Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A privacy-preserving and non-interactive federated learning scheme for regression training with gradient descent

Fengwei Wang [a,b], Hui Zhu [a,c,*], Rongxing Lu [b], Yandong Zheng [b], Hui Li [a]

[a] *National Key Laboratory of Integrated Networks Services, Xidian University, Xi'an, China*
[b] *Faculty of Computer Science, University of New Brunswick, New Brunswick, Canada*
[c] *Peng Cheng Laboratory, Shenzhen, China*

ARTICLE INFO

Article history:
Received 22 August 2020
Received in revised form 5 October 2020
Accepted 7 December 2020
Available online 16 December 2020

Keywords:
Regression training
Privacy-preserving
Secure data aggregation
Gradient descent

ABSTRACT

In recent years, the extensive application of machine learning technologies has been witnessed in various fields. However, in many applications, massive data are distributively stored in multiple data owners. Meanwhile, due to the privacy concerns and communication constraints, it is difficult to bridge the data silos among data owners for training a global machine learning model. In this paper, we propose a privacy-preserving and non-interactive federated learning scheme for regression training with gradient descent, named VANE. With VANE, multiple data owners are able to train a global linear, ridge or logistic regression model with the assistance of cloud, while their private local training data can be well protected. Specifically, we first design a secure data aggregation algorithm, with which local training data from multiple data owners can be aggregated and trained to a global model without disclosing any private information. Meanwhile, benefit from our data pre-processing method, the whole training process is non-interactive, i.e., there is no interaction between data owners and the cloud. Detailed security analysis shows that VANE can well protect the local training data of data owners. The performance evaluation results demonstrate that the training performance of our VANE is around 10^3 times faster than existing schemes.

1. Introduction

With the rapid growth of data volume in information society, the use of machine learning technology is rapidly expanding in various fields, and bringing great conveniences to people's life [21,7,28]. As one of the fundamental machine learning algorithms, regression model has attracted considerable attention since it is an essential tool in decision making systems such as policy making, health care, law enforcement, and finance [33,36,24]. For example, in e-healthcare, regression model is already being employed by care management organizations for constructing medical pre-diagnosis systems, which can save an immense amount of diagnosis time for patients [19,15]. In general, training a regression model often requires staggering quantities of data, and in many applications, data are stored distributively in different data owners. As shown in Fig. 1, in such a distributed scenario, a sole data owner collecting only a small dataset cannot train a regression model with high quality (i.e, the trained local regression model is not of high accuracy). Therefore, data owners expect to collaboratively train a

* Corresponding author at: National Key Laboratory of Integrated Networks Services, Xidian University, Xi'an, China.
E-mail address: zhuhui@xidian.edu.cn (H. Zhu).

https://doi.org/10.1016/j.ins.2020.12.007
0020-0255/© 2020 Elsevier Inc. All rights reserved.

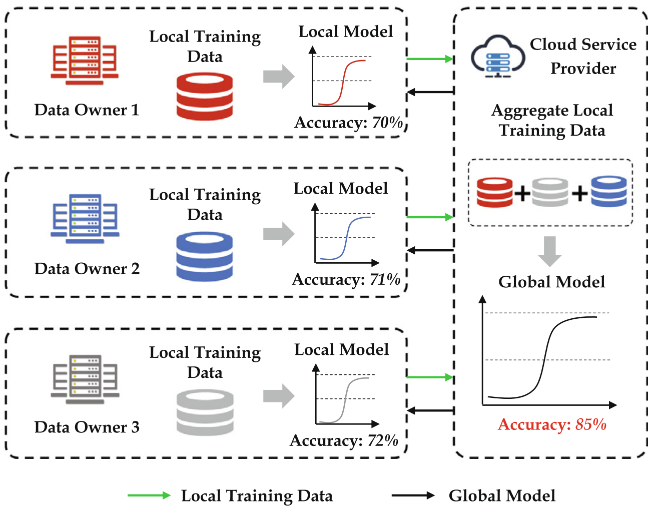


Fig. 1. Collaborative regression training.

regression model with the assistance of the cloud service provider, so that each of the parties can enjoy a global model with higher accuracy than what it can achieve alone.

Unfortunately, due to the privacy concerns and communication constraints, there are still many challenges lying ahead of collaborative regression training [9,4,2,25]. On the one hand, data owners' local training data commonly contain massive private information of individuals or organizations. Specifically, local training data can be divided into two categories. The first one is the raw dataset collected from individuals, which contains massive personal private information (i.e., health condition, income situation, etc.). Once these data are compromised, it may lead to computer-assisted crime by adversaries. The second category of local training data is the parameter of the local trained model, which generally contains massive statistical data of an organization. Leakage of these data may disclose the corporate secrets (i.e., operating conditions, management information, etc.) of an organization, which may further result in an economic loss. On the other hand, the iterative operation of regression training brings huge communication overhead among data owners and the cloud service provider, which is hard to handle in practice. Thus, how to achieve efficient collaborative regression training while protecting multiple data owners' sensitive data has attracted considerable interest recently.

To address the above-mentioned challenges, plenty of federated learning schemes for regression training have been proposed [8,18,5], which mainly rely on homomorphic encryption techniques and gradient descent algorithm. In detail, based on the homomorphic properties, existing schemes are able to execute gradient descent algorithm over ciphertexts for training a regression model, which can protect local training data of data owners. However, since gradient descent is an iterative algorithm, multiple interactions and repeated time-consuming homomorphic operations are inevitable, which brings massive extra computation and communication overhead. Moreover, for securely training a global regression model, most existing schemes apply two-cloud architecture to aggregate local datasets of data owners, which will spend more resources for employing two clouds in reality.

In this paper, we propose a privacy-preserving and non-interactive federated learning scheme for regression training, named VANE. With VANE, multiple data owners are able to train a global regression model with the assistance of the cloud service provider, while their local training data can be well protected. Moreover, during the whole training process, interactions between data owners and the cloud service provider are not required. Specifically, the main contributions of this paper are threefold.

- *First*, VANE achieves non-interactive regression training and model updating. In VANE, local datasets of data owners are first pre-processed into local training data, with which the global regression model can be trained without interactions between data owners and the cloud service provider. Moreover, we also design strategies for updating the trained global model regularly.
- *Second*, VANE is privacy-preserving in regression training. Based on Paillier cryptosystem, we propose a secure data aggregation algorithm under the single-cloud architecture, which can securely aggregate local training data of multiple data owners for training a global regression model. Thus, the sensitive information of data owners can be well protected and only one cloud is required in our proposed scheme.
- *Third*, we analyze the security of VANE and conduct experiments to evaluate its performance. The results show that with our modified Paillier cryptosystem, the local training data of data owners can be well protected. In addition, our scheme is indeed efficient in terms of computation cost and communication overhead.

The remainder of this paper is organized as follows. In Section 2, we formalize the models and identify our design goal. In Section 3, we review the Paillier cryptosystem, linear, ridge, and logistic regressions as preliminaries. Then, we propose our VANE in Section 4, followed by the security analysis and performance evaluation in Sections 5 and 6. We also review some related works in Section 7. Finally, we draw our conclusion in Section 8.

2. Models, security requirements and design goal

In this section, we formalize the system model, threat model, and security requirements. Then, we identify our design goal.

2.1. System model

In our system model, we mainly focus on how to train a global regression model without disclosing the sensitive information of data owners. Each data owner is equipped with a workstation, which can store and pre-process the collected dataset, as well as connect the cloud service provider. Specifically, the system consists of three parts: 1) trusted authority (TA); 2) data owners (DOs); and 3) cloud service provider (CSP), as shown in Fig. 2.

- TA is a trusted authority (i.e., a government organization), which initializes the system via generating system parameters, and distributing keys to data owners and the cloud service provider.
- $\text{DOs} = \{\text{DO}_1, \ldots, \text{DO}_m\}$ is a set of m data owners. In our system, each $\text{DO}_i \in \text{DOs}$ has its own local dataset. Specifically, each DO_i first pre-processes its local dataset to generate local training data. Then, the local training data will be encrypted and outsourced to the cloud service provider for generating the global regression model.
- CSP is the cloud service provider, which has abundant storage space and powerful computing capability (e.g., Google, Microsoft, Apple). In our system, CSP is responsible for aggregating the ciphertexts of local training data from multiple data owners, decrypting the aggregated result, and training the global regression model for data owners. Moreover, the cloud service provider has a testing dataset (e.g., the open machine learning dataset [3]), with which the cloud service provider is able to estimate the quality of the global regression model.

2.2. Threat model and security requirements

In our threat model, we consider that DOs and CSP are *honest-but-curious* [10]. In specific, during the process of federated regression training, CSP honestly executes the data aggregation protocol and trains the global regression model credibly, but it is greedy about the local training data of DOs. Moreover, DOs honestly outsource their encrypted local training data without tampering. But for commercial interests, each DO is curious about other DOs' local training data. Besides, the global regression model should be updated regularly with continuously-generated data. Nevertheless, the data acquisition devices of DOs may be out of order, and even some DOs may be under control of adversaries. As a result, the global regression model will be corrupted by contaminated datasets. Therefore, we should also take fuzzy data into consideration during model updating. Note that there may be some other passive or active attacks (e.g. side channel attack and denial of service) during the federated learning. Since our target is to protect the sensitive data of DOs and guarantee the quality of trained global

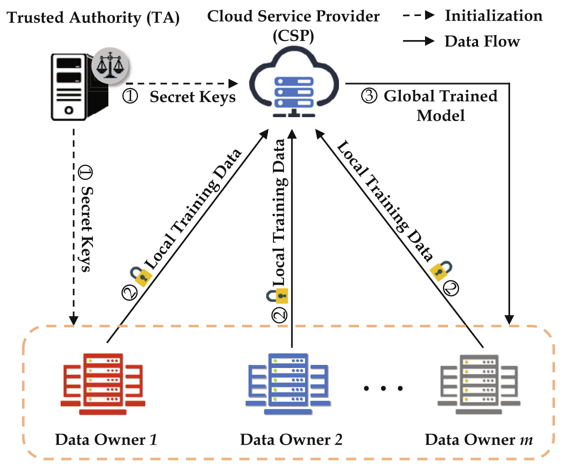


Fig. 2. System model under consideration.

model, these attacks are currently out of scope of this paper and will be considered in our future work. Considering the above security issues, the following security requirements should be satisfied.

- *Ensuring the privacy of DOs' local training data.* In general, the local training data consists of massive statistical data of DOs, which may contain sensitive information (i.e., business operation information) of enterprises. Therefore, during the federated regression training and model updating phases, the local training data of DOs should be protected.
- *Resisting fuzzy data in global regression model updating.* During the model updating process, some fuzzy data may be collected and contaminate the training dataset, which directly affects the quality of the global regression model. Therefore, after updating, the generated global model should be estimated for resisting fuzzy data and guaranteeing its accuracy.

2.3. Design goals

Based on the above-mentioned system model and security requirements, in this paper, our goal is to design a secure efficient federated learning scheme for regression training. Specifically, the following objectives should be achieved.

- *Guarantee security and privacy preservation.* Data privacy and security is always a non-negligible problem lying ahead of machine learning. Therefore, a fundamental goal of VANE is privacy preservation, i.e., during the federated regression training, the security of a DO's local training data should be guaranteed against CSP and other DOs.
- *Low computation and communication overhead.* In order to achieve privacy-preserving regression training, time-consuming computations (e.g., homomorphic operations) are inevitably incurred, which brings extra computation overhead. Moreover, in practice, it is hard for CSP to handle huge amounts of training data. Therefore, the proposed VANE should achieve high-efficiency in terms of computation cost and communication overhead.

3. Preliminaries

In this section, we review Paillier cryptosystem, linear, ridge, and logistic regressions, which serve as the basis of our scheme.

3.1. Paillier cryptosystem

We apply Paillier cryptosystem [22] as a building block of our scheme, which is a public key cryptography widely used in privacy-preserving machine learning techniques. Here, we briefly review the Paillier cryptosystem as follows.

- *Key Generation:* Given the security parameter κ and two big primes $|p| = |q| = \kappa$, compute $N = p \cdot q$ and $\lambda = lcm(p-1, q-1)$. Then, select a random number $g \in \mathbb{Z}^*_{N^2}$ satisfying $gcd(L(g^\lambda \bmod N^2), N) = 1$, where $L(x) = (x-1)/N$. Moreover, compute $\mu = (L(g^\lambda \bmod N^2))^{-1} \bmod N$. Then, the public key pk is (N, g), and the corresponding secret key sk is (μ, λ).
- *Encryption:* Given a message $m \in \mathbb{Z}_N$, the ciphertext can be computed with the public key pk as $c = E_{pk}(m) = g^m \cdot r^N \bmod N^2$, where r is a random number in $\mathbb{Z}^*_N$.
- *Decryption:* Given a ciphertext $c \in \mathbb{Z}^*_{N^2}$, the corresponding plaintext can be retrieved with the secret key sk through computing $m = D_{sk}(c) = L(c^\lambda \bmod N^2) \cdot \mu \bmod N$.

The additive homomorphism of Paillier can be described as: for two arbitrary ciphertexts $c_1 = E_{pk}(m_1)$ and $c_2 = E_{pk}(m_2)$, we have $c_1 \cdot c_2 = E_{pk}(m_1) \cdot E_{pk}(m_2) = g^{m_1+m_2}(r_1 r_2)^N \bmod N^2 = E_{pk}(m_1 + m_2)$.

Moreover, the multiple homomorphism of Paillier can be described as: for a ciphertexts $c_1 = E_{pk}(m_1)$ and a plaintext m_2, we have $c_1^{m_2} = E_{pk}(m_1)^{m_2} = g^{m_1 m_2} r_1^{m_2 N} \bmod N^2 = E_{pk}(m_1 m_2)$.

3.2. Linear regression

Linear regression [30] is one of the most popular machine learning algorithms for modeling the relationship between one or more input variables. In statistics, linear regression is extensively used for data prediction in many applications such as insurance or loan risk estimation, personalized medicine, mark analysis and so on.

Linear regression focuses on predicting the continuous target values of samples. Specifically, given a sample $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$, we have one hypothetical regression output $y = h(\boldsymbol{\theta}, \boldsymbol{x})$, where $\boldsymbol{\theta} = (\theta_0, \theta_1, \ldots, \theta_d)$ is the regression coefficients, and $h(\boldsymbol{\theta}, \boldsymbol{x}) = \theta_0 + \Sigma_{j=1}^{d} \theta_j x_j$ is the inner product of feature values and regression coefficients, respectively.

For training a linear regression model, assume that $\mathcal{D} = \{\boldsymbol{x}^{(k)}, y^{(k)}\}_{k=1}^{n}$ is a training dataset with n samples. Then, the loss function can be defined as follows

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{k=1}^{n}\left(h(\boldsymbol{\theta}, \boldsymbol{x}^{(k)}) - y^{(k)}\right)^2.$$

The goal of linear regression training is to minimize the loss function $\mathcal{L}(\boldsymbol{\theta})$ through adjusting the parameters $\boldsymbol{\theta}$. In general, gradient descent is commonly used to solve the above problem, which is an iterative optimization algorithm for minimizing the loss function. Specifically, with gradient descent, $\boldsymbol{\theta}$ is computed iteratively as

$$\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^{i} - \alpha\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}),$$

where $\boldsymbol{\theta}^0$ is initialized with a random value or all zero, α is the learning rate, and $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$ is the gradient of $\mathcal{L}(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$. Then, we have

$$\begin{aligned}\theta_0^{i+1} &= \theta_0^{i} - \tfrac{\alpha}{n}\sum_{k=1}^{n}(h(\boldsymbol{\theta}, \boldsymbol{x}^{(k)}) - y^{(k)})\\ \theta_j^{i+1} &= \theta_j^{i} - \tfrac{\alpha}{n}\sum_{k=1}^{n}(h(\boldsymbol{\theta}, \boldsymbol{x}^{(k)}) - y^{(k)})x_j^{(k)},\end{aligned}$$

where $j = 1, 2, \ldots, d$. Moreover, the iterative operation terminates when the loss function $\mathcal{L}(\theta)$ converges or the maximum iteration is reached.

3.3. Ridge regression

Compared to linear regression, ridge regression [13] introduces an ℓ_2-norm regularization term in loss function to penalize large regression coefficients, which effectively alleviates the overfitting problem in linear regression. The loss function of ridge regression can be represented as

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{k=1}^{n}\left(h(\boldsymbol{\theta}, \boldsymbol{x}^{(k)}) - y^{(k)}\right)^2 + \lambda\|\boldsymbol{\theta}\|^2,$$

where $\lambda\|\boldsymbol{\theta}\|^2$ is the ℓ_2-norm regularization term, and λ is the regularization parameter.

Then, for training the ridge regression model, the gradient descent algorithm can also be used for computing $\boldsymbol{\theta}$. Specifically, the iterative process for ridge regression is computed as follows.

$$\begin{aligned}\theta_0^{i+1} &= (1 - 2\lambda\alpha)\theta_0^{i} - \tfrac{\alpha}{n}\sum_{k=1}^{n}(h(\boldsymbol{\theta}, \boldsymbol{x}^{(k)}) - y^{(k)})\\ \theta_j^{i+1} &= (1 - 2\lambda\alpha)\theta_j^{i} - \tfrac{\alpha}{n}\sum_{k=1}^{n}(h(\boldsymbol{\theta}, \boldsymbol{x}^{(k)}) - y^{(k)})x_j^{(k)},\end{aligned}$$

where α is the learning rate, $j = 1, 2, \ldots, d$. Similar to linear regression, the iterative operation terminates when the loss function $\mathcal{L}(\boldsymbol{\theta})$ converges or the maximum iteration is reached.

3.4. Logistic regression

In order to resolve classification problem, logistic regression [23] is proposed, which maps the prediction results into a probability value in $[0, 1]$ as follows:

$$y = \frac{1}{1 + e^{-h(\boldsymbol{\theta}, \boldsymbol{x})}}$$

For the logistic function, the loss function can be approximated by second-order Taylor series and ℓ_2-norm regularization [11] as

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{k=1}^{n}\left(\log 2 - \frac{1}{2}y_i \cdot h(\boldsymbol{\theta}, \boldsymbol{x}) + \frac{1}{8}h(\boldsymbol{\theta}, \boldsymbol{x})^2\right) + \lambda\|\boldsymbol{\theta}\|^2.$$

Then, the gradient descent for training a logistic regression model can be executed as follows.

$$\begin{aligned}\theta_0^{i+1} &= (1 - 2\lambda\alpha)\theta_0^{i} - \tfrac{\alpha}{n}\sum_{k=1}^{n}(\tfrac{1}{4}h(\boldsymbol{\theta}, \boldsymbol{x}^{(k)}) - \tfrac{1}{2}y^{(k)})\\ \theta_j^{i+1} &= (1 - 2\lambda\alpha)\theta_j^{i} - \tfrac{\alpha}{n}\sum_{k=1}^{n}(\tfrac{1}{4}h(\boldsymbol{\theta}, \boldsymbol{x}^{(k)}) - \tfrac{1}{2}y^{(k)})x_j^{(k)},\end{aligned}$$

where α is the learning rate, $j = 1, 2, \ldots, d$, and the iterative operation terminates when the loss function $\mathcal{L}(\boldsymbol{\theta})$ converges or the maximum iteration is reached.

4. Proposed privacy-preserving scheme

In this section, we propose our VANE scheme. Specifically, we first introduce the main idea of VANE. Then, the secure data aggregation algorithm and a detailed description of VANE are presented. Moreover, we extract our secure data aggregation and training (SDAT) algorithm from VANE, and carefully prove its correctness.

4.1. Main idea of the proposed scheme

In traditional federated regression training schemes [8,18], for each iteration of gradient descent algorithm, CSP and DOs need to collaboratively compute local gradients for updating global model parameters. Therefore, multiple interactions between DOs and CSP are necessary to train the global regression model.

In VANE, we observe that the operations $\sum_{k=1}^{n}(h(\boldsymbol{\theta}, \boldsymbol{x}^{(k)}) - y^{(k)})$ and $\sum_{k=1}^{n}(h(\boldsymbol{\theta}, \boldsymbol{x}^{(k)}) - y^{(k)})x_j^{(k)}$ in gradient descent can be converted to $h(\boldsymbol{\theta}, \sum_{k=1}^{n} \boldsymbol{x}^{(k)}) - \sum_{k=1}^{n} y^{(k)}$ and $h(\boldsymbol{\theta}, \sum_{k=1}^{n} \boldsymbol{x}^{(k)} \cdot x_j^{(k)}) - \sum_{k=1}^{n} y^{(k)} \cdot x_j^{(k)}$, respectively. Base on this conversion, in VANE, each DO can locally pre-process its local dataset into a matrix as local training data through computing $\sum_{k=1}^{n} \boldsymbol{x}^{(k)}, \sum_{k=1}^{n} y^{(k)}, \sum_{k=1}^{n} \boldsymbol{x}^{(k)} \cdot x_j^{(k)}$, and $\sum_{k=1}^{n} y^{(k)} \cdot x_j^{(k)}$. Moreover, via aggregating matrixes from multiple data owners, CSP is able to generate global training data for executing gradient descent algorithm while local gradients are not required. Therefore, the global regression model can be trained non-interactively, which significantly reduces the computation cost and communication overhead in federated regression training.

In addition, to protect local training data of DOs, we propose a secure data aggregation algorithm under single-cloud architecture, which is introduced in the following.

4.2. Secure data aggregation algorithm

In this section, we modify Paillier cryptosystem and propose our secure data aggregation algorithm, which consists of four functions as follows.

Key Generation: $KeyGenerate(\kappa) \to (PP, SK_{DO_i}, SK_{CSP})$. Given a security parameter κ, TA first executes $Gen(\kappa)$ to generate parameters of Paillier cryptosystem, which includes the secret key $SK_p = (\mu, \lambda)$ and the public key $PK_p = (N, g)$. Note that in Paillier cryptosystem, the order of g is a multiple of N, i.e., $ord(g) = kN$, where k is an integer. Then, TA selects a large random integer γ satisfying $|\gamma| < \kappa/2$ and $gcd(k, \gamma) = 1$, and computes $h = g^{\gamma} \bmod N^2$. Finally, TA publishes the public parameters $PP = < \kappa, N, g, h >$.

For generating secret keys for DOs, TA first splits N into m random numbers $\{n_1, n_2, \ldots, n_m\}$, s.t., $\sum_{i=1}^{m} n_i = N$, where m is the number of DOs in our system. Then, TA chooses a random number $R_t \in \mathbb{Z}_N^*$ as the task ID for every data aggregation task, and computes $SK_{DO_i} = R_t^{n_i} \bmod N^2$ as the secret key for each DO$_i$.

Finally, TA generates the secret key $SK_{CSP} = < \lambda, \mu, \gamma >$ for CSP.

Data Encryption: $Encrypt(x^{(i)}, SK_{DO_i}) \to [[x^{(i)}]]$. Given a private message $x^{(i)} \in \mathbb{Z}_N$ of DO$_i$, it can be encrypted by DO$_i$ through computing

$$[[x^{(i)}]] = g^{x^{(i)}} \cdot h^{r_i} \cdot SK_{DO_i} \bmod N^2, \tag{1}$$

where r_i is a random number which satisfies $|r_i| < \frac{\kappa}{2}$.

Data Aggregation: $Aggregate([[x^{(1)}]], \ldots, [[x^{(m)}]]) \to [[x]]$. Given m ciphertexts from m DOs, CSP can aggregate these ciphertexts through computing

$$[[x]] = \prod_{i=1}^{m} [[x^{(i)}]] \bmod N^2. \tag{2}$$

Aggregated Result Decryption: $Decrypt([[x]], SK_{CSP}) \to x$. After aggregating the ciphertexts from m DOs, CSP is able to decrypt the aggregated result with SK_{CSP} through computing

$$x = \sum_{i=1}^{m} x^{(i)} = (L([[x]]^{\lambda} \bmod N^2 \cdot \mu) \bmod N) \bmod \gamma, \tag{3}$$

where $L(x) = \frac{x-1}{N}$.

The correctness of our secure data aggregation algorithm is proved in Section 4.4.

4.3. Description of our proposed scheme

In this section, we detailedly introduce our proposed scheme, which mainly consists of four phases: 1) *System initialization;* 2) *Local training data generation and encryption;* 3) *Secure data aggregation and training*; and 4) *Model updating and estimation*. The overview is sketched in Fig. 3. At first, TA bootstraps the system through distributing keys and system parameters to DOs and CSP. Then, each DO$_i$ pre-processes its local dataset and computes the encrypted local training data, which is further aggregated by CSP to generate global training data. Finally, CSP trains the global regression model and estimates its accuracy. To describe VANE more clearly, we give the description of used notations in Table 1.

4.3.1. System Initialization

In the system initialization phase, TA first generates system parameters, and distributes keys to DOs and CSP, respectively. Moreover, data normalization is executed by DOs.

• Step 1. System Parameters and Keys Distribution

TA first executes *KeyGenerate*(κ) in our secure data aggregation algorithm to generate public parameters and secret keys for DOs and CSP.

• Step 2. Data Normalization

To improve the efficiency of federated regression training, data normalization on local datasets should be executed at first. Specifically, assume that the local dataset of DO$_i$ is represented as

$$\mathcal{D}^{(i)} = \{\boldsymbol{x}^{(ik)}, y^{(ik)}\}_{k=1}^{n^{(i)}},$$

where $\boldsymbol{x}^{(ik)} = (x_1^{(ik)}, \ldots, x_d^{(ik)})$ is the feature vector, $y^{(ik)}$ is the target variable, and $n^{(i)}$ is the number of samples collected by DO$_i$.

Then, for each feature, DO$_i$ computes maximum and minimum values in $\mathcal{D}^{(i)}$ to generate vectors

$$\boldsymbol{x}_{max}^{(i)} = (max\{x_1^{(ik)}\}_{k=1}^{n^{(i)}}, \ldots, max\{x_d^{(ik)}\}_{k=1}^{n^{(i)}})$$
$$\boldsymbol{x}_{min}^{(i)} = (min\{x_1^{(ik)}\}_{k=1}^{n^{(i)}}, \ldots, min\{x_d^{(ik)}\}_{k=1}^{n^{(i)}}).$$

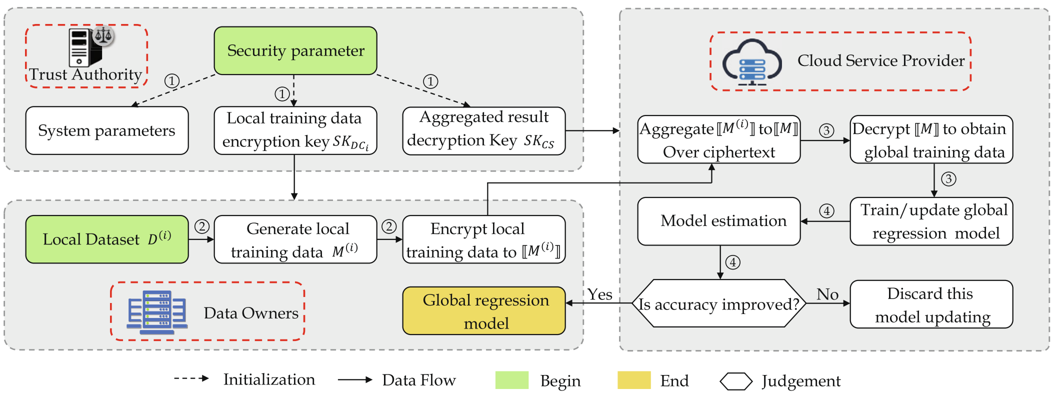
After this, DO$_i$ sends $< \boldsymbol{x}_{max}^{(i)}, \boldsymbol{x}_{min}^{(i)}, n^{(i)} >$ to TA.

Upon receiving all vectors from DOs, TA can compute the global maximum and minimum vectors $\boldsymbol{x}^{(max)} = (x_1^{max}, \ldots, x_d^{max})$ and $\boldsymbol{x}^{(min)} = (x_1^{min}, \ldots, x_d^{min})$ through extracting the maximum and minimum values from $\boldsymbol{x}_{max}^{(i)}$ and $\boldsymbol{x}_{min}^{(i)}$ in each dimension, where $i = 1, \ldots, m$. In addition, for security consideration, TA disturbs $\boldsymbol{x}^{(max)}$ and $\boldsymbol{x}^{(min)}$ as follows

$$\boldsymbol{x}^{(max)} \leftarrow (x_1^{max} + \varepsilon_1, \ldots, x_d^{max} + \varepsilon_d)$$
$$\boldsymbol{x}^{(min)} \leftarrow (x_1^{min} - \varepsilon'_1, \ldots, x_d^{min} - \varepsilon'_d),$$

where ε_j and $\varepsilon'_j, j = 1, \ldots, d$, are small random numbers. In addition, TA computes the total number of samples collected from DOs $n = \sum_{i=1}^{m} n^{(i)}$. Finally, TA returns $< \boldsymbol{x}^{(max)}, \boldsymbol{x}^{(min)} >$ to DOs, and sends n to CSP.

After receiving $< \boldsymbol{x}^{(max)}, \boldsymbol{x}^{(min)} >$ from TA, each DO$_i$ normalizes its local dataset through computing $x_j^{(ik)} \leftarrow \frac{x_j^{(ik)} - x_j^{min}}{x_j^{max} - x_j^{min}}$, where $j = 1, \ldots, j$ and $k = 1, \ldots, n^{(i)}$.



Trust Authority
Security parameter
①
System parameters
Local training data encryption key SK_{DC_i}
Aggregated result decryption Key SK_{CS}
Cloud Service Provider
Aggregate ⟦$M^{(i)}$⟧ to ⟦M⟧ Over ciphertext
③
Decrypt ⟦M⟧ to obtain global training data
Local Dataset $D^{(i)}$
②
Generate local training data $M^{(i)}$
Encrypt local training data to ⟦$M^{(i)}$⟧
Model estimation
④
Train/update global regression model
Data Owners
Global regression model
Yes
Is accuracy improved?
No
Discard this model updating
Initialization
Data Flow
Begin
End
Judgement

Fig. 3. Overview of VANE.

Table 1
Definition of Notations in VANE.

Notation	Definition
κ	Security parameter.
$\|m\|$	The bit length of m.
$[[m]]$	The ciphertext of m with our modified Paillier encryption.
SK_p	Secret key of Paillier cryptosystem.
PK_p	Public key of Paillier cryptosystem.
SK_{DO_i}	Secret key of DO$_i$.
SK_{CSP}	Secret key of CSP.
$\mathcal{D}^{(i)}$	Local dataset of DO$_i$.
$\boldsymbol{X}^{(ik)}$	Collected data from a sample of DO$_i$.
$M^{(ik)}$	Extended matrix from $\boldsymbol{X}^{(ik)}$.
$M^{(i)}$	Local training data of DO$_i$.
$[[M^{(i)}]]$	Encrypted local traning data of DO$_i$.
$[[M]]$	The aggregated ciphertext of $[[M^{(i)}]]$.
M	Global training data.
$AX_{jj'}, AY_j$	Elements in $M, j, j' = 1, \ldots, d$.
$\boldsymbol{\theta}$	Regression model.
α	Learning rate of gradient descent.
λ	Regularization parameter.
M_u	Updated global training data.
$\mathcal{D}'$	Testing dataset of CSP.
RSS	Residual Sum of Squares.
MAE	Mean Absolute Error.

4.3.2. Local training data generation and encryption

In this phase, every DO$_i$ first pre-processes its local dataset to generate local training data. Moreover, with the secret key SK_{DO_i}, each DO$_i$ encrypts its local training data before outsourcing to CSP.

• **Step 1. Data Pre-processing**

Based on above description, each sample in $\mathcal{D}^{(i)}$ is represented as a vector $\boldsymbol{X}^{(ik)} = (x_1^{(ik)}, \ldots, x_d^{(ik)}, y^{(ik)})$. DO$_i$ first extends each $\boldsymbol{X}^{(ik)}$ to a $(d+1) \times (d+1)$ matrix as follows

$$M^{(ik)} = \begin{bmatrix} x_1^{(ik)}, & \cdots, & x_d^{(ik)}, & y^{(ik)} \\ \left(x_1^{(ik)}\right)^2, & \cdots, & x_d^{(ik)} \cdot x_1^{(ik)}, & y^{(ik)} \cdot x_1^{(ik)} \\ \vdots & \ddots & \vdots & \vdots \\ x_1^{(ik)} \cdot x_d^{(ik)}, & \cdots, & \left(x_d^{(ik)}\right)^2, & y^{(ik)} \cdot x_d^{(ik)} \end{bmatrix}.$$

Then, from $k = 1$ to $n^{(i)}$, DO$_i$ computes $M^{(i)} = \sum_{k=1}^{n^{(i)}} M^{(ik)}$ as its local training data.

• **Step 2. Local Training Data Encryption**

Before sending to CSP, each DO$_i$ encrypts its local training data with secret key SK_{DO_i}. Specifically, for each element $a^{(i)} \in M^{(i)}$, DO$_i$ encrypts it through executing $Encrypt(a^{(i)})$ in our secure data aggregation algorithm. Note that $M^{(i)}$ is a rational number matrix, and for executing encryption operations, we should transfer $M^{(i)}$ into integer matrix through computing $\lfloor c \cdot a^{(i)} \rfloor$, where c can be selected as 1000 or bigger integers for achieving higher accuracy.

Finally, DO$_i$ obtains the encrypted local training data $[[M^{(i)}]]$, and sends $< [[M^{(i)}]] >$ to CSP.

4.3.3. Secure data aggregation and training

In this phase, CSP aggregates the encrypted local training data from DOs, decrypts the aggregated results with the secret key SK_{CSP}, and trains the global regression model with gradient descent algorithm.

• **Step 1. Local Training Data Aggregation**

Once the encrypted local training data from all DOs are received, DO first aggregates them over ciphertexts. In specific, for each encrypted element $[[a^{(i)}]]$ in $[[M^{(i)}]]$, where $i = 1, \ldots, m$, CSP executes $Aggregate([[a^{(1)}]], \ldots, [[a^{(m)}]])$ in our secure data aggregation algorithm. In other words, CSP computes

$$[[M]] = \odot \prod_{i=1}^{m} [[M^{(i)}]], \tag{4}$$

where $\odot$ represents hadamard product between matrixes. Then, for each element $[[a]] \in [[M]]$, CSP decrypts it through executing $Decrypt([[a]])$. Finally, the global training data M can be obtained by CSP. For simplify, we use $AX_{jj'}$ and AY_j, where $j = 0, \ldots, d$ and $j' = 1, \ldots, d$, to present the elements in M. Then, the matrix M can be represented as

$$M = \begin{bmatrix} AX_{01}, & \cdots, & AX_{0d}, & AY_0 \\ AX_{11}, & \cdots, & AX_{1d}, & AY_1 \\ \vdots & \ddots & \vdots & \vdots \\ AX_{d1}, & \cdots, & AX_{dd}, & AY_d \end{bmatrix}.$$

• **Step 2. Global Regression Training**

After generating the global training data, CSP is able to train the global regression model for DOs. In specific, with M, CSP first computes vectors

$$\begin{aligned} \boldsymbol{AX}_0 &= (n, AX_{01}, \ldots, AX_{0d}) \\ \boldsymbol{AX}_j &= (AX_{0j}, AX_{j1}, \ldots, AX_{jd}), \end{aligned}$$

where $j = 1, \ldots, d$. Then, CSP selects the learning rate α, initializes regression parameters $\boldsymbol{\theta} = (\theta_0, \theta_1, \ldots, \theta_d)$ as random numbers or zeros, and trains the global linear and ridge as follows.

For linear regression training, CSP executes the gradient descent algorithm

$$\begin{aligned} \theta_0^{i+1} &= \theta_0^i - \tfrac{\alpha}{n}(\boldsymbol{\theta} \cdot \boldsymbol{AX}_0^T - AY_0) \\ \theta_j^{i+1} &= \theta_j^i - \tfrac{\alpha}{n}(\boldsymbol{\theta} \cdot \boldsymbol{AX}_j^T - AY_j), \end{aligned} \tag{5}$$

where $j = 1, \ldots, d$.

For ridge regression training, CSP executes the gradient descent algorithm

$$\begin{aligned} \theta_0^{i+1} &= (1 - 2\lambda\alpha)\theta_0^i - \tfrac{\alpha}{n}(\boldsymbol{\theta} \cdot \boldsymbol{AX}_0^T - AY_0) \\ \theta_j^{i+1} &= (1 - 2\lambda\alpha)\theta_j^i - \tfrac{\alpha}{n}(\boldsymbol{\theta} \cdot \boldsymbol{AX}_j^T - AY_j), \end{aligned} \tag{6}$$

where $j = 1, \ldots, d$.

For logistic regression training, note that target variables $y^{(ik)} \in \{-1, 1\}$, negative values may exist in the $(d+1)$-th column of $M^{(i)}$. Therefore, DO$_i$ should add $n^{(i)}$ in the $(d+1)$-th column for encrypting $M^{(i)}$ correctly. Moreover, CSP can also obtain the global training data through subtracting n in the $(d+1)$-th column in M. Then, CSP is able to train the global logistic regression model through computing

$$\begin{aligned} \theta_0^{i+1} &= (1 - 2\lambda\alpha)\theta_0^i - \tfrac{\alpha}{n}\left(\tfrac{1}{4}\boldsymbol{\theta} \cdot \boldsymbol{AX}_0^T - \tfrac{1}{2}AY_0\right) \\ \theta_j^{i+1} &= (1 - 2\lambda\alpha)\theta_j^i - \tfrac{\alpha}{n}\left(\tfrac{1}{4}\boldsymbol{\theta} \cdot \boldsymbol{AX}_j^T - \tfrac{1}{2}AY_j\right), \end{aligned} \tag{7}$$

where $j = 1, \ldots, d$. Finally, CSP obtains the linear, ridge or logistic regression model $\boldsymbol{\theta}$, and returns it to DOs.

Algorithm 1: SDAT: Secure Data Aggregation and Training

```
  Input: The local training data of DOs: $M^{(i)}$.
  Output: The global regression model: $\boldsymbol{\theta}$.
1 foreach element $a^{(i)}$ in local training data $M^{(i)}$ do
2     DO$_i$ uses its secret key $SK_{DO_i}$ to encrypt $a^{(i)}$
          $[\![a^{(i)}]\!] \leftarrow Encrypt(a^{(i)}, SK_{DO_i})$;
3 end
4 DO$_i$ obtains the encrypted local training data $[\![M^{(i)}]\!]$;
5 foreach $[\![M^{(i)}]\!]$, $i = 1, 2, \cdots, m$ do
6     CSP aggregates each element $[\![a^{(i)}]\!]$ in $[\![M^{(i)}]\!]$ over ciphertexts
          $[\![a]\!] \leftarrow Aggregate([\![a^{(i)}]\!])$;
7 end
8 CSP uses its secret key $SK_{CSP}$ to decrypt aggregated elements
  $[\![a]\!]$
          $a \leftarrow Decrypt([\![a]\!], SK_{CSP})$;
9 With $a$, CSP generates matrix $M$ as the global training data, and
  trains the global regression model $\boldsymbol{\theta}$ with $M$ based on gradient
  descent algorithm
          $\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^i - \alpha\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, M)$;
```

4.3.4. Model updating and estimation

In practice, the regression model should be updated regularly, and there are mainly two strategies for updating the regression model: *updating with new coming samples* and *updating with new coming data owner*. Moreover, for ensuring the accuracy, CSP is responsible for estimating the updated global regression model after updating.

• **Model Updating**

On the one hand, the samples in local datasets are continuously collected by DOs. Therefore, the global regression model can be updated with *new coming samples*. On the other hand, some new data owners may participate in the system. Then, the global regression model can be updated with *new coming data owners*.

Specifically, for updating the model with new coming samples, TA first renews the random number R_t for updating the task ID, and generates new secret keys for DOs. In addition, TA updates normalization parameters through checking the maximum and minimum feature values of new coming samples. Then, similar to Section 4.2, each DO pre-processes its new collected samples and generates the encrypted fresh training data $[[M_f^{(i)}]]$ with its new secret key. Finally, CSP aggregates $[[M_f^{(i)}]]$, where $i = 1, \ldots, m$ to generate M_f through executing the same operations in Section 4.3, computes $M_u = M + M_f$, and updates the global regression model to $\boldsymbol{\theta}_u$ via retraining it with M_u.

For updating the model with a new coming data owner DO'_i, TA first regenerates secret keys for all DOs through selecting $m+1$ random numbers $< n_1, \ldots, n_{m+1} >$, s.t., $\sum_i^{m+1} n_i = N$, and updates normalization parameters with the maximum and minimum feature values in the local dataset of DO'_i. Then, each DO generates its encrypted local training data and outsources it to CSP. As a result, the local training data of DO'_i is contained in the new aggregated global training data M_u, and CSP can update the global regression model to $\boldsymbol{\theta}_u$ via retraining it with M_u.

• **Model Estimation**

During the regression model updating, it is nonnegligible that some fuzzy data may mix in the training dataset. Thus, CSP is responsible for estimating the generated global regression model after updating to ensure its accuracy.

Specifically, in our system, we assume CSP has a testing dataset (i.e., the open machine learning dataset) containing u samples, which is presented as $\mathcal{D}' = \{\boldsymbol{x}'^{(l)}, y'^{(l)}\}_{l=1}^{u}$. Then, the modeling error of global regression model can be defined with Residual Sum of Squares (RSS) as follows.

$$RSS(\boldsymbol{\theta}) = \sum_{l=1}^{u} \left(\mathrm{Pre}(y'^{(l)}) - y'^{(l)}\right)^2,$$

where $\mathrm{Pre}(y'^{(l)})$ is the predicted value of $\boldsymbol{x}'^{(l)}$ with the generated model.

Finally, CSP can estimate the contribution of new coming data through computing

$$C = \frac{RSS(\boldsymbol{\theta}_u)}{RSS(\boldsymbol{\theta})}.$$

In detail, $C \geqslant 1$ means that new coming data reduce the accuracy of global regression model, and CSP can discard this model updating. While $C < 1$, it means that new coming data are effective for improving the accuracy of global regression model, and CSP will accept this model updating.

4.4. Correctness of the proposed scheme

In this section, we first extract SDAT Algorithm 1 of our proposed VANE, and prove the correctness of SDAT in terms of secure data aggregation and global regression training, which is sufficient to verify the correctness of VANE.

4.4.1. Correctness of secure data aggregation

In SDAT, local training data are first aggregated with our secure data aggregation algorithm. Thus, we prove the correctness of our secure data aggregation algorithm as follows.

In Eq. (2), for aggregating ciphertext $[[x^{(i)}]]$, CSP computes

$$\begin{aligned}
[[x]] &= \prod_{i=1}^{m} [[x^{(i)}]] \bmod N^2 \\
&= \prod_{i=1}^{m} g^{x^{(i)}} \cdot h^{r_i} \cdot R_t^{n_i} \bmod N^2 \\
&= g^{\sum_{i=1}^{m} x^{(i)}} \cdot h^{\sum_{i=1}^{m} r_i} \cdot R_t^{\sum_{i=1}^{m} n_i} \bmod N^2 \\
&= g^{\sum_{i=1}^{m} x^{(i)} + \gamma \cdot \sum_{i=1}^{m} r_i} \cdot R_t^{N} \bmod N^2.
\end{aligned}$$

It can be seen that the in the aggregated ciphertext $[[x]]$, parameter N is retrieved. Moreover, since the lengths of γ and r_i are less than κ, $\sum_{i=1}^{m} x^{(i)} + \gamma \cdot \sum_{i=1}^{m} r_i$ is within the plaintext space of Paillier cryptosystem. Therefore, CSP is able to decrypt $[[x]]$ with the secret key SK_{CSP}. Specifically, in Eq. (3), we have

$$\begin{aligned} x &= (L([[x]]^{\lambda} \bmod N^2 \cdot \mu) \bmod N) \bmod \gamma \\ &= \left(\sum_{i=1}^{m} x^{(i)} + \gamma \cdot \sum_{i=1}^{m} r_i\right) \bmod \gamma \\ &= \sum_{i=1}^{m} x^{(i)}. \end{aligned}$$

As a result, the correctness of our secure data aggregation algorithm is verified, and it can be inferred that the global training data is the summation of all samples from DOs, which is presented as

$$M = \sum_{i=1}^{m} M^{(i)} = \sum_{i=1}^{m}\sum_{k=1}^{n^{(i)}} M^{(ik)}.$$

4.4.2. Correctness of global regression training

With global training data, CSP is able to train the global regression model based on gradient descent algorithm. Since global training data M is the summation of all samples from DOs, in Eq. (4), it can be derived that

$$\begin{aligned} \boldsymbol{\theta} \cdot \boldsymbol{AX}_0^T - AY_0 &= \boldsymbol{\theta} \cdot \sum_{i=1}^{m}\sum_{k=1}^{n^{(i)}} (1, x_1^{(ik)}, \ldots, x_d^{(ik)})^T - \sum_{i=1}^{m}\sum_{k=1}^{n^{(i)}} y^{(ik)} = h\left(\boldsymbol{\theta}, \sum_{i=1}^{m}\sum_{k=1}^{n^{(i)}} (1, x_1^{(ik)}, \ldots, x_d^{(ik)})\right) - \sum_{i=1}^{m}\sum_{k=1}^{n^{(i)}} y^{(ik)} \\ &= \sum_{i=1}^{m}\sum_{k=1}^{n^{(i)}} (h(\boldsymbol{\theta}, \boldsymbol{x}^{(ik)}) - y^{(ik)}) \end{aligned}$$

$$\begin{aligned} \boldsymbol{\theta} \cdot \boldsymbol{AX}_j^T - AY_j &= \boldsymbol{\theta} \cdot \sum_{i=1}^{m}\sum_{k=1}^{n^{(i)}} (x_j^{(ik)}, x_1^{(ik)} \cdot x_j^{(ik)}, \ldots, x_d^{(ik)} \cdot x_j^{(ik)})^T - \sum_{i=1}^{m}\sum_{k=1}^{n^{(i)}} y^{(ik)} \cdot x_j^{(ik)} \\ &= h(\boldsymbol{\theta}, \sum_{i=1}^{m}\sum_{k=1}^{n^{(i)}} (x_j^{(ik)}, x_1^{(ik)} \cdot x_j^{(ik)}, \ldots, x_d^{(ik)} \cdot x_j^{(ik)})) - \sum_{i=1}^{m}\sum_{k=1}^{n^{(i)}} y^{(ik)} \cdot x_j^{(ik)} = \sum_{i=1}^{m}\sum_{k=1}^{n^{(i)}} (h(\boldsymbol{\theta}, \boldsymbol{x}^{(ik)}) - y^{(ik)}) x_j^{(ik)}, \end{aligned}$$

where $h(\boldsymbol{\theta}, \boldsymbol{x}^{(ik)}) = \theta_0 + \Sigma_{j=1}^{d} \theta_j x_j^{(ik)}$.

Finally, it can be verified that the gradient descent algorithm of Eq. (4) is executed with all samples from DOs. For linear regression, we have

$$\begin{aligned} \theta_0^{i+1} &= \theta_0^i - \tfrac{\alpha}{n} \sum_{i=1}^{m}\sum_{k=1}^{n^{(i)}} (h(\boldsymbol{\theta}, \boldsymbol{x}^{(ik)}) - y^{(ik)}) \\ \theta_j^{i+1} &= \theta_j^i - \tfrac{\alpha}{n} \sum_{i=1}^{m}\sum_{k=1}^{n^{(i)}} (h(\boldsymbol{\theta}, \boldsymbol{x}^{(ik)}) - y^{(ik)}) x_j^{(ik)}. \end{aligned}$$

Correspondingly, the correctness for ridge and logistic regression can also be verified easily.

5. Security analysis

In this section, we analyze the security of our VANE. Specifically, corresponding to the security requirements discussed in Section 2, our analysis mainly focuses on the privacy of DOs' local training data and the accuracy of global regression considering fuzzy data.

• **The privacy of DOs' Local Training Data is Achieved.** In our VANE, since the privacy of DOs' local training data relies on our modified Paillier cryptosystem, we prove its security firstly.

In original Paillier cryptosystem, the secret key SK_P is strong to decrypt arbitrary ciphertexts encrypted with the public key PK_P. Therefore, to achieve secure data aggregation with original Paillier cryptosystem, double-cloud model is applied in [17,35]. Specifically, one cloud is responsible for aggregating the ciphertexts from data owners, while the other one is to decrypt the aggregated results for training the global model. Nevertheless, double-cloud model brings extra communication overhead, and it requires that two clouds are non-colluded, which is difficult to achieve in reality.

In our modified Paillier cryptosystem, we split N in public key PK_P to m parts $\{n_1, \ldots, n_m\}$, and compute secret keys $SK_{DO_i} = R_t^{n_i} \bmod N^2$ for each DO$_i$. Then, the local training data of DO$_i$ is encrypted as $[[a^{(i)}]] = g^{a^{(i)}} \cdot h^{r_i} \cdot SK_{DO_i} \bmod N^2$ in Eq.

(1). As a result, the secret key SK_P is unable to decrypt ciphertexts encrypted with SK_{DO_i}. In Eq. (2), each aggregated element of $[[M]]$ is $[[a]] = g^{\sum_{i=1}^{m} a^{(i)}} \cdot h^{\sum_{i=1}^{m} r_i} \cdot R_t^N \bmod N^2$, in which N is retrieved through aggregation operation. Therefore, the secret key SK_{CSP} can only be used to decrypt aggregated results, and only one cloud is required in our system. Besides, for maintaining the semantic security of Paillier cryptosystem, we add a term $h^{r_i} = (g^{\gamma})^{r_i}$ in $[[a^{(i)}]]$, where r_i is a random number to achieve indistinguishability of ciphertexts. In order to break the semantic security, CSP needs to find an ω satisfying $h^{\omega} = 1 \bmod N^2$ to break indistinguishability at first. Specifically, with ω, CSP is able to compute $[[a^{(i)}]]^{\omega} = g^{\omega \cdot a^{(i)}} \cdot SK_{DO_i}^{\omega} \bmod N^2$ to eliminate the random number r_i in the ciphertext. Meanwhile, it can be seen that $g^{\omega} \neq 1 \bmod N^2$ should also be satisfied for ensuring that the plaintext $a^{(i)}$ is not eliminated. Therefore, ω should satisfy both $h^{\omega} = 1 \bmod N^2$ and $g^{\omega} \neq 1 \bmod N^2$. In the following, we prove that such an ω is nonexistent.

Claim. In our modified Pailler cryptosystem, it is impossible to find a ω satisfying $h^{\omega} = 1 \bmod N^2$ and $g^{\omega} \neq 1 \bmod N^2$ to break the indistinguishability.

Proof. In order to achieve the first condition $h^{\omega} = g^{\gamma \cdot \omega} = 1 \bmod N^2$, since the order of g is a multiple of N (i.e., $ord(g) = k \cdot N$, where k is an integer), the following equation should be satisfied.

$$\gamma \cdot \omega = k' \cdot k \cdot N = k' \cdot k \cdot p \cdot q,$$

where k' and k are integers. Then, we have

$$\omega = \frac{k' \cdot k \cdot p \cdot q}{\gamma}.$$

Since $gcd(\gamma, k) = 1$ and p, q are primes, k' should be a multiple of γ, i.e., $k' = k'' \cdot \gamma$. As a result,

$$\omega = k'' \cdot k \cdot p \cdot q = k'' \cdot ord(g),$$

which cannot satisfy the second condition $g^{\omega} \neq 1 \bmod N^2$. Therefore, CSP is impossible to find a ω for breaking indistinguishability of our modified Paillier Cryptosystem.

Based on above analysis, it can be verified the local training data of DOs are well protected with our encryption algorithm. Moreover, for inferring a DO's local training data, it is required for CSP to collude with other $m - 1$ DOs, which is impossible in reality. Therefore, collusion attack is resisted in our scheme. Besides, for each data aggregation task, the task id R_t is renewed by TA to update secret keys for each MC, with which the replay attack is also resisted.

• **The Accuracy of Global Regression Model is Ensured.** In practice, fuzzy data may sneak into local dataset of DOs, and even poisoning attacks may be executed by adversaries. As a result, the accuracy of trained global regression model is reduced.

In VANE, CSP is able to estimate the accuracy of global regression model through computing Residual Sum of Squares (RSS) $RSS(\boldsymbol{\theta}) = \sum_{l=1}^{u} (\text{Pre}(y'^{(l)}) - y'^{(l)})^2$ with a testing dataset. In addition, after each model updating, CSP compares the RSS between the original model $\boldsymbol{\theta}$ and the updated model $\boldsymbol{\theta}_u$ with $C = RSS(\boldsymbol{\theta}_u)/RSS(\boldsymbol{\theta})$. Specifically, $C < 1$ means that new coming data contributes positively to the global regression model. Thus, CSP accepts this model updating and returns the updated global regression model to DOs. In addition, while $C \geqslant 1$, it means that the accuracy of global regression model is reduced and fuzzy data may exist in training data. Then, CSP discards this model updating, with which the accuracy of global regression model can be guaranteed.

6. Performance evaluation

In this section, we analyze and test the performance of the proposed VANE in terms of accuracy, computation cost and communication overhead, and make a comparison with the PrivFL [18]. In detail, PrivFL is an integrated federated regression scheme supporting linear, ridge, and logistic training, which is constructed with homomorphic encryption and double-cloud architecture.

6.1. Evaluation environment

In order to measure the integrated performance, we conduct the experiment in Java running on the PC with one 2.2-GHz Intel Core i7, 16-GB memory, and Windows 10 system. Moreover, we set the security parameter as $\kappa = 1024$ in our modified Paillier cryptosystem. For PrivFL, we choose Paillier cryptosystem as the homomorphic encryption algorithm, and also set the security parameter as 1024. Note that the computation cost of data normalization is not considered during our evaluation, since it can be ignored in the whole training process. For testing the accuracy of our VANE, we first use 6 real datasets to train linear, ridge and logistic regression models. Besides, we also use a synthetic dataset to test all factors affecting the performance of our VANE, and make a comparison with PrivFL. The details of datasets are described as follows.

- Datasets for linear and ridge regression: We first use 3 real datasets for evaluating accuracy of the trained models with linear and ridge regression. In detail, the target variables in these datasets are continuous, which are suitable for training linear and ridge regression models. In detail, the 3 datasets are Auto MPG Dataset (AMD) [26], Boston Housing Dataset (BHD) [12], and Wine Quality Dataset (WQD) [6].
- Datasets for logistic regression: We also select other 3 real datasets for evaluating the accuracy of the trained model with logistic regression. Different from linear and ridge regression, the target variables in these datasets are binary variables, which can be used to train classification models with logistic regression. In detail, the 3 datasets are Breast Cancer Dataset (BCD) [29], Diabetes Dataset (DD) [27], and US Census Income Dataset (UCID) [16].
- Synthetic dataset: In order to test the integrated performance of the proposed VANE, we generate a synthetic dataset randomly, which consists of 2000 instances. Each instance contains 20 dimensions, and the value of each element is randomly picked with positive decimals.

6.2. Accuracy evaluation

As mentioned above, we evaluate the accuracy of our VANE on 6 different real datasets: 3 datasets for evaluating the linear/ridge regression training, and other 3 datasets for logistic regression training. The dimension of the datasets ranges from 8 to 12 for linear/ridge regression and from 9 to 14 for logistic regression. Specifically, the Mean Absolute Error (MAE) and confusion matrix are used to measure the accuracy of the trained model for linear/ridge and logistic regression, respectively. Moreover, for different datasets, we set different parameters (i.e., learning rate, regularization parameter, and iteration times) to train different models. Tables 2 and 3 show the test results, it can be seen that our VANE is available in practice. Note that, the privacy-preserving mechanism is not used in this section.

6.3. Computation cost

In this section, we analyze and test the computation cost of our VANE in terms of linear regression and logistic regression, and make a comparison with PrivFL. Note that in this section, ridge regression is not considered since its main computations are similar to linear regression in both VANE and PrivFL.

For the sake of simplicity, we only record complex arithmetical operations in this section. Specifically, we set t_{inv}, t_{exp} and t_{mul} to represent the computation cost of a modular inverse, a modular exponentiation and a modular multiplication operation, respectively. In addition, we also use m, d, n, and l to represent the number of data owners, data dimensions, all training samples, and iterations for training a regression model, respectively.

- **Computation cost of our VANE**

In VANE, for securely training a linear regression model, DOs encrypt their local training data through our modified Paillier cryptosystem, which costs $2m \cdot (d+1)^2 \cdot (t_{exp} + t_{mul})$. Then, CSP aggregates encrypted local training data from DOs, and decrypts aggregated results to obtain global train data, which costs $m \cdot (d+1)^2 \cdot t_{mul}$ and $(d+1)^2 \cdot (t_{inv} + 2t_{mul} + t_{exp})$, respectively. Therefore, the total computation cost of CSP is $(d+1)^2 \cdot (t_{inv} + (m+2) \cdot t_{mul} + t_{exp})$. For securely training a logistic regression model with VANE, since the most time-consuming operations are encryption, decryption, and modular multiplication, which are the same as linear regression. Thus, the computation cost of DOs and CSP are $2m \cdot (d+1)^2 \cdot (t_{exp} + t_{mul})$ and $(d+1)^2 \cdot (t_{inv} + (m+2) \cdot t_{mul} + t_{exp})$, respectively.

- **Computation cost of PrivFL**

In PrivFL, for securely training a linear regression model, the computation cost of DOs in one iteration is $(3n + 2nd) \cdot t_{mul} + 2 \cdot (n + d + nd + 1) \cdot t_{exp}$. Besides, the computation cost of CSP in one iteration is $(d + 2m + 1) \cdot t_{mul} + (2d + m + 2) \cdot t_{exp} + m \cdot t_{inv}$. Therefore, the total computation cost of DOs and CSP is $l \cdot ((n + d + 2nd) \cdot t_{mul} + 2 \cdot (n + d + nd + 1) \cdot t_{exp}))$ and $l \cdot ((d + 2m + 1) \cdot t_{mul} + (2d + m + 2) \cdot t_{exp} + m \cdot t_{inv}))$, respectively. For secure training a logistic regression model, it costs $(13n + 2nd) \cdot t_{mul} + (10n + 2d + 2nd + 2) \cdot t_{exp}$ for DOs in one iteration. Meanwhile, $(d + 2m + 4n + 1) \cdot t_{mul} + (2d + m + 5n + 2) \cdot t_{exp} + (m + n) \cdot t_{inv}$ should be executed by CSP. Therefore, the total computation cost of secure logistic training for DOs and CSP is $l \cdot ((13n + 2nd) \cdot t_{mul} + (10n + 2d + 2nd + 2) \cdot t_{exp})$ and $l \cdot ((d + 2m + 4mn + 1) \cdot t_{mul} + (2d + m + 5mn + 2) \cdot t_{exp} + (m + mn) \cdot t_{inv})$, respectively.

Table 2
Accuracy Evaluation of Linear and Ridge Regression

Dataset	d	n	α	λ	l	MAE (linear)	MAE (ridge)
AMD [26]	8	392	10^{-1}	10^{-2}	10^3	0.4623	0.4827
BHD [12]	14	506	10^{-1}	10^{-2}	10^3	3.266	3.649
WQD [6]	12	1599	10^{-1}	10^{-2}	10^3	0.5277	0.5614

d: dimension; n: the number of samples; α: learning rate; λ: regularization parameter;
l: the number of iterations.

Table 3
Accuracy Evaluation of Logistic Regression

Dataset	d	n	α	λ	l	Accuracy (logistic)
BCD [29]	9	699	10^{-1}	10^{-2}	10^3	95.70%
DD [27]	9	768	10^{-1}	10^{-3}	10^4	77.08%
UCID [16]	14	48842	10^{-1}	10^{-3}	5×10^3	81.56%

d: dimension; n: the number of samples; α: learning rate; λ: regularization parameter;
l: the number of iterations.

• **Comparison**

In Tables 4 and 5, we present the comparison of computation cost for our VANE and PrivFL theoretically. Specifically, for VANE, it can be seen that the computation cost of both linear and logistic regression is independent of the number of training samples n since the local datasets are pre-processed by data owners. Besides, the computation cost of VANE is also independent of the number of iterations since interactions are not required in the whole training process. Nevertheless, in PrivFL, each sample should be computed with the encrypted global model parameters in every iteration, therefore, the computation cost of PrivFL increases linearly with the number of training samples n and dimensions d. Moreover, since the operations should be repeated in PrivFL for every iteration of gradient descent, the computation cost also increases with the number of iterations l. As a result, the computation cost of our VANE is only related with the number of dimensions d and data owners m, while PrivFl is related with the number dimensions d, data owners m, training samples n, and iterations l, which makes VANE more efficient than PrivFL.

In Fig. 4), we further plot the computation cost of our VANE and PrivFL with the generated synthetic dataset. Specifically, we set the security parameter $\kappa = 1024$, the number of data owners $m = 3$, the iterations of gradient descent $l = 100$, the number of dimensions d varying from 10 to 15, and the number of all training samples n varying from 1200 to 2000. From Fig. 4 (a), (c) and (d), we can see that the computation cost of VANE is not affected with n while that of PrivFL increases linearly, and the computation cost of PrivFL is much higher than VANE. Note that in Fig. 4 (b), the computation cost of cloud service provider in PrivFl also maintains a constant with different number training samples, which is owing to the cloud service provider only needs to process the aggregated results of training samples. However, our VANE can also perform better than PrivFL in this case.

6.4. Communication overhead

In this section, we analyze and test the communication overhead of our VANE in terms of linear regression and logistic regression, and make a comparison with PrivFL.

• **Communication overhead of our VANE**

In both of secure linear and logistic regression training of VANE, interactions are not required between data owners and the cloud service provider. Specifically, each DO$_i$ first pre-processes its local dataset to a $(d+1) \times (d+1)$ matrix $M^{(i)}$, and encrypted elements in $M^{(i)}$ with our modified Paillier cryptosystem. Finally, each DO$_i$ outsources $[[M^{(i)}]]$ to the cloud service provider. Note that the length of every encrypted element in $[[M^{(i)}]]$ is 2κ bits, where κ is the security parameter. Therefore, the total communication overhead between DOs and CSP in VANE is $2m \cdot (d+1)^2 \cdot \kappa$ bits.

• **Communication overhead of PrivFL**

In PrivFL, for securely training a linear regression model, it spends $2 \cdot (d+1) \cdot \kappa$ bits for CSP to public the encrypted global model parameters in every iteration. Then, each DO$_i$ returns its encrypted local gradient descent to CSP, which spends $2 \cdot \kappa$ bits. Therefore, the total communication overhead of secure linear regression training between DOs and CSP is $2ml \cdot (d+2) \cdot \kappa$ bits. For securely training a linear regression model, more interactions should be executed between each DO$_i$ and CSP in every iteration, which spends extra $6n\kappa$ bits. Therefore, the total communication overhead of secure logistic training is $2l \cdot (2m + 3n + md) \cdot \kappa$ bits.

• **Comparison**

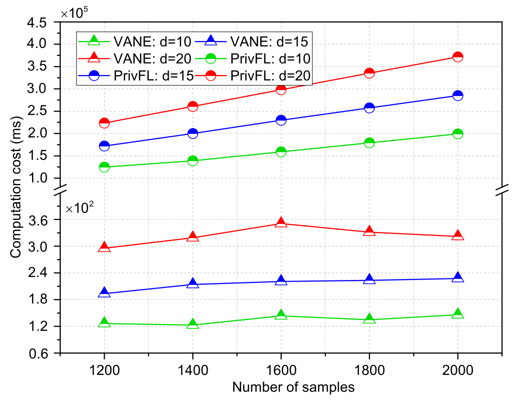
In Table 6, we present the comparison of communication overhead for our VANE and PrivFL theoretically. Similar to computation cost, in secure linear regression training, the communication overhead of VANE only increases with the number of

Table 4
Computation cost of VANE vs PrivFL (Linear)

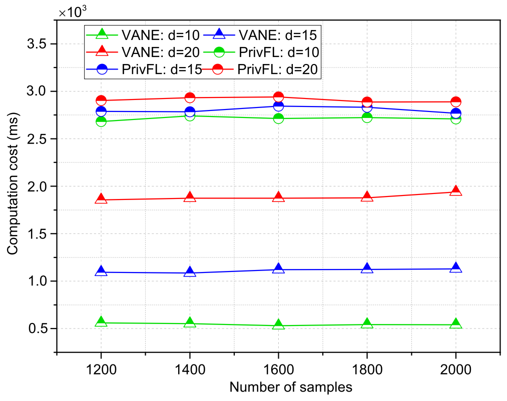
Linear regression	VANE	PrivFL
Data owners	$2m \cdot (d+1)^2 \cdot (t_{exp} + t_{mul})$	$l \cdot ((n + d + 2nd) \cdot t_{mul} + 2 \cdot (n + d + nd + 1) \cdot t_{exp}))$
cloud service provider	$(d+1)^2 \cdot (t_{inv} + (m+2) \cdot t_{mul} + t_{exp})$	$l \cdot ((d + 2m + 1) \cdot t_{mul} + (2d + m + 2) \cdot t_{exp} + m \cdot t_{inv}))$

Table 5
Computation cost of VANE vs PrivFL (Logistic)

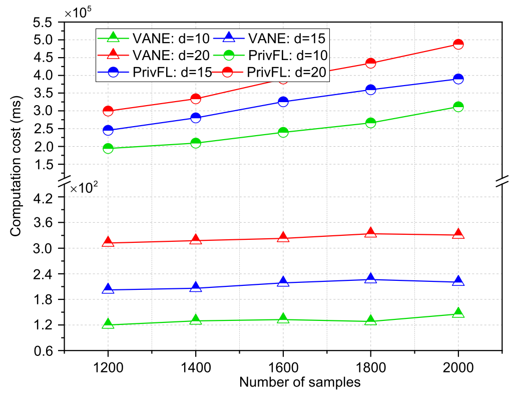
Logistic regression	VANE	PrivFL
Data owners	$2m \cdot (d+1)^2 \cdot (t_{exp} + t_{mul})$	$l \cdot ((13n + 2nd) \cdot t_{mul} + (10n + 2d + 2nd + 2) \cdot t_{exp})$
cloud service provider	$(d+1)^2 \cdot (t_{inv} + (m+2) \cdot t_{mul} + t_{exp})$	$l \cdot ((d + 2m + 4n + 1) \cdot t_{mul} + (2d + m + 5n + 2) \cdot t_{exp} + (m+n) \cdot t_{inv})$



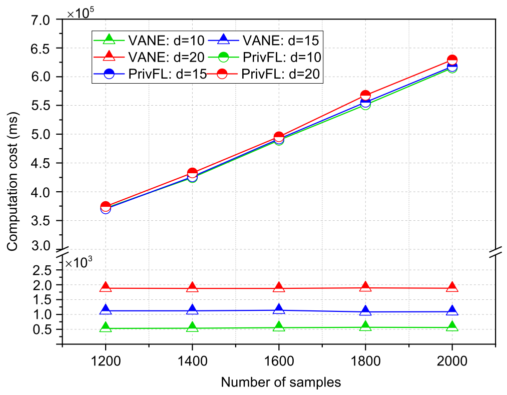
(a) Computation cost of data owners for linear regression.



(b) Computation cost of cloud service provider for linear regression.



(c) Computation cost of data owners for logistic regression.



(d) Computation cost of cloud service provider for logistic regression.

Fig. 4. Computation cost of VANE vs PrivFL.

Table 6
Communication overhead of VANE vs PrivFL

	VANE	PrivFL
linear regression	$2m \cdot (d+1)^2 \cdot \kappa$	$2ml \cdot (d+2) \cdot \kappa$
logistic regression	$2m \cdot (d+1)^2 \cdot \kappa$	$2l \cdot (2m + 3n + md) \cdot \kappa$

dimensions d and data owners m, while PrivFL increases with the number of dimensions d, data owners m and iterations l. In secure logistic regression training, the communication overhead of VANE is the same as linear regression training, and PrivFL increases with the number of dimensions d, data owners m, training samples n, and iterations l. Therefore, our proposed VANE performs much better than PrivFL especially when the size of training dataset is large.

In Fig. 5, the communication overhead of our VANE and PrivFL are plotted with different number of dimensions and training samples. Specifically, in Fig. 5 (a), we can see that the communication overheads of both VANE and PrivFl are not changed

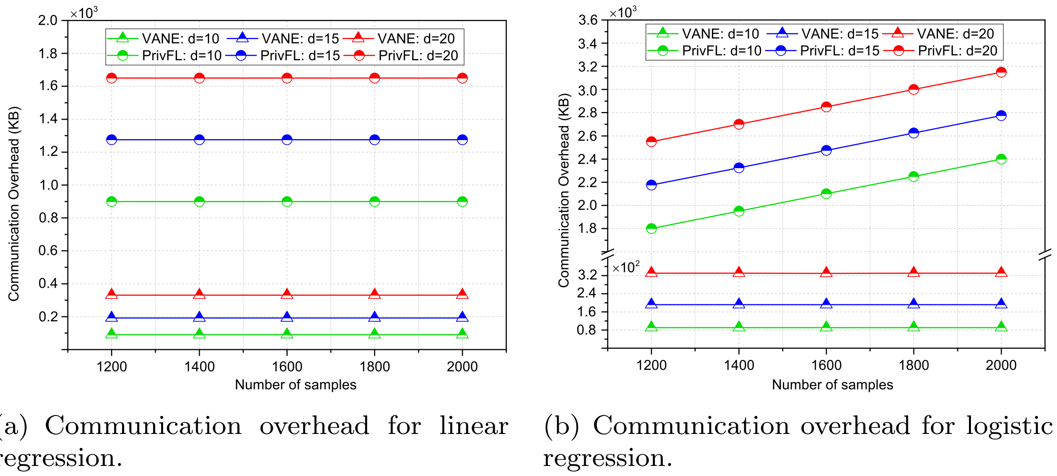


Fig. 5. Communication overhead of VANE vs PrivFL.

with the varying number of training samples, but PrivFL consumes more communication overhead than that of VANE. Moreover, in Fig. 5 (b), VANE maintains low communication overhead with different number of training samples while PrivFL linearly increases. As a result, it can be concluded that VANE is more efficient than PrivFL in terms of communication overhead, and can be applied in practice.

7. Related work

In this section, we briefly discuss some related works about federated learning and privacy-preserving regression training.

Federated learning. Recently, federated learning has been widely researched with the rapid growth of internet data. Zheng et al. [37] designed a non-interactive skyline points extracting scheme over ciphertexts, which is able to merge the skyline bounds from multiple data owners without disclosing sensitive information. Based on secret sharing and homomorphic function, Xu *et al.* [32] presented a secure and verifiable federated learning scheme, with which federated deep learning is achieved and the final learning results are verifiable. Xie *et al.* [31] proposed a new robust distributed optimization algorithm with efficient communication and attack tolerance, which achieves provable convergence and is robust in practice. Ang *et al.* [1] proposed a novel federated learning scheme for declining the effect of data noise, which can tackle the unavailable maxima or minima noise during the process of federated learning. Based on Paillier cryptosystem, Yang *et al.* [35] presented an efficient federated learning scheme for distributed Naïve bayesian training, which improves the efficiency of secure Naïve bayesian training through introducing super-increasing sequence. Liu *et al.* [17] proposed a new medical data aggregation algorithm based on double-cloud architecture, and further constructed a federated Naïve Bayesian training scheme for distributed clinical decision support system. Although Paillier cryptosystem is extensively used in federated learning scheme, it can only be deployed in double-cloud model to aggregate local data from data owners, which brings extra communication overhead and is hard to achieve in reality. Moreover, most above-mentioned distributed machine learning allows to training a more precious model with multiple data owners' local data. However, high computation complexity and complicated architecture are inevitable in these schemes, which makes them hardly to be applied in the real environment.

Privacy-preserving regression training. Nowadays, privacy-preserving regression training has also attracted much attention in the area of machine learning. Yang et al. [34] proposed a secure gradient descent computation method with homomorphic encryption, and constructed an efficient and privacy-preserving linear regression protocol over outsourced encrypted data. Jagielski *et al.* [14] introduced a fast statistical attack of linear training, meanwhile, they also designed a novel defense method which can resilient against poisoning attacks in regression training. Cock *et al.* [5] designed a secure two-party protocol to securely execute ordinary least square (OLS), and which achieved high-efficiency of linear regression training in online phase. Maddal *et al.* [18] considered the data normalization problem before model training, and proposed a comprehensive privacy-preserving method for linear, ridge, and logistic regression training. Specifically, cubic polynomial is used to approximate the loss function of logistic regression, which makes additive homomorphic encryption available in their scheme. Mohassel *et al.* [20] proposed a novel system which can train a linear, logistic or neural network model with double-cloud model, which is a general method for machine learning algorithms. Du *et al.* [8] proposed a framework enabling multiple parties to train a learning model collaboratively, which can achieve accurate model training and ϵ-differential privacy. However, most above-mentioned schemes rely on multi-round communication to achieve privacy-preserving gradient descent algorithm, which brings massive extra communication overhead.

Table 7
Functionality comparison

	[8]	[18]	[5]	[34]	[20]	VANE
Multi-party training	✓	✓	×	×	✓	✓
Intergraded training	✓	✓	×	×	✓	✓
Non-interactive	×	×	✓	✓	✓	✓
Single cloud model	✓	×	✓	✓	×	✓
Model estimation	×	×	×	×	×	✓
High-efficiency	×	✓	×	✓	×	✓

Different from above works, our proposed scheme aims at non-interactive model training and privacy issues. Based on data pre-computation, interactions between data owners and the cloud service provider are not required, and the local training data can be well protected during global regression model training. Moreover, only one cloud is required in our system. In detail, we make a detailed comparison of VANE and existing schemes in Table 7. From the table, it can be seen that our VANE is more practical in the real environment.

8. Conclusion

In this paper, we have proposed a secure and non-interactive federated learning scheme for regression training, called VANE. Based on the proposed secure data aggregation algorithm, in VANE, the CSP can securely aggregate local training data from multiple DOs over ciphertexts, and train a global regression model with the aggregated result. In the whole training process, interactions are not required between DOs and CSP. Moreover, the proposed scheme also greatly improved the efficiency of secure federated regression training through data pre-processing. Detailed security analysis showed its security strength and privacy-preserving ability, and extensive experiments were conducted to verify its high efficiency.

CRediT authorship contribution statement

Fengwei Wang: Conceptualization, Writing - original draft, Software. **Hui Zhu:** Funding acquisition, Project administration. **Rongxing Lu:** Methodology. **Yandong Zheng:** Writing - review & editing. **Hui Li:** Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by National Key Research and Development Program of China (2017YFB0802200), National Natural Science Foundation of China (61972304, 61932015), Natural Science Foundation of Shaanxi Province (2019ZDLGY12-02).

References

[1] F. Ang, L. Chen, N. Zhao, Y. Chen, W. Wang, F.R. Yu, Robust federated learning with noisy communication, IEEE Transactions on Communications 68 (2020) 3452–3464, https://doi.org/10.1109/TCOMM.2020.2979149.
[2] Y. Aono, T. Hayashi, L.T. Phong, L. Wang, Privacy-preserving logistic regression with distributed data sources via homomorphic encryption, IEICE Transactions on Information and Systems 99-D (2016) 2079–2089, https://doi.org/10.1587/transinf.2015INP0020.
[3] A. Asuncion, D. Newman, Uci machine learning repository, 2007. URL:https://archive.ics.uci.edu/ml/index.php..
[4] Y. Chen, A. Rezapour, W. Tzeng, Privacy-preserving ridge regression on distributed data, Information Sciences 451–452 (2018) 34–49, https://doi.org/10.1016/j.ins.2018.03.061.
[5] M.D. Cock, R. Dowsley, A.C.A. Nascimento, S.C. Newman, Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data, in: Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security ACM, 2015, pp. 3–14, https://doi.org/10.1145/2808769.2808774.
[6] Cortez, P., 2009. Wine quality dataset. URL:https://archive.ics.uci.edu/ml/datasets/wine+quality..
[7] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, J. Qin, A survey on application of machine learning for internet of things, International Journal of Machine Learning and Cybernetics 9 (2018) 1399–1417, https://doi.org/10.1007/s13042-018-0834-5.
[8] W. Du, A. Li, Q. Li, Privacy-preserving multiparty learning for logistic regression, in: 14th International Conference on Security and Privacy in Communication Networks, Springer, 2018, pp. 549–568, https://doi.org/10.1007/978-3-030-01701-9_30.
[9] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahur, D. Evans, Privacy-preserving distributed linear regression on high-dimensional data, Proceedings on Privacy Enhancing Technologies 2017 (2017) 345–364, https://doi.org/10.1515/popets-2017-0053.
[10] O. Goldreich, S. Micali, A. Wigderson, How to play any mental game, or a completeness theorem for protocols with honest majority, in: Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali, ACM, 2019, pp. 307–328. doi: 10.1145/3335741.3335755..
[11] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, B. Thorne, Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption, 2017. CoRR abs/1711.10677. URL:http://arxiv.org/abs/1711.10677, arXiv:1711.10677..
[12] D. Harrison, Jr., D.L. Rubinfeld, Boston housing dataset, 1978. URL:http://lib.stat.cmu.edu/datasets/boston..

[13] A.E. Hoerl, R.W. Kennard, Ridge regression: Biased estimation for nonorthogonal problems, Technometrics 12 (1970) 55–67, https://doi.org/10.1080/00401706.1970.10488634.
[14] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, B. Li, Manipulating machine learning: Poisoning attacks and countermeasures for regression learning, in: 2018 IEEE Symposium on Security and Privacy, IEEE Computer Society, 2018, pp. 19–35, https://doi.org/10.1109/SP.2018.00057.
[15] H. Kikuchi, C. Hamanaga, H. Yasunaga, H. Matsui, H. Hashimoto, C. Fan, Privacy-preserving multiple linear regression of vertically partitioned real medical datasets, Journal of Information Processing Systems 26 (2018) 638–647, https://doi.org/10.2197/ipsjjip.26.638.
[16] R. Kohavi, B. Becker, Us census income dataset, 1996. URL:https://archive.ics.uci.edu/ml/datasets/census+income..
[17] X. Liu, R. Lu, J. Ma, L. Chen, B. Qin, Privacy-preserving patient-centric clinical decision support system on naïve bayesian classification, IEEE Journal of Biomedical and Health Informatics 20 (2016) 655–668, https://doi.org/10.1109/JBHI.2015.2407157.
[18] K. Mandal, 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, ACM, 2019, pp. 57–68, https://doi.org/10.1145/3338466.3358926.
[19] P. Mohapatra, S. Chakravarty, P.K. Dash, Microarray medical data classification using kernel ridge regression and modified cat swarm optimization based gene selection system, Swarm and Evolutionary Computation 28 (2016) 144–160, https://doi.org/10.1016/j.swevo.2016.02.002.
[20] P. Mohassel, Y. Zhang, Secureml: A system for scalable privacy-preserving machine learning, in: IEEE Symposium on Security and Privacy, IEEE Computer Society, 2017. pp. 19–38. doi: 10.1109/SP.2017.12..
[21] F. Musumeci, C. Rottondi, A. Nag, I. Macaluso, D. Zibar, M. Ruffini, M. Tornatore, An overview on application of machine learning techniques in optical networks, IEEE Communications Surveys & Tutorials 21 (2019) 1383–1408, https://doi.org/10.1109/COMST.2018.2880039.
[22] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: Stern, J. (Ed.), Advances in Cryptology – EUROCRYPT '99, Springer, 1999. pp. 223–238. doi: 10.1007/3-540-48910-X_16..
[23] C.Y.J. Peng, K.L. Lee, G.M. Ingersoll, An introduction to logistic regression analysis and reporting, The Journal of Educational Research 96 (2002) 3–14, https://doi.org/10.1080/00220670209598786.
[24] L. Petrella, V. Raponi, Joint estimation of conditional quantiles in multivariate linear regression models with an application to financial distress, Journal of Multivariate Analysis 173 (2019) 70–84, https://doi.org/10.1016/j.jmva.2019.02.008.
[25] G. Qiu, X. Gui, Y. Zhao, Privacy-preserving linear regression on distributed data by homomorphic encryption and data masking, IEEE Access 8 (2020) 107601–107613, https://doi.org/10.1109/ACCESS.2020.3000764.
[26] R. Quinlan, Auto mpg dataset, 1993. URL:https://archive.ics.uci.edu/ml/datasets/auto+mpg..
[27] V. Sigillito, Diabetes dataset, 1990. URL:https://datahub.io/machine-learning/diabetes..
[28] Y. Sun, M. Peng, Y. Zhou, Y. Huang, S. Mao, Application of machine learning in wireless networks: Key techniques and open issues, IEEE Communications Surveys & Tutorials 21 (2019) 3072–3108, https://doi.org/10.1109/COMST.2019.2924243.
[29] W.H. Wolberg, Breast cancer dataset, 1995. URL:https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original)..
[30] S. Wold, A. Ruhe, H. Wold, W. Dunn III, The collinearity problem in linear regression. the partial least squares (pls) approach to generalized inverses, SIAM Journal on Scientific and Statistical Computing 5 (1984) 735–743, https://doi.org/10.1137/0905052.
[31] C. Xie, O. Koyejo, I. Gupta, SLSGD: secure and efficient distributed on-device machine learning, in: European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2019, pp. 213–228, https://doi.org/10.1007/978-3-030-46147-8_13.
[32] G. Xu, H. Li, S. Liu, K. Yang, X. Lin, Verifynet: Secure and verifiable federated learning, IEEE Transactions on Information Forensics and Security 15 (2020) 911–926, https://doi.org/10.1109/TIFS.2019.2929409.
[33] Y. Yan, D. Tang, S. Zhan, R. Dai, J. Chen, N. Zhu, Low-rate dos attack detection based on improved logistic regression, in: 21st IEEE International Conference on High Performance Computing and Communications, IEEE, 2019, pp. 468–476, https://doi.org/10.1109/HPCC/SmartCity/DSS.2019.00076.
[34] H. Yang, W. He, Q. Zhou, H. Li, Efficient and secure outsourced linear regression, in: J. Vaidya, J. Li (Eds.), 18th International Conference on Algorithms and Architectures for Parallel Processing, Springer, 2018, pp. 89–102, https://doi.org/10.1007/978-3-030-05057-3_7.
[35] X. Yang, R. Lu, J. Shao, X. Tang, H. Yang, An efficient and privacy-preserving disease risk prediction scheme for e-healthcare, IEEE Internet of Things Journal 6 (2019) 3284–3297, https://doi.org/10.1109/JIOT.2018.2882224.
[36] Y. Yu, D. Yi, Z. Tang, D. Ou, A. Zeng, J. Li, A regression prediction model based on incremental iteration for big industrial data, in: 21st IEEE International Conference on High Performance Computing and Communications, IEEE, 2019, pp. 1886–1893, https://doi.org/10.1109/HPCC/SmartCity/DSS.2019.00260.
[37] Y. Zheng, R. Lu, B. Li, J. Shao, H. Yang, K.R. Choo, Efficient privacy-preserving data merging and skyline computation over multi-source encrypted data, Information Sciences 498 (2019) 91–105, https://doi.org/10.1016/j.ins.2019.05.055.