IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 28, 2020, accepted June 4, 2020, date of publication June 8, 2020, date of current version June 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3000764

Privacy-Preserving Linear Regression on Distributed Data by Homomorphic Encryption and Data Masking

GUOWEI QIU[1,2], XIAOLIN GUI[1,2], AND YINGLIANG ZHAO[1]
[1]School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China
[2]Key Laboratory of Computer Network of Shaanxi Province, Xi'an 710049, China

Corresponding authors: Guowei Qiu (qwqiu@mail.xjtu.edu.cn) and Xiaolin Gui (xlgui@mail.xjtu.edu.cn)

This work was supported in part by the National Key Research and Development Project under Grant 2018YFB1800304, and in part by the Key Research and Development Project of Shaanxi Province under Grant 2017ZDXM-GY-011 and Grant 2019GY-005.

ABSTRACT Linear regression is a basic method that models the relationship between an outcome value and some explanatory values using a linear function. Traditionally, this method is conducted on a clear dataset provided by one data owner. However, in today's ever-increasingly digital world, the data for regression analysis are likely distributed among multiple parties and even contain sensitive information about the data owners. In this case, data owners are not willing to share their data unless data privacy is guaranteed. In this paper, we propose a novel protocol for conducting privacy-preserving linear regression (PPLR) on horizontally partitioned data. Our system architecture includes multiple clients and two noncolluding servers. In our protocol, each client submits its data in encrypted form to a server, and two servers collaboratively determine the regression model on pooled data without learning its contents. We construct our protocol with Paillier homomorphic encryption and a new data masking technique. This data masking technique can perturb data by multiplying a rational number while the data are encrypted. Due to the use of the data masking technique, the efficiency of our protocol is greatly improved. We provide an error bound of the protocol and prove it rigorously. We also provide security analysis of the protocol. Finally, we implement our system in C++ and Java, and then we evaluate our protocol using real datasets provided by UCI. The experiments show our protocol is one of the most effective approaches to date and has negligible errors compared with performing linear regression on clear data.

INDEX TERMS Privacy-preserving regression, linear regression, homomorphic encryption, data masking, multiplicative perturbation.

I. INTRODUCTION

With the extensive use of computer technology, many institutions and individuals have accumulated a large amount of data. This causes an increasing demand on collaborative mining among multiple data owners. Because many data owners lack professional skills and computing resources for data mining, they have to outsource this work to a service provider. If all data owners agree to submit their clear data to a service provider for mining information of common interested on the pooled data, the data mining over distributed data can be easily accomplished. However, in many cases, data owners are unwilling to share their data because some sensitive information of data owners may be contained in the data.

For example, some medical institutions want to evaluate the effect of different treatment plans for patients with a particular disease. Each treatment plan could be combination of several drugs in specific proportions. Under the premise that patients' privacy is well protected, all the medical institutions are willing to contribute their data for analysis on the combined dataset. Another example concerns personal credit evaluation. Personal credit is the basis of some individual economic activity, such as a mortgage loan, a car loan. Credit evaluations require data provided by banks, insurance companies, e-commerce firms, judicial offices, etc. All of these institutions have their own data usage policy. It is almost impossible for a third party to collect data from all these institutions in clear text form. This make comprehensive evaluation of personal credit become a difficult problem. These examples belong to distributed privacy-preserving data

The associate editor coordinating the review of this manuscript and approving it for publication was Sedat Akleylek.

mining. As we enter the age of big data, such demanding scenarios will become increasingly common.

In this paper, we focus on privacy-preserving linear regression (PPLR) on a distributed dataset. The approach addresses how several servers (service providers) conduct linear regression, where the data are divided among multiple clients (data owners), and any information about clients' data cannot be learned by any server. The PPLR is a problem that belongs to privacy-preserving data mining (PPDM) [1]. Thanks to the efforts of many experts in PPDM, now we have at least four techniques to contend with the PPLR problem. The first method is based on data perturbation [2]–[4]. In this method, each client first masks data with mathematical transforms, such as rotation, translation and swap. Then, a service provider performs a mining algorithm on the pooled dataset. High efficiency is the prominent advantage of this method. The downside is that a perturbation that is too large leads to a loss of data utility, wheres too small of a perturbation results in a privacy breach. The second method is linear secret sharing [5]–[7]. This method transforms each data element into the sum of several random parts. Each part is sent to a different server. The servers collaboratively perform a mining algorithm using these data parts. Because each server cannot expose its data part to other servers, the arithmetic operations become very complex [8]. The data privacy is assured unless the number of colluded servers exceeds a threshold. The third method is the garbled circuit [9]–[11]. In general, this method has two parties – Alice and Bob. Alice has data x and Bob has data y. They want to compute $f(a, b)$, where f is a public function. By using the garbled circuit technique, Alice and Bob can obtain $f(a, b)$ without learning anything about the other party's data. This method cannot be applied to the PPLR problem directly because Alice and Bob should play the role of service provider and they do not have any input data. Therefore, we must combine the garbled circuit method with other techniques to ensure the clients' privacy is not compromised. Garbled circuits are less efficient than other methods when solving the same problem. The fourth method is homomorphic encryption [12], which is a kind of public key encryption. The homomorphic encryption enables a server to perform computation directly on encrypted data without accessing a secret key, and the results of the computations remain in encrypted form. An attractive idea about using homomorphic encryption for the PPDM problem is one in which clients submit encrypted data to the server and the server conduct data mining on encrypted data directly. Until now, practical homomorphic encryption, known as partial homomorphic encryptions [13], [14], have only supported one kind of arithmetic operation - addition or multiplication. In contrast, fully homomorphic encryption [15], [16] supports both addition and multiplication. Although a major breakthrough had been made in fully homomorphic encryption, it is still far less efficient than other techniques. Most researchers still choose partial homomorphic encryption in their research. In this paper, we develop our approach also based on partial homomorphic encryption.

Privacy-preserving linear regression or ridge regression over distributed datasets has received considerable attention in recent years. Some approaches have been proposed. Each of them used one or more the aforementioned techniques.

In 2005, Karr *et al.* [22] proposed a protocol for PPLR based on secure summation. Their approach is highly efficient but does not conceal important intermediate values such as the covariance matrix. In 2011, Hall *et al.* [6] proposed a protocol for PPLR based on secret sharing and homomorphic encryption. Different from Karr *et al.*'s approach, their protocol conceals the covariance matrix. However, the iteration they used for computing the inverse matrix is inefficient. In 2017, Mohassel *et al.* [7] designed a computing framework for PPDM based on two-party secret sharing and provided fixed-point multiplication with O(1) complexity. Their approach is highly efficient at the phase of online computing but still requires expensive offline precomputation to prepare for online multiplication.

In 2013, Nikolaenko *et al.* [10] proposed a protocol to address privacy-preserving ridge regression on distributed records. They used additive homomorphic encryption to generate the covariance matrix and right-hand-side vector, and then used garbled circuit to determine the final result. Experiments show their hybrid protocol is more efficient than Hall *et al.*'s protocol. Gascón *et al.* [17] extended Nikolaenko *et al.*'s protocol to vertically partitioned datasets in 2017. They designed a secure inner product protocol for data aggregation and garbled circuits of the conjugate gradient decent algorithm for solving linear equations.

Hu *et al.* [18] in 2017 proposed a protocol for privacy-preserving ridge regression that was completely based on additive homomorphic encryption. They design a packed secure multiplication protocol, which is used to construct the secure Gauss elimination and Jacobi iteration algorithm. These algorithms are used in solving linear equations. In 2018, Chen *et al.* [19] designed a protocol for privacy-preserving ridge regression. In their protocol, multiplicative and additive homomorphic encryption are used together to implement data aggregation and solving equations. Their experiments show that the protocol based on homomorphic encryption alone is more efficient than the hybrid protocol using garbled circuits.

In 2018, Giacomelli *et al.* [20] proposed a protocol based on homomorphic encryption and a data masking technique. By using homomorphic encryption, they mask data in encrypted form, and then decrypt it and solve the masked linear equations. Their approach is similar to ours, but our data masking method is different. Data masking can provide a sufficient level of security and greatly reduces the computational cost.

A. OUR CONTRIBUTIONS

Our work in this paper follows the research line of homomorphic encryption. Our system framework is shown in figure 1. The Evaluator and Crypto-Service provider (CSP) are two servers, and multiple clients are data owners. In our protocol,

each client submits its data in encrypted form to the Evaluator, and the Evaluator cooperates with CSP to determine the data model on the pooled data without learning its contents. Our protocol is based on a semi-honest model, and the two servers do not collude with each other.

Our contribution can be summarized as follows:

- We propose a new protocol for conducting privacy-preserving linear regression on horizontally partitioned data. By combining homomorphic encryption and a data masking technique, two servers determine the data model and cannot learn any information about the input data. We design a new data masking technique that makes our protocol simple and effective. Furthermore, each client can be offline after submitting its data because the client does not participate in any subsequent computations.
- We derived an error bound of the protocol and prove it with rigorous matrix theory. Furthermore, we conduct a security analysis that shows our protocol is secure in a statistical sense.
- We implement our protocol in C++ and Java language, and then evaluate its accuracy and efficiency by performing experiments on real datasets provided by UCI. We also compare our protocol with the state-of-the-art solution. The experiments show our protocol is not only one of the most effective approaches to date but also sufficiently accurate with only negligible error.

II. PROBLEM STATEMENT

We first give some notations used in this paper.

- Bold uppercase – matrix (e.g., $\mathbf{X}$, $\mathbf{A}$, $\mathbf{U}$, $\mathbf{V}$)
- Bold lowercase – vector (e.g., $\mathbf{y}$, $\mathbf{b}$, $\mathbf{x}_i$, $\hat{\mathbf{b}}_1$, $\hat{\mathbf{b}}_2$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$)
- Normal lowercase – real number or integer (e.g., a_{ij}, y_i)
- $\lfloor f \rfloor$ – round a real number f down to the nearest integer
- $\lceil f \rceil$ – round a real number f up to the nearest integer
- $\lfloor f \rceil$ – round f to the integer that is the floor or ceiling of f
- (p, q)–bit length of the integral and fractional part of a fixed-point number
- pk, sk – public key and private key
- $E(\cdot)$, $D(\cdot)$ – encryption and decryption function
- $E(\mathbf{A})$ – a matrix that consists of elements $E(a_{ij})$
- $E(\mathbf{b})$ – a vector that consists of elements $E(b_i)$

We use x_{ij} to denote the (i, j) element of $\mathbf{X}$ or $\mathbf{x}_i$. Similarly, α_i is an element of $\boldsymbol{\alpha}$, etc.

A. ARCHITECTURE AND OUR GOAL

Our system architecture consists of three entities – a CSP, an Evaluator and r clients, as shown in Figure 1. Suppose there are m data pairs $(\mathbf{x}_i, y_i)$ used in regression, where $1 \leq i \leq m$, $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$.

- *Clients*: Each client holds part of the data pairs in our system. $\mathbf{x}_i$ includes d explanatory values, and its corresponding output is y_i. $(\mathbf{x}_i, y_i)$ contains some sensitive information of the client.

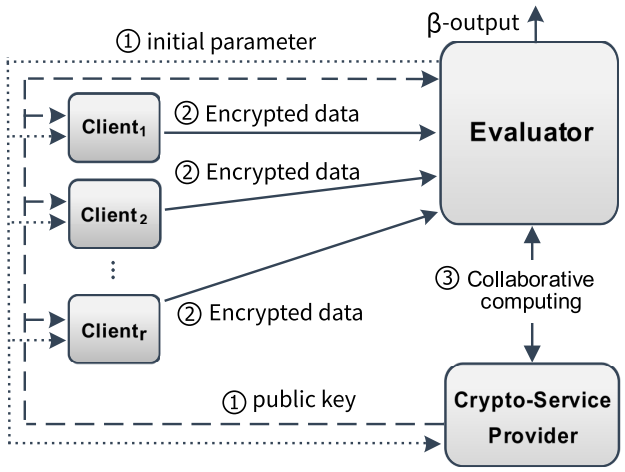

FIGURE 1. Distributed data analysis system.

- *CSP*: The CSP generates public/private key pair of homomorphic encryption and sends the public key to the Evaluator and clients in the initialization phase.
- *Evaluator*: The Evaluator sets up parameters for our protocol in the initialization phase and performs privacy-preserving regression on the combined dataset of all clients with the help of CSP. Finally, it outputs the result of regression analysis.

We call d the number of data features in this paper.

Our goals are threefold: to build a model without revealing any input data or important intermediate value to servers, to free clients from complicated data mining tasks, and to design a highly efficient protocol that is suitable for practical use. Here, the *important intermediate values* are the covariance matrix and its corresponding right-hand-side vector in the linear regression (ref. Section III. A). Some research [6], [23] has shown that revealing the covariance matrix or the corresponding right-hand-side vector may be a cause of privacy leakage.

We assume that all parties are honest-but-curious and the Evaluator and CSP do not collude with each other. This means all participators follow the instruction of protocol but try to learn privacy information through observing the protocol execution.

B. INPUT DATA DISTRIBUTION

In our system, input data are horizontally partitioned among r clients. This means each data $(\mathbf{x}_i, y_i)$ cannot be split and belongs to one client. Therefore, we can find some integers l_k that satisfy $0 = l_0 < l_1 < \cdots < l_r = m$, and client k $(1 \leq k \leq r)$ holds the data

$$(\mathbf{x}_{l_{k-1}+1}, y_{l_{k-1}+1}), (\mathbf{x}_{l_{k-1}+2}, y_{l_{k-1}+2}), \cdots, (\mathbf{x}_{l_k}, y_{l_k})$$

We can denote the client k's data as matrix $\mathbf{X}_k$ and vector $\mathbf{y}_k$ as follows.

$$\mathbf{X}_k = [\mathbf{x}_{l_{k-1}+1}, \mathbf{x}_{l_{k-1}+2}, \cdots, \mathbf{x}_{l_k}]^\mathsf{T}$$
$$\mathbf{y}_k = [y_{l_{k-1}+1}, y_{l_{k-1}+2}, \cdots, y_{l_k}]^\mathsf{T}$$

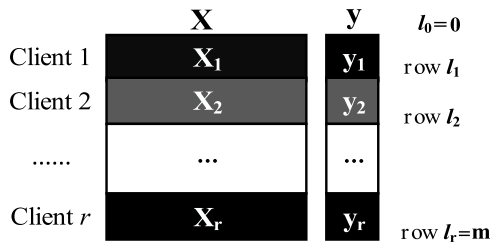

FIGURE 2. Horizontally partitioned data.

Additionally, all input data can be denoted as matrix $\mathbf{X}$ and vector $\mathbf{y}$ as follows.

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m]^\mathsf{T}, \quad \mathbf{y} = [y_1, y_2, \cdots, y_m]^\mathsf{T}$$

Therefore, we have

$$\mathbf{X}^\mathsf{T} = [\mathbf{X}_1^\mathsf{T}, \cdots, \mathbf{X}_r^\mathsf{T}], \quad \mathbf{y}^\mathsf{T} = [\mathbf{y}_1^\mathsf{T}, \cdots, \mathbf{y}_r^\mathsf{T}]$$

Figure 2 shows the relations between $\mathbf{X}$ and $\mathbf{X}_k$, $\mathbf{y}$ and $\mathbf{y}_k$.

III. BACKGROUND

A. LINEAR REGRESSION

Linear regression is one of most widely used approaches for prediction. Its theory is classical and can be found in textbook [29]. Linear regression takes a large number of data as input and outputs a best-fit linear equation for these data.

Given a set of $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{id})^\mathsf{T} \in \mathbb{R}^d$ and its corresponding values $y_i \in \mathbb{R}$, for $1 \le i \le m$. Linear regression finds $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_d)^\mathsf{T} \in \mathbb{R}^d$ and $\alpha \in \mathbb{R}$ that makes $y_i \simeq \boldsymbol{\beta}^\mathsf{T}\mathbf{x}_i + \alpha$. To simplify the problem, denote $\mathbf{x}_i = (x_{i1}, \cdots, x_{id}, 1)^\mathsf{T}$ and $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_d, \alpha)^\mathsf{T}$. Then, the goal of linear regression is to find the best-fit function $y = \boldsymbol{\beta}^\mathsf{T}\mathbf{x}$.

The common way to compute $\boldsymbol{\beta}$ is the least square method, which needs to solve the linear system

$$\mathbf{A}\boldsymbol{\beta} = \mathbf{b}$$

where $\mathbf{A} = \mathbf{X}^\mathsf{T}\mathbf{X} \in \mathbb{R}^{(d+1)\times(d+1)}$ is the covariance matrix, its corresponding right-hand-side vector $\mathbf{b} = \mathbf{X}^\mathsf{T}\mathbf{y} \in \mathbb{R}^{(d+1)}$, and

$$\mathbf{X} = \begin{bmatrix} x_{11} & \cdots & x_{1d} & 1 \\ x_{21} & \cdots & x_{2d} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & \cdots & x_{md} & 1 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

The element of $\mathbf{A}$ and $\mathbf{b}$ is

$$a_{ij} = \sum_{n=1}^{m} x_{ni}x_{nj}, \quad b_i = \sum_{n=1}^{m} x_{nd}y_n \quad (1 \le i, j \le d+1)$$

In particular,

$$\begin{aligned} a_{(d+1)(d+1)} &= m \\ a_{(d+1)j} &= a_{j(d+1)} = \sum_{n=1}^{m} x_{nj} \quad (j \ne d) \\ b_{(d+1)} &= \sum_{n=1}^{m} y_n \end{aligned}$$

Because our input data are horizontally partitioned, client k has $\mathbf{X}_k \in \mathbb{R}^{(l_k - l_{k-1})\times(d+1)}$ and $\mathbf{y}_k \in \mathbb{R}^{(l_k - l_{k-1})}$. From $\mathbf{X}^\mathsf{T} = [\mathbf{X}_1^\mathsf{T}, \cdots, \mathbf{X}_r^\mathsf{T}]$ and $\mathbf{y}^\mathsf{T} = (\mathbf{y}_1^\mathsf{T}, \cdots, \mathbf{y}_r^\mathsf{T})$, we obtain

$$\mathbf{A} = \sum_{k=1}^{r} \mathbf{X}_k^\mathsf{T}\mathbf{X}_k \quad \mathbf{b} = \sum_{k=1}^{r} \mathbf{X}_k^\mathsf{T}\mathbf{y}_k$$

Let $\mathbf{A}_k = \mathbf{X}_k^\mathsf{T}\mathbf{X}_k \in \mathbb{R}^{(d+1)\times(d+1)}$ and $\mathbf{b}_k = \mathbf{X}_k^\mathsf{T}\mathbf{y}_k \in \mathbb{R}^{(d+1)}$. Obviously, client k can compute $\mathbf{A}_k$ and $\mathbf{b}_k$ locally.

Later in this article, we always assume that the matrix $\mathbf{X}$ or $\mathbf{X}_k$ contain $d+1$ columns.

B. THE PAILLIER CRYPTOSYSTEM

The Paillier cryptosystem [13] is the core encryption used in our protocol. It is a semantically secure [21] and additive homomorphic encryption scheme. Semantic security makes it impossible for any polynomial algorithm to gain extra information about a plaintext when given only its ciphertext and public key. As an asymmetric encryption, the Paillier cryptosystem has a public/private key pair (pk, sk).

$$pk := (n, g) \quad \text{and} \quad sk := \lambda(n) = lcm(p_1 - 1, q_1 - 1)$$

where $n = p_1 \cdot q_1$, p_1 and q_1 are distinct large primes, $g \in \mathbb{Z}_{n^2}^*$ and n divides the order of g, $\lambda(n)$ is the Carmichael's function on n.

Encryption: Given plaintext $m \in \mathbb{Z}_n^*$, select random $r \in \mathbb{Z}_n^*$. The ciphertext c is

$$c = E_{pk}(m, r) = g^m \cdot r^n \mod n^2$$

where E_{pk} is the encryption function with public key pk.

Decryption: Given ciphertext $c \in \mathbb{Z}_{n^2}^*$, its plaintext m is

$$m = D_{sk}(c) = \frac{L(c^\lambda \mod n^2)}{L(g^\lambda \mod n^2)} \mod n$$

where D_{sk} is the decryption function with secret key sk and $L(u) := (u-1)/n$.

Given $a, b \in \mathbb{Z}_n$, the Paillier encryption scheme satisfies the following properties:

$$E_{pk}(a+b) = E_{pk}(a) \cdot E_{pk}(b) \bmod n^2$$

As a special case, for a positive integer k

$$E_{pk}(ka) = E_{pk}(a)^k \bmod n^2$$

In this paper, each client encrypts its data with Paillier encryption before submitting them to a service provider. To simplify the problem description, we use $E(\cdot)$ and $D(\cdot)$ instead of $E_{pk}(\cdot)$ and $D_{pk}(\cdot)$ elsewhere in this paper. In addition, we also omit the "mod" suffix when we describe the homomorphic addition later in this paper.

C. DATA REPRESENTATION

Clients' input data could be real numbers, but Paillier encryption only works on nonnegative integers; therefore, data conversion must be done before using Paillier encryption. In our protocol, a fixed-point number is used to realize conversion

between a real number and an integer. The fixed-point number has a q-bit fractional part and p-bit integral part. Because q is fixed, we can drop the binary point and use a binary integer to represent the fixed-point number.

Furthermore, the integer used in Paillier encryption is nonnegative, so the clients need to convert all $\mathbf{x}_i$ and y_i into nonnegative numbers locally at the beginning of the protocol. One solution is two's complement representation, which represents a negative integer with a positive integer. Our solution is find a large enough number $\mathcal{M}$ that makes $x_{ij}+\mathcal{M} \geq 0$ and $y_i + \mathcal{M} \geq 0$ for all data. In the initial phase of the protocol, each client converts a real number a to a nonnegative integer by the following formula

$$\mathbf{ufix}(a) = \lfloor (a + \mathcal{M}) \cdot 2^q \rfloor$$

where $\mathbf{ufix}(a)$ is an unsigned fixed number corresponding to a. Some fixed-point arithmetic operations are also used in our protocol as follows:

- Addition: $\mathbf{ufix}(a + b) = \mathbf{ufix}(a) + \mathbf{ufix}(b)$
- Multiplication: $\mathbf{ufix}(a \cdot b) = \mathbf{ufix}(a) \cdot \mathbf{ufix}(b)/2^q$
- Division: $\mathbf{ufix}(a/b) = \mathbf{ufix}(a) \cdot 2^q/\mathbf{ufix}(b)$

Parameter q is important because it influences the accuracy of calculations. p is trivial in most cases because the range of numbers in practice is far less than the upper limit of the encryption scheme.

IV. SECURE LINEAR REGRESSION PROTOCOL

A. THE KEY IDEA OF OUR PROTOCOL

The execution process of our protocol includes three phase: initialization, aggregation and regression (see figure 1). The main task of each phase is as follows.

(1) *initialization*: The CSP generates key pair (pk,sk), and sends pk to others; the Evaluator sets (p,q) for a fixed-point number and sends (p,q) to others. Client k converts data to nonnegative integers.
(2) *aggregation*: Client k computes $E(\mathbf{A}_k)$, $E(\mathbf{b}_k)$ locally and submits $E(\mathbf{A}_k)$ and $E(\mathbf{b}_k)$ to the Evaluator. The Evaluator merges data into $E(\mathbf{A})$ and $E(\mathbf{b})$.
(3) *regression*:
   – Suppose $M(\cdot)$ is a data masking algorithm; the Evaluator uses $M(\cdot)$, $E(\mathbf{A})$ and $E(\mathbf{b})$ to determine $E(M(\mathbf{A}))$, $E(M(\mathbf{b}))$ with the help of the CSP.
   – The CSP decrypts $E(M(\mathbf{A}))$, $E(M(\mathbf{b}))$, solves the linear equations related to $M(\mathbf{A})$ and $M(\mathbf{b})$, and returns the masked model $M(\boldsymbol{\beta})$ to the Evaluator.
   – The Evaluator obtains the true $\boldsymbol{\beta}$ by eliminating the data mask.

In phase 3, $M(\cdot)$ can mask each element in $\mathbf{A}$ and $\mathbf{b}$ and generates the data masked matrix and vector. There are two critical problems in this protocol:

(a) how do we design $M(\cdot)$?
(b) how do we introduce $M(\cdot)$ into $E(\mathbf{A})$ and $E(\mathbf{b})$?

Let us first illustrate our design about $M(\cdot)$ in plaintext. We randomly choose three invertible diagonal matrices $\mathbf{V}$, $\mathbf{S}$ and $\mathbf{T}$. Then, we obtain the diagonal matrix $\mathbf{U} = \mathbf{S}+\mathbf{T}$. Transform the equation $\mathbf{A}\boldsymbol{\beta} = \mathbf{b}$ as follows.

$$\mathbf{UAVV}^{-1}\boldsymbol{\beta} = \mathbf{Ub} = \mathbf{Sb} + \mathbf{Tb}$$

We choose two random numbers w_1, w_2 and let $\hat{\mathbf{A}} := \mathbf{UAV}$, $\hat{\mathbf{b}} := \mathbf{Ub}$, $\hat{\mathbf{b}}_1 := w_1\mathbf{Sb}$, $\hat{\mathbf{b}}_2 := w_2\mathbf{Tb}$; hence,

$$\boldsymbol{\beta} = \mathbf{V}\hat{\mathbf{A}}^{-1}\hat{\mathbf{b}} = \mathbf{V}\left(\frac{\hat{\mathbf{A}}^{-1}\hat{\mathbf{b}}_1}{w_1} + \frac{\hat{\mathbf{A}}^{-1}\hat{\mathbf{b}}_2}{w_2}\right)$$

Then, we construct two linear equations as follows

$$\hat{\mathbf{A}}\boldsymbol{\xi} = \hat{\mathbf{b}}_1, \quad \hat{\mathbf{A}}\boldsymbol{\eta} = \hat{\mathbf{b}}_2$$

Solving these equations, we can obtain the final result

$$\boldsymbol{\beta} = \mathbf{V}\left(\frac{\boldsymbol{\xi}}{w_1} + \frac{\boldsymbol{\eta}}{w_2}\right)$$

The masked $\hat{\mathbf{A}}$, $\hat{\mathbf{b}}_1$ and $\hat{\mathbf{b}}_2$ prevent information leakage about $\mathbf{A}$ or $\mathbf{b}$. In our protocol, the Evaluator first chooses $\mathbf{U}$, $\mathbf{V}$, w_1 and w_2, then manages to obtain $E(\hat{\mathbf{A}})$, $E(\hat{\mathbf{b}}_1)$, $E(\hat{\mathbf{b}}_2)$, and sends them to the CSP. The CSP decrypts the received data, solves the equations $\hat{\mathbf{A}}\boldsymbol{\xi} = \hat{\mathbf{b}}_1$ and $\hat{\mathbf{A}}\boldsymbol{\eta} = \hat{\mathbf{b}}_2$, and sends back the solution $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$. The Evaluator uses $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ to determine $\boldsymbol{\beta}$.

Now, let us show how to introduce $M(\cdot)$ into $E(\mathbf{A})$ and $E(\mathbf{b})$. Given u_i, v_j, and $E(a_{ij})$, they are elements of matrix $\mathbf{U}$, $\mathbf{V}$ and $E(\mathbf{A})$ respectively. How do we compute $E(u_ia_{ij}v_j)$? If u_i and v_j are integers, we can use $E(u_ia_{ij}v_j) = E(a_{ij})^{u_iv_j}$ directly. However, this is not secure because the CSP will obtain $u_ia_{ij}v_j$ when solving the data masked equations. If a_{ij}, u_i and v_j are prime numbers, it is not difficult for the CSP to guess a_{ij}. In our protocol, the elements in $\mathbf{U}$ and $\mathbf{V}$ are rational numbers that have the same denominator. That mean the elements in $\mathbf{U}$ and $\mathbf{V}$ are u_i/e and v_j/e, where u_i, v_j and e are integers. To obtain $E(u_ia_{ij}v_j/e^2)$, we first compute $E(u_ia_{ij}v_j) = E(a_{ij})^{u_iv_j}$. Then, we use an encrypted division technique (in protocol 2) to obtain $E(\lfloor u_ia_{ij}v_j/e^2 \rceil)$. The $\lfloor u_ia_{ij}v_j/e^2 \rceil$ is the integer near $u_ia_{ij}v_j/e^2$. Actually we cannot compute $E(u_ia_{ij}v_j/e^2)$ in most cases because $u_ia_{ij}v_j/e^2$ may not be an integer. Replacing $u_ia_{ij}v_j/e^2$ with $\lfloor u_ia_{ij}v_j/e^2 \rceil$ only causes negligible error.

B. INPUT DATA CONVERSION

In the inintialization phase, the client needs to convert its data to nonnegative integers. If all clients' primary data are nonnegative, each client can simply convert data to integers by using formula $\lfloor a \cdot 2^q \rfloor$, where a represents primary data. If there are some negative input data, our method is to pass a positive real number $\mathcal{M}$ to all clients, who can convert the local data to nonnegative numbers by using it, and then convert the data to integers. We propose the protocol 1 to complete data conversion when there are negative data.

In protocol 1, $E(\lceil -z_k \cdot 2^q \rceil)$ are collected by the Evaluator before they are sent to the CSP. We do this to keep more information about the client secrete. The $\mathcal{M}$ is known to all clients and the CSP in protocol 1. This means $\mathcal{M}$ is almost public information. If $\mathcal{M}$ is sensitive information, each client

Protocol 1 Clients' Data Conversion

Parties:	*Clients*	*Evaluator*	*CSP*
Input:	x_{ij}, y_i	(p, q)	pk
Output:	$\bar{x}_{ij}, \bar{y}_i \in \mathbb{N}$		

1: Client k finds the minimum value z_k in the local negative dataset $Ng = \{x_{ij}, y_i | x_{ij} \le 0, y_i \le 0, l_{k-1} < i \le l_k, 1 \le j \le d\}$, where $k = 1, 2, \cdots, r$.
2: Client k computes $E(\lceil -z_k \cdot 2^q \rceil)$ and sends it to Evaluator.
3: Evaluator sends all $E(\lceil -z_k \cdot 2^q \rceil)$ to CSP.
4: CSP decrypts all $E(\lceil -z_k \cdot 2^q \rceil)$, finds the maximum value $\mathcal{M}$ in $\{\lceil -z_1 \cdot 2^q \rceil / 2^q, \lceil -z_2 \cdot 2^q \rceil / 2^q, \cdots, \lceil -z_r \cdot 2^q \rceil / 2^q\}$ and passes it to all clients.
5: Each client converts local data to nonnegative integers by using formula $\bar{x}_{ij} = \lfloor (x_{ij} + \mathcal{M}) \cdot 2^q \rfloor$ and $\bar{y}_i = \lfloor (y_i + \mathcal{M}) \cdot 2^q \rfloor$.

Protocol 2 (Data Masking) Compute $E(\lfloor xc/d \rceil)$ From Given $E(x)$, c and d

Parties:	$\mathcal{A}$	$\mathcal{B}$
Input:	$E(x)$, integer c, integer d	pk
Output:	$E(\lfloor xc/d \rceil)$	

1: $\mathcal{A}$ computes $E(xc) = E(x)^c$
2: $\mathcal{A}$ chooses a random r, computes $E(xc+r) = E(xc) \cdot E(r)$. Then $\mathcal{A}$ sends $E(xc + r)$ to $\mathcal{B}$.
3: $\mathcal{B}$ decrypts $E(xc+r)$, computes $\lfloor (xc+r)/d \rfloor$, then returns $E(\lfloor (xc + r)/d \rfloor)$ to $\mathcal{A}$.
4: $\mathcal{A}$ computes $E(\lfloor r/d \rfloor)$, then computes

$$\begin{aligned} E(\lfloor xc/d \rceil) &= E(\lfloor (xc + r)/d \rfloor - \lfloor r/d \rfloor) \\ &= E(\lfloor (xc + r)/d \rfloor) \cdot E(\lfloor r/d \rfloor)^{-1} \end{aligned}$$

can add a random positive integer to $\lceil -z_k \cdot 2^q \rceil$ before it is encrypted and passed to the Evaluator.

C. DATA MASKING TECHNIQUE

In our system, the Evaluator holds some data that can be viewed as $E(x)$, integer c and public integer d. To mask the data x, the Evaluator needs to compute $E(\lfloor xc/d \rceil)$. It is easy to compute $E(xc) = E(x)^c$. The problem is how to compute $E(\lfloor xc/d \rceil)$ using $E(xc)$ and d. Until now, there is no way for the Evaluator to complete this task alone. In our data masking protocol, we use Veugen's [26] approach to perform division between $E(xc)$ and a public divisor d.

In protocol 2, the $E(\lfloor r/d \rfloor)^{-1}$ is a modular inverse of $E(\lfloor r/d \rfloor)$. In proposition 1, we prove the correctness of protocol 2.

Proposition 1: In protocol 2, the integer $\lfloor (xc + r)/d \rfloor - \lfloor r/d \rfloor$ *in step 4 is the integer* $\lfloor xc/d \rfloor$ *or* $\lfloor xc/d \rfloor + 1$.

Proof: Some basic formulas are

$$\begin{aligned} xc/d &= \lfloor xc/d \rfloor + xc\%d \\ r/d &= \lfloor r/d \rfloor + r\%d \\ (xc + r)/d &= \lfloor (xc + r)/d \rfloor + (xc + r)\%d \end{aligned}$$

Hence,

$$\begin{aligned} &\lfloor (xc + r)/d \rfloor - \lfloor r/d \rfloor \\ &= \lfloor xc/d \rfloor + xc\%d + r\%d - (xc + r)\%d \\ &= \lfloor xc/d \rfloor + xc\%d + r\%d - (xc\%d + r\%d)\%d \end{aligned} \quad (4.1)$$

From $0 \le xc\%d < d$ and $0 \le r\%d < d$
we obtain $0 \le xc\%d + r\%d < 2d$.
It can be divided into two cases as follows.

Case 1: $0 \le xc\%d + r\%d < d$
we have $xc\%d + \%d = (xc\%d + r\%d)\%d$
Combining (4.1), we obtain

$$\lfloor (xc + r)/d \rfloor - \lfloor r/d \rfloor = \lfloor xc/d \rfloor$$

Case 2: $d \le xc\%d + r\%d < 2d$
we have $xc\%d + r\%d = 1 + (xc\%d + r\%d)\%d$
Combining (4.1), we obtain

$$\lfloor (xc + r)/d \rfloor - \lfloor r/d \rfloor = \lfloor xc/d \rfloor + 1$$

□

In protocol 2, d is public. If we can design a protocol that can mask our data while c and d is only known to $\mathcal{A}$, the data masking protocol will be perfect. However, this is not easy work. Some attempts [24], [25] have been made to deal with the problem of computing $E(a/b)$ from $E(a)$ and $E(b)$. However, all of these methods are too complicated and time consuming. If our protocol spends too much time on data masking, our advantage in terms of efficiency will lost.

We include an example here to show why this data masking technique enjoys a high level of security and accuracy. For simplicity, we perform an arithmetic operation on fixed-point decimal numbers in this toy example.

Suppose input data are given as $x = 2871234561091231$; the low-order 8 digital numbers represent the fractional part. We choose an 8-digit prime number $d = 39999931$ and a random integer $c = 8233875439$. We mask x as follows and obtain y:

$$y = \lfloor xc/d \rfloor = 591035712841030450$$

or

$$y = \lceil xc/d \rceil = 591035712841030451$$

Let us consider security first. The CSP can see y after decrypting $E(y)$, and the CSP also knows divisor d; now the CSP wants to know x. What he can do is compute yd and guess x based on it. However,

$$yd = 23641387732177031968898950$$

or

$$yd = 23641387732177032008898881$$

If the CSP guesses x by factorization, he needs to know the exact xc.

$$xc = 23641387732177031969175409$$

Actually, there are 7 or 10 different digits between yd and xc in this example. It is difficult to guess xc based on yd.

Now, we consider the accuracy when the CSP uses y in computation. The exact computation can be done by using xc/d instead of y. Obviously,

$$|y - xc/d| = |\lfloor xc/d \rceil - xc/d| < 1$$

Because the low-order 8 digital numbers are the fractional part, the difference between y and xc/d is actually less than 10^{-8}. Obviously, we can obtain sufficient accuracy if the length of the fractional part is large enough.

D. OUR PROTOCOL FOR DISTRIBUTED DATA

Our protocol described here is the main contribution of this paper. The protocol comprises three phases: initialization, aggregation and regression. Our data masking protocol (protocol 2) is used in regression phase. Finally, the Evaluator outputs the regression model $\boldsymbol{\beta}$ in the clear.

In protocol 3, operator $\otimes$ denotes the elementwise product of two matrices or vectors.

Protocol 3 Secure Linear Regression for Distributed Data

Input:

Client k: $\mathbf{x}_i, y_i$ ($\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, $l_{k-1} < i \le l_k$), $k = 1, 2, \cdots, r$

Evaluator: p, q – bit length of integral and fractional part
e – a public denominator (prime number)

CSP: pk

Output:

Evaluator: regression model $\boldsymbol{\beta} \in \mathbb{R}^{(d+1)}$

1. *Initialization*
   a. Evaluator sets (p, q) and sends it to all clients; then it chooses a q-bit prime number e for data masking and passes it to CSP.
   b. If all data are nonnegative, each client converts their data to integers locally. Otherwise, all clients, the Evaluator and CSP execute protocol 1 to complete data conversion. Finally, all x_{ij} and y_i become nonnegative integers.
   c. CSP generates a key pair (pk, sk) of the Paillier encryption scheme and sends pk to all other parties.
2. *Aggregation*
   a. Client k computes $\mathbf{A}_k = \mathbf{X}_k^{\mathsf{T}}\mathbf{X}_k$, $\mathbf{b}_k = \mathbf{X}_k^{\mathsf{T}}\mathbf{y}_k$ and encrypts these data locally, then passes $E(\mathbf{A}_k)$ and $E(\mathbf{b}_k)$ to the Evaluator, where $1 \le k \le r$.
   b. Evaluator computes matrix $E(\mathbf{A})$ and vector $E(\mathbf{b})$:
   $E(\mathbf{A}) = E(\sum_{k=1}^{r} \mathbf{A}_k) = \otimes_{k=1}^{r} E(\mathbf{A}_k)$
   $E(\mathbf{b}) = E(\sum_{k=1}^{r} \mathbf{b}_k) = \otimes_{k=1}^{r} E(\mathbf{b}_k)$.
3. *Regression*
   a. Evaluator chooses random integer v_i, s_i, t_i, w_1, w_2 in $(e, 2^{10}e)$, and obtains $u_i = s_i + t_i$. where $1 \le i \le d+1$.
   b. Evaluator (role $\mathcal{A}$) and CSP (role $\mathcal{B}$) execute protocol 2 to mask data as follows.

Parties:	$\mathcal{A}$	$\mathcal{B}$
Input 1:	$E(a_{ij})$, $u_i v_j$, e^2	pk, e^2
Output 1:	$E(\lfloor u_i v_j a_{ij}/e^2 \rceil)$	
Input 2:	$E(b_i)$, $w_1 s_i$, e^2	pk, e^2
Output 2:	$E(\lfloor w_1 s_i b_i/e^2 \rceil)$	
Input 3:	$E(b_i)$, $w_2 t_i$, e^2	pk, e^2
Output 3:	$E(\lfloor w_2 t_i b_i/e^2 \rceil)$	

   Then, Evaluator sends the output 1,2,3 to CSP.
   d. CSP decrypts the data received and obtains

$$\widetilde{a}_{ij} = \lfloor \frac{u_i v_j a_{ij}}{e^2} \rceil, \quad \widetilde{b}_{i1} = \lfloor \frac{w_1 s_i b_i}{e^2} \rceil, \quad \widetilde{b}_{i2} = \lfloor \frac{w_2 t_i b_i}{e^2} \rceil$$

   then CSP solves two equations as follows, and returns $\widetilde{\boldsymbol{\xi}}, \widetilde{\boldsymbol{\eta}}$ to Evaluator.

$$\widetilde{\mathbf{A}}\widetilde{\boldsymbol{\xi}} = \widetilde{\mathbf{b}}_1, \quad \widetilde{\mathbf{A}}\widetilde{\boldsymbol{\eta}} = \widetilde{\mathbf{b}}_2$$

   where
   $\widetilde{\mathbf{A}} := \{\widetilde{a}_{ij}\} \in \mathbb{R}^{(d+1)\times(d+1)}$, $\widetilde{\mathbf{b}}_1 := \{\widetilde{b}_{i1}\} \in \mathbb{R}^{(d+1)}$,
   $\widetilde{\mathbf{b}}_2 := \{\widetilde{b}_{i2}\} \in \mathbb{R}^{(d+1)}$.
   e. Evaluator computes the solution $\widetilde{\boldsymbol{\beta}}$ as follows

$$\widetilde{\boldsymbol{\beta}} = \mathbf{V}\left(\frac{e\widetilde{\boldsymbol{\xi}}}{w_1} + \frac{e\widetilde{\boldsymbol{\eta}}}{w_2}\right)$$

   where $\mathbf{V} = diag(v_1/e, v_2/e, \cdots, v_d/e)$.

V. PROTOCOL CORRECTNESS AND ERROR ANALYSIS

In this section, we discuss the correctness of protocol 3 and conduct error analysis. First, we define diagonal matrix $\mathbf{S}$, $\mathbf{T}$, $\mathbf{U}$, $\mathbf{V}$, matrix $\hat{\mathbf{A}}$, vector $\hat{\mathbf{b}}_1$, $\hat{\mathbf{b}}_2$ as follows

$$\mathbf{S} = diag(s_1/e, s_2/e, \cdots, s_{d+1}/e)$$
$$\mathbf{T} = diag(t_1/e, t_2/e, \cdots, t_{d+1}/e)$$
$$\mathbf{U} = \mathbf{S} + \mathbf{T} = diag(u_1/e, u_2/e, \cdots, u_{d+1}/e)$$
$$\mathbf{V} = diag(v_1/e, v_2/e, \cdots, v_{d+1}/e) \tag{5.1}$$
$$\hat{\mathbf{A}} = \mathbf{UAV}, \quad \hat{\mathbf{b}}_1 = \frac{w_1}{e}\mathbf{Sb}, \quad \hat{\mathbf{b}}_2 = \frac{w_2}{e}\mathbf{Tb} \tag{5.2}$$

where s_i, t_i, u_i, v_i, w_1, w_2, and e are random integers in protocol 3.

According to definition (5.2), the element of $\hat{\mathbf{A}}$, $\hat{\mathbf{b}}_1$ and $\hat{\mathbf{b}}_2$ is $u_i v_j a_{ij}/e^2$, $w_1 s_i b_i/e^2$ and $w_2 t_i b_i/e^2$. Obviously, $\hat{\mathbf{A}}$, $\hat{\mathbf{b}}_1$ and $\hat{\mathbf{b}}_2$ is the accurate version of $\widetilde{\mathbf{A}}$, $\widetilde{\mathbf{b}}_1$ and $\widetilde{\mathbf{b}}_2$ in protocol 3. We construct functions as follows

$$\hat{\mathbf{A}}\boldsymbol{\xi} = \hat{\mathbf{b}}_1, \quad \hat{\mathbf{A}}\boldsymbol{\eta} = \hat{\mathbf{b}}_2$$

Then, we use $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ to compute $\boldsymbol{\beta}$ in the same way as we compute $\widetilde{\boldsymbol{\beta}}$ in protocol 3. We obtain

$$\begin{aligned}
\boldsymbol{\beta} &= \mathbf{V}\left(\frac{e\boldsymbol{\xi}}{w_1} + \frac{e\boldsymbol{\eta}}{w_2}\right) \\
&= \mathbf{V}\hat{\mathbf{A}}^{-1}\left(\frac{e\hat{\mathbf{b}}_1}{w_1} + \frac{e\hat{\mathbf{b}}_1}{w_2}\right) \\
&= \mathbf{V}\hat{\mathbf{A}}^{-1}(\mathbf{Sb} + \mathbf{Tb}) \\
&= \mathbf{V}(\mathbf{V}^{-1}\mathbf{A}^{-1}\mathbf{U}^{-1})\mathbf{Ub} \\
&= \mathbf{A}^{-1}\mathbf{b}
\end{aligned}$$

This means $\boldsymbol{\beta}$ is the true solution when we ignore the error in protocol 3. This process is just the same as what we described in the *key idea* section.

Now, the problem is how do small changes to $\hat{\mathbf{A}}$, $\hat{\mathbf{b}}$ affect the solution $\widetilde{\boldsymbol{\beta}}$. From the classical textbook of numerical analysis [27], a theorem can be found as follows.

Lemma 1: Given two linear systems of equations $Ax = b$ *and* $(A + \Delta A)\hat{x} = b + \Delta b$, *where* ΔA *and* Δb *are small changes, let* $\Delta x \equiv \hat{x} - x$; *then, the estimation holds up as follows*

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\mathcal{K}(A)}{1 - \mathcal{K}(A)\|\Delta A\|/\|A\|}\left(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|}\right)$$

where $\mathcal{K}(A) = \|A\|\|A^{-1}\|$ *is the condition number that measures how sensitive the linear system is to changes in A and b.* $\|\cdot\|$ *is the matrix and vector norm.*

Lemma 1 is used in our error analysis of protocol 3. Now, we present our conclusion as follows.

Proposition 2 (Error Estimation): Given two linear systems of equations as

$$\hat{\mathbf{A}}\mathbf{V}^{-1}\boldsymbol{\beta} = \frac{e}{w_1}\hat{\mathbf{b}}_1 + \frac{e}{w_2}\hat{\mathbf{b}}_2$$
$$\widetilde{\mathbf{A}}\mathbf{V}^{-1}\widetilde{\boldsymbol{\beta}} = \frac{e}{w_1}\widetilde{\mathbf{b}}_1 + \frac{e}{w_2}\widetilde{\mathbf{b}}_2$$

where $\widetilde{\mathbf{A}}$, $\widetilde{\mathbf{b}}_1$, *and* $\widetilde{\mathbf{b}}_2$ *are defined in the regression phase of protocol 3.* $\mathbf{V}$, $\hat{\mathbf{A}}$, $\hat{\mathbf{b}}_1$, *and* $\hat{\mathbf{b}}_2$ *are defined as (5.1) and (5.2), and* e, w_1, *and* w_2 *are random integers used in protocol 3. Let* $\|\Delta\boldsymbol{\beta}\| = \|\widetilde{\boldsymbol{\beta}} - \boldsymbol{\beta}\|$, *given the condition* $m \geq d + 1$ *and* $\sum_{i=1}^{m} y_i \geq 2^q$, *the error caused by perturbation* $(\hat{\mathbf{A}}\mathbf{V}^{-1} - \widetilde{\mathbf{A}}\mathbf{V}^{-1})$ *and* $((e\hat{\mathbf{b}}_1/w_1 + e\hat{\mathbf{b}}_2/w_2) - (e\widetilde{\mathbf{b}}_1/w_1 + e\widetilde{\mathbf{b}}_2/w_2))$ *can be estimated as follows*

$$\frac{\|\Delta\boldsymbol{\beta}\|}{\|\boldsymbol{\beta}\|} \leq \frac{3 \cdot \mathcal{K}(\mathbf{A})}{2^{2q-9} - \mathcal{K}(\mathbf{A})}$$

where q is the bit length of the fractional part, d is the number of data features, and $\mathcal{K}(\mathbf{A})$ *is the condition number of the matrix.*

Proof: Obviously, $\widetilde{\boldsymbol{\beta}}$ is the approximate solution we obtain in protocol 3, and $\boldsymbol{\beta}$ is the true solution of $\mathbf{A}\boldsymbol{\beta} = \mathbf{b}$.

First, consider the error of the right-hand items. Using the infinity-norm, we have

$$\|\widetilde{\mathbf{b}}_1 - \hat{\mathbf{b}}_1\| = \max_{1\leqslant i\leqslant d+1}\left|\frac{w_1 s_i b_i}{e^2} - \lfloor\frac{w_1 s_i b_i}{e^2}\rceil\right| < 1$$
$$\|\widetilde{\mathbf{b}}_2 - \hat{\mathbf{b}}_2\| = \max_{1\leqslant i\leqslant d+1}\left|\frac{w_1 t_i b_i}{e^2} - \lfloor\frac{w_1 t_i b_i}{e^2}\rceil\right| < 1$$

Let

$$\hat{\mathbf{g}} = \frac{e}{w_1}\hat{\mathbf{b}}_1 + \frac{e}{w_2}\hat{\mathbf{b}}_2, \quad \widetilde{\mathbf{g}} = \frac{e}{w_1}\widetilde{\mathbf{b}}_1 + \frac{e}{w_2}\widetilde{\mathbf{b}}_2$$

In protocol 3, we have $w_1 > e$ and $w_2 > e$. Hence,

$$\begin{aligned}\|\Delta\hat{\mathbf{g}}\| &= \|\widetilde{\mathbf{g}} - \hat{\mathbf{g}}\| \\ &= \|\frac{e}{w_1}\widetilde{\mathbf{b}}_1 + \frac{e}{w_2}\widetilde{\mathbf{b}}_2 - \frac{e}{w_1}\hat{\mathbf{b}}_1 - \frac{e}{w_2}\hat{\mathbf{b}}_2\| \\ &\leq \frac{e}{w_1}\|\widetilde{\mathbf{b}}_1 - \hat{\mathbf{b}}_1\| + \frac{e}{w_2}\|\widetilde{\mathbf{b}}_2 - \hat{\mathbf{b}}_2\| \\ &< 2 \end{aligned} \tag{5.3}$$

From the definition of $\hat{\mathbf{b}}_1$, $\hat{\mathbf{b}}_2$, and $s_i > e$, $t_i > e$, we obtain

$$\begin{aligned}\|\hat{\mathbf{g}}\| &= \|\frac{e}{w_1}\hat{\mathbf{b}}_1 + \frac{e}{w_2}\hat{\mathbf{b}}_2\| \\ &= \|\mathbf{S}\mathbf{b} + \mathbf{T}\mathbf{b}\| \\ &= \max_{1\leqslant i\leqslant d+1}\left|\frac{s_i + t_i}{e}b_i\right| \\ &> 2\left|b_{(d+1)}\right| \quad (\text{let } i = d+1) \\ &= 2 \cdot \left|\sum_{n=1}^{m} 2^q \cdot y_n\right| \quad (x_{n(d+1)} = 2^q) \\ &> 2^{2q+1} \end{aligned} \tag{5.4}$$

where $x_{n(d+1)} = 2^q$ $(1 \leq n \leq m)$ corresponds to 1 in primary data.

From (5.3) and (5.4) we obtain

$$\frac{\|\Delta\hat{\mathbf{g}}\|}{\|\hat{\mathbf{g}}\|} < \frac{1}{2^{2q}} \tag{5.5}$$

Second, consider the error of the coefficient matrix. For each element of $\widetilde{\mathbf{A}} - \hat{\mathbf{A}}$, we have

$$\Delta\hat{a}_{ij} = \left|\lfloor\frac{u_i v_j a_{ij}}{e^2}\rceil - \frac{u_i v_j a_{ij}}{e^2}\right| < 1 \quad (1 \leq i, j \leq d+1)$$

and we also know $v_j > e$; hence,

$$\begin{aligned}\|\Delta(\hat{\mathbf{A}}\mathbf{V}^{-1})\| &= \|\widetilde{\mathbf{A}}\mathbf{V}^{-1} - \hat{\mathbf{A}}\mathbf{V}^{-1}\| \\ &= \|(\widetilde{\mathbf{A}} - \hat{\mathbf{A}})\mathbf{V}^{-1}\| \\ &= \max_{1\leqslant i\leqslant d+1}\sum_{j=1}^{d+1}\left|\Delta\hat{a}_{ij}\frac{e}{v_j}\right| \\ &< d+1 \end{aligned} \tag{5.6}$$

From $x_{n(d+1)} = 2^q$ $(1 \leq n \leq m)$, we obtain $a_{(d+1)(d+1)} = 2^{2q} \cdot m$. From the definition of $\hat{\mathbf{A}}$, $\mathbf{V}$ and $u_i > 2e$, we obtain

$$\begin{aligned}\|\hat{\mathbf{A}}\mathbf{V}^{-1}\| &= \max_{1\leqslant i\leqslant d+1}\sum_{j=1}^{d+1}\left|\frac{e}{v_j}\hat{a}_{ij}\right| \\ &= \max_{1\leqslant i\leqslant d+1}\sum_{j=1}^{d+1}\left|\frac{e}{v_j}\frac{u_i a_{ij} v_j}{e \cdot e}\right| \\ &> 2 \cdot \max_{1\leqslant i\leqslant d+1}\sum_{j=1}^{d+1}|a_{ij}| \\ &\geq 2 \cdot \sum_{j=1}^{d+1}|a_{(d+1)j}| \qquad (\text{let } i = d+1) \\ &> 2 \cdot a_{(d+1)(d+1)} = 2^{2q+1} \cdot m \end{aligned} \tag{5.7}$$

From (5.6), (5.7) and $m > d + 1$, we obtain

$$\frac{\|\Delta(\hat{\mathbf{A}}\mathbf{V}^{-1})\|}{\|\hat{\mathbf{A}}\mathbf{V}^{-1}\|} < \frac{1}{2^{2q+1}} \tag{5.8}$$

According to lemma 1, (5.5), and (5.8), we obtain

$$\frac{\|\Delta\boldsymbol{\beta}\|}{\|\boldsymbol{\beta}\|} \leq \frac{\mathcal{K}(\hat{\mathbf{A}}\mathbf{V}^{-1})}{1 - \mathcal{K}(\hat{\mathbf{A}}\mathbf{V}^{-1})(1/2^{2q+1})}\left(\frac{1}{2^{2q+1}} + \frac{1}{2^{2q}}\right)$$

$$= \frac{3 \cdot \mathcal{K}(\mathbf{UAVV}^{-1})}{2^{2q+1} - \mathcal{K}(\mathbf{UAVV}^{-1})}$$
$$\leq \frac{3 \cdot \mathcal{K}(\mathbf{U})\mathcal{K}(\mathbf{A})}{2^{2q+1} - \mathcal{K}(\mathbf{U})\mathcal{K}(\mathbf{A})}$$
$$= \frac{3 \cdot \|\mathbf{U}\|\|\mathbf{U}^{-1}\|\mathcal{K}(\mathbf{A})}{2^{2q+1} - \|\mathbf{U}\|\|\mathbf{U}^{-1}\|\mathcal{K}(\mathbf{A})} \tag{5.9}$$

Since $2 < u_i/e = (s_i + t_i)/e < 2^{11}$ and $\mathbf{U}$, $\mathbf{U}^{-1}$ are diagonal matrices. We have

$$\|\mathbf{U}\| = \max_{1 \leqslant i \leqslant d} \left|\frac{u_i}{e}\right| < 2^{11}$$
$$\|\mathbf{U}^{-1}\| = \max_{1 \leqslant i \leqslant d} \left|\frac{e}{u_i}\right| < \frac{1}{2}$$

Combining (5.9), we obtain

$$\frac{\|\Delta\boldsymbol{\beta}\|}{\|\boldsymbol{\beta}\|} \leq \frac{3 \cdot \mathcal{K}(\mathbf{A})}{2^{2q-9} - \mathcal{K}(\mathbf{A})}$$

□

In most cases, protocol 3 provides a solution with enough precision. For example, suppose $\mathbf{A}$ is well conditioned when $\mathcal{K}(\mathbf{A}) < 2^{50} (\approx 1.1 \cdot 10^{15})$. If we choose $q = 40$, then

$$\frac{\|\Delta\boldsymbol{\beta}\|}{\|\boldsymbol{\beta}\|} \leq \frac{3 \cdot 2^{50}}{2^{71} - 2^{50}} \approx 1.4 \cdot 10^{-6}$$

In fact, our experiments in section VII show this is a rather conservative estimate. This estimate shows the larger the q we choose, the more accurate the solution we obtain.

Remark: There are two preconditions for the establishment of proposition 2. The first one is $m \geq d + 1$ and is easy to satisfy because the number of data used in analysis is often greater than the number of data features in practice. The second one is $\sum_{i=1}^{m} y_i \geq 2^q$ and indicates that the sum of all y_i in the primary data is no less than 1. This is also easy to satisfy.

VI. SECURITY ANALYSIS

Our system is secure provided all participators are honest-but-curious. At first, all clients contribute their own data and cannot see any intermediate result, it is impossible for clients to learn some information about others except that which is revealed by the final result. Therefore, we only analyze two servers.

A. SEMI-HONEST EVALUATOR

The Evaluator only have some ciphertext, such as $E(a_{ij})$, $E(b_i)$, $E(\lfloor u_i v_j a_{ij}/e^2 \rceil)$, $E(\lfloor w_1 s_i b_i/e^2 \rceil)$, and $E(\lfloor w_2 t_i b_i/e^2 \rceil)$. Moreover, he also knows some relations between the plaintexts corresponding to these ciphertexts, such as

$$a_{ij} \cdot \lfloor u_i v_j/e^2 \rceil \approx \lfloor u_i v_j a_{ij}/e^2 \rceil$$
$$b_i \cdot \lfloor w_1 s_i/e^2 \rceil \approx \lfloor w_1 s_i b_i/e^2 \rceil$$

The relations are unhelpful except that an Evaluator can use them to deduce one plaintext based on knowledge of the other. For example, if the Evaluator knows a_{ij}, he can compute $\lfloor u_i v_j a_{ij}/e^2 \rceil$, or vice versa (ignoring small difference). Therefore, the essential problem is still how to crack the unrelated ciphertext. However, the Paillier cryptosystem is semantically secure. It is impossible for the Evaluator to gain information about a plaintext when given only its ciphertext and public key. Hence, the security of the Paillier cryptosystem can guarantee the Evaluator cannot learn anything about input data and important intermediate values.

B. SEMI-HONEST CSP

In our data masking protocol, the CSP can obtain some results, such as $a_{ij} + r_{ij}$, $b_i + q_i$, where r_{ij} and q_i are random integers and used only once. Therefore, the client's data cannot be disclosed to the CSP in this procedure.

In the regression phase, the CSP obtains a masked matrix $\widetilde{\mathbf{A}}$ and right-hand-side vector $\widetilde{\mathbf{b}}_1$, $\widetilde{\mathbf{b}}_2$. The element of them is $\lfloor u_i v_j a_{ij}/e^2 \rceil$, $\lfloor w_1 s_i b_i/e^2 \rceil$, $\lfloor w_2 t_i b_i/e^2 \rceil$. Here, we ignore the influence of operation $\lfloor \cdot \rceil$ and only assess the security of multiplicative perturbation. Suppose the CSP has data $u_i v_j a_{ij}/e^2$, $w_1 s_i b_i/e^2$, $w_2 t_i b_i/e^2$. If the CSP wants to guess the value of a_{ij} and b_i, he can use the final result $\widetilde{\boldsymbol{\beta}}$ to confirm his conjecture. The CSP must guess the value of all a_{ij} and b_i and then compute a new model $\bar{\boldsymbol{\beta}}$; if $\bar{\boldsymbol{\beta}} \approx \widetilde{\boldsymbol{\beta}}$, then his guess is likely to be right.

Because the value of random v_i, s_i, t_i, w_1, and w_2 are limited to any integer between e and $2^{10}e$ in protocol 3, the probability of guessing one of them correctly is $1/(1023e)$ for the CSP. Let us investigate a simple case: given m plane points (x_1, y_1), (x_2, y_2), $\cdots$, (x_m, y_m) for a linear regression. In such a case, the number of data features is $d = 1$ and the dimension of $\widetilde{\mathbf{A}}$ is 2; therefore, the CSP have two linear system as follows.

$$\begin{bmatrix} u_1 v_1 a_{11} & u_1 v_2 a_{12} \\ u_2 v_1 a_{21} & u_2 v_2 m \end{bmatrix} \begin{bmatrix} \widetilde{\xi}_1 \\ \widetilde{\xi}_2 \end{bmatrix} = \begin{bmatrix} w_1 s_1 b_1 \\ w_1 s_2 b_2 \end{bmatrix}$$
$$\begin{bmatrix} u_1 v_1 a_{11} & u_1 v_2 a_{12} \\ u_2 v_1 a_{21} & u_2 v_2 m \end{bmatrix} \begin{bmatrix} \widetilde{\eta}_1 \\ \widetilde{\eta}_2 \end{bmatrix} = \begin{bmatrix} w_2 t_1 b_1 \\ w_2 t_2 b_2 \end{bmatrix}$$

Suppose m is known to all parties. $\mathbf{A}$ is a symmetric matrix; this means $a_{12} = a_{21}$. The CSP can crack the secret as follows.

(1) Suppose the CSP knows v_1 and v_2; he can computes u_2 from the value of $u_2 v_2 m$.
(2) Then, from the value of $u_1 v_2 a_{12}/u_2 v_1 a_{21}$, the CSP determines u_1.
(3) Next, if the CSP also knows s_1 and s_2, he can obtain t_1 and t_2 because $u_1 = s_1 + t_1$, $u_2 = s_2 + t_2$.
(4) Finally, if the CSP also know w_1 and has the final answer $\widetilde{\boldsymbol{\beta}}$ from the Evaluator, he can determine w_2 from $\widetilde{\boldsymbol{\beta}} = \mathbf{V}\left(e\widetilde{\boldsymbol{\xi}}/w_1 + e\widetilde{\boldsymbol{\eta}}/w_2\right)$ by using v_1, v_2, w_1 and $\widetilde{\boldsymbol{\beta}}$.

Therefore, there are 5 independent random variables: v_1, v_2, s_1, s_2, and w_1. The probability that the CSP can guess $\mathbf{A}$ and $\mathbf{b}$ correctly is $1/(1023e)^5$. If e is a 40-bit long prime, the odds are approximately $1/2^{245}$. For a regression analysis with d data features, the probability that the CSP can determine $\mathbf{A}$ and $\mathbf{b}$ is $1/(1023e)^{2d+3}$. From the above analysis, it can

ascertained that our protocol is statistically secure by using multiplicative perturbation. Moreover, the greater the e is, the safer the data become.

VII. EXPERIMENT AND COMPARISON

We assess our privacy-preserving linear regression protocol by a set of numerical experiments on some real datasets. All experiments are performed on a Dell Mobile Workstation M6800 with an 8-core CPU at 2.70GHz and 32 GB RAM, running Ubuntu Linux 16.04. We implement our protocol using C++ with the NTL library and Java on the OpenJDK 1.8 platform. The Paillier encryption scheme with a 1024-bit modulus is adopted in our implementation.

A. DATASETS

We choose 6 datasets from the UCI repository [28] for our experiments. These datasets are listed below.

- **Auto MPG**
  This dataset contains 398 records about cars regarding attempts to predict miles per gallon for each. We removed the car name attribute and 6 records that have missing values of horsepower. In the end, the dataset has 1 target and 7 predictive attributes, and contains 392 records.
- **Wine Quality**
  It contain 4898 records of wine used for predicting wine quality. We choose a dataset about white wine that has 11 predictive attributes and 1 target attribute.
- **Bike Sharing**
  This dataset contains 17379 records about bike rentals. We remove the record index, the date, count of casual users and count of registered users from the dataset *hour.csv*. We leave 12 attributes to predict the count of total rental bikes.
- **Forest Fires**
  The dataset contains 517 records used in predicting the burned area of the forest. We remove the month attribute and alter the content of the weekday attribute from 'mon' to 1, 'tue' to 2, etc. Finally, we have 11 predictive attributes and 1 target attribute.
- **Communities and Crime**
  This dataset contains information from 1994 communities. We remove 5 nonpredictive attributes and use 122 attributes to predict the number of crimes per capita for each community. All missing values are replaced with 0s.
- **YearPredictionMSD**
  This dataset describes 515344 songs with 90 audio features for each one. It is used for predicting the release year of each song. To speed up the process, only the top 100000 records are used.

For every dataset, we add integer 1 at the end of each line to determine the constant term of the regression equation. We modify some datasets only for obtaining usable data for our experiment, not for serious prediction research. Furthermore, all data has not been normalized.

B. EVALUATION OF ACCURACY

When we calculate something like $\hat{a} = u_i v_j a_{ij}/e^2$ in protocol 3, we obtain $\tilde{a} = \lfloor u_i v_j a_{ij}/e^2 \rceil$ instead. This procedure causes very small calculation error. Because $\hat{a}$ and $\tilde{a}$ are integers corresponding to fixed point numbers, $|\hat{a} - \tilde{a}| < 1$ means the real gap between these two data is less than 2^{-q}. This indicates that the accuracy of the data masking procedure depends on q. Furthermore, because q is the length of the fractional part of the fixed-point number, it also has a great influence on the truncation error of our system. Consequently, the accuracy of the protocol depend on q. The larger the q is, the more accurate the calculation is.

Let $\boldsymbol{\beta}$ be the model learned in the clear data and $\widetilde{\boldsymbol{\beta}}$ learned in our system. We define the relative error of our system as:

$$Err_{\widetilde{\boldsymbol{\beta}}} = \frac{\|\widetilde{\boldsymbol{\beta}} - \boldsymbol{\beta}\|_2}{\|\boldsymbol{\beta}\|_2}$$

Table 1 shows the accuracy of $\widetilde{\boldsymbol{\beta}}$ is improved with the increasing of q. Moreover, the results also indicate there is no obvious correlation between m and $Err_{\widetilde{\boldsymbol{\beta}}}$, or between d and $Err_{\widetilde{\boldsymbol{\beta}}}$. Although the condition number $\mathcal{K}(\mathbf{A})$ is used in evaluating the error bound in section V, there is no direct correlation between $\mathcal{K}(\mathbf{A})$ and the accuracy of the output.

It must be pointed out that the change of q almost has no impact on the running time of the protocol. In our experiments listed in table 1, the running time has no identifiable differences between $q = 10$ and $q = 50$ for any dataset. In fact, after clients encrypted a data, the corresponding cipher is always a 1024-bit long integer, no matter what the value of q is. Moreover, when two servers execute our protocol, they always perform computation with encrypted data, except that the CSP will solve two data masked linear systems. Because the time spent on a normal multiplication or division is less than 1% of that spent on an encryption (see table 5), our protocol spends almost all running time on crypto operations.

C. EVALUATION OF RUNNING TIME

In this section, we focus on the time spent on online computation. Because computation in the initial phase of our protocol is mainly offline computation, we just evaluate the time efficiency of aggregation and regression phase.

Figure 3 shows how the time spent on different phases varies with the increasing of d. Dataset *YearPredictionMSD* is used in this experiment and only the top 60000 records are preserved. This means the number of all data $m = 60000$. In addition, m is constant in this experiment. We assume all data are divided evenly among 10 clients and all clients execute step (a) of aggregation in parallel on similar hardware. Therefore, we recalculate the running time of aggregation as follows: 10% of execution time on step (a) of aggregation plus execution time on step (b) of aggregation. The result of this experiment shows the time spent on aggregation and regression increases at a speed of $O(d^2)$ with the increasing

TABLE 1. Accuracy Evaluation based on *q*-bit length of Fractional Part.

Dataset Name	m^*	d^{**}	$\mathcal{K}(\mathbf{A})^{***}$	$Err_{\tilde{\beta}}$				
				$q=10$	$q=20$	$q=30$	$q=40$	$q=50$
Auto MPG	392	7	$7.3\cdot10^{9}$	$1.66\cdot10^{-4}$	$1.07\cdot10^{-7}$	$1.58\cdot10^{-10}$	$1.03\cdot10^{-13}$	$2.05\cdot10^{-16}$
Forest Fires	517	11	$1.9\cdot10^{8}$	$1.08\cdot10^{-4}$	$1.73\cdot10^{-7}$	$1.03\cdot10^{-10}$	$1.65\cdot10^{-13}$	$1.04\cdot10^{-18}$
Wine Quality	4898	11	$1.4\cdot10^{11}$	0.725	$3.52\cdot10^{-3}$	$2.44\cdot10^{-6}$	$2.96\cdot10^{-9}$	$9.58\cdot10^{-13}$
Bike Sharing	17379	12	$6.2\cdot10^{5}$	$3.01\cdot10^{-3}$	$6.73\cdot10^{-8}$	$1.52\cdot10^{-10}$	$2.56\cdot10^{-14}$	$4.55\cdot10^{-17}$
Communities	1994	122	$8.1\cdot10^{8}$	1.062	$1.39\cdot10^{-3}$	$1.21\cdot10^{-6}$	$8.01\cdot10^{-9}$	$1.35\cdot10^{-11}$
YearPredictionMSD	100000	90	$2.7\cdot10^{18}$	$5.90\cdot10^{-5}$	$6.72\cdot10^{-8}$	$5.42\cdot10^{-11}$	$6.67\cdot10^{-14}$	$1.39\cdot10^{-16}$

* m is the number of input data.
** d is the number of data feature.
*** $\mathcal{K}(\mathbf{A})$ is the condition number of matrix $\mathbf{A}$

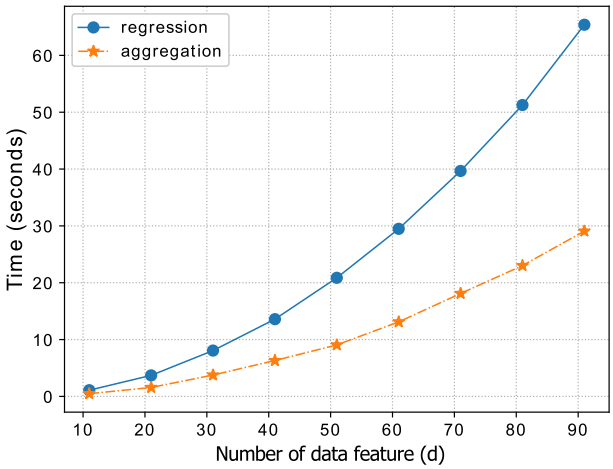

FIGURE 3. Running time variation with increasing of *d*.

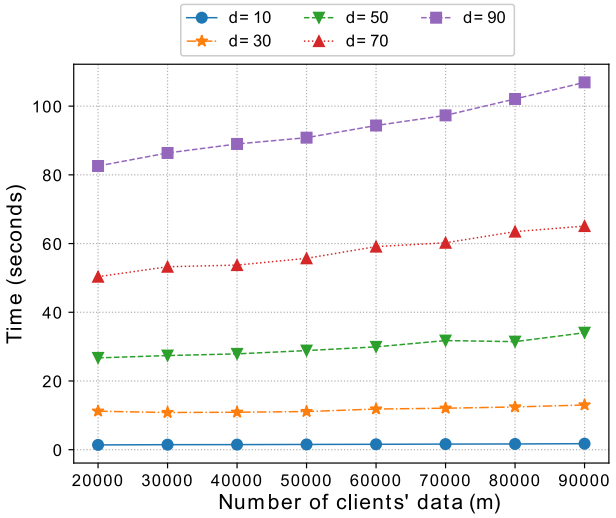

FIGURE 4. Running time variation with increasing of *m* and *d*.

of d. Moreover, the running time of the regression phase grows more quickly.

The results of another experiment are shown in figure 4. In this experiment, both m and d are taken as variable. We set d to 10, 30, 50, 70, and 90, and for each value of d, we survey the running time of the protocol (including the aggregation and regression phase) when m is varied from 20000 to 90000. The results reveal that the running time of the protocol increases with the increasing of m or d, but d has a more obvious influence on the running time. For example, if d is fixed at 90, the running time only rises by 30% when m runs from 20000 to 90000. However, when m is set to 60000, the running time rises by 700% when d increases from 30 to 90. The reason this occurs is that except that each client costs $O(md^2)$ arithmetic operations in step (a) of the aggregation, m is unrelated to all other computation. Thus, our protocol is relatively more sensitive to the variation of d.

D. COMPARISON WITH OTHER PROTOCOLS

In recent years, some protocols have been proposed to tackle the privacy-preserving linear regression or ridge regression on distributed data. Most of these solutions fall into two categories: protocols that combine garble circuits and homomorphic encryption and protocols based on homomorphic encryption alone.

1) COMPARISON WITH NIKOLAENKO *et al.*'s HYBRID PROTOCOL

In Nikolaenko *et al.*'s [10] protocol, garbled circuits are used to solve a linear system. In table 2, we list some experimental results of Nikolaenko *et al.*'s protocol and our protocol running on UCI datasets. The platform descriptions of experiments are as follows.

- *Their platform*: Server (CPU 1.9 GHz, 64 G RAM), Ubuntu 12.04, JDK 1.7
- *Our platform*: Mobile workstation (CPU 1.9 GHz, 32 G RAM), Ubuntu 16.4, OpenJDK 1.8

We confine our laptop to running at 1.9 GHz to make the comparison more convincing. It clearly shows that the protocol using garble circuits is still less efficient than that based on homomorphic encryption alone.

2) COMPARISON WITH CHEN *et al.*'s PROTOCOL

Chen *et al.*'s [19] protocol uses Paillier's and ElGamel's encryption to generate and solve linear equations in encrypted form. Table 3 shows the comparison of experimental results. Due to the adoption of data masking technique, our protocol is more efficient than their protocol under the similar experimental conditions.

TABLE 2. Experimental results of Nikolaenko *et al.*'s and ours.

Dataset	q	Server's running time(sec.)	
		NWIJBT *	Ours
Forest Fires	20	46	5.95
Wine Quality(white)	23	45	5.80
Communities**	23	122	16.98

* The data are reported in paper [10]
** Number of data features d is limited to 20.

TABLE 3. Experimental results of Chen *et al.*'s and ours.

	Chen et al.'s*	Ours
Servers' running time	217.3 sec.	64.35 sec.
Dataset parameter	$m = 10^4, d = 20, q = 30$	
Hardware platform	CPU 2.8 GHz, 4 G RAM Desktop PC	CPU 2.7 GHz, 32 G RAM Mobile Workstation
Software platform	Windows 7, JDK 1.7	Ubuntu 16.4, OpenJDK 1.8

* The data are reported in paper [19]

TABLE 4. Computational cost of the two protocol's regression phase.

	Giacomelli et al.	Ours
Enc	0	$3d^2$
Dec	$d^2 + d$	$2d^2 + 2d$
Exp	$d^3 + d^2$	$d^2 + 2d$
Arithmetic	$(d-1)^3 + d^2 + O(d^3)^*$	$3d^2 + O(d^3)^*$

* $O(d^3)$ is the cost of solving data masked equations

3) COMPARISON WITH GIACOMELLI *et al.*'s PROTOCOL

Finally, we compare our protocol with Giacomelli *et al.*'s [20] protocol, which is also based on homomorphic encryption alone. Moreover, we implement Giacomelli *et al.*'s protocol in C++ language. Therfore, the results in this section all come from our experiment. Both of the protocols take advantage of different data masking techniques. In the aggregation phase, these two protocols adopt the same method for horizontally partitioned data. To compare the two protocols, we only need to focus on the regression phase.

Table 4 shows the computational cost of the two protocol's regression phase; the cost counts all operations of the Evaluator and CSP. The *Enc*, *Dec*, and *Exp* in table 4 represent encryption, decryption and the modular-exponentiation operation in Paillier's cryptosystem. The *Arithmetic* operations not only include the basic arithmetic operations, but also the modular multiplication (*Mul*) in Paillier's cryptosystem.

Table 5 shows the average ratio of time spent on other operations to that on encryption. Obviously, the cost on *Arithmetic* operations is much less than that on other crypto operations. This means the running time of the protocols is mainly determined by *Enc*, *Dec* and *Exp*. Among these operations, *Exp* is a special one because the operation cost rises with the increase of the exponent. When increasing the bit-length of the exponent from 60 to 500, the value of Exp/Enc rises from 6% to 50% approximately.

TABLE 5. Ratio of other operations' cost to encryption's cost.

Dec/Enc	Exp/Enc^*	Mul/Enc	$division/Enc$
90.3%	6%	0.16%	0.06%

* Exponent is a 60-bit integer in the *Exp* operation

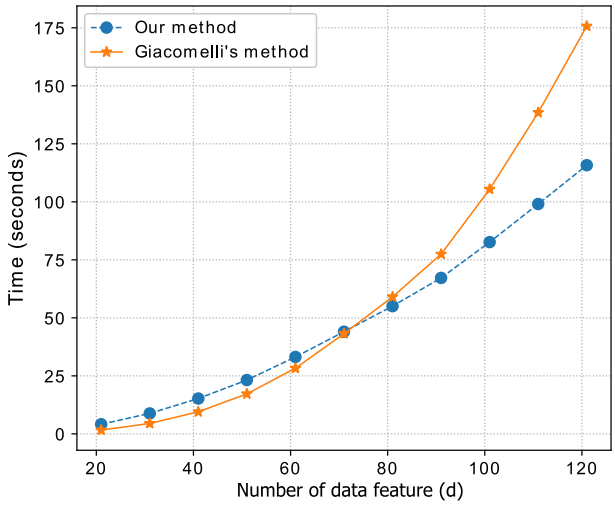

FIGURE 5. Comparison of running time between protocols.

Because *Exp* spends much less time than *Enc* or *Dec*, the running time of our protocol is longer than Giacomelli *et al.*'s protocol when d is small enough. However, the running time of their protocol increases more rapidly because their protocol costs $O(d^3)$ *Exp* operations, but our protocol costs $O(d^2)$ *Exp* operations (see table 4). Figure 5 shows Giacomelli *et al.*'s protocol is more efficient than ours when d is less than 70, but when d is greater than 70 our protocol is better. The top 5000 records of the *YearPredictionMSD* dataset are used in this experiment, and some columns are added to the dataset to fit the experiment. The bit-length of the exponent in the *Exp* operation is designated as 60. Obviously, if a larger number is set for the exponent, the performance of our protocol can exceed their protocol earlier.

When comparing the communication cost, only the regression phase needs to be considered because the approach adopted for aggregation in both protocols is the same. For the regression phase, the communication cost of Giacomelli *et al.*'s protocol is d^2+2d, while our protocol is $3d^2+4d$. The reason for this is the linear equations need to be transmitted only once in Giacomelli *et al.*'s protocol, but three times in our protocol.

Although efficiency differences exist between Giacomelli *et al.*'s protocol and ours, the difference is far smaller than that between protocols belonging to different types. We believe both of protocols are fit for solving the same problem in practice.

VIII. CONCLUSION

In this paper, we propose a protocol that learns a linear regression model over distributed clients' data without leaking any information of the client to the service provider. Theoretical analysis and numerical experiments have been performed

to verify its efficiency, accuracy and security. By taking advantage of the data masking technique, our protocol can be more efficient than most existing protocols. By combining the merits of homomorphic encryption and data masking, our protocol is able to realize a high level of security and accuracy. These advantages make our protocol ideally suited for practical application, especially for realizing a regression module in a privacy-preserving machine learning task.

As future work, we believe the idea about introducing rational number data masking into encrypted data is a basic technique, and we are interested in extending this technique to other privacy-preserving machine learning methods.

REFERENCES

[1] C. C. Aggarwal, P. S. Yu, "A General Survey of Privacy-Preserving Data Mining Models and Algorithms." in *Privacy-Preserving Data Mining*. Advances in Database Systems, vol 34. C. C. Aggarwal, P. S. Yu, Ed. Boston, MA, USA: Springer, 2008, pp 11–52

[2] C. M. O'Keefe, N. M. Good, "Regression output from a remote analysis server." in *Data & Knowledge Engineering*, Vol. 68, no. 11, 2009, pp 1175–1186.

[3] A. R. Khan, C. M. O'Keefe, "Disclosure risk reduction for generalized linear model output in a remote analysis system," in *Data & Knowledge Engineering*, Vol. 111, 2017, pp 90–102

[4] K. Chen, L. Liu. "Privacy-Preserving Multiparty Collaborative Mining with Geometric Data Perturbation." *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 12, 2009, pp 1764–1776.

[5] A. Shamir, "How to Share a Secret." *Comm ACM*, vol. 22, no. 11, 1979, pp 612–613

[6] R. Hall, S. E. Fienberg, Y. Nardi, "Secure multiple linear regression based on homomorphic encryption." *J. Official Statistics*, vol. 24, no. 4, 2011, pp 669–691

[7] P. Mohassel, Y. Zhang. 2017. "SecureML: A System for Scalable Privacy-Preserving Machine Learning." In *IEEE Symposium on Security and Privacy*, 2017, pp 19–38

[8] O. Catrina, A. Saxena, "Secure computation with fixed-point numbers." In *International Conference on Financial Cryptography and Data Security*, 2010, pp 35–50

[9] A. C. Yao, "How to generate and exchange secrets." in *Annual Symposium on Foundations of Computer Science (Proceedings) 27th*, 1986, pp 162–167.

[10] V. Nikolaenko, U. Weinsberg, *et al.* "Privacy-Preserving Ridge Regression on Hundreds of Millions of Records." in *Proc. IEEE Symp. Secur. Privacy*, 2013, pp 334–348

[11] M. Naor, B. Pinkas, "Efficient oblivious transfer protocols." in *Proc Annu ACM SIAM Symp Discrete Algorithms*, January, 2001, pp 448–457.

[12] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," in *Foundations of Secure Computation*, R. Demillo, D. Dobkin, A. Jones, and R. Lipton, Eds. New York: Academic, 1978, pp 169–180.

[13] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes." in *EUROCRYPT 1999 - 7th International Conference*, 1999, pp 223–238

[14] T. Elgamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," in *Advances in Cryptology - Proceedings of CRYPTO 84*, 1985, pp 10–18.

[15] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices." In *41st ACM Symposium on Theory of Computing (STOC)*, 2009, pp 169–178

[16] T. Graepel, K. Lauter, M. Naehrig, "ML Confidential: Machine Learning on Encrypted Data." in *ICISC 2012 - 15th International Conference*, 2012, pp 1–21

[17] A. Gascón, P. Schoppmann, B. Balle, *et al.* "Privacy preserving distributed linear regression on high-dimensional data." in *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 4, 2017, pp 345–364.

[18] S. Hu, W. Qian, J. Wang, *et al.* "Securing Fast Learning! Ridge Regression over Encrypted Big Data." in *2016 IEEE TrustCom-BigDataSE-ISPA*, 2016, pp 19–26

[19] Y. R. Chen, A. Rezapour, W. G. Tzeng, "Privacy-preserving ridge regression on distributed data." in *Information Sciences*, vol. 451-452, 2018, pp 34–49.

[20] I. Giacomelli, S. Jha, M. Joye, *et al.* "Privacy-Preserving Ridge Regression with only Linearly-Homomorphic Encryption." in *Applied Cryptography and Network Security. ACNS 2018.*, pp 243-261

[21] S. Goldwasser, S. Micali, "Probabilistic encryption," in *J. Comput. Syst. Sci.*, vol. 28, no. 2, 1984, pp. 270–299.

[22] A. F. Karr, X. Lin, P. S. Ashish, *et al.* "Secure Regression on Distributed Databases." in *Journal of Computational and Graphical Statistics*, vol. 14, no. 2, 2005, pp 263–279.

[23] A. F. Karr, X. Lin, J. P. Reiter, *et al.* "Privacy Preserving Analysis of Vertically Partitioned Data Using Secure Matrix Products". in *Journal of Official Statistics*, vol. 25, no. 1, 2009, pp 125–138.

[24] M. Dahl, C. Ning, T. Toft, "On secure two-party integer division." in *Financial Cryptography and Data Security*. 2012. pp 164–178

[25] C. Ugwuoke, E. Zekeriya, R. L. Lagendijk,"Secure Fixed-point Division for Homomorphically Encrypted Operands". in *13th International Conference - ARES 2018*, 2018, pp 1–10

[26] T. Veugen. "Encrypted integer division and secure comparison." in *International Journal of Applied Cryptography*, vol. 3, no. 2, 2014, pp 166–180.

[27] N. J. Higham, "Chapter 7 Perturbation Theory for Linear Systems." in *Accuracy and Stability of Numerical Algorithms*, SIAM, USA, 1997,pp 131–150

[28] UCI, "Machine Learning Repository." [Online]. Available: http://archive.ics.uci.edu/ml/datasets.html

[29] J. O. Rawlings, S. G. Pantula, *et al.* "Applied regression analysis: a research tool.", Springer, New York, 1998, pp 75–93

GUOWEI QIU received the B.Sc. degree in computational mathematics and the M.Sc. degree in mechanical and electronic engineering from Xi'an Jiaotong University, China, in 1992 and 2000, respectively.

He worked as a Software Engineer with the Xi'an Institute of Aeronautical Computing Technology, from 1992 to 1997. Since 2000, he has been a Lecturer with the School of Electronic and Information Engineering, Xi'an Jiaotong University. Early in his career, he participated in some aeronautical technology research projects and had been awarded second prize at ministerial level. He is the author of more than ten books. His current research interests include information security, data mining, and machine learning.

XIAOLIN GUI received the B.Sc., M.Sc., and Ph.D. degrees in computer science from Xi'an Jiaotong University (XJTU), China, in 1993 and 2001, respectively.

Since 1988, he has been an Active Researcher in network computing, network security, and wireless networks with XJTU, where he is currently a Professor. He has been the Director of the Key Laboratory of Computer Network, XJTU, since 2008. He is the author of more than 100 research articles and nine books and had won several awards at the ministerial or national level. His current research interests include the secure computation of open network systems, including grid, P2P, and cloud; dynamic trust management theory; and development on community networks.

YINGLIANG ZHAO received the B.Sc., M.Sc., and Ph.D. degrees in computational mathematics from Xi'an Jiaotong University (XJTU), China, in 1989, 1992, and 1998, respectively.

In 2002, he became a member of the Computer Teaching and Experiment Center, XJTU. He is currently a Professor with the School of Computer Science and Technology, XJTU. His current research interests include the secure computation of open network systems, dynamic trust management theory, and the IoT applications.