,2}, \cdots, y_{m-1,2}\right]\right)\\
[\![\mathbf{x}_{l-1,n-1}^T]\!] &= \mathrm{Enc}\left(\left[x_{m-(l-1)\cdot N/2,n-1}, x_{m-(l-1)\cdot N/2+1,n-1}, \cdots, y_{m-1,n-1}\right]\right).
\end{aligned}
\tag{2}
$$

calls the Algorithm 1 to encrypt the initial weight $\left\{w_0^{(0)}, w_1^{(0)}, \cdots, w_{n-1}^{(0)}\right\}$ into n ciphertexts

$$
\begin{aligned}
[\![\mathbf{w}_0^{(0)}]\!] &= \mathrm{Enc}\left(\left[\underbrace{w_0^{(0)}, w_0^{(0)}, \cdots, w_0^{(0)}}_{N/2}\right]\right),\\
[\![\mathbf{w}_1^{(0)}]\!] &= \mathrm{Enc}\left(\left[\underbrace{w_1^{(0)}, w_1^{(0)}, \cdots, w_1^{(0)}}_{N/2}\right]\right),\\
[\![\mathbf{w}_{N-1}^{(0)}]\!] &= \mathrm{Enc}\left(\left[\underbrace{w_1^{(0)}, w_1^{(0)}, \cdots, w_1^{(0)}}_{N/2}\right]\right),
\end{aligned}
\tag{3}
$$

calls the Algorithm 1 to encrypt the learning rate α into one ciphertext

$$
[\![\frac{\boldsymbol{\alpha}}{\mathbf{m}}]\!] = \mathrm{Enc}\left(\left[\underbrace{\alpha/m, 0, 0, \cdots, 0}_{N/2}\right]\right). \tag{4}
$$

and sends $\mathbf{y}_0^T$, $\mathbf{y}_1^T$, $\cdots$, $\mathbf{y}_{l-1}^T$, $\mathbf{x}_{0,1}^T$, $\mathbf{x}_{0,2}^T$, $\cdots$, $\mathbf{x}_{0,n-1}^T$, $\mathbf{x}_{1,1}^T$, $\mathbf{x}_{1,2}^T$, $\cdots$, $\mathbf{x}_{1,n-1}^T$, $\mathbf{x}_{l-1,1}^T$, $\mathbf{x}_{l-1,2}^T$, $\cdots$, $\mathbf{x}_{l-1,n-1}^T$, $w_0^{(0)}$, $\mathbf{w}_1^{(0)}$, $\cdots$, $\mathbf{w}_{n-1}^{(0)}$, $\alpha/\mathbf{m}$, N, Q, sk, pk, rk, gk, t to SP.

(2) SP computes ciphertexts

$$
\begin{aligned}
[\![\mathbf{x}_{i,0}^T]\!] &= Enc\left(\left[\underbrace{1, 1, \cdots, 1}_{N/2}\right]\right),\ (i = 1, 2, \cdots, l-2),\\
[\![\mathbf{x}_{l-1,0}^T]\!] &= \mathrm{Enc}\left(\left[\underbrace{1, 1, \cdots, 1}_{m-(l-1)\cdot N/2}\right]\right),
\end{aligned}
\tag{5}
$$

and sets the lists

$$\begin{aligned}
\mathrm{Y} &= \left\{[\![\mathbf{y}_0^T]\!], [\![\mathbf{y}_1^T]\!], \cdots, [\![\mathbf{y}_{l-1}^T]\!]\right\},\\
[\![\mathrm{X}_0]\!] &= \left\{[\![\mathbf{x}_{0,0}^T]\!], [\![\mathbf{x}_{0,1}^T]\!], [\![\mathbf{x}_{0,2}^T]\!], \cdots, [\![\mathbf{x}_{0,n-1}^T]\!]\right\},\\
[\![\mathrm{X}_1]\!] &= \left\{[\![\mathbf{x}_{1,0}^T]\!], [\![\mathbf{x}_{1,1}^T]\!], [\![\mathbf{x}_{1,2}^T]\!], \cdots, [\![\mathbf{x}_{1,n-1}^T]\!]\right\},\\
[\![\mathrm{X}_{l-1}]\!] &= \left\{[\![\mathbf{x}_{l-1,0}^T]\!], [\![\mathbf{x}_{l-1,1}^T]\!], [\![\mathbf{x}_{l-1,2}^T]\!], \cdots, [\![\mathbf{x}_{l-1,n-1}^T]\!]\right\},\\
[\![\mathrm{W}^{(0)}]\!] &= \left\{[\![\mathbf{w}_0^{(0)}]\!], [\![\mathbf{w}_1^{(0)}]\!], \cdots, [\![\mathbf{w}_{n-1}^{(0)}]\!]\right\}.
\end{aligned} \tag{6}$$

Next, SP calls the Algorithm 11, and returns the ciphertext result $W^{(t)}$ to DO.

(3) DO calls the Algorithm 2 to decrypt the ciphertext result $W^{(t)}$ into the result $\{w_0^{(t)}, w_1^{(t)}, \cdots, w_{n-1}^{(t)}\} = \mathrm{Dec}(\mathrm{W}^{(t)})$. Next, DO judges whether $\{w_0^{(t)}, w_1^{(t)}, \cdots, w_{n-1}^{(t)}\}$ has met the requirements. If so, terminates the training. Otherwise, DO calls the Algorithm 1 to encrypt $\{w_0^{(t)}, w_1^{(t)}, \cdots, w_{n-1}^{(t)}\}$ into n ciphertexts

$$\begin{aligned}
[\![\mathbf{w}_0^{(0)}]\!] &= \mathrm{Enc}\left(\left[\underbrace{w_0^{(t)}, w_0^{(t)}, \cdots, w_0^{(t)}}_{N/2}\right]\right),\\
[\![\mathbf{w}_1^{(0)}]\!] &= \mathrm{Enc}\left(\left[\underbrace{w_1^{(t)}, w_1^{(t)}, \cdots, w_1^{(t)}}_{N/2}\right]\right),\\
[\![\mathbf{w}_{n-1}^{(0)}]\!] &= \mathrm{Enc}\left(\left[\underbrace{w_{n-1}^{(t)}, w_{n-1}^{(t)}, \cdots, w_{n-1}^{(t)}}_{N/2}\right]\right).
\end{aligned} \tag{7}$$

and sends $w_0^{(0)}$, $\mathbf{w}_1^{(0)}$, $\cdots$, $\mathbf{w}_{n-1}^{(0)}$ to SP to continue training.

5. Performance Evaluation

We implement all experiments on a 32-core Intel Xeon CPU with 256 GB RAM. We compare the performance of the proposed P^{2}OLR with the related P^{2}OLR [17, 18, 22]. We employ 5-fold cross-validation method to obtain the validity of the experimental results. For [17, 18], the implementations are publicly available at [35, 36], respectively, which use the HEAAN library [33] to provide HE cryptographic operations. For [22] and the proposed P^{2}OLR, we employ the Microsoft SEAL library [26] for the HE cryptographic operations. For all experiments, we set the learning rate $\alpha = 0.01$, random initial weight vector $\{w_0^{(0)}, w_1^{(0)}, \cdots, w_{n-1}^{(0)}\}$ maximum number of iterations $\lambda = 20$, and scaling factor $\Delta = 2^{40}$. To guarantee $\kappa = 128$ bit security, the scheme [17] takes the polynomial-modulus-degree $N = 2^{17}$, coefficient-modulus Q around 2204 to 2406 bits; the scheme [18] sets the $N = 2^{16}$, $Q = 1176$ bits; the scheme [22] chooses the $N = 2^{15}$, $Q = 320$ bits; For the proposed P^{2}OLR, we select $N = 2^{15}$, $Q = 512$ bits. Using the three datasets [18]: D_1—Umaru Impact Study, D_2—Myocardial Infarction Study from Edinburgh, D_3—Nhanes III, we compare the proposed P^{2}OLR with the related P^{2}OLR [17, 18, 22] in terms of the encryption time (E. time) and decryption time (D. time) of DO, storage of encrypted training data, and training time (T. time), accuracy, precision, recall, F1-score and AUC of model. All comparison results are shown as an average of 10 experiments. The performance comparisons of the proposed P^{2}OLR and the related P^{2}OLR [17, 18, 22] are shown in Table 2.

From Table 2, we can see that, compared with the related P^{2}OLR [17, 18, 22], the proposed P^{2}OLR has a better performance. Specifically, as shown in Figure 3, under the training dataset D_1, the encryption time of DO in the proposed P^{2}OLR is 2.01 s, which is reduced by nearly 71.4%, 7.8%, and 93.3% respectively compared with the encryption time of DO in [17, 18, 22]; under the training dataset D_2, the encryption time of DO in the proposed P^{2}OLR is 2.16 s, which is reduced by nearly 73.6%, 2.3%, and 96.8% respectively compared with the encryption time of DO in [17, 18, 22]; under the training dataset D_3, the encryption time of DO in the proposed P^{2}OLR is 3.49 s, which is reduced by nearly 75.9%, 81.6%, and 75.0% respectively compared with the encryption time of DO in [17, 18, 22].

As can be seen in Figure 4, under the training dataset D_1, the decryption time of DO in the proposed P^{2}OLR is 0.23 s, which is reduced by almost 95.3% and 41.0% respectively in comparison to the decryption time of DO in [17, 18]; under the training dataset D_2, the decryption time of DO in the proposed P^{2}OLR is 0.26 s, which is reduced by almost 95.0% and 36.6% respectively in comparison to the decryption time of DO in [17, 18]; under the training dataset D_3, the decryption time of DO in the proposed P^{2}OLR is 0.45 s, which is reduced by almost 96.1% and 6.1% respectively in comparison to the decryption time of DO in [17, 18]. The decryption time of DO in [22] is smaller in comparison to that of the proposed P^{2}OLR.

As described in Figure 5, under the training dataset D_1, the storage of encrypted training data in the proposed P^{2}OLR is 72.00 MB, compared with the storage of encrypted training data in [17, 22], which is reduced by nearly 88.9% and 95.0%; under the training dataset D_2, the storage of encrypted training data in the proposed P^{2}OLR is 80.00 MB, compared with the storage of encrypted training data in [17, 22], which is reduced by nearly 89.0% and 97.4%; under the training dataset D_3, the storage of encrypted training data in the proposed P^{2}OLR is 128.00 MB, compared with the storage of encrypted training data in [17, 18, 22], which is reduced by nearly 89.4%, 13.0% and 99.7% respectively. Although the storage of encrypted training data for dataset D_1 and D_2 in [18] is smaller than that of the proposed P^{2}OLR, as the number of samples m and features n increases, for dataset D_3, the storage of encrypted training data in the proposed P^{2}OLR is smaller than that of [22].

As displayed in Figure 6, under the training dataset D_1, the training time of model in the proposed P^{2}OLR is 2.64 min, which is reduced by almost 96.6%, 73.8%, and 90.1% respectively than the training time of model in [17, 18, 22]; under the training dataset D_2, the training time of model in the proposed P^{2}OLR is 2.91 min, which is reduced by almost 96.5%, 71.7%, and 95.0% respectively than the training time of model in [17, 18, 22]; under the training dataset D_3, the training time of model in the proposed P^{2}OLR is 4.21 min, which is reduced by almost 96.5%, 79.8%, and 99.4% respectively than the training time of model in [17, 18, 22].

Input: Y, X_0, X_1, $\cdots$, X_{l-1}, $w^{(0)}$, $\alpha/\mathbf{m}$, N, Q,., sk, pk, rk, gk, t
Output: $W^{(t)}$

(1) **for** ($k = 0$ to $t-1$) **do**
(2) **for** ($i = 0$ to $l-1$) **do**
(3) **for** ($j = 0$ to $n-1$) **do**
(4) $C_{ij} = \mathrm{Mul}(W^{(k)}{}_j, X_{ij})$
(5) **end for**
(6) $O_i = 0$
(7) **for** ($j = 0$ to $n-1$) **do**
(8) $O_i = \mathrm{Add\_Inplace}(O_i, C_{ij})$
(9) **end for**
(10) $G_i = \mathrm{Sigmoid\_Approximation}(O_i)$
(11) $G\prime_i = \mathrm{Sub}(G_i, Y_i)$
(12) **for** ($j = 0$ to $n-1$) **do**
(13) $C'_{ij} = \mathrm{Mul}(G\prime_i, X_{ij})$
(14) **end for**
(15) **end for**
(16) **for** ($j = 0$ to $n-1$) **do**
(17) $Z_i = 0$
(18) **for** ($i = 0$ to $l-1$) **do**
(19) $Z_j = \mathrm{Add\_Inplace}(Z_j, C'_{ij})$
(20) **end for**
(21) $Z\prime_j = \mathrm{Rotate\_Sum}(Z_j)$
(22) $Z\prime_j = \mathrm{Mul}(Z\prime_j, \alpha/m)$
(23) $W^{(k+1)}{}_j = \mathrm{Sub}(W^{(k)}{}_j, Z\prime\prime_j)$
(24) $W^{'(k+1)}{}_j = \mathrm{Mul\_Plain}(W^{(k+1)}{}_j, 1)$
(25) $W''^{(k+1)}{}_j = \mathrm{Rotate\_Sum}(W'^{(k+1)}{}_j)$
(26) **end for**
(27) **end for**
(28) **return**: $W^{(t)}$

Algorithm 11: P^{2}OLR.

Table 2: Performance comparisons.

Dataset	m	n	λ	Scheme	E. time (s)	D. time (s)	Storage (MB)	T. time (min)	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	AUC
D_1	575	8	20	[17]	7.04	4.93	648.56	77.35	74.8	92.3	71.4	80.5	0.68
				[18]	2.18	0.39	36.75	10.09	74.4	90.9	71.4	80.0	0.65
				[22]	30.28	0.06	1438.75	26.67	74.4	90.9	71.4	80.0	0.66
				P^{2}OLR	2.01	0.23	72.00	2.64	80.6	95.6	77.4	85.5	0.73
D_2	1253	9	20	[17]	8.17	5.19	726.86	83.57	81.6	89.7	82.4	85.9	0.82
				[18]	2.21	0.41	36.75	10.28	83.0	90.4	83.5	86.8	0.86
				[22]	68.35	0.06	3133.75	57.35	82.7	90.4	82.9	86.5	0.86
				P^{2}OLR	2.16	0.26	80.00	2.91	90.6	95.1	90.6	92.8	0.88
D_3	15649	15	20	[17]	14.48	11.65	1203.00	121.95	79.1	50.0	61.2	55.0	0.83
				[18]	13.96	0.48	147.00	20.86	79.2	50.2	61.3	55.2	0.71
				[22]	823.56	0.06	39123.75	718.25	77.9	52.4	62.2	56.9	0.71
				P^{2}OLR	3.49	0.45	128.00	4.21	83.7	60.3	64.2	62.2	0.85

As illustrated in Figure 7, under the training dataset D_1, the average accuracy of model in the proposed P^{2}OLR is 80.6%, which has nearly 5.8%, 6.2%, and 6.2% improvement respectively compared with the average accuracy of model in [17, 18, 22]; under the training dataset D_2, the average accuracy of model in the proposed P^{2}OLR is 90.6%, which has nearly 9.0%, 7.6%, and 7.9% improvement respectively compared with the average accuracy of model in [17, 18, 22]; under the training dataset D_3, the average accuracy of model in the proposed P^{2}OLR is 83.7%, which has nearly 4.6%, 4.5%, and 5.8% improvement respectively compared with the average accuracy of model in [17, 18, 22].

As illustrated in Figure 8, under the training dataset D_1, the average precision of model in the proposed P^{2}OLR is 95.6%, which has nearly 3.3%, 4.7%, and 4.7% improvement respectively compared with the average precision of model in [17, 18, 22]; under the training dataset D_2, the average precision of model in the proposed P^{2}OLR is 95.1%, which has nearly 5.4%, 4.7%, and 4.7% improvement respectively compared with the average precision of model in [17, 18, 22];

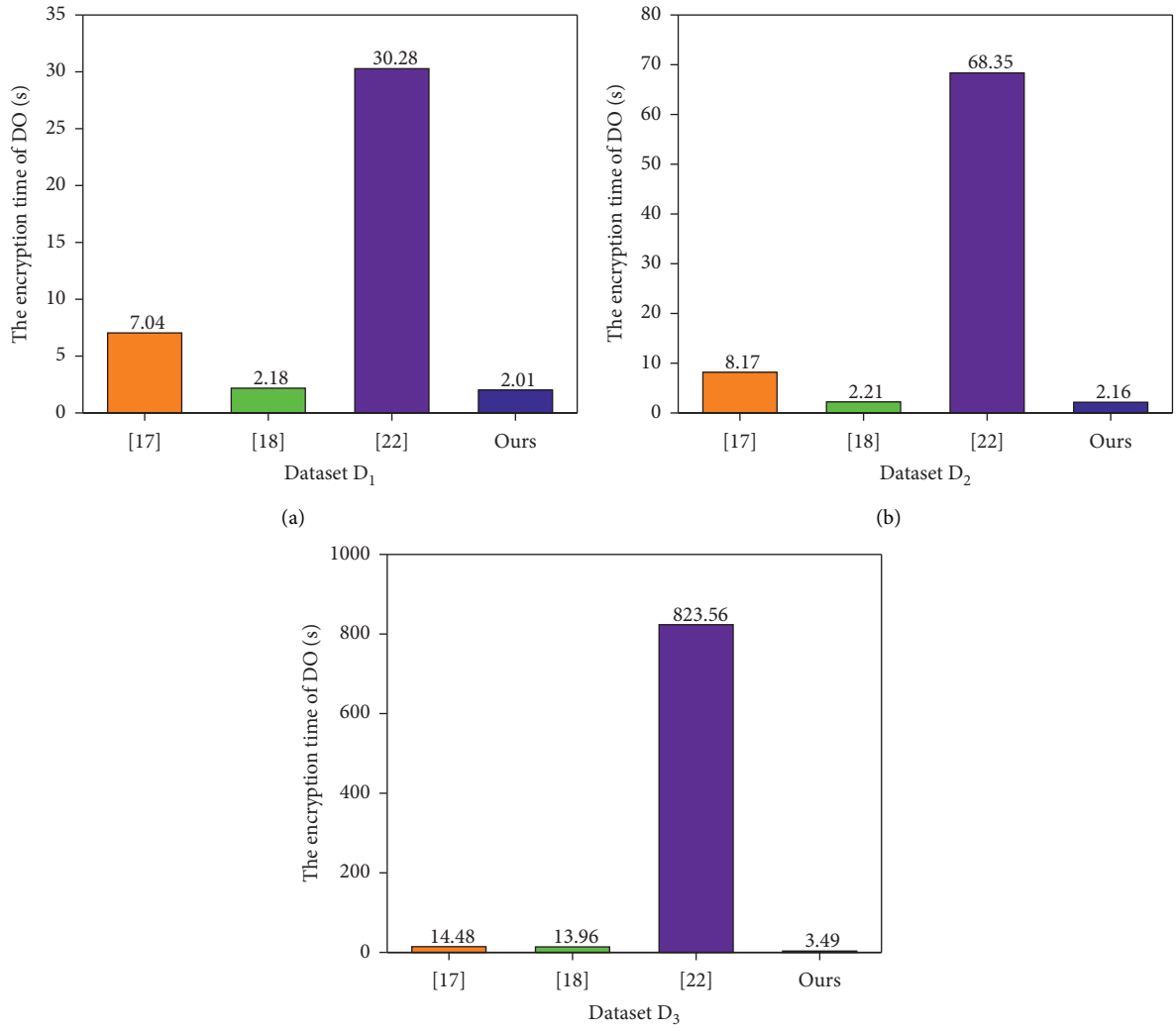


Figure 3: The encryption time of DO.

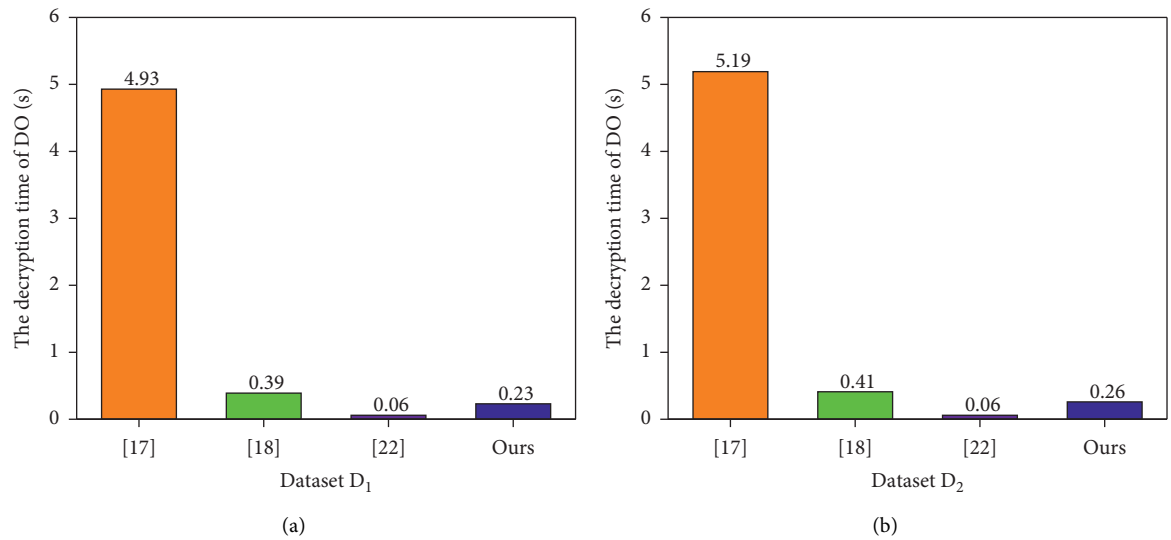
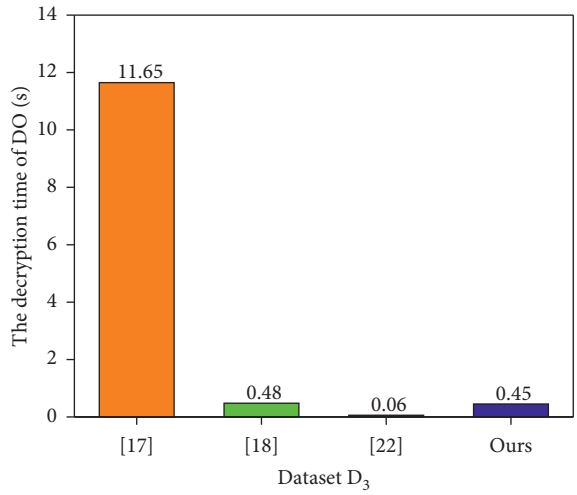


Figure 4: Continued.



(c)

Figure 4: The decryption time of DO.

under the training dataset D_3, the average precision of model in the proposed P^{2}OLR is 60.3%, which has nearly 10.3%, 10.1%, and 7.9% improvement respectively compared with the average precision of model in [17, 18, 22].

As illustrated in Figure 9, under the training dataset D_1, the average recall of model in the proposed P^{2}OLR is 77.4%, which has nearly 6.0%, 6.0%, and 6.0% improvement respectively compared with the average recall of model in [17, 18, 22]; under the training dataset D_2, the average recall of model in the proposed P^{2}OLR is 90.6%, which has nearly 8.2%, 7.1%, and 7.7% improvement respectively compared with the average recall of model in [17, 18, 22]; under the training dataset D_3, the average recall of model in the proposed P^{2}OLR is 64.2%, which has nearly 3.0%, 2.9%, and 2.0% improvement respectively compared with the average recall of model in [17, 18, 22].

As illustrated in Figure 10, under the training dataset D_1, the average F1-score of model in the proposed P^{2}OLR is 85.5%, which has nearly 5.0%, 5.5%, and 5.5% improvement respectively compared with the average F1-score of model in [17, 18, 22]; under the training dataset D_2, the average F1-score of model in the proposed P^{2}OLR is 92.8%, which has nearly 6.9%, 4.0%, and 4.3% improvement respectively compared with the average F1-score of model in [17, 18, 22]; under the training dataset D_3, the average F1-score of model in the proposed P^{2}OLR is 62.2%, which has nearly 7.2%, 7.0%, and 5.3% improvement respectively compared with the average F1-score of model in [17, 18, 22].

As demonstrated in Figure 11, under the training dataset D_1, the AUC of model in the proposed P^{2}OLR is 0.73, compared with the AUC of model in [17, 18, 22], which has nearly 0.05, 0.08, and 0.07 improvement respectively; under the training dataset D_2, the AUC of model in the proposed P^{2}OLR is 0.88, compared with the AUC of model in [17, 18, 22], which has nearly 0.06, 0.02, and 0.02 improvement respectively; under the training dataset D_3, the AUC of model in the proposed P^{2}OLR is 0.85, compared with the AUC of model in [17, 18, 22], which has nearly 0.02, 0.14, and 0.14 improvement respectively.

6. Security Analysis

In a semi-honest adversary model, we assume that DO and SP hold the public key pk, relinearization key rk, galois key gk, and only DO holds the secret key sk. For our P^{2}OLR that evaluates deterministic function f, following the simulation-based paradigm [37], we consider the security model for security analysis, namely, DO encrypts its private data $\mathbf{x}$ and sends $\mathbf{x}$ to SP. SP performs the homomorphic operations on $\mathbf{x}$ to obtain $\mathbf{y}$, homomorphically evaluates $f(\mathbf{x})$ on $\mathbf{x}$ to obtain $f(\mathbf{x})$, and sends $f(\mathbf{x})$ to DO. DO decrypts $f(\mathbf{x})$ and obtains $f(\mathbf{x})$.

Theorem 1. *We assume that SP is a semi-honest entity and assume that DO and SP do not collude with each other. Let* $\mathbf{x}$ *be a private data of DO. If the HE scheme [23] provides semantic security, after performing the homomorphic operations on* $\mathbf{x}$ *and the evaluation of* $f(\mathbf{x})$ *on* $\mathbf{x}$*, DO learns* $f(\mathbf{x})$ *but nothing else, SP learns nothing.*

Security Proof. The security proof of the proposed P^{2}OLR follows the simulation-based paradigm [37]. Let the view of DO and SP during the evaluation be $\mathcal{V}_{DO}$ *and* $\mathcal{V}_{SP}$*, respectively. The view* $\mathcal{V}_{SP}$ *of SP consists of* $\{pk, rk, gk, \mathbf{x}, \mathbf{y}, f(\mathbf{x})\}$*. We construct a simulator* $\mathcal{S}_{SP}$ *as*

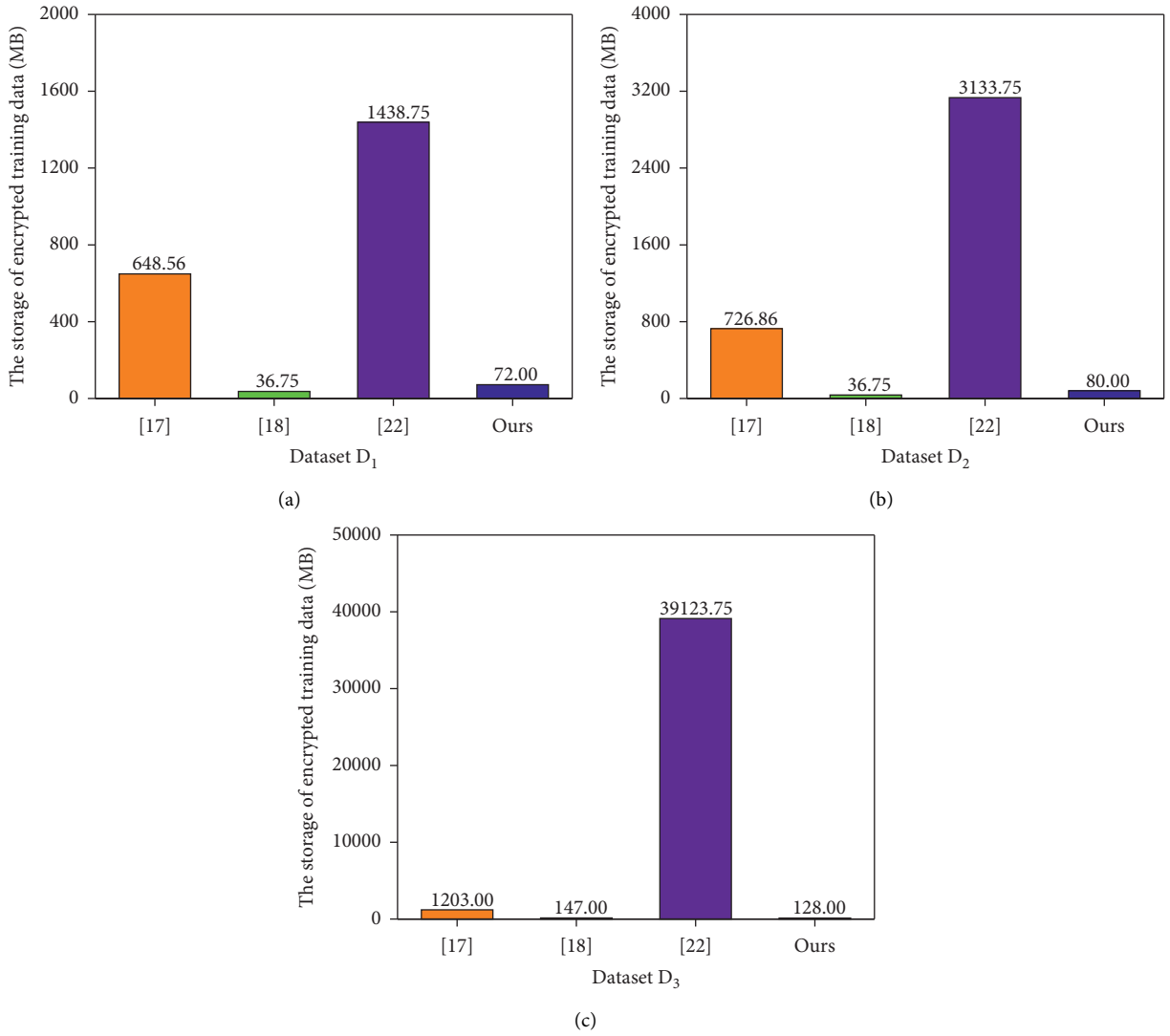


Figure 5: The storage of encrypted training data.

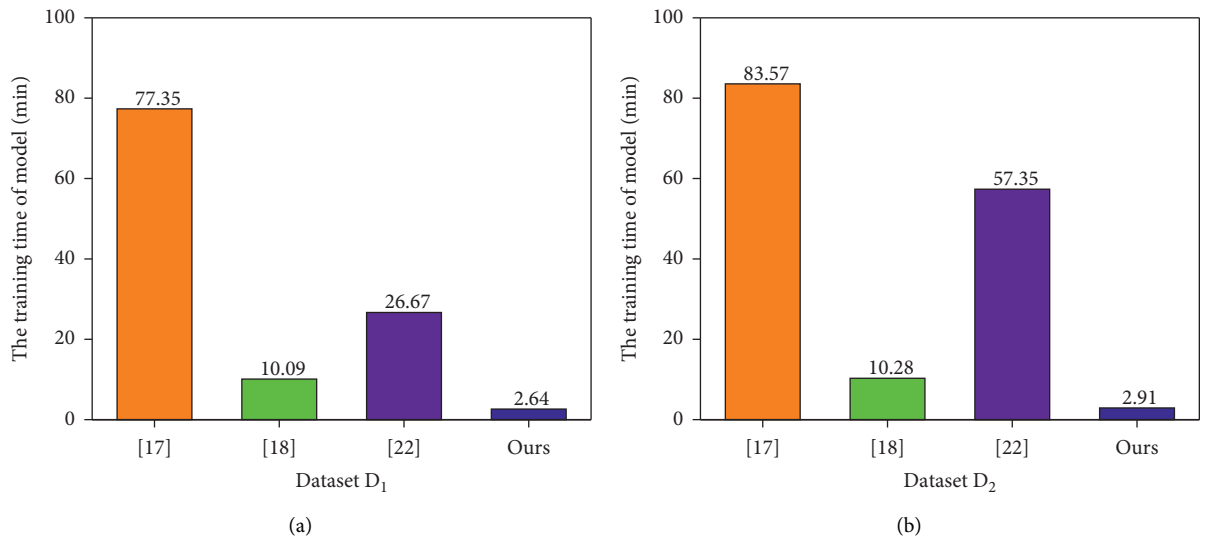
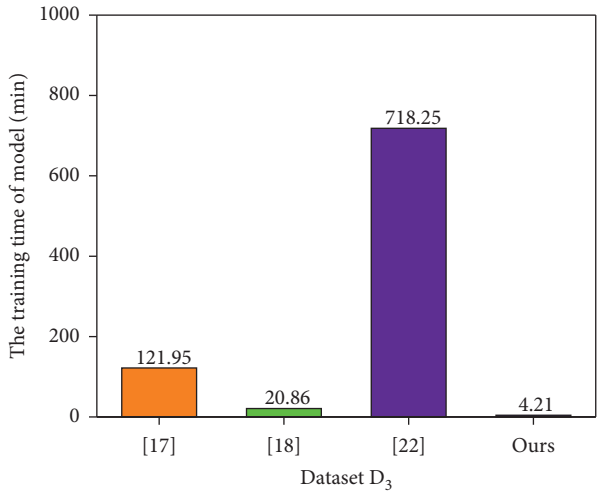
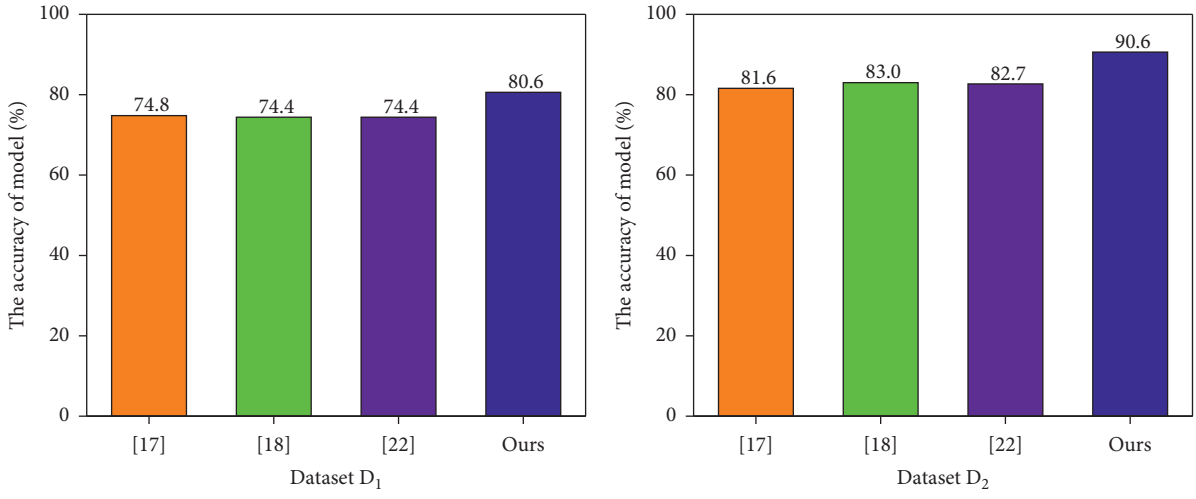


Figure 6: Continued.

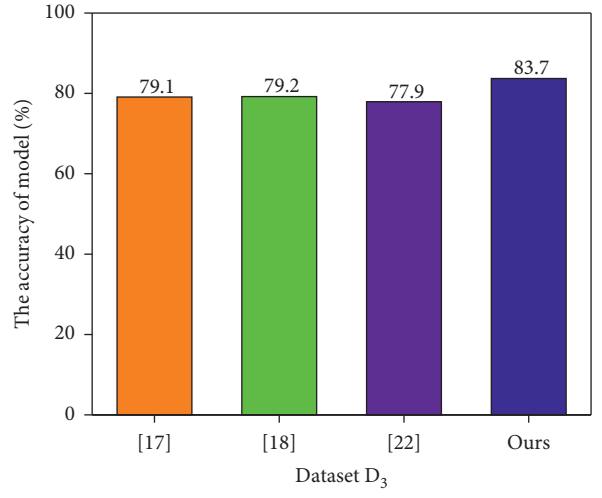


(c)

Figure 6: The training time of model.



(a) (b)



(c)

Figure 7: The accuracy of model.

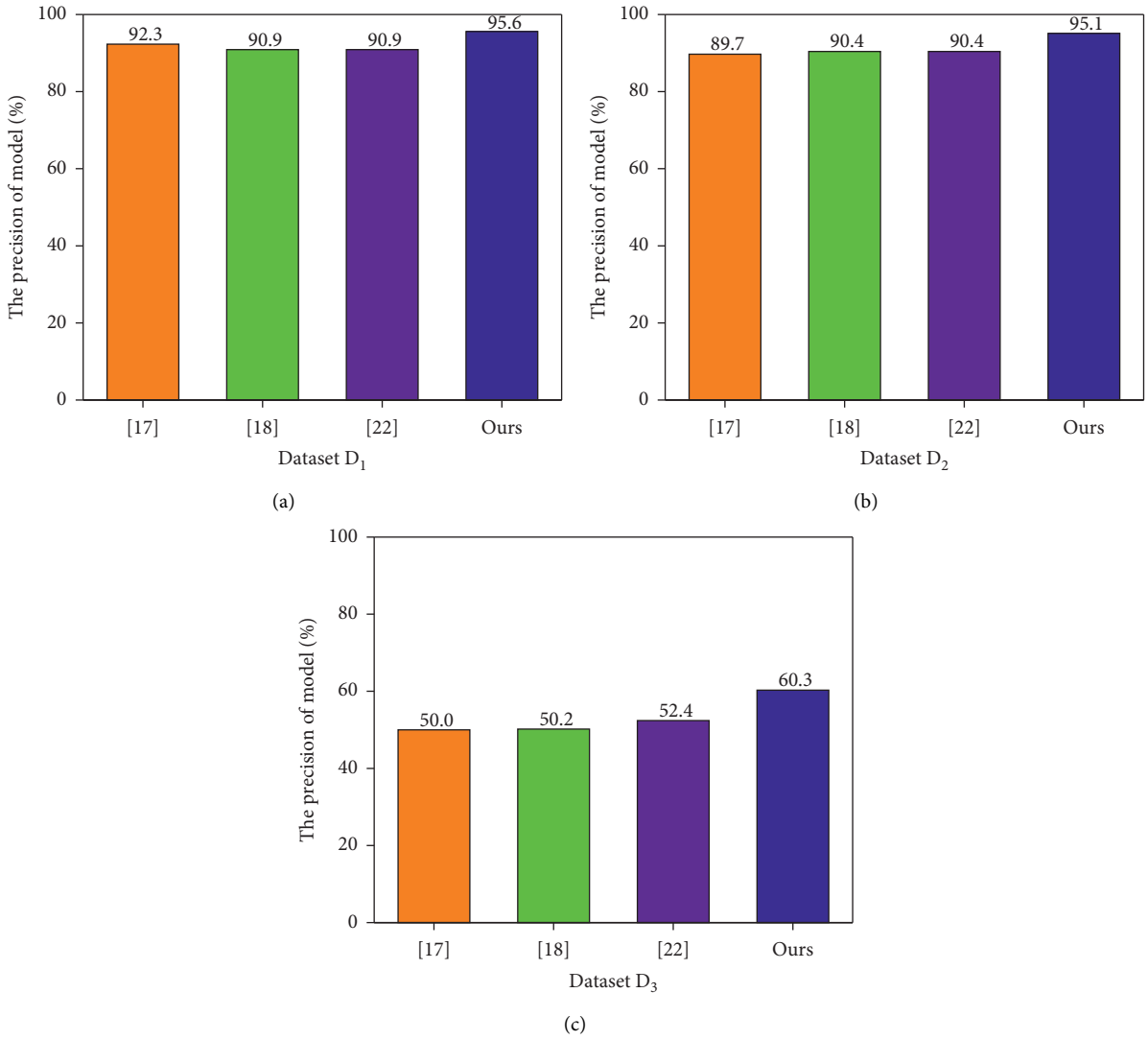


Figure 8: The precision of model.

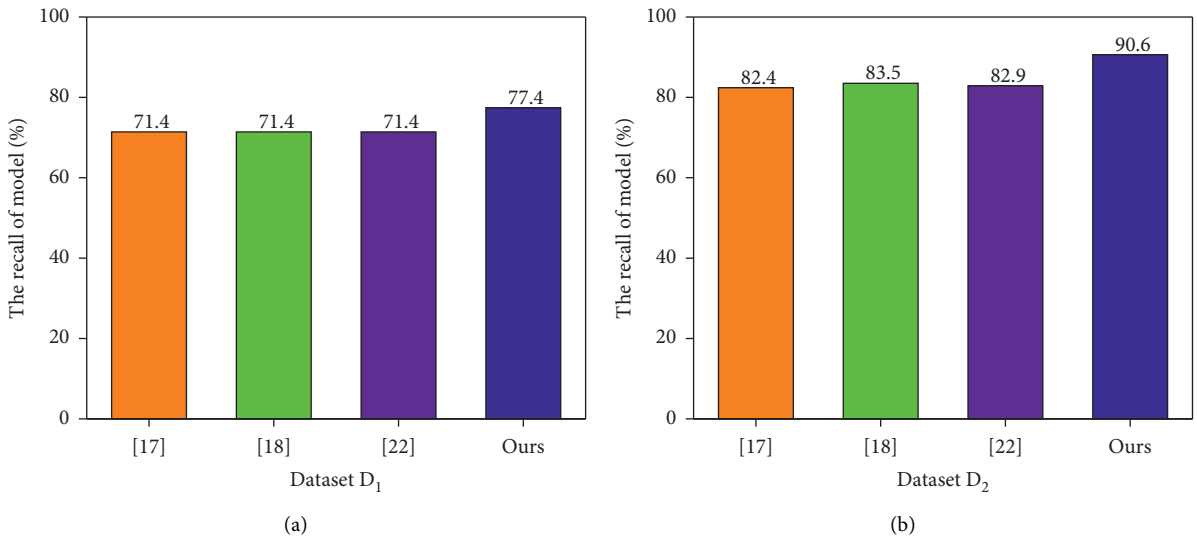
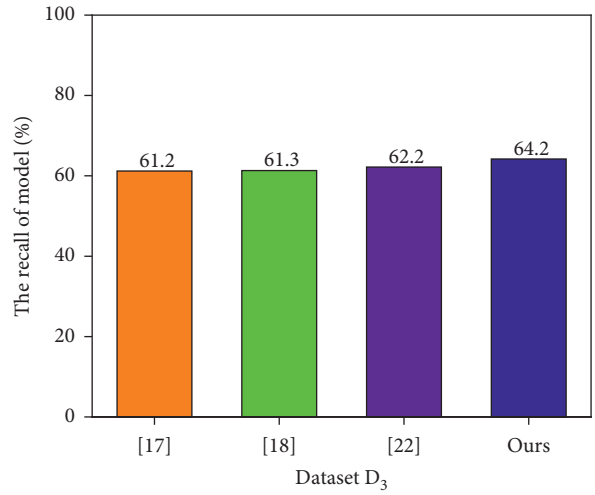
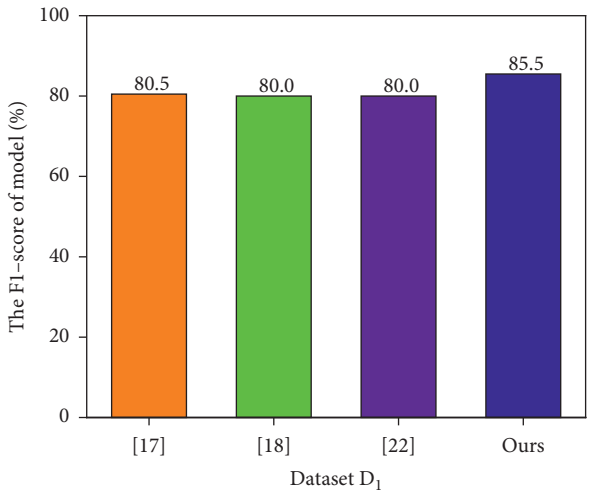


Figure 9: Continued.

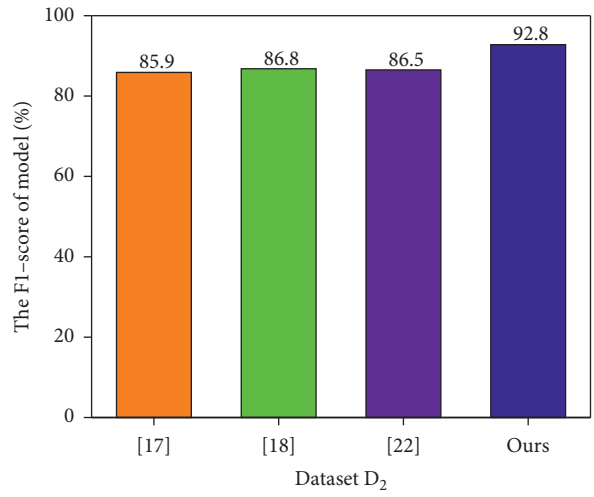


(c)

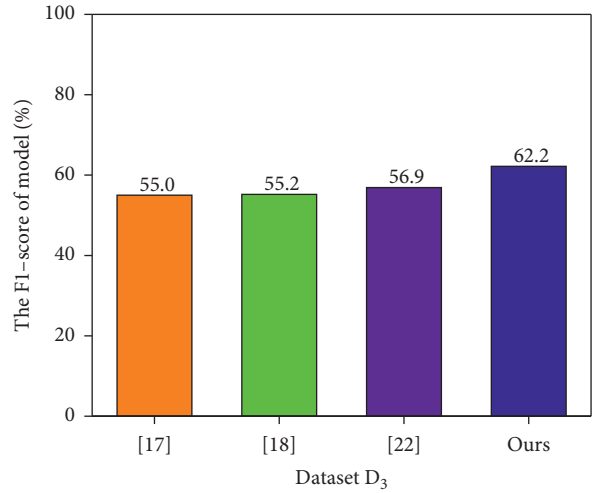
Figure 9: The recall of model.



(a)



(b)



(c)

Figure 10: The F1-score of model.

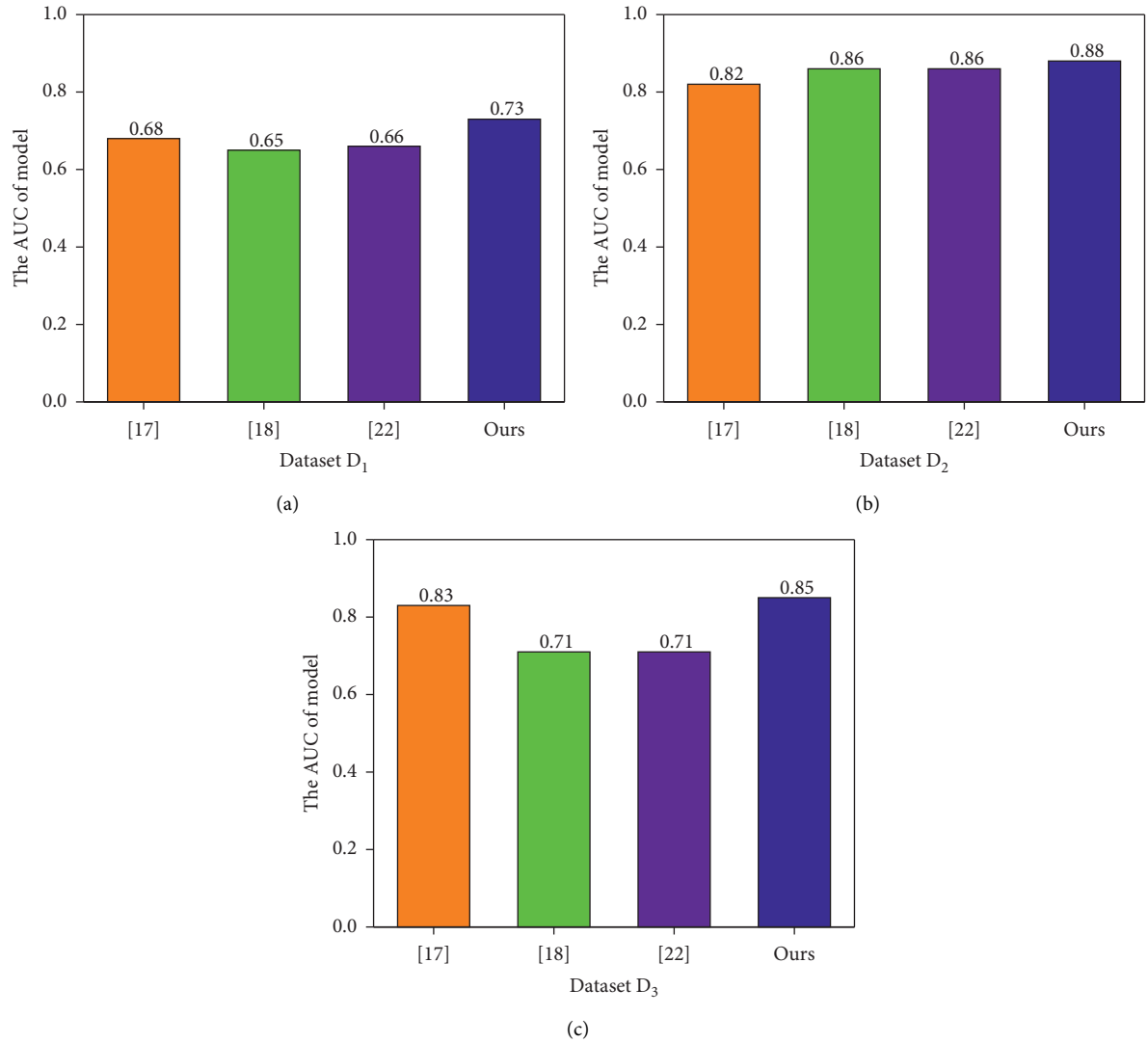


Figure 11: The AUC of model.

follows. $\mathcal{S}_{SP}$ *randomly chooses input data* $\mathbf{x}', \mathbf{y}', f(\mathbf{x}')$. *Then,* $\mathcal{S}_{SP}$ *simulates* $\mathcal{V}_{SP}$ *by* $\mathcal{V}_{SP}' = \{pk, rk, gk, \mathbf{x}', \mathbf{y}', f(\mathbf{x}')\}$. *Since the HE scheme [23] provides semantic security by assumption,* $\mathcal{V}_{SP}$ *and* $\mathcal{V}_{SP}'$ *are indistinguishable. Therefore, the proposed P*^{2}*OLR is secure against a semi-honest SP.*

7. Conclusion

In this paper, we present a method for achieving a P^{2}OLR on encrypted training data, which enables data owners to utilize the powerful storage and computing resources of cloud service providers for logistic regression analysis without exposing the privacy of training data. We take advantage of the batching technique and SIMD mechanism in HE to speed up the training progress. On the three public datasets, compared with the related P^{2}OLR schemes [17, 18, 22], the model training time of the proposed P^{2}OLR is reduced by more than 71.7%, and the proposed P^{2}OLR has over 4.5%, 3.3%, 2.0%, 4.0%, and 0.02 performance in terms of the accuracy, precision, recall, F1-score, and AUC of model. There are still some limitations in applying our scheme to arbitrary datasets and performing arbitrary number of iterations on encrypted training data. In the future, we will extend our scheme to efficiently support P^{2}OLR with arbitrary number of iterations.

Appendix

(1) key_generation(*params*) $\longrightarrow$ {*sk*, *pk*, *rk*, *gk* }: Given the poly_modulus_degree N and coeff_modulus Q, it returns the secret_key *sk*, public_key *pk*, relinearization_key *rk*, galois_key *gk*.

(2) encode_double($\mathbf{x}$, Δ, $\mathbf{x}$): Given the message vector $\mathbf{x} \in \mathbb{C}^{N/2}$ and scaling factor Δ, it expands $\mathbf{x}$ to $\mathbb{H}$ by $\pi^{-1}(\mathbf{x})$, scales $\pi^{-1}(x)$ by $\Delta \cdot \pi^{-1}(\mathbf{x})$, and outputs the plaintext $\mathbf{x} = \sigma^{-1}(\Delta \cdot \pi^{-1}(\mathbf{x})) \in \mathcal{R}$.

(3) decode_double($\mathbf{x}$, $\mathbf{x}$): Given the plaintext $\mathbf{x}$, it computes
$\sigma \cdot \mathbf{x} = \sigma \cdot \sigma^{-1}(\Delta \cdot \pi^{-1}(\mathbf{x})) = \Delta \cdot \pi^{-1}(\mathbf{x}) \in \mathbb{H}$,
$\Delta^{-1} \cdot [\Delta \pi^{-1}(\mathbf{x}) \approx \pi^{-1}(\mathbf{x})]$, and outputs the message vector $\mathbf{x} = \pi \cdot \pi^{-1}(\mathbf{x}) \in \mathbb{C}^{N/2}$.

(4) encrypt (**x**, **x**): Given the plaintext **x**, it encrypts **x** into a ciphertext **x**, and outputs the ciphertext **x**.

(5) decrypt (**x**, **x**): Given a ciphertext **x**, it decrypts **x** into a plaintext **x**, and outputs the plaintext **x**.

(6) add (**x**, **y**, **x** + **y**): Given two ciphertexts **x** and **y**, it computes **x** + **y** and saves the result as a new ciphertext **x** + **y**.

(7) add_inplace(**x**, **y**): Given two ciphertexts **x** and **y**, it computes **x** + **y** and saves the result in ciphertext **x**.

(8) add_plain(**x**, **y**, **x** + **y**): Given a ciphertext **x** and a plaintext **y**, it computes **x** + **y** and saves the result as a new ciphertext **x** + **y**.

(9) sub(**x**, **y**, **x** − **y**): Given two ciphertexts **x** and **y**, it computes **x** − **y** and saves the result as a new ciphertext **x** − **y**.

(10) multiply(**x**, **y**, **x** ∗ **y**): Given two ciphertexts **x** and **y**, it computes **x** ∗ **y** and saves the result as a new ciphertext **x** ∗ **y**.

(11) multiply_plain(**x**, **y**, **x** ∗ **y**): Given a ciphertext **x** and a plaintext **y**, it computes **x** ∗ **y** and saves the result as a new ciphertext **x** ∗ **y**.

(12) mod_switch_to_inplace(**x**\ **x**, **y**.parms_id()): Given a ciphertext/plaintext **x**\ **x** and a levels **y**.parms_id() of ciphertext **y**, it switches the levels of **x**\ **x** to **y**.parms_id().

(13) relinearize_inplace(**x**, *rk*): Given a ciphertext **x** and a relinearization_key *rk*, it relinearizes **x** and saves the result in ciphertext **x**.

(14) rescale_to_next_inplace(**x**): Given a ciphertext **x**, it switches the modulo of **x** to the next levels, reduces the length of the plaintext accordingly, and saves the result in ciphertext **x**.

(15) set_scale(Δ): Given a scaling factor Δ, it scales the ciphertext **x** by computing **x**.set_scale(Δ), and outputs the ciphertext **x**.

(16) rotate_vector(**x**, *k*, *gk*, **y**): Given a ciphertext $\mathbf{x} = [x_0, x_1, \cdots, x_{N/2-1}]$, a rotation value *k*, and galois_key *gk*, it rotates **x** left by *k*, and saves the result as a new ciphertext $\mathbf{y} = [x_k, x_{k+1}, \cdots, x_{N/2-1}, x_0, x_1, \cdots, x_{k-1}]$.

Data Availability

Previously reported Umaru Impact Study, Myocardial Infarction dataset from Edinburgh and Nhanes III datasets were used to support this study and are available at https://doi.org/10.1186/s12920-018-0401-7. These prior studies (and datasets) are cited at relevant places within the text as references [18].

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant no. U19B2021.

References

[1] Y. Jiang, J. Hamer, C. Wang et al., "SecureLR: secure logistic regression model via a hybrid cryptographic protocol," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 16, no. 1, pp. 113–123, 2019.

[2] V. V. P. Wibowo, Z. Rustam, A. R. Laeli, and A. A. Said, "Logistic regression and logistic regression-genetic algorithm for classification of liver cancer data," in *Proceedings of theInternational Conference on Decision Aid Sciences and Application*, pp. 244–248, Sakheer, Bahrain, December 2021.

[3] B. Liu, L. Lu, Q. Zeng, and Y. Li, "Implementation of credit scoring card model based on logistic regression and lightgbm," in *Proceedings of the International Conference on Control Science and Electric Power Systems*, pp. 175–178, Shanghai, China, May 2021.

[4] Z. Han, L. Lu, and H. Liu, "A differential privacy preserving approach for logistic regression in genome-wide association studies," in *Proceedings of the International Conference on Networking and Network Applications*, pp. 181–185, Daegu, Korea (South), October, 2019.

[5] X. Jiang, M. Kim, K. Lauter, and Y. Song, "Secure outsourced matrix computation and application to neural networks," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pp. 1–23, Toronto, Canada, October 2018.

[6] J. M. Cortas-Mendoza, A. Tchernykh, M. Babenko, L. B. Pulido-Gaytan, and A. Avetisyan, "Privacy-preserving logistic regression as a cloud service based on residue number system," in *Proceedings of the 6th Russian Supercomputing Days*, pp. 598–610, Moscow, Russia, September 2020.

[7] P. Mohassel and Y. Zhang, "SecureML: "A system for scalable privacy-preserving machine learning," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 19–38, San Jose, CA, USA, May 2017.

[8] J. Feng, L. Liu, Q. Pei, and K. Li, "Min-max cost optimization for efficient hierarchical federated learning in wireless edge networks," *IEEE Transactions on Parallel and Distributed Systems*, p. 1, 2022.

[9] J. Feng, W. Zhang, Q. Pei, J. Wu, and X. Lin, "Heterogeneous computation and resource allocation for wireless powered federated edge learning systems," *IEEE Transactions on Communications*, vol. 70, no. 5, pp. 3220–3233, 2022.

[10] S. Mao, L. Liu, N. Zhang et al., "Reconfigurable intelligent surface-assisted secure mobile edge computing networks," *IEEE Transactions on Vehicular Technology*, p. 1, 2022.

[11] A. C. Yao, "Protocols for secure computations," in *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pp. 1–5, Chicago, Illinois, USA, November 1982.

[12] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st Symposium on Theory of Computing*, pp. 169–178, Bethesda, Maryland, USA, June 2009.

[13] B. Charlotte and V. Frederik, "Privacy-preserving logistic regression training," *BMC Medical Genomics*, vol. 11, no. 4, pp. 13–21, 2018.

[14] H. Chen, R. Gilad-Bachrach, K. Han et al., "Logistic regression over encrypted data from fully homomorphic encryption," *BMC Medical Genomics*, vol. 11, no. S4, p. 81, 2018.
[15] Z. Li and M. Sun, "Privacy-preserving classification of personal data with fully homomorphic encryption: an application to high-quality ionospheric data prediction," *Machine Learning for Cyber Security*, pp. 437–446, 2020.
[16] S. Carpov, N. Gama, M. Georgieva, and J. R. Troncoso-Pastoriza, "Privacy-preserving semi-parallel logistic regression training with fully homomorphic encryption," *BMC Medical Genomics*, vol. 13, no. S7, p. 88, 2020.
[17] M. Kim, Y. Song, S. Wang, Y. Xia, and X. Jiang, "Secure logistic regression based on homomorphic encryption: design and evaluation," *JMIR Medical Informatics*, vol. 6, no. 2, p. e19, 2018.
[18] A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon, "Logistic regression model training based on the approximate homomorphic encryption," *BMC Medical Genomics*, vol. 11, no. S4, p. 83, 2018.
[19] J. H. Cheon, D. Kim, Y. Kim, and Y. Song, "Ensemble method for privacy-preserving logistic regression based on homomorphic encryption," *IEEE Access*, vol. 6, pp. 46938–46948, 2018.
[20] F. Bergamaschi, S. Halevi, T. T. Halevi, and H. Hunt, "Homomorphic training of 30,000 logistic regression models," *Applied Cryptography and Network Security*, vol. 11464, pp. 592–611, 2019.
[21] Q. Wei, Q. Li, Z. Zhou, Z. Ge, and Y. Zhang, "Privacy-preserving two-parties logistic regression on vertically partitioned data using asynchronous gradient sharing," *Peer-to-Peer Networking and Applications*, vol. 14, no. 3, pp. 1379–1387, 2020.
[22] Y. Fan, J. Bai, X. Lei et al., "Privacy preserving based logistic regression on big data," *Journal of Network and Computer Applications*, vol. 171, p. 102769, 2020.
[23] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proceedings of the Advances in Cryptology - ASIACRYPT 2017: 23rd International Conference on the Theory and Application of Cryptology and Information Security*, vol. 10624, pp. 409–437, Hong Kong, China, December 2017.
[24] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption", IACR Cryptology Eprint Archive," 2012, https://eprint.iacr.org/2012/144.
[25] "FV-NFLlib," 2016, https://github.com/CryptoExperts/FV-NFLlib.
[26] "Seal," 2021, https://github.com/microsoft/SEAL.
[27] C. Ilaria, N. Gama, M. Georgieva, and M. Izabachne, "Faster fully homomorphic encryption: bootstrapping in less than 0.1 seconds," in *Proceedings of the Advances in Cryptology ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security*, vol. 10031, pp. 3–33, Hanoi, Vietnam, December 2016.
[28] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science*, pp. 309–325, New Yark, USA, June 2012.
[29] "HElib," 2021, https://github.com/homenc/HElib.
[30] C. Boura, N. Gama, M. Georgieva, and D. Jetchev, "Chimera: combining ring-lwe-based fully homomorphic encryption schemes," *Journal of Mathematical Cryptology*, vol. 14, no. 1, pp. 316–338, 2020.
[31] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: fast fully homomorphic encryption over the torus," *Journal of Cryptology*, vol. 33, no. 1, pp. 34–91, 2020.
[32] "Tfhe," 2021, https://github.com/tfhe/tfhe.
[33] "Heaan," 2022, https://github.com/snucrypto/HEAAN.
[34] "Heaan," 2019, https://github.com/kimandrik/HEAAN.
[35] "Helr," 2019, https://github.com/K-miran/HELR.
[36] "Heml," 2018, https://github.com/kimandrik/IDASH2017.
[37] R. Küsters, A. Datta, J. C. Mitchell, and A. Ramanathan, "On the relationships between notions of simulation-based security," *Journal of Cryptology*, vol. 21, no. 4, pp. 492–546, 2008.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 5498058, 14 pages
https://doi.org/10.1155/2022/5498058

Research Article

Post-Quantum Secure Identity-Based Encryption Scheme using Random Integer Lattices for IoT-enabled AI Applications

Dharminder Dharminder,[1] Ashok Kumar Das,[2,3] Sourav Saha,[2] Basudeb Bera,[2] and Athanasios V. Vasilakos[4]

[1]*Department of Computer Science and Engineering, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Chennai, 601 103, India*
[2]*Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad, 500 032, India*
[3]*Virginia Modeling, Analysis and Simulation Center, Old Dominion University, Suffolk, VA 23435, USA*
[4]*Center for AI Research (CAIR), University of Agder (UiA), Grimstad, Norway*

Correspondence should be addressed to Ashok Kumar Das; iitkgp.akdas@gmail.com

Received 21 April 2022; Accepted 2 June 2022; Published 6 July 2022

Academic Editor: Wenbo Shi

Identity-based encryption is an important cryptographic system that is employed to ensure confidentiality of a message in communication. This article presents a provably secure identity based encryption based on post quantum security assumption. The security of the proposed encryption is based on the hard problem, namely Learning with Errors on integer lattices. This construction is anonymous and produces pseudo random ciphers. Both public-key size and ciphertext-size have been reduced in the proposed encryption as compared to those for other relevant schemes without compromising the security. Next, we incorporate the constructed identity based encryption (IBE) for Internet of Things (IoT) applications, where the IoT smart devices send securely the sensing data to their nearby gateway nodes(s) with the help of IBE and the gateway node(s) secure aggregate the data from the smart devices by decrypting the messages using the proposed IBE decryption. Later, the gateway nodes will securely send the aggregated data to the cloud server(s) and the Big data analytics is performed on the authenticated data using the Artificial Intelligence (AI)/Machine Learning (ML) algorithms for accurate and better predictions.

1. Introduction

According to [1], it is projected by 2027 the market of Internet of Things (IoT) industry will grow by $2 trillion annually, which has already a market of $520 billion in 2022. In the connected world, the IoT makes an environment where various smart devices are interconnected with each other. The advancement of information and communications technology (ICT) makes the IoT technologies and their solutions rich that have great impact to the society for improving the human life advanced and easy. There are enormous applications of IoT, such as Industrial IoT (IIoT), smart cities, healthcare monitoring, smart home, and so on. In an IIoT, various IoT smart devices are connected in an industry to collect manufacturing data in order to predict the failure rates to increase productivity and efficiency [2]. In healthcare application, various smart devices like smartwatches and medical sensors are connected in the body of a patient to collect vital information and provide appropriate health condition of that person. Furthermore, in recent days, smart home application is in limelight where the smart

devices like smart locks and home appliances are connected with each other via the internet and they can be also controlled via the mobile devices. Though IoT has transformed the human life easier, there are various serious threats associated with IoT applications. For instance, it was found by HP that 70% of the devices connected IoT devices are vulnerable to various attacks [3].

In IoT applications, the smart devices exchange the sensitive data among each other and also with various other entities. In such a scenario, an unauthorized user or an attacker may take the advantage to compromise the data by eavesdropping, modifying, updating and deleting the information during the communication [4]. According to broadcom [5] in the year 2017, it was found that there was an approximately 600% hike in attacks against IoT devices in various applications. Therefore, there is a great need to design a secure IoT system to protect the data from the attackers [6].

Once the sensing information from the deployed smart devices in an IoT environment is aggregated by the nearby gateway node or access point, the gathered data needs to be also stored in semi-trusted cloud servers. Now, the stored data at the cloud is huge in volume and it needs data analytics. As a result, it is preferable to used some Big data analytics using traditional Artificial Intelligence (AI)/Machine Learning (ML) algorithms for accurate and better predictions [7, 8].

Ahanger et al. [9] provided various Machine Learning (ML) and Deep Learning (DL) based mechanisms for IoT paradigm. They also provided a taxonomy based on several IoT vulnerabilities, respective attackers and effects, as well as various threats. Iwendi et al. [10] pointed out the important of deep learning (DL) for detecting attacks in IoT paradigm. They suggested DL based mechanism to detect cyber-attacks on IoT using a long short term networks classifier.

Omolara et al. [11] gave an IoT concept and then provided the deep insights into possible solutions to the IoT security challenges due to the heterogeneous nature of IoT, and the respective emerging issues, opportunities, gaps as well as recommendations. Mukhopadhyay et al. [12] pointed out that IoT sensors need to be reliable, safety as well as privacy-aware for the users interacting with them. Thus, they discussed that IoT sensors having advanced AI capabilities will have the potential for identifying, detecting, and avoiding performance degradation as well as discovering new patterns.

Public-key cryptosystem works under a pair of keys (public key and private key), whereas the public key is made public that is accessible by everyone during communication, and the private key is kept secret and only known to the owner (sender/signer). The notion of the "Identity-Based Encryption (IBE)" due to Shamir [13], solves the certificate management problem. The existing Shor's algorithm [14] is a big threat to the existing number-theoretic identity-based encryptions. The main difference of IBE from certificate based public-key encryption schemes lies in the way how the public and secret keys pair generated for a user. A private key generator, say $\mathscr{PKG}$ handles the process of secret key generation, but it executes user authentication process to confirm the validity of a legitimate user. In IBE process, a public key may be an information such as the user's email address or mobile number. The corresponding secret key is generated by embedding the user's identity with the $\mathscr{PKG}$'s master secret. This process removes the need of certificate that is required for verification of a legitimate recipient's public key. The IBE process also solves the problems related to key generations and distributions in a multi-user settings. In case of limited resources, it can also offer the potential solution to make the process resource efficient.

In the literature, we have three important classes of identity-based encryptions (IBE) (see in Figure 1): 1) IBE based on bilinear pairings [15–18], 2) IBE based on quadratic residue [19, 20], and 3) IBE based on lattices [21]. To the best of our knowledge, most of the constructions proposed in the standard model relies on bilinear pairings.

Chamola et al. [22] reviewed that the disruption which the quantum computers have caused in the cryptographic field. They pointed out that the existing public key encryption schemes can be broken by the quantum computers, and as a result, there is a requirement for hunting the new cryptographic mechanisms that need to be secure in the post-quantum era. Hassija et al. [23] provided a review on several quantum computing applications that can be applied in different computer science areas, including "cryptography", "machine learning", "deep learning" as well as "quantum simulations". They also provided several real-life case studies on "risk analysis", "logistics", and "satellite communication". Hassija et al. [24] also discussed that with the help of online cloud services, the first generation of quantum computers can be programmed and accessed using the available software development kits. Moreover, they presented a growing trend in both the investments as well as patents in the quantum computing field. In recent years, the lattice-based cryptography has played a very important role in the post-quantum era for various real-life applications, such as "Vehicular Ad Hoc Network (VANET)" [25], " medical Cyber-Physical Systems (CPS)" [26] and "mobile communications" [27].

1.1. Research Contributions. There are two reasons to move towards post quantum secure lattice based cryptography: a) simple algebraic operations that are based on matrix multiplication and b) secure against existing quantum assisted algorithms. The main contributions of the work are listed below:

- This article presents a new identity-based encryption based on lattices without using the random oracles. The proposed encryption is anonymous in nature [28], which means that the cipher does not reveal the recipient's identity.
- Our proposed encryption is selective-ID secure [29], and can be converted to an adaptive-ID secure [15, 16, 30] by

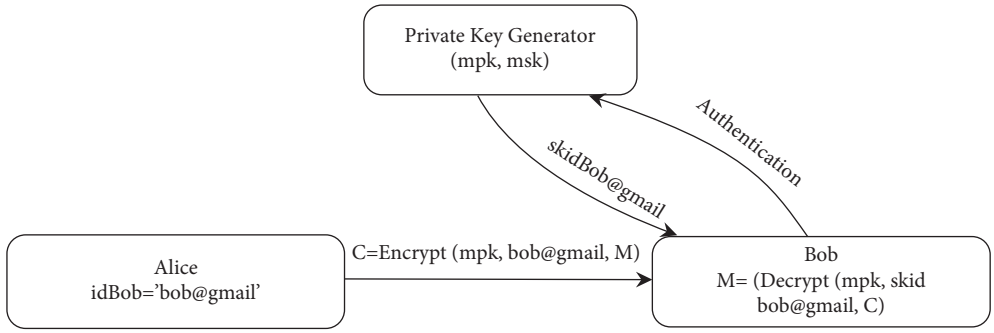


Figure 1: A communication model for IBE.

taking the bit-wise decomposition of the corresponding identities.

- The proposed encryption is inspired from the Water's [18] encryption and signature that use only non-zero positions of bits in the decomposition of the corresponding identity. The encryption is secure under "learning with errors" assumption without the random oracles.
- If κ is an appropriate security parameter and $O(\kappa^2)$ is the size of a public key, we can relate the computation time in terms of security parameter complexity as $O(\kappa^2)$, and it can be compared with the size of classic public key (such as RSA [31] and ElGamal [32] cryptosystems) as $O(\kappa)$ and computation time in terms of security parameter as $O(\kappa^3)$, respectively [33].
- We then incorporate the constructed identity based encryption (IBE) for IoT applications, where the smart devices send securely the sensing data to their nearby gateway nodes(s) with the help of IBE and the gateway node(s) secure aggregate the data from the smart devices by decrypting the messages using the proposed IBE decryption. Later, the authenticated data stored at the cloud server(s) will be used for accurate and better predictions with the help of AI/ML algorithms.

1.2. Paper Outline. In Section 2, the security of an Identity-Based Encryption (IBE) is discussed. Section 3 provides a discussion of basic preliminaries that are needed to analyze the proposed scheme in Section 4. In Section 5, we incorporating our proposed IBE scheme for IoT-enabled AI applications. Next, the security analysis of the proposed scheme under standard models is discussed in Section 6. A comparative study among the proposed scheme and other relevant schemes is given in Section 7. Some concluding remarks are then provided in Section 8.

2. Security of an Identity-Based Encryption

An identity-based encryption (IBE) [15] comprises of four phases (algorithms): a) **Set-up**, b) **Extraction**, c) **Encrypts**, and d) **Decrypts**. The **Set-up** algorithm is run under the public parameters and a secret master key. The **Extraction** algorithm makes use of the master key to create a secret key respective to the given identity. The **Encrypts** algorithm encrypts a message using the identity. Finally, the **Decrypts** algorithm decrypts a ciphertext with the help of the corresponding private key.

2.1. Both Selective and Adaptive Encryption. The security model of an IBE [15] defines the "indistinguishable adaptive chosen cipher and chosen identity (**IND-ID-CCA2**)" security. It allows a probabilistic polynomial time-adversary, say $\mathscr{A}$ to pick an identity on which it wants to target. A weaker version of an IBE security [34] restricts the adversary $\mathscr{A}$ to announce the target or identity at advance, that is known as the "indistinguishable adaptive chosen cipher and selective identity (**IND-sID-CCA2**)" security. We have described this system as a selective identity and chosen cipher secure identity-based encryption. In this version of encryption, we will not allow the adversary $\mathscr{A}$ to process decryption queries on the target identity, which implies a weaker notion of the "indistinguishable against adaptive chosen cipher and chosen identity (**IND-ID-CCA2**) and indistinguishable adaptive chosen cipher and selective identity (**IND-sID-CCA2**)", respectively. Another important notion is the "indistinguishable cipher against chosen plaintext attack (**IND-CPA**)", which is also called semantic security.

2.2. Security Model. We now define an IBE semantic security under the **IND-sID-CCA2** with the help of a game that is played between a challenger, say $\mathscr{C}$ and an adversary $\mathscr{A}$. The description of the game is given below.

1. **Target-phase:** $\mathscr{A}$ declares the target identity ID^* in advance.
2. **Set-up-phase:** $\mathscr{C}$ executes the **Set-up-phase**, generates the public parameters for $\mathscr{A}$, and keeps the master key as secret.

Phase-1. $\mathscr{A}$ submits queries $q_1, q_2, \ldots, q_m$ corresponding to the identities $\mathrm{ID}_1, \mathrm{ID}_2, \ldots, \mathrm{ID}_m$, respectively, where $\mathrm{ID}_i \neq \mathrm{ID}^*$ for $1 \leq i \leq m$. Now, $\mathscr{C}$ runs an algorithm, called **Extraction** $(\mathrm{Mk}, \mathrm{ID}_i)$ with the master key Mk and identity ID_i to obtain the private key D_i corresponding to the identity ID_i, ID_i, which is the public key. Then, it sends D_i to $\mathscr{A}$, where all the queries are processed adaptively meaning that

$\mathcal{A}$ can make queries with the knowledge of the previous queries.

3. **Challenge-phase:** After completion of **Phase-1**, $\mathcal{A}$ submits two messages m_0 and m_1 from the message space on which it executes the challenge. The challenger $\mathcal{C}$ then picks $b \in \{0, 1\}$ randomly, and outputs $c =$**Encrypts**(paramts, ID*, m_b) and sends it to $\mathcal{A}$, where paramts, *paramts* are the parameters relevant to encryption.

Phase-2. $\mathcal{A}$ submits the adaptive extraction queries q_{m+1}, q_{m+2}, . . ., q_n corresponding to ID_{m+1}, ID_{m+2}, . . ., ID_n, where $\mathrm{ID}_i \neq \mathrm{ID}^*$, respectively. Next, $\mathcal{C}$ replies as in **Phase-1**.

4. **Guess-phase:** Finally, $\mathcal{A}$ requires to guess a bit $b' \in \{0, 1\}$. The game is won by $\mathcal{A}$ if $b' = b$; otherwise, $\mathcal{A}$ looses the game.

We call such an adversary $\mathcal{A}$ as an **IND-sID-CPA**-adversary, and define the advantage of $\mathcal{A}$ attacking the identity-based encryption, say $\mathcal{P}$ as

$$A\ DV_{\mathcal{A}}(\mathcal{P}) = \left|\Pr[b = b'] - \frac{1}{2}\right|. \tag{1}$$

We can also describe an adaptive phase to the above notion by excluding the target phase, and permitting $\mathcal{A}$ to wait for the challenge phase to declare ID* as challenge identity. $\mathcal{A}$ can submit the arbitrary key extraction queries as in **Phase-1**, and then select an identity ID*, ID^* as a target. But, the only condition imposed here is that $\mathcal{A}$ cannot submit extraction query on ID*, ID^* in **Phase-1**, and the resulting notion is called as **IND-ID-CPA** security. In **Cipher-Anonymity** along with semantic security, we have another important notion of cipher anonymity under chosen plaintext attack.

3. Preliminaries

Let $\mathbb{R}$ be a set of real numbers and $x \in \mathbb{R}$ be a real number. We denote $\lfloor x \rfloor$ as the largest integer, but not greater than x, whereas $\llcorner x \urcorner = \llcorner x + 1/2 \urcorner$ denotes the integer closest to x, with ties broken upward. We apply a bold big letter **A** to denote a matrix and a bold small letter **x** to denote a column vector of the matrix **A**, where [**A**|**x**] denotes concatenation of the matrix **A** with a vector **x**. Let $\mathbb{Z}$ denote the set of all integers and $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ be a quotient ring under integers modulo a prime q, that is, a collection of the (left or right) cosets $a + q\mathbb{Z}$ with addition and multiplication operations in the quotient ring $\mathbb{Z}_q$. It is worth noticing that $y = z \,(\mathrm{mod} q)$ if and only if $y + q\mathbb{Z} = z + q\mathbb{Z}$, which is an obvious fact about the equality of cosets.

3.1. Lattice. A lattice Δ is defined with the following two properties: 1) it is an additive subgroup which implies $0 \in \Delta$, and $-x$, $x + y \in \Delta$ for all x, $y \in \Delta$, and 2) it is discrete that implies every $x \in \Delta$ possesses a neighborhood in $\mathbb{R}^n$ in which x is the only lattice point in the neighborhood. More specifically, the i^{th} successive minima $\lambda_i(\Delta)$ is the smallest

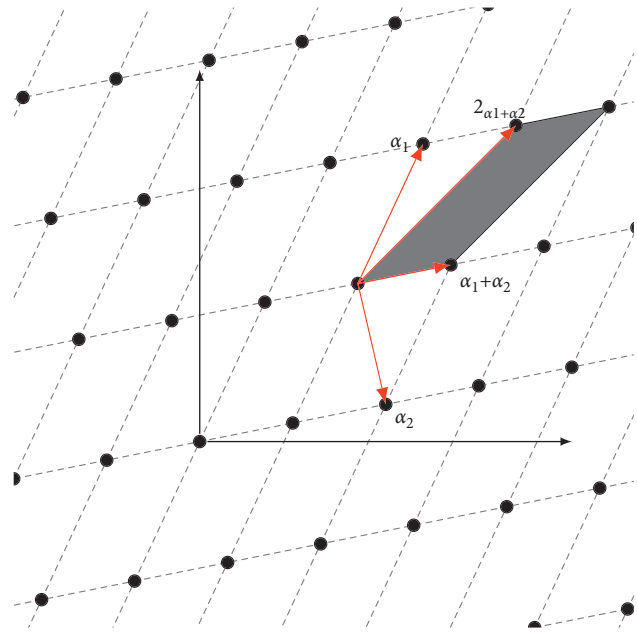


Figure 2: A two-dimensional lattice with bad basis.

Euclidean norm ℓ such that Δ possesses i number of linearly independent vectors of norm less than or equal to ℓ. Due to properties of a discrete group, one can observe that the quotient group $\mathbb{R}^n/\Delta$ of cosets $c + \Delta = \{c + v: v \in \Delta\}$, $c \in \mathbb{R}^n$, under the usual addition: $(c_1 + \Delta) + (c_2 + \Delta) = (c_1 + c_2) + \Delta$ in the quotient group. A fundamental domain of Δ is a set $\mathbb{F} \subset \mathbb{R}^n$ that contains exactly one representative $\widehat{c} \in (c + \Delta) \cap \mathbb{F}$ of each coset $c + \Delta$.

3.2. Bases and Fundamental Parallelepiped. A lattice (see Fig. 2) is generated by a basis $B = \{b_1, b_2, \ldots, b_m\}$ and the integer linear combination of the linearly independent vectors $b_1, b_2, \ldots, b_m$ in the basis as $\Delta = \Delta(B) = \{\sum_{i=1}^{m} z_i b_i : z_i \in \mathbb{Z}\}$. The positive integer m is the rank of the basis and n represents the dimension of the space under consideration. We can consider $m = n$ to represent a full rank lattice. A lattice possesses infinitely many bases, because if B is a basis then BU is also a basis for a unimodular matrix. If B is a basis of the lattice Δ, the fundamental domain is the parallelepiped $\mathbb{P}(B) = B[-1/2, 1/2)$ centered at the origin. Note that parallelepiped is formed by "six parallelogram sides to result in a three-dimensional figure" or a "Prism", which contains a parallelogram base.

Definition 1. *Let $b_1, b_2, \ldots, b_m \in \mathbb{R}^n$ be linearly independent tuples, a lattice Δ generated by a basis $B = \{b_1, b_2, \ldots, b_m\}$ is denoted $\Delta(b_1, b_2, \ldots, b_m) = \{\Sigma_i z_i b_i : z_i \in \mathbb{Z}\}$. The integers m and n denote the rank of the concerned matrix and the dimension of given lattice, respectively.*

Definition 2. *Let $b_1, b_2, \ldots, b_m \in \mathbb{R}^n$ be linearly independent tuples that generate a lattice $\Delta(b_1, b_2, \ldots, b_m) = \{\Sigma_i z_i b_i : z_i \in \mathbb{Z}\}$, its dual lattice be $\Delta^* = \{z \in \mathbb{Z} | \forall\, y \in \Delta, \langle z, y \rangle \in \mathbb{Z}\}$, where Δ can be represented as*

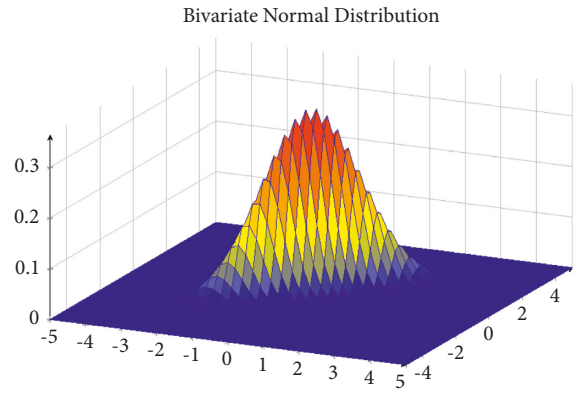


Figure 3: Gaussian distribution in multi dimensions.

$$\Delta = \mathbf{A} \cdot \mathbf{z} = \begin{bmatrix} | & | & . & | \\ b_1 & b_2 & \dots & b_m \\ | & | & . & | \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix}. \tag{2}$$

3.3. q-ary Lattice. The **q-ary lattice** satisfying $\mathbb{Z}_q^m \subseteq \Delta \subseteq \mathbb{Z}_q^n$, for some integer q, is called q-ary lattice because q times vectors of lattice also belongs to it. Given a matrix modulo $q = \text{poly}(n)$ (depends only dimension of lattice), denoted $A \in \mathbb{Z}_q^{n \times m}$, there are n-dimensional q-ary lattice $\Delta_q^{\perp} = \{z \in \mathbb{Z}^n : Az = 0 (\text{mod} q)\}$ and a coset of the lattice as $\Delta_q^u = \{z \in \mathbb{Z}^n : u = Az (\text{mod} q) | z \in \mathbb{Z}^m\}$, where m, n and q are integers and $m > n$. Here, $\mathbf{A} \cdot \mathbf{z} = 0$ implies that:

$$\begin{bmatrix} | & | & . & | \\ b_1 & b_2 & \dots & b_m \\ | & | & . & | \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{3}$$

and $\mathbf{A} \cdot \mathbf{z} = u$ implies that:

$$\begin{bmatrix} | & | & . & | \\ b_1 & b_2 & \dots & b_m \\ | & | & . & | \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix}. \tag{4}$$

These q-ary lattices are applied in the construction of cryptographic techniques. Now, if the matrix $\mathbf{A}$ is chosen randomly, solving the short vector problem on $\Delta^{\perp}(A)$ is equivalent to solve a hard problem in random lattice.

3.4. Gaussian Measures. Let $x, c \in \mathbb{R}^n$ and $\sigma > 0$ be arbitrary. Then, $\rho_{\sigma,c}(x) = e^{-\pi \|(x-c)\|^2/\sigma^2}$ defines a Gaussian distribution function (see Fig. 3) with center c and scaling σ, where the total measure corresponding to $\rho_{\sigma,c}$ is given by $\int_{x \in \mathbb{R}^n} \rho_{\sigma,c}(x) \mathrm{d}x = \sigma^n$. We can define the discrete Gaussian distribution as $D_{\Delta,\sigma,c}(z) = \rho_{\sigma,c}(z)/\rho_{\sigma,c}(\Delta)$, where $z \in \Delta$ is an arbitrary lattice point. Note that $D_{\Delta,\sigma,c}(z) = D_{\sigma,c}(z)/D_{\sigma,c}(\Delta) = \rho_{\sigma,c}(z)/\rho_{\sigma,c}(\Delta)$.

We now introduce an advanced lattice parameter (called the smoothing parameter [35]) related to the Gaussian measures on random lattices as follows.

Definition 3. Let Δ be a lattice of dimension n and $\varepsilon > 0$ be an arbitrary small real number. The smoothing parameter is defined by $\xi_\varepsilon(\Delta)$ to be the smallest $\sigma > 0$ such that $\sum_{z \neq 0 \in \Delta^*} \rho_{1/\sigma,0}(z) \leq \varepsilon$ holds.

3.5. Hard Assumptions Based on Learning with Errors. The "learning with errors" was introduced by Regev [36], which is secure against quantum computing. In the following, we state the assumption with respect to the Gaussian error distribution [35] and its parameterizations.

Definition 4 (see [36]). *Let $n \in \mathbb{N}$, $q = q(n) > 2$, and $s \in \mathbb{Z}_q^n$ be a secret. Then, $LWE_{s,\xi}$ is a distribution of $\langle b, b^t s + z \rangle$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ with $b \in \mathbb{Z}_q^n$ is an arbitrary random and $z \in \mathbb{Z}_q$ is chosen from ξ, where ξ is the Gaussian distribution.*

Definition 5 (see [37]). *The "Learning with Errors" decision problem is to distinguish between $LWE_{s,\xi}$ which is the distribution of $\langle b, b^t s + z \rangle$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ with randoms $\in \mathbb{Z}_q^n$ and the uniform random distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$, given access to the random samples from the given distribution.*

Regev [37] proved that the decision problem (learning with errors) under a suitable prime modulus q and Gaussian distribution ξ is as hard as solving the worst-case lattice problem, known as "short independent vector problem" and "decision short vector problem" in Euclidean norm, using quantum algorithms. Suppose $\mathbb{R}/\mathbb{Z} = [0, 1)$ is a group with respect to modulo one operation. Let Φ_α be the Gaussian distribution on $\mathbb{R}/\mathbb{Z}$ with mean 0 and standard deviation $\alpha/2\pi$, under modulo one, where $\alpha > 0$ is a real number.

Theorem 1 (see [37]). *Let $\alpha = \alpha(n) \in (0, 1)$ be a real number, and $q = q(n) > 0$ be a prime such that $\alpha q > 2n$ holds. If there exists a quantum algorithm that can solve LWE_{q,Φ_α}, there also exists a quantum algorithm to solve "short independent vectors problem" and approximate "decision short vector problem", in Euclidean norm, under the worst-case with in $O(n/\alpha)$ factors.*

3.6. Regev's Dual Cryptosystem. If Δ is a lattice, its dual is the set Δ^* consisting of tuples $z \in L(\Delta)$, that is, a linear span of Δ such that inner product $\langle z, y \rangle$ is an integer for all $y \in \Delta$. Following the definition, one can easily observe that the dual of $\mathbb{Z}^n$ is $\mathbb{Z}^n$. The inner product between two n-tuples x and y is defined as $\langle x, y \rangle = x^t y = \sum_{i=1}^n x_i y_i$, where $x = (x_1, x_2, \dots, x_n)$ and $y = (y_1, y_2, \dots, y_n)$ are tuples with the real entries.

The dual space Δ^* has the same dimension as its primal space Δ, and both are essentially isomorphic to each other. Therefore, a dual space Δ^* lies in the same space as the primal Δ, and not necessarily be a sub-lattice of Δ. The lattice Δ^* contains non-integers even Δ contains only integers

entries. The dual space is necessarily defined as follows in abstract vector space. If $V \subseteq \mathbb{R}^n$ is a vector space, a function Ψ: $V \longrightarrow R$ is called a linear function if it satisfies the following conditions: 1) $\Psi(\vartheta_1 + \vartheta_2) = \Psi(\vartheta_1) + \Psi(\vartheta_2)$, and 2) $\Psi(a\vartheta_1) = a\Psi(\vartheta_1)$, where $a, b \in \mathbb{R}$ and $\vartheta_1, \vartheta_2 \in V$. The dual space of an abstract vector space V is then the set of all linear functions, where a function Ψ is represented as a tuple $\vartheta \in V$ such that $\Psi(x) = \langle \vartheta, x \rangle$, whereas the dual lattice is considered on the set of integers $\mathbb{Z}$ instead set of reals one $\mathbb{R}$. The dual of lattice Δ is the collection of linear functions of the forms: Ψ: $V \longrightarrow \mathbb{Z}$ represented as tuples in span (Δ). Each vector $\vartheta \in \Delta^*$ generates a linear function $\Psi_\vartheta(x) = \langle \vartheta, x \rangle$ satisfying $\Psi_\vartheta(\Delta) \subseteq \mathbb{Z}$ and partitions Δ into the layers as $\Delta = \cup_{i \in \mathbb{Z}} \{\varrho \in \mathbb{Z}: \Psi_\vartheta(\varrho) = i\}$, where each layer $\Psi_\vartheta^{-1}(i) = \{\varrho \in \Delta: \Psi_\vartheta(\varrho) = i\}$ is necessarily a shifted copy of $\Delta \cap \vartheta^\perp = \{\varrho \in \Delta: \langle \vartheta, \varrho \rangle = 0\}$, that is, a lower dimensional sub-lattice orthogonal to ϑ with distance between layers $1/\|\vartheta\|$ implies that the sparser lattice has denser dual and vice-versa. Therefore, the dual of $c\Delta$ is $1/c\Delta$, where $c > 0$ is an arbitrary real.

Under the hard assumption "learning with errors", one can construct a public key cryptosystem under indistinguishable property of pseudo-random tuple $\langle b, b^t s + z \rangle$ from a random sample. The pseudo-random $b^t s + z \in \mathbb{Z}_q$ is used to mask a bit of the message in Regev's cryptosystem [37]. Furthermore, the dual Regev's cryptosystem consists of three phases: a) D-key-Gen, b) D-Encrypt, and c) D-Decrypt, which are discussed below.

1. **D − key − Gen**: Let $A \in \mathbb{Z}_q^{n \times m}$ be a random matrix, where $m \geq 2n \log(q)$, $f_A : \mathbb{Z}^m \longrightarrow \mathbb{Z}^n$: $e \mapsto Ae \pmod q$. Choose an error $e \leftarrow D_{\mathbb{Z}^m, \sigma}$. It then computes its syndrome as $u = f_A(e)$, where the secret vector $e \in \mathbb{Z}_q^m$ and the public key is $u \in \mathbb{Z}_q^n$.
2. **D − Encrypt**: Let $b \in \{0, 1\}$ be a bit to be encrypted. Choose a random $s \in \mathbb{Z}_q^n$ with an error scalar $x \in \xi$ and an error vector $y \in \xi^m$. It then outputs $c = \langle c_0, c_1 \rangle$, where $c_0 = u^t s + x + b \cdot \lfloor q/2 \rfloor$ and $c_1 = A^t s + y$.
3. **D − De crypt**: To perform the decryption on $c = \langle c_0, c_1 \rangle$ using the secret e under the matrix A, this phase computes $b = c_0 - e^t c_1 \in \mathbb{Z}_q$ and outputs 1 if b is closer to $\lfloor q/2 \rfloor$; else, it is 0.

3.7. Pre-image Samplable Family of Functions. Gentry *et al.* [21] defined a family of pre-image samplable functions that plays a very important role in the construction of the proposed encryption described in Section 4.

Definition 6. A family of pre-image samplable functions [21] *consists of three phases: a)* ***Trap-Gen*** *, b)* ***Sample-Dom*** *, and c)* ***Sample-Pre*** *, which are given below.*

- ***Trap-Gen*** (1^κ)***:*** ***Trap-Gen*** *takes input as the parameter* κ*, and outputs a pair* $\langle A, T \rangle$*, where* A *is utilized in the function* $f_A : D_\kappa \longrightarrow R_\kappa$ *with recognizable domain* D_κ *and range* R_κ*, and* T *is a trapdoor for the function* f_A*.*
- ***Sample-Dom*** (A)***:*** *Under function description A, it will sample* $x \leftarrow \xi$ *over the domain* D_κ *in such a way the distribution of* $f_A(x)$ *is uniform over* R_κ*, and outputs* x *accordingly.*
- ***Sample-Pre*** (T, y)***:*** *Under a trapdoor* T *and a value* $y \in R_\kappa$*, it will sample an element* $x \in D_\kappa$ *from the distribution* ξ *under the criteria that* $f_A(x) = y$*, and it then outputs* x*.*

3.7.1. Correctness. It is worth noticing that **Sample-Dom** samples $x \leftarrow \xi$ over the domain D_κ such that $f_A(x)$ follows a uniform distribution over the range R_κ, and **Sample-Pre** samples $x \in D_\kappa \leftarrow \xi$ as in **Sample-Dom** under condition $f_A(x) = y$.

3.7.2. Security. The security of the pre-image samplable functions [21] is discussed below. The samplable functions [21] must satisfy the following properties:

1. **One-way without trapdoor:** If $\mathcal{A}$ is a probabilistic polynomial time adversary, the advantage $\mathcal{A}(1^\kappa, A, y) \leftarrow f_A^{-1}(y) \subset D_\kappa$ is negligible, where the advantage is considered over all the possible choices of A, the value $y \leftarrow R_\kappa$ is random, and $\mathcal{A}$ tosses the coin randomly.
2. **Pre-image minimum entropy:** If $y \leftarrow R_\kappa$, the conditioned minimum entropy of $x \leftarrow \text{Sample} - \text{Dom}(A)$ is least under the condition $f_A(x) = y$.
3. **Collision-free without trapdoor:** If $\mathcal{A}$ is a probabilistic polynomial time adversary, the advantage $\mathcal{A}(1^\kappa, A)$ results in the distinct $x, x' \leftarrow D_\kappa$ with $f_A(x) = f_A(x')$ is negligible.

Theorem 2 (see [37]). *If* $q = poly(n)$ *is an arbitrary large prime and* $m \geq 5n \log(q)$*, there exists a probabilistic polynomial time algorithm [38] that takes input as* 1^n*, and outputs a matrix* $A \in \mathbb{Z}_q^{n \times m}$ *and a full rank set* $S \subset \Delta^\perp(A)$*, where the distribution corresponding to A is statistically close to a uniform distribution over* $\mathbb{Z}_q^{n \times m}$ *under the length* $\|S\| \leq m^{2.5}$*.*

Another algorithm is known as the sampling Gaussian, denoted by Sample-Gauss, discussed by Gentry *et al.* [21], plays a very important role in cryptographic construction. The Sample-Gauss (B, σ, c) uses a random basis B in sampling from the Gaussian distribution centered at c with the standard deviation σ over the lattice $\Delta(B)$.

Theorem 3 (see [37]). *The probabilistic polynomial time algorithm provided in [21] with inputs as a basis* B*, a lattice* $\Delta(B)$*, an appropriate parameter* $\sigma \geq \|B^*\| \cdot \omega(\sqrt{\log(m)})$ *and arbitrary* $c \in \mathbb{R}^m$*, results in a sample distribution that is statistically close to* $D_{\Delta, \sigma, c}$*.*

The function defined in [21] is a sample pre-image consisting of three phases: a) **Trap-Gen**, b) **Sample-Dom**

and c) **Sample-Pre**. Let κ be a security parameter, $n = \Theta(\kappa)$, $q = \text{poly}(n)$ be a large prime, $m \geq 5n \log(q)$, $L = m^{2.5}$ and Gaussian parameter $\sigma \geq L\omega(\sqrt{\log(m)})$, respectively. Then,

- **Trap-Gen** $(1^\kappa, \sigma)$**:** Under the algorithm in Theorem 3, choose a matrix $A \in \mathbb{Z}_q^{n\times m}$ and a trapdoor $T \in \Delta^\perp(A)$. Consider $D_\kappa = \{e \in A\mathbb{Z}_q^m: \|e\| \leq \sigma\sqrt{m}\}$ and $R_\kappa = \mathbb{Z}_q^n$ and $f_A: D_\kappa \longrightarrow R_\kappa$ such that $f_A(e) = Ae(\text{mod}q)$. This phase then results $\langle A, T\rangle$.
- **Sample-Dom** (A, σ)**:** Assuming B' as a standard basis for $\mathbb{Z}^m$, use **Sample-Gauss** $(B', \sigma, 0)$ to get sample from $D_{\mathbb{Z}^m,\sigma}$.
- **Sample-Pre** (T, σ, y)**:** Let $k \in \mathbb{Z}^m$ be an arbitrary number under condition $\text{Ak} = y(\text{mod}q)$. Then, use **Sample-Gauss** $(T, \sigma - k)$ [21] to sample v from $D_{\Delta^\perp(A),\sigma,-k}$.

Theorem 4 (see [37]). *Assume that the columns of $A \in \mathbb{Z}_q^{n\times m}$ span $\mathbb{Z}_q^n, \varepsilon \in (0, 1/2)$, and $\sigma \geq \eta_\varepsilon(\Delta^\perp(A))$. Then, the syndrome's distribution $u = Ae(\text{mod}q)$ differs by a statistically distance equal to at most 2ε from the uniform distribution over $\mathbb{Z}_q^n$.*

To prove the correctness of the distribution $\xi = D_{\mathbb{Z}^m,\sigma}$, for a given $u \leftarrow \mathbb{Z}_q^n$ and $k \leftarrow \mathbb{Z}^m$ is a solution to $\text{Ak} = u(\text{mod}q)$, the conditional probability distribution of $e \leftarrow D_{\mathbb{Z}^m,\sigma}$ under $Ae = u(\text{mod}q)$ matches perfectly with $k + D_{\Delta^\perp(A),\sigma,-k}$. The correctness of the distribution is as follows. It can be observed that $f_A(e) = Ae(\text{mod}q)$ is indistinguishable from the uniform distribution over $R_\kappa = \mathbb{Z}_q^n$, assuming the columns of $A \in \mathbb{Z}_q^{n\times m}$ spans $\mathbb{Z}_q^n$ [21] with the probability $1 - q^{-n}$. Since $\sigma \geq L\omega(\sqrt{\log(m)})$, and $\|T\| \leq L$, the result in [21] implies $\sigma \geq \eta_\varepsilon(\Delta^\perp(A))$. Thus, as a result, **Sample-Pre** $(v + k)$ is distributed under $D_{\mathbb{Z}^m,\sigma}$ under the condition $A(v + k) = y(\text{mod}q)$.

In the proof of security, we use the functions described in [21], which are one-way and collision resistant functions. A brief discussion of these two properties are given below.

- **One-way without trapdoor:** The process of inversion of f_A under a uniform random $u \leftarrow R_n$ is equivalent to solving "in-homogeneous short integer solution" problem, say $\text{ISIS}_{q,m,\sigma\sqrt{m}}$ [21].
- **Pre-image minimum entropy:** Since all the pre-images follow the discrete Gaussian, it has minimum entropy [21].
- **Collision-free without trapdoor:** Let $z, z' \leftarrow D_\kappa$. Then, a collision implies $A(z - z') = 0(\text{mod}q)$, which actually solves the "short integer solution" problem, say $\text{SIS}_{q,m,2\sigma\sqrt{m}}$.

4. Proposed Identity-Based Encryption (IBE) Scheme in Standard Model

In this section, we propose a new provably secure identity-based encryption scheme. Note that such a scheme has a *compact* public key and also achieves adaptive security in the standard model [39].

Our proposed identity-based encryption scheme consists of four phases: a) **Set-up**, b) **Extraction**, c) **Encrypts** and d) **Decrypts**. We take an identity ID as an arbitrary k-bits string $\{0, 1\}^k$, where $k = \Theta(\kappa)$ for a given security parameter κ. In the following, we now discuss the details of these four phases.

4.1. Set-up Phase. It includes the function **Set-up** (1^κ). First, choose a suitable large prime q, a smoothing parameter σ depending on the security parameter κ and an arbitrary random matrix $A \in \mathbb{Z}_q^{n\times m}$, under a short basis for $\Delta^\perp(A)$, that is, T_A with the help of Ajtai's construction [38]. Let f_A: $\mathbb{Z}_q^m \longrightarrow \mathbb{Z}_q^n$ be a function defined as $f_A(e) = Ae(\text{mod}q)$. Next, pick a tuple $u_0 \in \mathbb{Z}_q^n$ and a random matrix $H_{i,b} \in \mathbb{Z}_q^{n\times\ell}$, where $\ell = m$ and $\widehat{H} = \{\langle i, b, H_{i,b}\rangle: 1 \leq i \leq 2, 0 \leq b \leq 1\}$ is the ordered set. The public parameters are $\{A, u_0, \widehat{H}\}$, whereas T_A is considered as the master secret.

4.2. Extraction Phase. This phase is accomplished by the function **Extraction** $(A, u_0, \widehat{H}, \text{ID}, T_A)$. A decryption key is extracted related to the identity $\text{ID} \in \{0, 1\}^k$ under the master secret T_A as the trapdoor. The following steps need to be executed:

- Let $S = H(\text{ID})$ and U be the set of non-zero positions in the string S. After that, assemble an $n \times \ell$ matrix $H_{\text{ID}} = [H_{i_1\text{mod}2,bi_1\text{mod}2}|H_{i_2\text{mod}2,bi_2\text{mod}2}|\cdots|H_{i_\ell\text{mod}2,bi_\ell}\text{mod}2] \in \mathbb{Z}_q^{n\times\ell}$, where $H_{i_1\text{mod}2,bi_1\text{mod}2}|H_{i_2\text{mod}2,bi_2\text{mod}2}|\cdots|H_{i_\ell\text{mod}2,bi_\ell\text{mod}2} \in \widehat{H}$ as $H_{i_1,b1}$ or $H_{i_1,b0}$ is according to either $i_1(\text{mod}2) = 0$ or $i_1(\text{mod}2) = 1$, respectively.
- Now, sample $r_i \leftarrow \mathbb{Z}_q^\ell$ under $\text{Sample} - \text{Dom}(H_{i,bi\text{mod}2}, \sigma)$, where $1 \leq i \leq k$, and consider $r \leftarrow \mathbb{Z}_q^\ell$ such that $r^t = [r_1^t|r_2^t|\cdots| - r_\ell^t]$.
- Let $u = u_0 + H_{\text{ID}}r \in \mathbb{Z}_q^n$. It can be observed as $u = u_0 + \sum_{i=1}^{\ell} H_{i\text{mod}2,b_{i(\text{mod}2)}}r_i$, where i is the non-zero position in the string S.
- Next, apply the **Sample-Pre** (T_A, σ, u) under the trapdoor T_A to find the pre-image e of u satisfying $u = Ae(\text{mod}q)$, and outputs the private key $\langle e, r\rangle$.

4.3. Encrypts Phase. In this phase, we involve the function **Encrypts** $(A, u_0, \widehat{H}, \text{ID}, b)$. In order to process the encryption on a bit $b \in \{0, 1\}$ under the identity $\text{ID} \in \{0, 1\}^k$ using the master key T_A, the following steps are necessary:

- Let $H_{\text{ID}} = [H_{i_1\text{mod}2,bi_1\text{mod}2}|H_{i_2\text{mod}2,bi_2\text{mod}2}|\cdots|H_{i_\ell\text{mod}2,bi_\ell}\text{mod}2] \in \mathbb{Z}_q^{n\times\ell}$, where $H_{i_1\text{mod}2,bi_1\text{mod}2}|H_{i_2\text{mod}2,bi_2\text{mod}2}|\cdots|H_{i_\ell\text{mod}2,bi_\ell\text{mod}2} \in \widehat{H}$ because $H_{i_1\text{mod}2,b1}$ or $H_{i_1\text{mod}2,b0}$ is based on either $i_1(\text{mod}2) = 0$ or $i_1(\text{mod}2) = 1$.
- Choose an arbitrary $s \in \mathbb{Z}_q^n$.
- Pick $x \in \mathbb{Z}_q$, $y = \langle y_1, y_2, \ldots, y_m\rangle \in \mathbb{Z}_q^m$ and $z = \langle z_1, z_2, \ldots, z^\ell\rangle \in \mathbb{Z}_q^\ell$ which are sampled from the distributions ξ, ξ^m, and ξ^ℓ, respectively, based on the Regev's cryptosystem.

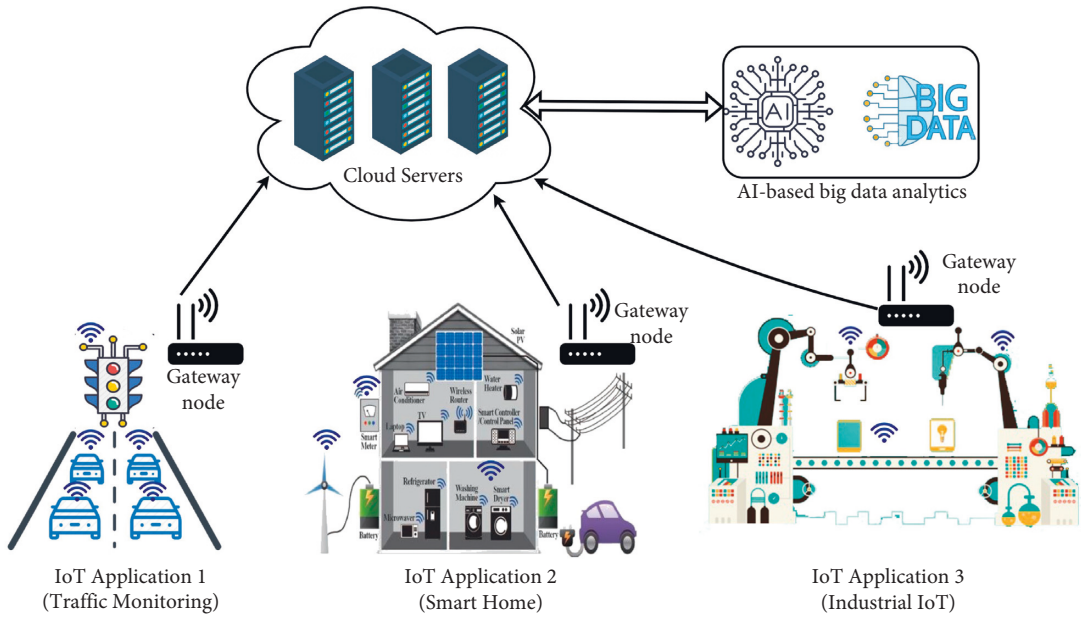


Figure 4: Network model for IoT-enabled AI applications.

- Now, calculate $c_0 = u_0^t s + x + b\lfloor q/2 \rfloor \in \mathbb{Z}_q$, $c_1 = A^t s + y \in \mathbb{Z}_q^m$ and $c_2 = H_{\mathrm{ID}}^t s + z \in \mathbb{Z}_q^\ell$.
- Finally, the initiator sends the output as the cipher $c = \langle c_0, c_1, c_2 \rangle$ to the responder.

4.4. Decrypts Phase. This phase is implemented by the function **Decrypts**(A,u_0,$\widehat{H}$,ID,k,c). After receiving the cipher $c = \langle c_0, c_1, c_2 \rangle$, with the private key $\langle e, r \rangle \in \mathbb{Z}_q^{m+\ell}$, the responder executes the following steps:

- Compute $v = c_0 - e^t c_1 + r^t c_2 \in \mathbb{Z}_q$, and then compare v with $\lfloor q/2 \rfloor$ in $\mathbb{Z}$.
- If $|v - q/2| \le q/4$, it results bit $b = 1$; else, it outputs the bit $b = 0$.

5. Incorporating Proposed IBE Scheme for IoT-Enabled AI Applications

In this section, we first discuss the network model for IoT-enabled AI applications, which is used for incorporating our proposed IBE scheme described in Section 4. Next, we describe the various phases where the proposed IBE scheme has been applied for IoT.

5.1. Network Model. The network model considered for IoT-enabled AI applications using our proposed IBE scheme is presented in Figure 4. The model expresses various applications of IoT, such as traffic monitoring, smart home, and IIoT. In this model, different types of smart sensors, say $\{\mathrm{SS}_i | i = 1, 2, 3, \ldots, n_{\mathrm{ss}}\}$ are connected with each other via the nearby gateway node(s) $\{\mathrm{GWN}_j | j = 1, 2, 3, \ldots, n_{\mathrm{gwn}}\}$, where n_{ss} and n_{gwn} denote the number of smart sensors and gateway nodes to be deployed for each IoT application, respectively. Note that there might be multiple nodes that are connected with a particular application and the gateways GWN_j are further connected with the cloud server(s), say $\{\mathrm{CLS}_k | k = 1, 2, 3, \ldots, n_{\mathrm{cls}}\}$, where n_{cls} is the number of cloud servers. Before initiating any secure communications between GWN_j and CLS_k, they need to complete their registration process which is performed by a fully-trusted registration authority (RA). Similarly, the RA also performs the registration of each smart sensor node to be deployed in various IoT applications. Next, a gateway node needs to perform the secure data aggregation where the data is collected through secure communication among the smart sensors and the gateway node. In this case, we apply the proposed IBE scheme for encryption/decryption of the data. After that the gateway nodes send the data securely to the cloud server(s) for secure data storage purpose. Finally, the cloud servers CLS_k can perform the Big data analytics using AI/ML techniques with the data stored at CLS_k.

5.2. Description of Various Phases. We have the following phases:

- In the *pre-deployment of IoT devices phase*, the trusted RA will perform the registration of each IoT smart device prior to their deployment in respective application. After deployment of the IoT devices, they need to communicate with their nearby gateway node(s). For avoiding various attacks by an adversary, we use the proposed IBE scheme for secure data transfer among the sensor nodes and their gateway node(s).
- In the *registration of gateway nodes and cloud servers phase*, the RA, *RA* also performs the registration of the deployed gateway nodes and cloud servers. For secure communication, we again use the proposed IBE scheme for secure data transfer among the gateway nodes and the cloud servers.

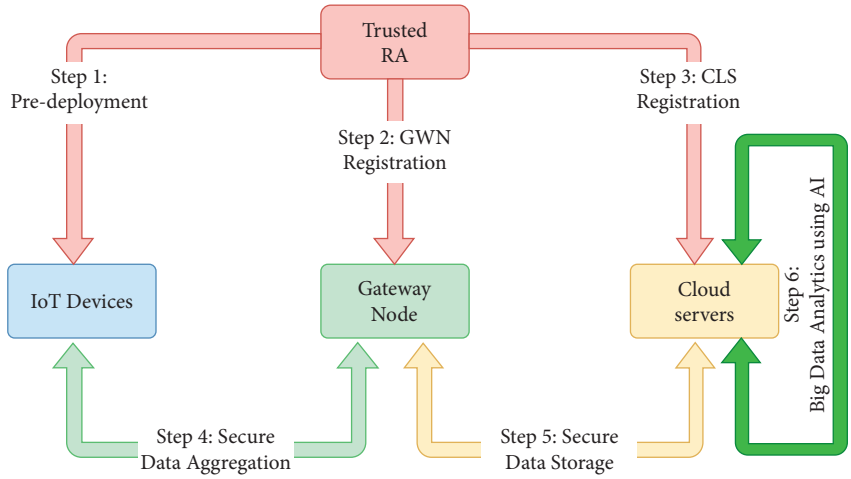


Figure 5: High-level overview of various phases.

- The *secure data aggregation at gateway phase* allows a gateway node to collect the data from its associated IoT smart devices securely using the proposed IBE scheme.
- The *secure data storage at cloud servers phase* permits storage of data at the cloud servers securely from the gateway nodes with the help of the proposed IBE scheme.
- Finally, the *Big data analytics using AI phase* is needed because the cloud servers store a huge volume of data from various IoT applications. Since the Big data analytics provides numerous advantages, such as better decision making and preventing fraudulent activities, it is preferable to do the Big data analytics on the data stored at the cloud servers.

A high-level description of various phases related to IoT-enabled AI applications is given in Figure 5.

5.2.1. Pre-deployment of IoT Devices. Before deploying the IoT smart devices (smart sensors) SS_i in their respective application, the trusted RA, *RA* executes the Set-up phase described in Section 4.1 in order to select the system parameters. The steps are as follows:

- Step 1. The selected public parameters are $\{A, u_0, \widehat{H}\}$, whereas T_A is as the master secret.
- Step 2. For each SS_i, the RA, *RA* assigns a unique identity ID_{SS_i}.
- Step 3. Next, for each SS_i, the RA, *RA* executes the Extraction phase described in Section 4.2 to extract a decryption key related to ID_{SS_i} under the trapdoor master secret T_A. The private key for SS_i is considered as (e_{SS_i}, r_{SS_i}).

5.2.2. Registration of Gateway Nodes and Cloud Servers. The registration process for the deployed gateway nodes GWN_j and cloud servers CLS_k is also based on the execution of the Set-up phase, where the public parameters are $\{A, u_0, \widehat{H}\}$, and T_A is the trapdoor master secret. This phase involves the following steps:

- Step 1. For each GWN_j, the RA, *RA* assigns a unique identity ID_{GWN_j}. In a similar way, for each CLS_k, the RA, *RA* also assigns a unique identity ID_{CLS_k}.
- Step 2. For each GWN_j and CLS_k, the RA, *RA* executes the Extraction phase. After executing this process, the private keys for GWN_j and CLS_k are selected as (e_{GWN_j}, r_{GWN_j}) and (e_{CLS_k}, r_{CLS_k}), respectively.

5.2.3. Secure Data Aggregation at Gateway. In this phase, the following steps are involved:

- Step 1. Suppose the IoT smart sensors SS_i are deployed in their respective IoT applications as shown in Figure 4. The gateway nodes GWN_j and cloud servers CLS_k are also placed accordingly in the network. Let a smart sensor SS_i sense the information (data), say $Data_{SS_i}$ from its deployment area and want to communicate it securely with its gateway node GWN_j, GWN_j. For this purpose, the SS_i, SS_i generates a current timestamp, say TS_{SS_i}, prepares a message of the type $Msg_{SS_i} = \{ID_{SS_i}, TS_{SS_i}, Data_{SS_i}\}$ and encrypts Msg_{SS_i} bit wise using the public parameters, identity of GWN_j, GWN_j and trapdoor master key T_A to create the ciphertext $C_{SS_i} = \{C_0, C_1, C_2\}$ as done in the Encrypts phase described in Section 4.3, where C_0, C_1 and C_2 are the encrypted bit strings corresponding to the bit strings of the Msg_{SS_i}. Next, SS_i sends the encrypted message $\{C_{SS_i}, TS_{SS_i}\}$ to its destination GWN_j, GWN_j via a public channel.

5.2.4. Secure Data Storage at Cloud Servers. In this phase, a cloud server CLS_k, CLS_k receives the encrypted data from the respective gateway nodes GWN_j residing in an IoT application, and stores the encrypted data in its database for further processing. In order to do this, the following steps are executed by the CLS_k, CLS_k:

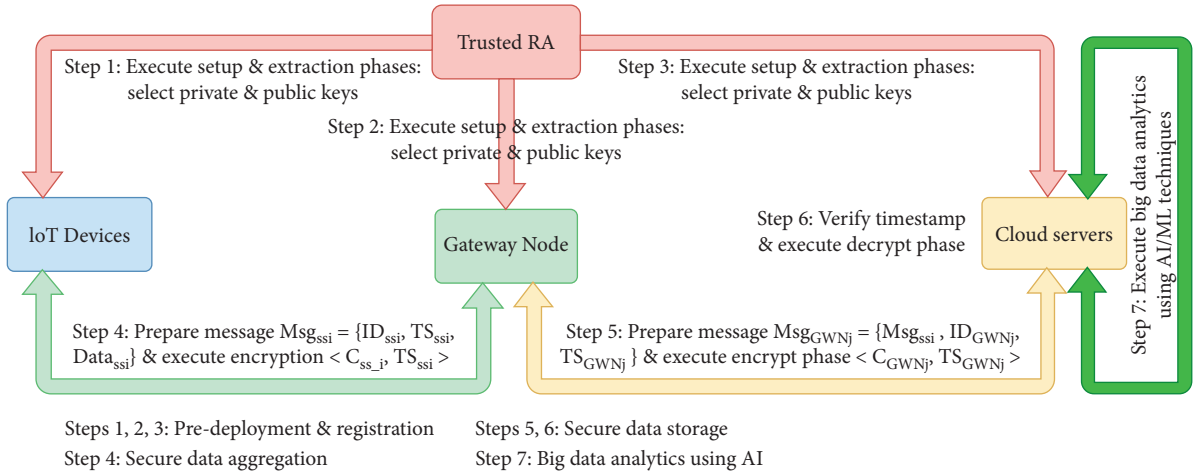


FIGURE 6: Overall mechanism of the proposed IBE scheme for IoT-based AI applications.

- Step 1. Once the message $\{C_{GWN_j}, TS_{GWN_j}\}$, is received at time $TS^*_{GWN_j}$, for checking replaying attacks, CLS_k, CLS_k checks the validity of the received timestamp by the condition: $|TS_{GWN_j} - TS^*_{GWN_j}| < \Delta T$. If the condition fails, the process is immediately terminated.
- If the timestamp validation is satisfied, the encrypted data C_{GWN_j} is then stored in the database of CLS_k, CLS_k.

5.2.5. Big Data Analytics using AI. It is worth noticing that a cloud server CLS_k, CLS_k receives the encrypted data generated by the IoT smart sensors residing in various applications via the aggregator nodes (gateway nodes). CLS_k, CLS_k can then decrypt the stored data bit wise using its own private key (e_{CLS_k}, r_{CLS_k}) and performs the Big Data analytics steps using AI/ML techniques, such as "data acquisition and filtering", "data extraction", "data aggregation and representation", "data analysis" as well as "data visualization". The results of this phase will provide some useful conclusions and predictions on the stored data.

The overall mechanism of the proposed IBE scheme for IoT-based AI applications is also illustrated in Figure 6. The pre-deployment and registration phases are performed through the steps 1, 2 and 3. Step 4 explains about the data aggregation phase. While the steps 5 and 6 are about secure data storage, Step 7 explains the Big data analytics using the AI techniques.

- Step 2. After receiving the message from SS_i, GWN_j, GWN_j first checks the validity of the received timestamp by the condition: $|TS_{SS_i} - TS^*_{SS_i}| < \Delta T$, where $TS^*_{SS_i}$ and ΔT represent the time when the message was received and the maximum transmission delay, respectively. If the condition is satisfied, GWN_j, GWN_j proceeds to decrypt C_{SS_i} bit wise using its private (secret) key (e_{GWN_j}, r_{GWN_j}) with the help of the Decrypts phase described in Section 4.4 to obtain $\{ID_{SS_i}, TS_{SS_i}', Data_{SS_i}\}$. After that if the checking condition: $TS_{SS_i}' = TS_{SS_i}$ is valid, GWN_j, GWN_j considers the data is fresh. Thus, no replay attack has been there during this process with the timestamping mechanism. Of course, for this purpose, it is reasonable to assume that the network entities are synchronized with their clocks [8].
- Step 3. Now, GWN_j, GWN_j generates a current timestamp TS_{GWN_j}, encrypts the prepared message $Msg_{GWN_j} = \{Msg_{SS_i}, ID_{GWN_j}, TS_{GWN_j}\} = \{(ID_{SS_i}, TS_{SS_i}, Data_{SS_i}), ID\ GWN_j, TS_{GWN_j}\}$ bit wise using the public key of its corresponding cloud server CLS_k to obtain the ciphertext $C_{GWN_j} = \{C_0', C_1', C_2'\}$ as done in the Encrypts phase, and sends the encrypted message $\{C_{GWN_j}, TS_{GWN_j}\}$ to its respective CLS_k, CLS_k via a public channel, where C_0', C_1' and C_2' are the encrypted bit strings corresponding to the bit strings of the Msg_{GWN_j}.

6. Security Analysis

In this section, we analyze the security of the proposed encryption scheme by using a sequence of games played between an adversary, say $\mathcal{A}$ and a challenger, say $\mathcal{B}$, namely the games $\mathcal{G}_l$, for $l = 0, 1, 2, 3, 4$. The initial game $\mathcal{G}_0$ is considered as the real attack, whereas the final game $\mathcal{G}_4$ is the game that cannot be cracked by the adversary $\mathcal{A}$. Each transition from a game $\mathcal{G}_i$ to another game $\mathcal{G}_{i+1}$ is indistinguishable with a negligible advantage under some hard assumption. If there are polynomial time games, each of the transitions is also indistinguishable with the negligible advantage meaning that the advantage of $\mathcal{A}$ in real attack is negligible. We now define the games in order to ensure the indistinguishable transitions.

6.1. Games Descriptions. The following games are discussed below.

- **Game** $(\mathcal{G}_0)$**:** This game is played between the adversary $\mathcal{A}$ and the challenger $\mathcal{B}$ with both honest and indistinguishable properties under the **IND-sID-CPA** property. We have defined as earlier that, under "selective identity chosen plaintext attack

IND-sID-CPA" property, $\mathscr{A}$ needs to submit target identity at advance to the $\mathscr{B}$, before $\mathscr{B}$ runs the Set-up algorithm.

- **Game** $(\mathscr{G}_1)$**:** This game is same as $\mathscr{G}_0$ except in the **Set-up** phase, $\mathscr{B}$ computes the matrices $H_{i,b}$, for $1 \le i \le k$ and $b \in \{0,1\}$ not directly, but as an arbitrary public key of random **GPV** trapdoors [21] corresponding to the trapdoor $T_{i,b}$.
- **Game** $(\mathscr{G}_2)$**:** This game is same as $\mathscr{G}_1$, except $\mathscr{B}$ neither uses the master secret T_A nor the **Extraction** phase to answer the queries to private keys, but it uses another **Trapdoor-Extraction** phase and trapdoors $T_{i,b}$ for $1 \le i \le k$ and $b \in \{0,1\}$. The trapdoors are represented as $\widehat{T} = \{\langle i, b, T_{i,b}\rangle : 1 \le i \le k, 0 \le b \le 1\}$.

Trapdoor-Extraction $\langle A, u_0, \widehat{H}, \mathrm{ID}, i^*, T_{i^*,b^*}\rangle$: A key that corresponds to decryption is extracted for the identity ID, with the help of the trapdoor:

1. Let $b_i = \mathrm{bit}_{i(\mathrm{mod}2)}(\mathrm{ID})$ be the position of non zero bit for $1 \le i \le k$ and $b \in \{0, 1\}$. Assemble an $n \times \ell$ matrix $H_{\mathrm{ID}} = [H_{i_1,bi_1 \bmod 2} | H_{i_2,bi_2 \bmod 2} | \cdots, \ | H_{i_\ell,bi_\ell \bmod 2}] \in \mathbb{Z}_q^{n\times\ell}$, where $H_{i_1,bi_1 \bmod 2} | H_{i_2,bi_2 \bmod 2} | \cdots, \ | H_{i_\ell,bi_\ell \bmod 2} \in \widehat{H}$ because $H_{i_1,b1}$ or $H_{i_1,b0}$ is according to either $i_1 \pmod 2 = 0$ or $i_1 \pmod 2 = 1$.
2. Sample $r_i \leftarrow \mathbb{Z}_q^\ell$ under **Sample-Dom** $(H_{i,bi \bmod 2}, \sigma)$, where $i \in \{1, 2, \ldots, (k\} - \{i^*\}$, that is, from the set $\mathbb{Z}_q^\ell$.
3. Let $\widehat{u} = u_0 + H_{\mathrm{ID}}\widehat{r} \in \mathbb{Z}_q^n$. It can be then observed as $\widehat{u} = u_0 + \sum_{i \in \{1,2,\ldots,k\}-\{i^*\}}^{\ell'} H_{i,b_{i(\mathrm{mod}2)}} r_i$, where i is the non zero position in the string $S = H(\mathrm{ID})$, and $\widehat{r}$ is the concatenation of all r_i s, except 0, which follows the distribution r_{i^*}.
4. Using the distribution $D_{\mathbb{Z}^m,\sigma}$, sample $e \leftarrow \mathbb{Z}_q^m$ under the **Sample-Dom** (A, σ) algorithm.
5. Compute $u = Ae \in \mathbb{Z}_q^n$ and $\overrightarrow{u} = u - \widehat{u}$, and then use the **Sample-Pre(**T_{i^*,b^*}**,**σ**,**$\widehat{u}$**)** to sample $r_{i^*} \leftarrow \mathbb{Z}_q^\ell$ such that $\widehat{u} = H_{i^*,b^*} r_{i^*}$.
6. Let $r^t = [r_1^t | r_2^t | \cdots | r_\ell^t]$ including r_{i^*} such that $u = u_0 + H_{\mathrm{ID}} r$. Output a private key $K = \langle e, r\rangle$.

- **Game** $(\mathscr{G}_3)$**:** This game is same as $\mathscr{G}_2$, except $\mathscr{B}$ computes $\widetilde{H}$ with the trapdoors $\widetilde{T}$. It knows only the trapdoor of i^{th} index, but not corresponding to i^{th}, i^{th}-bit of the target ID*.

1. Let $\mathrm{bit}_{i(\mathrm{mod}2)}(\mathrm{ID}^*)$ for $i \in \{1, 2, \ldots, (k\}$, be the modulo of non-zero i^{th}, i^{th} position declared by $\mathscr{A}$ to $\mathscr{B}$ in the Set-up phase.
2. $\mathscr{B}$ generates $\widetilde{H}$ by taking $b \in \{0,1\}$, $i \in \{1, 2, \ldots, (k\}$ such that $b \ne \mathrm{bit}_{i(\mathrm{mod}2)}(\mathrm{ID}^*)$, and executes **GPV** trapdoors [21] as in the $\mathscr{G}_2$ to obtain $H_{i,b}$ corresponding to $T_{i,b}$. Furthermore, it takes $b \in \{0,1\}$, $b \in \{0,1\}$ such that $b = \mathrm{bit}_{i(\mathrm{mod}2)}(\mathrm{ID}^*)$ for a random $i \in \{1, i \in \{1, 2, \ldots, (k\}, (k\}$, and takes $H_{i,b} \in \mathbb{Z}_q^{n\times\ell}$ with $T_{i,b} = \perp$.
3. To extract the private key for $\mathrm{ID} \ne \mathrm{ID}^*$, $\mathscr{B}$ repeats the game $\mathscr{G}_2$, except i^* is picked such that $\mathrm{bit}_{i(\mathrm{mod}2)}(\mathrm{ID}) \ne \mathrm{bit}_{i(\mathrm{mod}2)}(\mathrm{ID}^*)$ and i^* corresponding to a legal query. If $b^* = \mathrm{bit}_{i^*(\mathrm{mod}2)}(\mathrm{ID})$ and $T_{i^*,b^*} \ne \perp$, $\mathscr{B}$ executes **Trapdoor-Extraction** $\langle A, u_0, \widehat{H}, \mathrm{ID}, i^*, T_{i^*,b^*}\rangle$ to generate the private key.

The challenge cipher then is generated by **Encrypts** $(A, H_0, \overrightarrow{H}, \mathrm{ID}^*, b^*)$ for an arbitrary $b^* \in \{0, 1\}$, and outputs $c = \langle c_0, c_1, c_2\rangle$ as the challenge.

- **Game** $(\mathscr{G}_4)$**:** This game is also same as $\mathscr{G}_3$, except $\mathscr{B}$ gives a challenge to $\mathscr{A}$ that is not computed honestly, but it is a random cipher, that is, $c = \langle c_0, c_1, c_2\rangle$ is chosen randomly from $\mathbb{Z}_q^{1+m+\ell}$ distribution.

6.2. Games Transitions. In the following, we now show that each of the transitions between the successive games (**Game** $(\mathscr{G}_0)$, **Game** $(\mathscr{G}_1)$, **Game** $(\mathscr{G}_2)$, **Game** $(\mathscr{G}_3)$, **Game** $(\mathscr{G}_4)$) is indistinguishable as follows.

- **Transition: Game** $(\mathscr{G}_0)$, $(\mathscr{G}_0) \longrightarrow$ **Game** $(\mathscr{G}_1)$, $(\mathscr{G}_1)$: Both games are identical with respect to $\mathscr{A}$, and $\mathscr{B}$ possesses the information regarding trapdoor $T_{i,b}$ corresponding to $H_{i,b}$ which is not known to $\mathscr{A}$.
- **Transition: Game** $(\mathscr{G}_1)$, $(\mathscr{G}_1) \longrightarrow$ **Game** $(\mathscr{G}_2)$, $(\mathscr{G}_2)$: Both games are identical with respect to $\mathscr{A}$, and $\mathscr{B}$ possesses a different algorithm for key extraction and it is invisible to $\mathscr{A}$.
- **Transition: Game** $(\mathscr{G}_2)$, $(\mathscr{G}_2) \longrightarrow$ **Game** $(\mathscr{G}_3)$, $(\mathscr{G}_3)$: Both games are identical with respect to $\mathscr{A}$, and $\mathscr{B}$ knows only half of all the hash-trapdoors and answers if the extraction queries are known, and these are invisible to $\mathscr{A}$.
- **Transition: Game** $(\mathscr{G}_3)$, $(\mathscr{G}_3) \longrightarrow$ **Game** $(\mathscr{G}_4)$, $(\mathscr{G}_4)$: The views are not identical with respect to $\mathscr{A}$, but are indistinguishable under "learning with errors" assumption.

1. In the beginning, $\mathscr{B}$ receives $1 + m + \ell$ samples of "learning with errors" assumption $\langle a_j, b_j\rangle \in \mathbb{Z}_q^{n+1}$, for $1 \le j \le 1 + m + \ell$, with random $a_j \in \mathbb{Z}_q^n$, and either b_j for $1 \le j \le 1 + m + \ell$ are random or $b_j = a_j^t s + x_j$ for $1 \le j \le 1 + m + \ell$ with a random $s \in \mathbb{Z}_q^n$ and Gaussian $x_j \leftarrow \xi$.
2. In the beginning, $\mathscr{B}$ also receives ID* from $\mathscr{A}$ to be challenged. By applying the Set-up phase, $\mathscr{B}$ computes $\overrightarrow{H}$. $\mathscr{B}$ picks $b \in \{0,1\}$ such that $b \ne \mathrm{bit}_{i(\mathrm{mod}2)}(\mathrm{ID}^*)$ for $1 \le i \le k$, and executes **GPV** trapdoors [21] as in $\mathscr{G}_2$ to obtain random $H_{i,b}$ and its trapdoor $T_{i,b}$ as in another $\mathscr{G}_3$ and $\mathscr{G}_4$, respectively. Now, $\mathscr{B}$ picks $b \in \{0,1\}$ such that $b = \mathrm{bit}_{i(\mathrm{mod}2)}(\mathrm{ID}^*)$ for $1 \le i \le k$, random $H_{i,b}$ and its j^{th}-column "learning with errors" instance a_j, and then sets $T_{i,b} = \perp$.
3. $\mathscr{B}$ answers the private key queries as in the games $\mathscr{G}_3$ and $\mathscr{G}_4$ using the corresponding trapdoors. $\mathscr{B}$ picks random $b \in \{0, 1\}$ and computes a challenge cipher $c_0^* = b_1 + b\lfloor q/2\rfloor$, $c_1^* = \langle b_{1+i} : 1 \le i \le m\rangle \in \mathbb{Z}_q^m$, $c_2^* = \langle b_{1+m+i} : 1 \le i \le \ell\rangle \in \mathbb{Z}_q^\ell$.
4. Finally, $\mathscr{A}$ guesses a bit $b^* \in \{0, 1\}$, and $\mathscr{B}$ returns the correct $b^* = b$; else, returns a random bit as an answer to the "learning with errors" instances.

Table 1: A comparative study on recommended bit-size: Lattice *versus* classical discrete logarithm.

Protocol	Primitive	Recommended bit-size
DLP storage	$g \in \mathbb{Z}_p^*$	$k' = \log\ p$
Lattice storage	$A \in \mathbb{Z}_q^{m\times n}$, $\widehat{H}, u_0 \in \mathbb{Z}_q^n$	$16\kappa^2\log^3\ \kappa$
DLP communication	$g^r, Pg^{rx} \in \mathbb{Z}_p^*$	$k' = \log\ p$
Lattice communication	$u_0^t s + x + b\lfloor q/2\rfloor \in \mathbb{Z}_q$, $A^t s + y \in \mathbb{Z}_q^m$, $H_{\mathrm{ID}}^t s + z \in \mathbb{Z}_q^\ell$	$8\kappa\ \log^2\ \kappa$

It is thus worth noticing that $\mathscr{B}$ is indistinguishable in both the games $\mathscr{G}_3$ and $\mathscr{G}_4$ with respect to view of $\mathscr{A}$, excluding the challenge cipher. The "learning with errors" instance is random for the challenge cipher and components of c^* has same distribution as in the game $\mathscr{G}_3$, and so they will be the components in $\mathscr{G}_4$.

6.3. Anonymous Cipher and Indistinguishablity. In this section, we discuss the notion of semantic security that is discussed in Section 3. It is observed that the proposed identity-based encryption scheme provides indistinguishable property of the ciphers from random strings of equal lengths, although an adversary can presume the identity of the receiver. The challenge cipher is then pseudo-random under the "learning with errors" assumption, which implies indistinguishability.

7. Performance Comparison

This section provides computation costs and recommended bit-size of the proposed identity-based encryption scheme and compares them with the other relevant approaches, such as discrete logarithm-based schemes, RSA public key cryptosystem [31] and ElGamal cryptosystem [32].

7.1. Comparison on Recommended Bit-size. Let κ be an appropriate security parameter and $O(\kappa^2)$ be the size of public key. We can then relate the computation time in terms of security parameter complexity $O(\kappa^2)$, $O(\kappa^2)$. It can be compared with the size of classic public key cryptosystems (RSA and ElGamal) which is $O(\kappa)$ and computation time in terms of security parameter as $O(\kappa^3)$ [33, 36].

We take $q = O(\kappa^2)$, $m = O(\kappa\log(q))$ and $n = O(\kappa\log(q))$ as the parameters, where κ is the security parameter. Furthermore, we consider $q = \kappa^2$, $m = \kappa\log(q)$ and $n = \kappa\log(q)$ as the parameters to simplify the computation. The storage cost is $mn\log(q) = 16\kappa^2\log^3(q)$ and the communication cost is $8\kappa\log^2(\kappa)$ in the proposed scheme. The cipher is computed as $c_0 = u_0^t s + x + b\lfloor q/2\rfloor \in \in \mathbb{Z}_q$, $c_1 = A^t s + y \in \mathbb{Z}_q^m$ and $c_3 = H_{\mathrm{ID}}^t s + z \in \mathbb{Z}_q^\ell$, that is, in the form of triplet $c = \langle c_0, c_1, c_2\rangle$. The size of public keys involves the security parameters $A \in \mathbb{Z}_q^{m\times n}$, $\widehat{H}, u_0 \in \mathbb{Z}_q^n$, which is roughly $16\kappa^2\log^3(\kappa)$, that is, $O(m^2\log(q)) = O(\kappa^2\log(\kappa))$. In Table 1, a comparative study on recommended bit-size with respect to Lattice and classical discrete logarithm due to the "discrete logarithm problem (DLP)" intractability.

Table 2: Key length and key generation time comparative study: RSA *versus* Lattice based cryptosystem.

Approach	Key-length (in bits)	Key generation time (in milliseconds)
	512	360
RSA	1024	1280
	2048	4195
	1169	4
Lattice-based	1841	7.5
	4024	17.5

Table 3: Encryption and decryption costs comparative study: RSA *versus* Lattice based cryptosystem.

Approach	Key-length (in bits)	Message encryption (blocks per second)	Message decryption (blocks per second)
	512	2440	120
RSA	1024	930	20
	2048	310	3
	1169	5940	2820
Lattice-based	1841	3680	1620
	4024	1470	610

7.2. Comparison on Computation Costs. In Table 2, the relationship between the length of keys in bits and the key generation time in milliseconds has been shown. Based on the results reported in [40], in RSA-based public cryptosystem, the key lengths of 512, 1024 and 2048 bits take 360, 1280 and 4195 milliseconds, respectively. On the other hand, in the proposed lattice-based scheme, the key lengths of 1170, 1841 and 4024 bits require the generation time having 4, 7.5 and 17.5 milliseconds, respectively [40]. This clearly shows that the lattice-based IBE scheme requires less computational time for key generation part as compared to other public key cryptosystems, such as RSA.

Table 3 shows a comparative analysis on the key length in bits with the encryption and decryption speed in terms of blocks per second based on the results reported in [40]. It is noticed that when the key size is smaller, the encryption and decryption processing time for the blocks per second are less. However, the lattice-based cryptosystem performs better than RSA-based public key cryptosystem even if the key size is large.

7.3. Comparison on Security. A comparative study on the key length size and the security aspect between the RSA-based public key cryptosystem and lattice-based cryptosystem has been presented in Table 4 based on the results reported in [40]. Million instructions per second (MIPS) is taken as an "approximate measure of a computer's raw processing power", which is considered in the comparative study. It is observed that in both the cases when the key size is large, the

Table 4: Key length and security comparative study: RSA *versus* Lattice based cryptosystem.

Approach	Key-length (in bits)	Security (MIPS per year)
RSA	512	4×10^5
	1024	3×10^{12}
	2048	3×10^{21}
	1169	2×10^6
Lattice-based	1841	4.6×10^{14}
	4024	3.4×10^{35}

security of the system increases. Moreover, even for a smaller key size the lattice based cryptosystem provides significantly better security as compared to that for an RSA-based cryptosystem.

In summary, the lattice-based cryptosystem has several advantages, such as: (a) "cryptographic resistance compared to RSA", (b) "faster key generation", and (c) "faster encryption and decryption of the messages". In addition, the prime advantage of the lattice-based cryptosystem is its resistance to quantum computer attacks.

8. Concluding Remarks

In this work, we attempted to design an advanced identity-based encryption that is a very important cryptographic tool to ensure confidentiality in the current quantum era. The proposed encryption is a provably post-quantum secure without random oracles. Since lattices depends on algebraic operations that are typically matrix addition and multiplication, they make the encryption much efficient as compared to other public key cryptosystems, such as RSA. In addition, the proposed scheme is also anonymous and it produces the pseudo-random ciphers. Finally, we incorporated the constructed identity based encryption (IBE) scheme for IoT applications and described how the Big data analytics using the AI/ML techniques will be helpful in such applications.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

The authors would like to thank the anonymous reviewers and the Associate Editor for their valuable comments and suggestions which helped us to improve the presentation and quality of the paper.

References

[1] B. Eshghi, "IoT Market Outlook for 2022 & beyond," 2022, https://research.aimultiple.com/iot-future/.

[2] TE CONNECTIVITY, "Smart Factory Sensors and Industrial Internet of Things," 2022, https://www.te.com/usa-en/industries/sensor-solutions/applications/iot-sensors/industry-4-0.html.

[3] Hp Internet of Things Security Study, "Smartwatches," 2017, https://www.ftc.gov/system/files/documents/public_comments/2015/10/00050-98093.pdf.

[4] A. K. Das, M. Wazid, A. R. Yannam, J. J. P. C. Rodrigues, Y. Park, and Y. Park, "Provably secure ECC-based device access control and key agreement protocol for IoT environment," *IEEE Access*, vol. 7, no. 1, pp. 55382–55397, 2019.

[5] ISTR, "Internet Security Threat Report (Istr)," 2018, https://docs.broadcom.com/doc/istr-23-2018-en.

[6] M. Rana, Q. Mamun, and R. Islam, "Lightweight cryptography in IoT networks: a survey," *Future Generation Computer Systems*, vol. 129, pp. 77–89, 2022.

[7] G. S. Aujla, R. Chaudhary, N. Kumar, A. K. Das, J. J. P. C. Rodrigues, and P. C. Rodrigues, "SecSVA: secure storage, verification, and auditing of big data in the cloud environment," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 78–85, 2018.

[8] J. Srinivas, A. K. Das, M. Wazid, and A. V. Vasilakos, "Designing secure user authentication protocol for big data collection in IoT-based intelligent transportation system," *IEEE Internet of Things Journal*, vol. 8, no. 9, pp. 7727–7744, 2021.

[9] T. Ahamed Ahanger, A. Aljumah, and M. Atiquzzaman, "State-of-the-art survey of artificial intelligent techniques for IoT security," *Computer Networks*, vol. 206, Article ID 108771, 2022.

[10] C. Iwendi, S. U. Rehman, A. R. Javed, S. Khan, and G. Srivastava, "Sustainable security for the internet of things using artificial intelligence architectures," *ACM Transactions on Internet Technology*, vol. 21, no. 3, 2021.

[11] A. E. Omolara, A. Alabdulatif, O. I. Abiodun et al., "The internet of things security: a survey encompassing unexplored areas and new insights," *Computers & Security*, vol. 112, Article ID 102494, 2022.

[12] S. C. Mukhopadhyay, S. K. S. Tyagi, N. K. Suryadevara, V. Piuri, F. Scotti, and S. Zeadally, "Artificial intelligence-based sensors for Next generation IoT applications: a review," *IEEE Sensors Journal*, vol. 21, no. 22, pp. 24920–24932, 2021.

[13] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Workshop on the Theory and Application of Cryptographic Techniques*, pp. 47–53, Springer, Berlin, Germany, 1984.

[14] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Review*, vol. 41, no. 2, pp. 303–332, 1999.

[15] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Proceedings of the Annual international cryptology conference*, pp. 213–229, Springer, Santa Barbara, CA, USA, August 2001.

[16] D. Boneh and X. Boyen, "Efficient selective-id secure identity-based encryption without random oracles," in *Proceedings of the International conference on the theory and applications of cryptographic techniques*, pp. 223–238, Springer, Interlaken, Switzerland, May 2004.

[17] G. Craig, "Practical identity-based encryption without random oracles," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 445–464, Springer, Petersburg Russia, June 2006.

[18] B. Waters, "Efficient identity-based encryption without random oracles," *Lecture Notes in Computer Science*, Springer, in *Proceedings of the Annual International Conference on the*

Theory and Applications of Cryptographic Techniques, pp. 114–127, May 2005.

[19] D. Boneh, G. Craig, and M. Hamburg, "Space-efficient identity based encryptionwithout pairings," in *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pp. 647–657, IEEE, Providence, RI, USA, October 2007.

[20] C. Cocks, "An identity based encryption scheme based on quadratic residues," in *Proceedings of the IMA international conference on cryptography and coding*, pp. 360–363, Springer, Cirencester, UK, December 2001.

[21] G. Craig, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pp. 197–206, Victoria British Columbia Canada, May 2008.

[22] V. Chamola, A. Jolfaei, V. Chanana, P. Parashari, and V. Hassija, "Information security in the post quantum era for 5G and beyond networks: threats to existing cryptography, and post-quantum cryptography," *Computer Communications*, vol. 176, pp. 99–118, 2021.

[23] V. Hassija, V. Chamola, A. Goyal, S. S. Kanhere, and N. Guizani, "Forthcoming applications of quantum computing: peeking into the future," *IET Quantum Communication*, vol. 1, no. 2, pp. 35–41, 2020.

[24] V. Hassija, V. Chamola, V. Saxena et al., "Present landscape of quantum computing," *IET Quantum Communication*, vol. 1, no. 2, pp. 42–48, 2020.

[25] Q. Li, D. He, Z. Yang, Q. Xie, and K.-K. R. Choo, "Lattice-based conditional privacy-preserving authentication protocol for the vehicular Ad Hoc network," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 4336–4347, 2022.

[26] Z. Xu, D. He, P. Vijayakumar, K.-K. R. Choo, and L. Li, "Efficient NTRU lattice-based certificateless signature scheme for medical cyber-physical systems," *Journal of Medical Systems*, vol. 44, no. 5, p. 92, 2020.

[27] F. Qi, D. He, S. Zeadally, N. Kumar, and K. Liang, "Ideal lattice-based anonymous authentication protocol for mobile devices," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2775–2785, 2019.

[28] X. Boyen and B. Waters, "Anonymous hierarchical identity-based encryption (without random oracles)," in *Proceedings of the Annual International Cryptology Conference*, pp. 290–307, Springer, Santa Barbara, CA, USA, August 2006.

[29] C. Ran, S. Halevi, and J. Katz, "A forward-secure public-key encryption scheme," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 255–271, Springer, Warsaw Poland, May 2003.

[30] D. Boneh and X. Boyen, "Secure identity based encryption without random oracles," in *Proceedings of the Annual International Cryptology Conference*, pp. 443–459, Springer, Santa Barbara, CA, USA, August 2004.

[31] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[32] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[33] O. Goldreich, S. Goldwasser, and S. Halevi, "Public-key cryptosystems from lattice reduction problems," in *Proceedings of the Annual International Cryptology Conference*, pp. 112–131, Springer, Santa Barbara, CA, USA, August 1997.

[34] C. Ran, S. Halevi, and J. Katz, "A forward-secure public-key encryption scheme," *Journal of Cryptology*, vol. 20, no. 3, pp. 265–294, 2007.

[35] D. Micciancio and O. Regev, "Worst-case to average-case reductions based on Gaussian measures," *SIAM Journal on Computing*, vol. 37, no. 1, pp. 267–302, 2007.

[36] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM*, vol. 56, no. 6, p. 34, 2009.

[37] O. Regev, "Lattice-based cryptography," in *Proceedings of the 26th Annual International Conference on Advances in Cryptology (CRYPTO'06)*, pp. 131–141, Santa Barbara, CA, USA, August 2006.

[38] M. Ajtai, "Generating hard instances of the short basis problem," in *International Colloquium on Automata, Languages, and Programming (ICALP'99), Lecture Notes in Computer Science*vol. 1644, pp. 1–9, Springer, 1999.

[39] D. Apon, X. Fan, and F.-H. Liu, "Compact Identity Based Encryption from LWE," 2016, https://ia.cr/2016/125.

[40] A. Gagnidze, M. Iavich, and I. Giorgi, "Analysis of post quantum cryptography use in practice," *Bull. Georgian Natl. Acad. Sci*, vol. 11, no. 2, pp. 29–36, 2017.