# Efficient and Secure Outsourced Linear Regression

Haomiao Yang(✉), Weichao He(✉), Qixian Zhou(✉), and Hongwei Li

School of Computer Science and Engineering and Center for Cyber Security, University of Electronic Science and Technology of China, Chengdu, China
{haomyang,hongweili}@uestc.edu.cn, wechaohe@163.com, qxzhou1010@qq.com

**Abstract.** The linear regression, as a classical machine learning algorithm, is often used to be a predictor. In the era of big data, the data owner can outsource their linear regression task and data to the cloud server, which has powerful calculation and storage resources. However, outsourcing data may break the privacy of the data. It is a well-known method to encrypt them prior to uploading to the cloud by using the homomorphic encryption (HE). Nevertheless, it is a difficult problem to apply the linear regression protocol in the encrypted domain. With this observation, we propose an efficient and secure linear regression protocol over outsourced encrypted data by using the vector HE, named *ESLR*, and in our protocol, we further present a privacy-preserving gradient descent method. Security analysis shows that our protocol can guarantee the confidentiality of data. And compared to the linear regression over plaintexts, our proposal can achieve almost the same accuracy and efficiency over ciphertexts.

**Keywords:** Machine learning · Homomorphic encryption
Linear regression · Gradient descent

## 1 Introduction

Predictive modeling is an essential tool in decision making processes in domains such as policy making, medicine, law enforcement, and finance. Considering a hospital would like to use a cloud service which provide predictive service to analyze the patient's condition so as to improve the quality of care and reduce costs. Due to ethical and legal requirements, the hospital might be restricted to use such service [3,4,12]. Like the hospital, many organizations are collecting ever-increasing data for mining to improve decision-making and productivity. However, they may have no powerful resources to deal with such large-scale data. To solve this problem, an attractive business model is that a service provider, which has powerful platforms and advanced analytic skills, provides such services. Organizations who need the calculation resource can outsource their computational tasks to such powerful service providers. However, because the data


J. Vaidya and J. Li (Eds.): ICA3PP 2018, LNCS 11336, pp. 89–102, 2018.
https://doi.org/10.1007/978-3-030-05057-3_7

may contain sensitive information, outsourcing data to public clouds directly raises privacy concerns.

In current implementations, the learning algorithm must see all user data in the clear in order to build the predictive model. In this paper, we consider whether the learning algorithm can operate in encrypted domains, thereby allowing users to retain control of their data. For medical data, this allows for a model to be built without affecting user privacy. For the book and movie preferences, letting users keep control of their data, can reduce the risk of future unexpected embarrassment in case of a data breach at the service provider.

Roughly speaking, there are three existing approaches to essure the privacy when the server mines the user data. The first lets users split their data among multiple servers by using secure multi-party computation [2,5,9]. These servers, then, run the learning algorithm using a distributed protocol. Privacy is assured as long as a majority of servers do not collude. The second is based on differential privacy protection, where the learning algorithm is executed over data containing noise [6,7,13]. And the third is based on homomorphic encryption, where the learning algorithm is executed over encrypted data [15].

Distributed linear regression is not suitable for outsourced model. In distributed linear regression, every party must take part in computation. Consequently, the secure multi-party computation may be inefficient. In addition, there may be a great loss of accuracy and can not fully guarantee the security of data by using the differential privacy protection. In this work, we choose homomorphic encryption for our privacy-preserving machine learning algorithm. As we know, homomorphic encryption (HE) allows operations on encrypted data, which provides a possible solution for linear regression over ciphertexts. In our work, we propose an efficient and secure linear regression protocol over encrypted data for outsourced environments, namely ESLR, where the cloud performs linear regression processing over encrypted data. The challenge is how to apply the linear regression algorithm over ciphertexts, while maintaining high accuracy and performance. To address these challenges, we exploit the vector HE (VHE) recently presented by Zhou and Wornell [17]. Unlike the fully HE (FHE), VHE only needs to support somewhat homomorphic encryption. As a result, it is much more efficient than many existing FHE schemes. For example, it is orders of magnitude faster than HELib [10], which is a very famous FHE implementation. Especially by designing ingeniously, VHE can be used in the privacy-preserving gradient descent. Different from existed works, our contributions are twofold as follows.

(1) Firstly, ESLR reconstructs linear regression clustering process in the domain of ciphertext by taking advantage of the vector encryption, which allows low computation and communication cost. What's more, we proposed a scheme that can apply privacy-preserving gradient descent method over ciphertext domain efficiently. To our best knowledges, it's very efficient for the optimization algorithm over encrypted data. Experiments shows that ESLR achieves almost the same accuracy compared to the plaintext algorithm.

(2) Secondly, security analysis demonstrates that ESLR achieves the confidentiality of data, ensuring the privacy of the data owner. In addition, we give the definition of loss function, which is needed for optimization over ciphertext domain.

This paper is organized as follows: The problem formulation is described in Sect. 2. The constructions of linear regression protocol are proposed in Sect. 3, followed by further discusses in Sect. 4. Then we give the security analysis and performance evaluation in Sects. 5 and 6, respectively. Finally, the conclusion is presented in Sect. 7.

## 2 Problem Statement

In this section, we give the problem statement, including system model and threat model, design goals, notations and preliminaries.

### 2.1 System Model and Threat Model

We give our system model concentrating on how to achieve secure liner regression over encrypted data in outsourced environments. As shown in Fig. 1, we proposed a classical outsourced system model, mainly consisting of two parties. The one is the data owner, and the other is the service provider. We primarily consider the service provider as an "honest-but-curious" server in our model. We assume the public matrix $\boldsymbol{H}$ and encrypted data $\boldsymbol{D}^{'}$ ($\boldsymbol{D}^{'}$ is the encryption of the data $\boldsymbol{D}$) have been outsourced to the cloud, and the confidentiality of the data

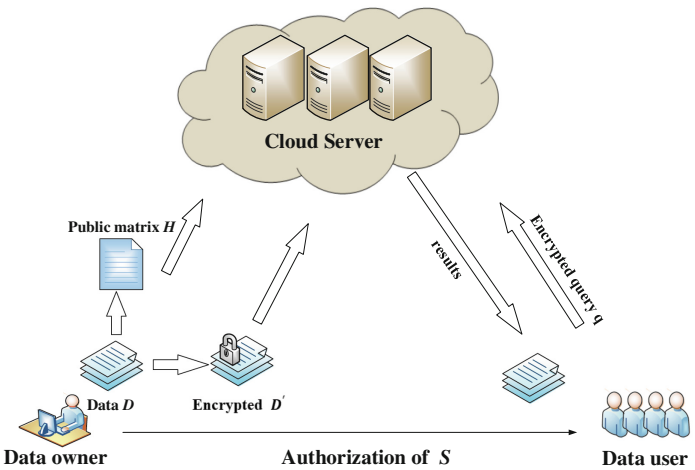



**Fig. 1.** System model

will be protected by the underlying encryption primitive. After that, the server will implement the regression algorithm based on $\boldsymbol{D}^{'}$. That is, the data owner outsources his encrypted data $\boldsymbol{D}^{'}$, and the service provider runs the proposed protocol over $\boldsymbol{D}^{'}$. Finally the service provider returns the predicted results to the data owner.

### 2.2 Design Goals

The overarching goal is to enable liner regression algorithm to be performed over encrypted data. What's more, for an efficient and secure liner regression protocol, we consider the following requirements to be necessary.

- Accuracy: Enable secure linear regression over encrypted data in outsourced environments and achieve high accuracy.
- Security: Protect privacy of linear regression process.
- Efficiency: Process large amount of data with practical performance.

### 2.3 Overview of Standard Linear Regression and Gradient Descent

In this section, we give a brief introduction about standard linear regression algorithm [16]. In statistics, linear regression equation is a regression analysis using least square function to find the relationship between one or more independent variables and dependent variables. This function is a linear combination of one or more model parameters called regression coefficients. Linear regression with only one independent variable is called simple regression, and it is called multiple regression with greater than one independent variable. Like all forms of regression analysis, linear regression also focuses on the probability distribution of $x$ and $y$. Given a random sample $(x_{i1}, x_{i2}, ..., x_{ip}, y_i)$, we have one hypothetical regression output $y_i$, and hypothetical regression inputs $x_{i1}, x_{i2}, ..., x_{ip}$. So a multivariate linear regression model is expressed as $y_i = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + b$.

For a data set $\boldsymbol{D} = [(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_n, y_n)]$, the goal of linear regression is to get the regression coefficients $\boldsymbol{\theta} = [w_1, w_2, \cdots, w_d, b]$ such that the loss function get the minimum value. We define the loss function as

$$J(\boldsymbol{\theta}) = (\frac{1}{2n}) \sum_{i=1}^{n} (\boldsymbol{\theta}^T \boldsymbol{x}_i^{'} - y_i)^2.$$

Further, we formulate the problem as Algorithm 1.

The gradient descent method [8] is one of the iterative methods, which can be used to solve the least squares problem. Gradient descent is one of the most commonly used methods in solving the model parameters of machine learning algorithm (unconstrained optimization problem). The other method commonly used is the least square method. When solving the minimum value of the loss function, we can get the minimum value of loss function and the model parameters through the gradient descent method.

**Algorithm 1.** Standard linear regression

**Input:** data set $\boldsymbol{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_n, y_n)\}$ and threshold $t$
**Output:** $\boldsymbol{\theta} = [w_1, w_2, \cdots, w_d, b]$
1: Define the loss function $J(\boldsymbol{\theta}) = (\frac{1}{2n}) \sum_{i=1}^{n} (\boldsymbol{\theta}^T \boldsymbol{x}_i - y_i)^2$
2: Generating the $\boldsymbol{\theta}^0$ randomly
3: **repeat**
4: $\quad \boldsymbol{\theta}^k = \boldsymbol{\theta}^{k-1} - \alpha \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$, where $\boldsymbol{\theta}^k$ is the value of $k^{th}$ iteration and $\alpha$ is the iteration step.
5: **until** $J(\boldsymbol{\theta}^{k+1}) - J(\boldsymbol{\theta}^k) < t$
6: **return** $\boldsymbol{\theta}$

### 2.4 Notations and Preliminaries

In this section, we review the preliminaries that are necessary for our work. First, we give notations used throughout the paper as illustrated in Table 1.

**Table 1.** Notations

| Notation | Meaning |
|---|---|
| $\lceil a \rfloor$ | To round $a$ to the nearest integer, for $a \in \mathbb{R}$ |
| $\lceil \boldsymbol{a} \rfloor$ | To round each entry $a_i$ to the nearest integer, for a vector $\boldsymbol{a} \in \mathbb{R}^n$ |
| $\boldsymbol{a}^*$ | To be a binary representation for a vector $\boldsymbol{a} \in \mathbb{Z}^n$ |

We outline the VHE scheme as suggested by Zhou and Wornell [17] that encrypts integer vectors to allow computation of arbitrary polynomials in the encrypted domain. For our purpose of ESLR, we only consider the fundamental operations below and more details are referred to [17].

- ***VHE.KG***$(\lambda)$: Input a security parameter $\lambda$, choose $l$, $m$, $n$, $p$, $q$, $w \in \mathbb{Z}$, and the distribution $\chi$ where $l = \lceil \log_2 (q-1) \rceil$, $w(p-1) < q$, $q \gg p$, and $m < n$, construct $\boldsymbol{S} = [\boldsymbol{I}, \boldsymbol{T}] \in \mathbb{Z}^{m \times n}$ with $\boldsymbol{I} \in \mathbb{Z}^{m \times m}$ as the identity matrix, and output the secret key $\boldsymbol{S}$ and the public parameters $\boldsymbol{Param} = (l, m, n, p, q, w, \chi)$.
- ***VHE.E***$(\boldsymbol{x}, \boldsymbol{S})$: Input a secret key $\boldsymbol{S} \in \mathbb{Z}^{m \times n}$ and a plaintext vector $\boldsymbol{x} \in \mathbb{Z}^m$, output a ciphertext $\boldsymbol{c} \in \mathbb{Z}^n$ that satisfies

$$\boldsymbol{S}\boldsymbol{c} = w\boldsymbol{x} + \boldsymbol{e}$$

where $w$ is a large integer, $\mid \boldsymbol{S} \mid \ll w$, and $\boldsymbol{e}$ is an error term with $|\boldsymbol{e}| < w/2$.

- ***VHE.D***$(\boldsymbol{c}, \boldsymbol{S})$: Input a ciphertext vector $\boldsymbol{c} \in \mathbb{Z}^n$ and a secret key $\boldsymbol{S} \in \mathbb{Z}^{m \times n}$, output a plaintext $\boldsymbol{x} \in \mathbb{Z}^m$ that satisfies $\boldsymbol{x} = \lceil \boldsymbol{S}\boldsymbol{c}/w \rfloor$.

For the VHE scheme, the key switching is an important operation in the encrypted domain. Given two secret keys $\boldsymbol{S} \in \mathbb{Z}^{m \times n}$ and $\boldsymbol{S}' \in \mathbb{Z}^{m \times n'}$, and

the ciphertext $\boldsymbol{c} \in \mathbb{Z}^n$ which decrypts to the plaintext $\boldsymbol{x} \in \mathbb{Z}^m$ with $\boldsymbol{S}$, we calculate a matrix $\boldsymbol{M} \in \mathbb{Z}^{n^{'} \times nl}$ producing a new ciphertext $\boldsymbol{c}^{'} \in \mathbb{Z}^{n^{'}}$ so as to decrypt $\boldsymbol{c}^{'}$ to the same $\boldsymbol{x}$ with $\boldsymbol{S}^{'}$. In specific, this key switching task can be divided two steps: $\boldsymbol{M} \leftarrow \boldsymbol{VHE.KSM}(\boldsymbol{S}, \boldsymbol{S}^{'})$ and $\boldsymbol{c}^{'} \leftarrow \boldsymbol{VHE.KS}(\boldsymbol{M,c})$.

Furthermore, as inferred by [17], for the plaintext $\boldsymbol{x}$, the ciphertext $\boldsymbol{c}$, and the key-switching matrix $\boldsymbol{M}$, the following equation holds.

$$\boldsymbol{c} = \boldsymbol{M}(w\boldsymbol{x})^*$$

In addition, it is obvious that VHE supports the operation of the addition in ciphertexts domain as

$$\boldsymbol{S}(\boldsymbol{c}_1 + \boldsymbol{c}_2 + \cdots + \boldsymbol{c}_n) = w(\boldsymbol{x}_1 + \boldsymbol{x}_2 + \cdots + \boldsymbol{x}_n) + \boldsymbol{e}.$$

### 2.5 Privacy-Preserving Inner Product

In this section, we present a new technique of computing the inner product of two vectors. For simplication, we can assume that there are two vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ which are encrypted to $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ using the vector homomorphic encryption of VHE. The challenge is how to calculate the inner product on ciphertext domain.

To tackle the problem, a matrix $\boldsymbol{H}$ is essential to be calculated. By solving equation $\boldsymbol{AM} = \boldsymbol{I}^*$, we have a matrix $\boldsymbol{A}$. Then we can get the matrix $\boldsymbol{H}$ from $\boldsymbol{H} = \boldsymbol{A}^T\boldsymbol{A}$. We can prove that

$$\boldsymbol{c}^T\boldsymbol{H}\,\boldsymbol{c} = w^2\boldsymbol{x}^T\boldsymbol{x}.$$

Hence, we can calculate the inner product in ciphertex domain, and will later discuss the security of this method.

## 3 Proposed Protocol

In this section, we will propose the protocol for linear regression over encrypted items in outsourced environments using VHE.

### 3.1 Reformulating the Problem

In this section, we give a brief introduction about our problem again. We supposed that the data owner owns a database $\boldsymbol{D}$ that can be thought to be a big table of $n$ records $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$. The record $\boldsymbol{x}_i = [x_{i_1} \cdots x_{i_m}]$ includes $m$ attributes. Because the resources of the data owner is limited, so the data owner encrypts his database $\boldsymbol{D}$ record-wise, and then outsources the encrypted database $\boldsymbol{D}^{'}$ to the cloud. After that, the service provider will apply the linear regression over encrypted data sets, and return back the results to the data owner. In this protocol, the service provider know nothing to the plaintext.

### 3.2 Linear Regression Over VHE

With the preparatory work ahead, we discuss the problem of regression over encrypted data firstly. In order to make our protocol faster and easier, We only consider the security of data properties. Supposed dataset $\boldsymbol{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_n, y_n)\}$ which is only known by data owner are encrypted to be $\boldsymbol{D}' = \{(\boldsymbol{c}_1, y_1), (\boldsymbol{c}_2, y_2), \cdots, (\boldsymbol{c}_n, y_n)\}$. The relation between plaintext and ciphertext satisfies $\boldsymbol{S}\boldsymbol{c}_i = w\boldsymbol{x}_i + \boldsymbol{e}_i$ where $i = 1, 2, \cdots, n$. When the service provider get the encrypted data sets $\boldsymbol{D}'$ from the data owner, he will apply linear regression protocol over $\boldsymbol{D}'$. The whole process is divided three phases: **Preparation**, **Regression**, and **BackResults**.

- ***Preparation***$(\boldsymbol{D}, \lambda)$. The security parameter $\lambda$ and the data sets $\boldsymbol{D}$ is taken as the input, and the data owner generates a secret key $\boldsymbol{S}$ and a key-switch Matrix $\boldsymbol{M}$ for every record which satisfies the following equation.

$$\boldsymbol{c} = \boldsymbol{M}(w\boldsymbol{x})^*,$$

where $\boldsymbol{c}$ is the ciphertext of $\boldsymbol{x}$. The data owner need to calculate the key-switch matrix $M$ only once and the data owner can use the key-switch $M$ to encrypted data sets $\boldsymbol{x}$. As we know, the scheme of VHE cost most is key-switch. If we use the same key-switch $M$ to encrypt data, We can save a lot of overhead on encryption. Then, the data owner need to calculate the matrix $H$, which is used to define the loss function over encrypted data. As we know, the following equation holds.

$$w\boldsymbol{x} = \boldsymbol{I}^*(w\boldsymbol{x})^*.$$

The data owner solve a matrix equation which satisfies:

$$\boldsymbol{A}\boldsymbol{M} = \boldsymbol{I}^*.$$

Then, the data owner obtains the matrix $\boldsymbol{A}$ from the equation. Finally, the data owner can get the matrix $\boldsymbol{H}$ as

$$\boldsymbol{H} = \boldsymbol{A}^T\boldsymbol{A}$$

Finally, the data owner upload the encrypted data set $\boldsymbol{D}'$ and the matrix $\boldsymbol{H}$ to the service provider.

- ***Regression***$(\boldsymbol{D}', \boldsymbol{H})$. The service provider get the encrypted data set $\boldsymbol{D}' = \{(\boldsymbol{c}_1, y_1), (\boldsymbol{c}_2, y_2), \cdots, (\boldsymbol{c}_n, y_n)\}$ and the matrix $\boldsymbol{H}$ from the data owner and apply the regression algorithm, which includes the steps as below:
  (1) Generate a vector $\boldsymbol{\theta}'$ randomly and choose a threshold $t$.
  (2) Define the loss function over encrypted data as

$$J'(\boldsymbol{\theta}') = (\frac{1}{2n})\sum\nolimits_{i=1}^{n}(\frac{1}{w^2}\boldsymbol{\theta}'^T\boldsymbol{H}\,\boldsymbol{c}_i - y_i)^2.$$

(3) Upload the $\boldsymbol{\theta}'$ based on gradient descent method as below:

$$\boldsymbol{\theta}'^{k} = \boldsymbol{\theta}'^{k-1} - \alpha \frac{\partial J'(\boldsymbol{\theta}')}{\partial \boldsymbol{\theta}'},$$

where $\boldsymbol{\theta}'^{k}$ is the value of the $k^{th}$ iteration.

(4) Repeat *step (3)* until the value of the loss function satisfies the condition as below:

$$|J'(\boldsymbol{\theta}'^{k}) - J'(\boldsymbol{\theta}'^{k-1})| < t.$$

– ***BackResults***$(\boldsymbol{\theta}')$. From ***Regression*** the cloud will get the encrypted parameters. Then, the cloud return it back to the data owner.

## 4 Discussion

We have shown how to achieve a basic protocol for linear regression over encrypted data in outsourced environment. In this section, we will give the correctness analysis of our protocol and give a brief introduction about how to use the encrypted results.

### 4.1 Loss Function over Encrypted Data

In this section, we introduce the correctness of loss function over encrypted data, and verify that the following equation holds.

$$\begin{aligned} J'(\boldsymbol{\theta}') &= (\frac{1}{2n}) \sum_{i=1}^{n} (\frac{1}{w^2} \boldsymbol{\theta}'^{T} \boldsymbol{H} \boldsymbol{c}_i - y_i)^2 \\ &= (\frac{1}{2n}) \sum_{i=1}^{n} (\frac{1}{w^2} w^2 \boldsymbol{\theta}^{T} \boldsymbol{c}_i - y_i)^2 \\ &= (\frac{1}{2n}) \sum_{i=1}^{n} (\boldsymbol{\theta}^{T} \boldsymbol{c}_i - y_i)^2 \\ &= J(\boldsymbol{\theta}) \end{aligned}$$

As we can see, the loss function on the encrypted data is equal to the loss function on the plaintext.

### 4.2 Encrypted Parameters

In this section, we will discuss the relationship between encrypted parameters $\boldsymbol{\theta}'$ and encrypted data. First of all, We analysis loss function of plaintext. The loss function of plaintext is shown as follow:

$$J(\boldsymbol{\theta}) = (\frac{1}{2n}) \sum\nolimits_{i=1}^{n} (\boldsymbol{\theta}^{T} \boldsymbol{x}_i - y_i)^2$$

Gradient descent is one of the most commonly used methods in solving the model parameters. When solving the minimum value of the loss function, we can get the minimum value of loss function and the model parameters. $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_d]$ where the iterative equation is given as below:

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \alpha \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix} := \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix} - \frac{\alpha}{n} \begin{bmatrix} \sum_{i=1}^{n} (\boldsymbol{\theta}^T \boldsymbol{x}_i - y_i) * x_{i1} \\ \sum_{i=1}^{n} (\boldsymbol{\theta}^T \boldsymbol{x}_i - y_i) * x_{i2} \\ \vdots \\ \sum_{i=1}^{n} (\boldsymbol{\theta}^T \boldsymbol{x}_i - y_i) * x_{id} \end{bmatrix}$$

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix} := \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix} - \frac{\alpha}{n} \sum_{i=1}^{n} (\boldsymbol{\theta}^T \boldsymbol{x}_i - y_i) \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{bmatrix}$$

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \frac{\alpha}{n} \sum_{i=1}^{n} (\boldsymbol{\theta}^T \boldsymbol{x}_i - y_i) \boldsymbol{x}_i,$$

where $\alpha$ is the iteration step. Note that $\boldsymbol{\theta}$ is a linear combination of $\boldsymbol{x}_i^{'}$ when the initial value is set to the vector $\mathbf{0}$. Linear combination is supported by Vector Homomorphic Encryption, and thus we can get the results on the encrypted domain.

## 5 Security Analysis

In this section, we give the security analysis for ESLR, focusing on the encrypted database $\boldsymbol{D}^{'} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_n\}$ and the matrix $\boldsymbol{H}$. The honest-but-curious cloud server could not threat the privacy of the data owner, i.e., the cloud could not recover plaintexts database $\boldsymbol{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}$.

First of all, $\boldsymbol{c}_i$ is the ciphertext of $\boldsymbol{x}_i$ by the encryption of VHE, for $i = 1, 2, \cdots, n$. For convenience, we omit the subscripts, denoting as $\boldsymbol{c} = \boldsymbol{VHE.E}(\boldsymbol{x}, \boldsymbol{S})$, where $\boldsymbol{S}$ is the secret key. Therefore, we can ensure the confidentiality of $\boldsymbol{x}$, only if the encryption scheme VHE is secure and the secret key $\boldsymbol{S}$ is not known by the cloud. Of course, we may suppose that the secret key $\boldsymbol{S}$ is stored privately by the data owner, and thus the cloud could not get it. Hence, we would focus on the security of VHE.

As shown in [17], the security of VHE could reduce to the problem of the learning with errors (*LWE*). It is well known the *LWE* problem is as hard to solve as several worst-case lattice problems [14]. As a resut, the intracibility of *LWE* assures the security of VHE.

However, in order to evaluate the distance of two ciphertexts vectors, we introduce a special matrix $\boldsymbol{H}$. It is natural to consider if $\boldsymbol{H}$ may bring certain

unknown privacy risk. For example, on one hand, to calculate $\boldsymbol{H}$, we first solve the equation $\boldsymbol{I}^* = \boldsymbol{AM}$ to obtain $\boldsymbol{A}$, then compute $\boldsymbol{H} = \boldsymbol{A}^T\boldsymbol{A}$ to get $\boldsymbol{H}$. On the other hand, according to VHE, for the ciphertext $\boldsymbol{c}$ and the plaintext $\boldsymbol{x}$, $\boldsymbol{c} = \boldsymbol{M}(w\mathbf{x})^*$ holds. As known, the cloud has $\boldsymbol{H}$ and $\boldsymbol{c}$. If the cloud combines the equations as follows, it seems that the cloud could recover the plaintext $\boldsymbol{x}$.

$$\begin{cases} \boldsymbol{H} = \boldsymbol{A}^T\mathbf{A} \\ \boldsymbol{I}^* = \boldsymbol{AM} \\ \boldsymbol{c} = \boldsymbol{M}(w\boldsymbol{x}) \end{cases}$$

In the following, We would give positive answer about the challenge. The analysis demonstrates that the cloud could not yet recover the plaintext $\boldsymbol{x}$ from the ciphertext $\boldsymbol{c}$ by exploiting $\boldsymbol{H}$.

As is known, for a random orthogonal matrix $\boldsymbol{Q}$, satisfying the relation $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$, where $\boldsymbol{I}$ is an identity matrix, we have

$$\begin{aligned} \boldsymbol{H} &= \boldsymbol{A}^T\boldsymbol{A} \\ &= \boldsymbol{A}^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{A} \\ &= \boldsymbol{A}^T\boldsymbol{I}\boldsymbol{A} \\ &= \boldsymbol{H} \end{aligned}$$

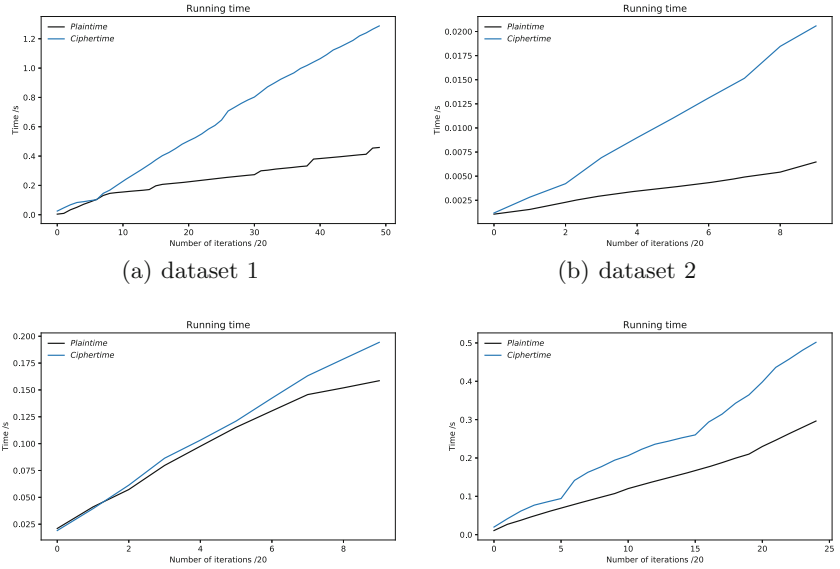
It is clear that the equation $\boldsymbol{H} = \boldsymbol{A}^T\boldsymbol{A}$ has infinite solutions for $\boldsymbol{A}$ since $\boldsymbol{Q}$ is randomly chosen. Therefore, the cloud could not extract the matrix $\boldsymbol{A}$ from the Norm-matrix $\boldsymbol{H}$. Futhermore, without knowing $\boldsymbol{A}$, the cloud could not yet get $\boldsymbol{M}$. And the cloud could not recover the plaintext $\boldsymbol{x}$ from the ciphertext $\boldsymbol{c}$. As a result, we achieve the privacy of the database $\boldsymbol{D}$.

## 6 Performance Evaluation

In this section, we evaluate the proposed linear regression protocol. Our data sets come from the UCI repository [1], and the experiment environment includes a data owner and a service provider. Python language is used on a Window 10 machine with i3-4130 CPU @1.40 GHz and 4 GB RAM for a user, and the server is a Linux machine with an Intel Xeon E5-2430 v2 CPU @2.5 GHz and 16 GB RAM running Ubuntu 14.04 LTS. The user acts as a data owner and a data user, and the server acts as a service provider. In the following, we will conduct the simulation experiments in terms of the time cost, accuracy, and communication overhead.

### 6.1 Time Cost and Accuracy

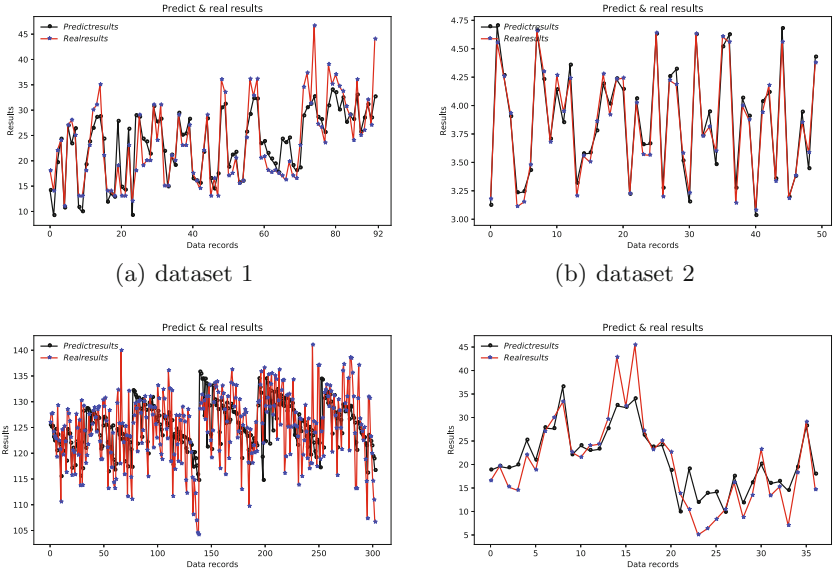
Firstly, we evaluate the time cost by the comparison of running time between plaintext and ciphertext. As illustrated in Fig. 2, we choose 4 data sets to verify our protocol from the UCI repository, and can see that the linear regression




(a) dataset 1 (b) dataset 2

(c) dataset 3 (d) dataset 4

**Fig. 2.** Comparison of running time between plaintext and ciphertext




(a) dataset 1 (b) dataset 2

(c) dataset 3 (d) dataset 4

**Fig. 3.** Comparison between real results and predicted results in encrypted domain

on ciphertext is a little slower than that on plaintext. However, the result is acceptable, and it has almost the same results for the data sets between the plaintext and the ciphertext.

Then, we show the comparison of accuracy between the real results and the predicted results of the four different data sets in the ciphertext domain. As illustrated in Fig. 3, we can see that the predicted results almost coincide the actual results in the ciphertext domain. Furthermore, we choose the Mean Squared Error, Root Mean Squard Error, Mean Absolute Error (MAE) and R Squared (R-S), as the indexes of linear regression to evaluate our model. As seen in Table 2, compared to results in the plaintext domain, our protocol has almost achieved the same prediction performance. This shows that our model has a good performance on the ciphertext domain.

**Table 2.** Clustering time and iterations

| Data | MSE | RMSE | MAE | R-S |
|---|---|---|---|---|
| Plaintext on dataset 1 | 9.964 | 3.156 | 2.427 | 0.838 |
| Ciphertext on dataset 1 | 15.253 | 3.905 | 2.873 | 0.768 |
| Plaintext on dataset 2 | 0.007 | 0.081 | 0.068 | 0.972 |
| Ciphertext on dataset 2 | 0.025 | 0.157 | 0.133 | 0.895 |
| Plaintext on dataset 3 | 23.999 | 4.899 | 3.790 | 0.497 |
| Ciphertext on dataset 3 | 24.987 | 4.999 | 3.897 | 0.475 |
| Plaintext on dataset 4 | 19.134 | 4.374 | 3.499 | 0.782 |
| Ciphertext on dataset 4 | 20.134 | 4.564 | 3.619 | 0.768 |

### 6.2 Communication Cost

In this section, we will discuss the communication cost of our protocol. In our protocol, the communication cost mainly come from ciphertext and the matrix $\boldsymbol{H}$ which is used to define the loss function. Firstly, for $n$ records and every record have $m$ dimensions, it will produce $\mathcal{O}(m(n+1))$ communication traffic overhead when the data items are encrypted. Secondly, it will generate $\mathcal{O}((n+1)^2)$ communication traffic overhead for matrix $\boldsymbol{H}$. That means that it will produce $\mathcal{O}(m+n+1)(n+1)$ communication traffic overhead totally on encrypted domain. On the other hand, the complexity of plaintext stage is $\mathcal{O}(mn)$ for the same data sets. In fact, m is always far greater than n because of dimension disaster problem [11]. So communication traffic overhead between plaintext and ciphertext is almost same when m is far greater than n and m is big enough.

## 7 Conclusion

In this paper we have proposed an efficient and secure linear regression protocol over encrypted data using the vector homomorphic encryption. Especially, we have given a good solution to the challenging problem of privacy-preserving gradient descent method. Performance evaluation shows that it has high accuracy and low computation and communication cost. As we know, many machine learning algorithm base on gradient descent method. In the future, we will use this method on other machine learning algorithms.

**Acknowledgement.** Our work is supported by of the National Key Research and Development Program of China (2017YFB0802003), the National Natural Science Foundation of China (U1633114) and the Sichuan Science and Technology Program (2018GZ0202).

## References

1. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
2. Ben-David, A., Nisan, N., Pinkas, B.: FairplayMP: a system for securemulti-party computation. In: Proceedings of the 15th ACM Conference on Computer and Communications Security, pp. 257–266. ACM (2008)
3. Dankar, F.K., El Emam, K.: The application of differential privacy to healthdata. In: Proceedings of the 2012 Joint EDBT/ICDT Workshops, pp. 158–166. ACM (2012)
4. Centers for Disease Control and Prevention, et al.: HIPAA privacy rule and public health. guidance from CDC and the us department of health and human services. MMWR Morb. Mortal. Wkly. Rep. **52**(Suppl. 1), 1–17 (2003)
5. Du, W., Atallah, M.J.: Secure multi-party computation problems and their applications: a review and open problems. In: Proceedings of the 2001 Workshop on New Security Paradigms, pp. 13–22. ACM (2001)
6. Dwork, C.: Differential privacy: a survey of results. In: Agrawal, M., Du, D., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 1–19. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79228-4_1
7. Dwork, C., Roth, A., et al.: The algorithmic foundations of differential privacy. Found. Trends® Theor. Comput. Sci. **9**(3–4), 211–407 (2014)
8. Fletcher, R., Powell, M.J.: A rapidly convergent descent method for minimization. Comput. J. **6**(2), 163–168 (1963)
9. Goldreich, O.: Secure multi-party computation. Manuscript. Preliminary version, pp. 86–97 (1998)
10. Halevi, S., Shoup, V.: Helib (2014). Retrieved from HELib: https://github.com.shaih/HElib
11. Huang, Z.: Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Min. Knowl. Discov. **2**(3), 283–304 (1998)
12. Lee, L.M., Gostin, L.O.: Ethical collection, storage, and use of public health data: a proposal for a national privacy protection. Jama **302**(1), 82–84 (2009)
13. McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: 48th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2007, pp. 94–103. IEEE (2007)

14. Regev, O.: On lattices, learning with errors, random linear codes, andcryptography. J. ACM **56**(6), 1–40 (2009)
15. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_2
16. Wold, S., Ruhe, A., Wold, H., Dunn III, W.: The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses. SIAM J. Sci. Stat. Comput. **5**(3), 735–743 (1984)
17. Zhou, H., Wornell, G.: Efficient homomorphic encryption on integer vectors and its applications. In: Information Theory and Applications Workshop (ITA), 2014, pp. 1–9. IEEE (2014)