Information Sciences 596 (2022) 119–136

Contents lists available at ScienceDirect

Information Sciences

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

Toward practical privacy-preserving linear regression

Wenju Xu [a,b], Baocang Wang [a,b,*], Jiasen Liu [c], Yange Chen [a,b], Pu Duan [d], Zhiyong Hong [e,f]

[a] The State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China
[b] Cryptographic Research Center, Xidian University, Xi'an 710071, China
[c] College of Cryptography Engineering, Engineering University of People's Armed Police, Xi'an 710086, China
[d] Ant Group, Hangzhou 310000, China
[e] Facility of Intelligence Manufacture, Wuyi University, Jiangmen 529020, China
[f] Yue-Gang-Ao Industrial Big Data Collaborative Innovation Center, Wuyi University, Jiangmen 529020, China

ARTICLE INFO

Article history:
Received 1 September 2021
Received in revised form 20 December 2021
Accepted 3 March 2022
Available online 7 March 2022

Keywords:
Linear regression
Rational numbers
Linearly homomorphic encryption
Multi-key
Fully homomorphic encryption

ABSTRACT

Linear regression is an ordinary machine learning algorithm that models the relation between the input values and the output ones with underlying linear functions. Giacomelli et al. (ACNS 2018) proposed the first system training the linear regression model over the rational numbers using only linearly homomorphic encryption. However, we find their system model is not applicable. A third authority generates the public key and secret key, which are used to encrypt and decrypt all the data sets. Then the privacy of data sets is in the risk of leakage even if the third authority is assumed to have no access to encrypted data sets. In this paper, we improve the system model in order to design a more practical linear regression algorithm over the rational numbers from the view of security. Concretely, every data owner generates his own public key and secret key, independent on a third authority. An improved multi-key fully homomorphic encryption over complex numbers is utilized to construct our linear regression algorithm with a preprocessing phase, which can directly encrypt rational numbers, support computations over ciphertexts under multi keys and obviate the rational reconstruction technique as Giacomelli et al.. Furthermore, performance analyses demonstrate that our algorithm is more feasible and practical.

1. Introduction

Linear regression is a popular statistical analysis algorithm training the data sets with linear functions, which is simple but very useful, such as in epidemiology, finance [5] and economics. In the aspect of finance, capital asset pricing model [20] uses the linear regression and Beta coefficient to analyze and calculate the systemic risk of investment. Then we can utilize the linear regression to quantitatively relate the investment and the risk and for future prediction of the risk in case of a investment.

To enhance the efficiency and accuracy of linear regression model, Euclidean norm can be applied to lessen the overfitting of ordinary least squares regression without adding computational cost. A large and diverse data sets from various data owners ensure the linear regression model more accurate and closed to the reality. Meanwhile, gradient descent [14] and its

* Corresponding author at: The State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China.
E-mail addresses: xuwenjuxwj@126.com (W. Xu), bcwang@xidian.edu.cn (B. Wang), 869550196@qq.com (J. Liu), ygchen428@163.com (Y. Chen), p.duan@antgroup.com (P. Duan), hzy_wyu@163.com (Z. Hong).

https://doi.org/10.1016/j.ins.2022.03.023
0020-0255/

extensions including batch, stochastic, mini-batch gradient descents [27] by iterations many times contribute to a local optimum model. However, the local optimum model may not be the exact one. Meanwhile in many cases, data owners are disinclined to share their data with any others since the data sets may contain some sensitive information of them, especially in the big data era. Hence, a privacy-preserving linear regression algorithm becomes more popular.

There are several non-cryptographic approaches for privacy-preserving linear regression including anonymization, diversity, perturbation and randomization [23,28]. However, some common attacks may break the validity of these methods. Moreover, the fact that a large number of information append to hide the data causes a lower utility of the data. On the basis of these cases, homomorphic encryption as an important cryptographic primitive can encrypt data and meanwhile allow direct computations over ciphertexts without access to the secret key, and hence is well suited for the privacy issue of linear regression.

The concept of homomorphism was proposed back to 1978. A homomorphic addition cryptosystem Paillier scheme was presented in [26], which is widely utilized. Barbosa et al. [6] put forward a primitive called labeled homomorphic encryption based on Paillier cryptosystem, which was applied to design a privacy-preserving linear regression algorithm by Nikolaenko et al. [25]. Until 2009, Gentry [16] constructed the first fully homomorphic encryption (FHE), which made a big progress for homomorphic cryptography. Since then, Dijk et al. [13] investigated the FHE scheme over the integers, Brakerski et al. [8] come up with a leveled FHE scheme without bootstrapping BGV scheme, which both support various computations over encrypted data. Furthermore, batching [8], Chinese Reminder Theorem and Single Instruction Multiple Data [30] were proposed in order to significantly improve the efficiency of FHE. Arimitsu et al. [4] utilized the relatively efficient BGV scheme, to design a fast, simple and exact privacy-preserving linear regression model compared with gradient descent method under a semi-honest model.

1.1. Related Work

In this section, we mainly bring in some related work with privacy-preserving linear regression algorithms from homomorphic encryption.

The privacy-preserving machine learning was first introduced by [1,22]. Later on, some meaningful researches about privacy-preserving linear regression come into being in [3,17,25]. Nikolaenko et al. [25] initially trained a privacy-preserving linear regression model with linearly homomorphic encryption (LHE) and Yao's garbled circuits in a two-server system on a horizontally-partitioned sample sets. The data sets are validly protected by a semantical security LHE scheme. However, Yao's two-party protocol with garbled circuits is inefficient. As Giacomelli et al. [17] reported, the lookup table for a gate represented by garbled circuits is of around 30 bytes (80-bit security).

Following the two-server system, Giacomelli et al. also proposed a novel system training a ridge linear regression model over the real interval $[-\delta, \delta]$ with at most l digits in the fractional part (the parameter is $l < 10$ in the implementation of [17].) discarding the inefficient Yao's two-party protocol. Concretely, the labeled homomorphic encryption in [6] was utilized to protect the sample sets of users both in the horizontal and vertical partitions. Compared with general linearly homomorphic encryption schemes, the labeled homomorphic encryption was well-suitable for linear regression model, since the homomorphic computations over the encrypted sample sets were transparent and could be expedited by offline. And the Lagrange-Gauss algorithm in [15] was available to achieve the transformations from integers to real numbers, hence the messages of real numbers can be represented by integers to be encrypted by data owners with LHE. As Giacomelli et al. claimed, for sample sets of 10 millions instances and 20 features in horizontal partition (into ten equal-sized parts), they ran a privacy-preserving regression model under 2 min[1], while [25] needed more than 50 min[2].

Motivated by Giacomelli et al., Akavia et al. [2] leveraged the FHE techniques Chinese Remainder Theorem and Single Instruction Multiple Data representations to accelerate the regularized linear regression models over the real interval $[0, 1)$. The BGV based on Ring Learning with Errors assumption as a linearly homomorphic encryption (not a fully homomorphic encryption) was utilized to significantly reduce both the runtime of homomorphic operations and runtime complexity on packed encrypted data. As they said for a linear regression task of data size 1000 with 40 features, they could learn a model in a total of 3 s for the homomorphic operations in HElib [18] library, compared to more than 100 s reported in [17].

However, we find two improper places surrounding the linear regression algorithms in [2,17]. One is the system model, the other is the parameter setting for the transformation from integers to rational numbers.

- *System model.* In [2,17], an auxiliary server (Crypto Service Provider, CSP) generates the public key and secret key for the data owners. Every data owner encrypts his data with the same public key. As a result, the ciphertexts of every data owner seem to be transparent for CSP, even the auxiliary server is assumed to have no access to the ciphertexts: he has the ability to decrypt them into plaintexts, then the data sets of all data owners are in the risk of leaking the information. This violates the original intuition of protection. Meanwhile, every data owner shares common pair of public key

[1] Timing on a 2.6 GHz 8 GB RAM machine running Linux 16.04 with 80-bit security.
[2] Timing on a 1.9 GHz 64 GB RAM machine running Linux 12.04 with 80-bit security.

and secret key, which may collide the will of every data owner and is not practical. They must interact with more communication and computation overheads to generate a common pair of keys. For practicality the situation where every data owner generates his own public key and secret key is much better.

- *Parameter setting.* The important parameter N largely depends on the rational reconstruction. As Giacomelli et al. analyzed, they thought only the output $\mathbf{A}^{-1}$ need to be reconstructed, and intermediate values such as $\mathbf{AM}$ do not affect the bound of N. That is, the values of $\det(\mathbf{A})$ and $||\text{adj}(\mathbf{A})\mathbf{b}||_\infty$ affect the bound of N, where $\det(\mathbf{A}), \text{adj}(\mathbf{A})$ denote the determinant and adjoint matrix of $\mathbf{A}$ respectively. But the intermediate values $\mathbf{AM}, \mathbf{b}^T + \mathbf{Ar}^T$ actually have effects on it due to the special rational reconstruction technique. The values of $\mathbf{M}, \mathbf{r}$ change the numerator and denominator of $\mathbf{A}^{-1}$. Hence only when

$$N > 2\det(\mathbf{AM}) \cdot ||\text{adj}(\mathbf{AM})\left(\mathbf{b}^T + \mathbf{Ar}^T\right)||_\infty$$

holds, the rational numbers can be recovered. Refer to Section 4.2 for more details.

1.2. Our Contribution

Motivated by [2,17], we propose a more practical privacy-preserving linear regression algorithm in an enhanced system model. Specifically, the contributions are unfolded below.

- *Improved system model.* In our model, every data owner generates his own public key and secret key encrypting and decrypting by himself to protect his data instances. In other words, we abort the auxiliary server CSP, and every data owner directly interacts with Machine-Learning Engine.
- *Improved CKKS scheme.* An improved CKKS scheme under multi keys is proposed to encrypt the rational data for every data owner. We extend the CKKS scheme in [12] into an FHE scheme under multi keys, which is well-suitable for our linear regression scenario to deal with the issue that every data owner is inclined to enjoy his own public key and secret key.
- *More efficient performance.* We make a preprocessing phase with the improved CKKS scheme, in order to generate an invertible $\mathbf{A}$ from all data sets. Some techniques of our improved CKKS scheme including batching, Chinese Reminder Theorem, Single Instruction Multiple Data and the matrix multiplications in [19] significantly improve the total computation overheads. Experimental analyses demonstrate that it takes 6 ms with $d = 40$ for our algorithm in LHE.op and almost 1932 and 4 times less than the one in [17,2] respectively. The total running time for a 1000×40 linear regression task is almost 170 times less than [17].

1.3. Paper Organization

The rest of this paper is organized as follows. In Section 2, we briefly introduce some preliminaries including the detailed linear regression and our improved CKKS scheme under multi keys. The linear regression algorithm proposed by Giacomelli et al. will be briefly presented in Section 3. In Section 4, we present some analyses about Giacomelli et al.'s linear regression algorithm and elaborate our algorithm in Section 5. The performance evaluations consisting of theoretical and experimental analyses are shown in Section 6. Finally, a conclusion is given in Section 7.

2. Preliminaries

Initially, we give some notations that will be used throughout this paper in Table 1. Then the linear regression, the probability of column full-rank matrix and our improved CKKS scheme under multi keys are illustrated.

2.1. Linear Regression

Linear regression is a widely-used statistical algorithm in machine learning [9,31] for modeling the relationship $\mathbf{w}$ between properties of data instances $\mathbf{x}_i \in \mathbb{Q}^{1\times d}$ and an outcome $y_i \in \mathbb{Q}$ such that $\mathbf{x}_i\mathbf{w}^T = y_i$ for $i \in [n]$. In a statistical sense, collecting more data instances from various data owners makes the model more accurate. Hence, we consider the following assumption: there are m data owners DO_j consisting of k_j items data $\left\{(\mathbf{x}_{j1}, y_{j1}), (\mathbf{x}_{j2}, y_{j2}), \ldots, \left(\mathbf{x}_{jk_j}, y_{jk_j}\right)\right\}$ for $j \in [m]$. Every $n_j (n_j < k_j)$ items from the j-th data owner will be picked out as the total n data instances satisfying $\sum_{j=1}^{m} n_j = n$. For simplicity, we denote $\mathbf{Xw}^T = \mathbf{y}^T$, where the matrix $\mathbf{X} \in \mathbb{Q}^{n\times d}$ is composed of the vector $\mathbf{x}_i$ as the i-th row and the vector $\mathbf{y}^T \in \mathbb{Q}^{n\times 1}$ consists of the value y_i as the i-th element.

One common way to compute such a model is to use the squared-loss function and a regularization term to avoid overfitting, that is, the model $\mathbf{w}^T$ will be computed by minimizing the function $||\mathbf{Xw}^T - \mathbf{y}^T||^2 + \lambda||\mathbf{w}^T||^2$. From the monotonicity of

Table 1
Notations.

Notation	Description
$[n]$	The set $\{1, 2, \ldots, n\}$
$\lfloor \delta \rceil$	The rounding of δ down the nearest integer
mod q	Least absolute remainder set $(-q/2, q/2]$
$\mathbf{x}$	Row vector
$\mathbf{x}[t]$	The t-th component of vector $\mathbf{x}$
$\mathbf{x}^T$	The transposition of row vector $\mathbf{x}$
$\|\|\mathbf{x}\|\|$	The 2-norm of $\mathbf{x}$
$\|\|\mathbf{x}\|\|_\infty$	The infinite norm of $\mathbf{x}$
$\mathbb{Q}$	Space of rational number
$\mathbb{C}$	Space of complex number
$\mathbb{Z}_q$	Integer set modulo q
$\mathbb{Q}[X]/\left(X^N+1\right)$	Rational polynomial ring modulo X^N+1
$\mathbf{x} \in \mathbb{Q}^d$	The row vectors $\mathbf{x}$ of 1-by-d over $\mathbb{Q}$
$\mathbf{X} \in \mathbb{Q}^{n\times d}$	A matrix $\mathbf{X}$ of n-by-d over $\mathbb{Q}$
$\mathbf{X} \in \mathbb{Z}_q^{n\times d}$	A matrix $\mathbf{X}$ of n-by-d over $\mathbb{Z}_q$
$\mathbf{X}[:, j]$	The j-th column of matrix $\mathbf{X}$
$\mathbb{R}_q$	The polynomials $\mathbb{Z}_q[X]/\left(X^N+1\right)$

derivative, we can attain $\mathbf{w}^T$ by solving the equation $\mathbf{A}\mathbf{w}^T = \mathbf{b}^T$, where $\mathbf{A} = \mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}, \mathbf{b}^T = \mathbf{X}^T\mathbf{y}^T$, and $\mathbf{I}$ is an identity matrix of order d. If the matrix $\mathbf{X}$ is column full-rank[3], implying $\mathbf{A}$ is invertible, then the model $\mathbf{w}^T$ can be solved by $\mathbf{w}^T = \mathbf{A}^{-1}\mathbf{b}^T$. As [17], we consider the data instances from the real interval $[-\delta, \delta]$ with l digits in the fractional part in this paper.

2.2. Our Improved CKKS Scheme under Multi Keys

Cheon et al. [12] proposed an FHE under a single key for approximate arithmetic called CKKS, which supports homomorphic evaluations with underlying real numbers. To deal with a practical scenario: data owners are with different secret keys in an linear regression model over $\mathbb{Q}$, we propose a multi-key FHE for approximate arithmetic based on [8,10], denoted by our improved CKKS scheme under multi keys.

Before introducing our improved CKKS scheme under multi keys, some necessary notations, formulas and algorithms are described briefly.

(1) Set $\mathbb{R}_q = \mathbb{Z}_q[X]/\left(X^{\mathcal{N}}+1\right)$ and with L decreasing modulus $q = q_L \gg q_{L-1} \gg \cdots \gg q_0$ for each level and a small integer p coprime with all $q_{L_i} = q_{L-i}$ for $i \in \{0, 1, \ldots, L\}$. Let $\beta_{L_i} = \lfloor \log\ q_{L_i} \rfloor + 1$ and χ be an error distribution over $\mathbb{R}$ with coefficient bound $[-B, B]$.

(2) The canonical embedding (bijection) is described as

$$\sigma : \mathbb{Q}_p[X]/\left(X^{\mathcal{N}}+1\right) \to \mathbb{C}^{\mathcal{N}/2}$$

by $\sigma(a) = \left(a(\zeta), a(\zeta^5), \ldots, a\left(\zeta^{2\mathcal{N}-3}\right)\right)$, where $\mathcal{N}$ is a power of 2 and $\zeta = \exp(\pi i/\mathcal{N})$ (i is the unit of complex number) is the principal $2\mathcal{N}$-th complex root of unity.

(3) The RGSW algorithm in [10] and the key switching technique in [8] are applied to generate the evaluation key and evaluation algorithm for our improved CKKS scheme under multi keys. Details are omitted here, but the relevant symbols and algorithms are used, such as BitDecomp, Powersof2, Key Switching algorithm, Modulus Switching algorithm, RGSW.Enc algorithm, RGSW.EncRand algorithm, MKFHE.CTExt algorithm, MKFHE.EVKGen algorithm, MKFHE.EvalAdd algorithm, and MKFHE.EvalMult algorithm.

The details of our improved CKKS scheme under multi keys are given below, including Key Generation, Encryption, Expand, Evaluation and Decryption.

- Key Generation(KeyGen) $(pk, sk, evk) \leftarrow (1^\kappa, m, L)$: Taking the security parameter κ, m number of parties and circuit depth L as input, generate the keys for the j-th party, where $j \in [m]$.

  1) Randomly pick $z_{L_i j} \leftarrow \chi$, set the private key as $sk_{L_i j} = \mathbf{s}_{L_i j} = (1, -z_{L_i j}) \in \mathbb{R}^2_{q_{L_i}}$ for $i \in \{0, 1, \ldots, L\}$.

[3] In general, the matrix $\mathbf{X}$ is assumed to be column full-rank since $d < n$ in real applications, as [2,17,25]. We will discuss how to achieve the column full-rank of X additionally in this paper.

2) Randomly choose $\mathbf{a}_{L_i j} \in \mathbb{R}_{q_{L_i}}^{2\beta_{L_i}}$ and $\mathbf{e}_{L_i j} \leftarrow \chi^{2\beta_{L_i}}$, compute $\mathbf{b}_{L_i j} = \mathbf{a}_{L_i j} \cdot z_{L_i j} + p\mathbf{e}_{L_i j} \left(\mathrm{mod}\, q_{L_i}\right)$. Set the public key as $pk_{L_i j} = (\mathbf{b}_{L_i j}, \mathbf{a}_{L_i j}) \in \mathbb{R}_{q_{L_i}}^{2\beta_{L_i} \times 2}$ for $i \in \{0, 1, \ldots, L\}$.

3) Compute $\mathrm{RGSW.Enc}\left(\mathrm{Powersof2}\left(\mathbf{s}_{L_i j}[t]\right), pk_{L_{i-1} j}\right)$ for $t \in \left[2\beta_{L_i}\right]$, get

$$\begin{aligned}\phi_{t,L_i j} = &\ \mathrm{RGSW.Enc}_{\mathbf{s}_{L_{i-1} j}}\left(\mathrm{Powersof2}\left(\mathbf{s}_{L_i j}[t]\right)\right)\\ = &\ r_{t,L_i j}\left(\mathbf{b}_{L_{i-1} j}, \mathbf{a}_{L_{i-1} j}\right) + p\mathbf{E}_{t,L_i j} + \mathrm{Powersof2}\left(\mathbf{s}_{L_i j}\right)[t]\mathbf{G},\end{aligned}$$

and $\mathbf{F} = \mathrm{RGSW.EncRand}\left(r_{t,L_i j}, pk_{L_{i-1} j}\right)$, also compute

$$\begin{aligned}\varphi_{t,L_i j} = &\ \mathrm{RGSW.Enc}_{\mathbf{s}_{L_{i-1} j}}\left(\mathrm{BitDecomp}\left(\mathbf{s}_{L_i j}[t]\right)\right)\\ = &\ r'_{t,L_i j}\left(\mathbf{b}_{L_{i-1} j}, \mathbf{a}_{L_{i-1} j}\right) + p\mathbf{E}'_{t,L_i j} + \mathrm{BitDecomp}\left(\mathbf{s}_{L_i j}\right)[t]\mathbf{G},\end{aligned}$$

together with

$$\mathbf{F}' = \mathrm{RGSW.EncRand}\left(r'_{t,L_i j}, pk_{L_{i-1} j}\right),$$

where $r_{t,L_i j}, r'_{t,L_i j} \leftarrow \chi, \mathbf{E}_{t,L_i j}, \mathbf{E}'_{t,L_i j} \leftarrow \chi^{2\beta_{L_i} \times 2}$ and $\mathbf{G}$ is the gadget matrix.
The evaluation key is

$$evk_j = \left\{\left(\phi_{t,L_i j}, \mathbf{F}\right), \left(\varphi_{t,L_i j}, \mathbf{F}'\right)\right\}_{t \in \left[2\beta_{L_i}\right], i \in \{0,1,\ldots,L\}}.$$

- Encryption(Enc) $ct \leftarrow \left(\mathbf{m}, pk_{L_i j}\right)$: Given a message $\mathbf{m} = (m_0, m_1, \ldots, m_{\mathcal{N}/2-1}) \in \mathbb{C}^{\mathcal{N}/2}$ and a scaling factor $\triangle$,

  1) First generate the plaintext polynomial[4]

  $$m(X) = \lfloor \triangle \cdot \sigma^{-1}(\mathbf{m}) \rceil \in \mathbb{R}_p,$$

  where $\lfloor \cdot \rceil$ represents the rounding to the nearest integer.

  2) Randomly sample $r, e, e\prime \leftarrow \chi$, then compute the following equation

  $$\mathbf{c} = (c^0, c^1) = \left(r\mathbf{b}_{Lj}[1] + pe + m(X), r\mathbf{a}_{Lj}[1] + pe\prime\right) \in \mathbb{R}_{q_L}^2.$$

  Output a tuple $ct = (\mathbf{c}, \{j\}, L)$ as the ciphertext of level-L for the j-th party, where $j \in [m]$.
- Expand $\overline{ct} \leftarrow (S, ct_1, \ldots, ct_m)$: Given a set S and m number of ciphertexts of level L_i, where $ct_j = (\mathbf{c}_j, \{j\}, L_i)$ for $i \in \{0, 1, \ldots, L\}, j \in [m], j \in S$, generate the expanded ciphertext for the j-th party as

  $$\overline{ct}_j = \left(\left(\mathbf{0}, \ldots, \underbrace{\mathbf{c}_j}_{j}, \ldots, \mathbf{0}\right), S, L_i\right).$$

  - Evaluation(Eval) $\left(\left(pk_{i_1}, \ldots, pk_{i_\theta}\right), evk_S, (ct_1, \ldots, ct_\alpha)\right)$: Assume the ciphertexts are at the same level L_i ($i \in \{0, 1, \ldots, L\}$, if not, use the Key Switching algorithm and Modulus Switching algorithm to achieve it). For $k \in [\alpha]$, parse $ct_k = (\mathbf{c}_k, S_k, L_i)$, let $|S_k| = \theta_k, S = \bigcup_{k=1}^{\alpha} S_k = \{i_1, \ldots, i_\theta\}, pk_S = \left(pk_{i_1}, \ldots, pk_{i_\theta}\right), sk_S = \left(sk_{i_1}, \ldots, sk_{i_\theta}\right)$, then the extended ciphertexts $\bar{\mathbf{c}}_k, \bar{\mathbf{c}}'_k$ under pk_S to evaluate the circuit as MKFHE.
    $\mathrm{EvalAdd}\left(evk_S, \bar{\mathbf{c}}_k, \bar{\mathbf{c}}'_k\right)$ and $\mathrm{MKFHE.EvalMult}\left(evk_S, \bar{\mathbf{c}}_k, \bar{\mathbf{c}}'_k\right)$.
    1) $\mathrm{MKFHE.EvalAdd}\left(evk_S, \bar{\mathbf{c}}_k, \bar{\mathbf{c}}'_k\right)$: the addition of the extended ciphertexts is the component-wise addition. Compute $\bar{\mathbf{c}}^+ = \bar{\mathbf{c}}_k + \bar{\mathbf{c}}'_k \mathrm{mod}\, q_{L_i}$ under the secret key sk_S, to generate a ciphertext at the level L_i. The Modulus Switching algorithm can be utilized to generate a corresponding ciphertext under the secret key sk_S at the level L_{i-1} from $\bar{\mathbf{c}}^+$.
    2) $\mathrm{MKFHE.EvalMult}\left(evk_S, \bar{\mathbf{c}}_k, \bar{\mathbf{c}}'_k\right)$: the multiplication of the extended ciphertexts is the tensor product. First, compute $\bar{\mathbf{c}}^\times = \bar{\mathbf{c}}_k \otimes \bar{\mathbf{c}}'_k \mathrm{mod}\, q_{L_i}$ under the secret key $sk_S \otimes sk_S$ at the level L_i. Second, use the Key Switching algorithm and RGSW algorithm over $\bar{\mathbf{c}}^\times$ to generate a ciphertext $\bar{\mathbf{c}}^{\times 1}$ under the secret key sk_S at the level L_i. Third, the Modulus Switching algorithm helps generate a ciphertext $\bar{\mathbf{c}}^{\times 2}$ under the secret key sk_S at the level L_{i-1}.
- Decryption(Dec) $\mathbf{m} \leftarrow (ct, sk_S)$: Upon the receipt of a ciphertext tuple $ct = (\mathbf{c}, S, L_i)$, where $\mathbf{c} = \left(\mathbf{c}_{i_1}, \ldots, \mathbf{c}_{i_\theta}\right)$, the decryption algorithm can be performed by two steps.

  1) A threshold decryption protocol accompanying every party, with $sk_S = \left(\mathbf{s}_{L_i, i_1}, \ldots, \mathbf{s}_{L_i, i_\theta}\right)$ to retrieve the corresponding plaintext $m(X)$ is first performed. The i_j-th party ($i_j \in S$) performs with his own secret key $\mathbf{s}_{L_i, i_j}$ and outputs[5]

[4] In this paper, the main operation over ciphertexts to be decrypted is the linear operation. Hence the approximate scaling factor $\Delta = 2^{40}$, and the corresponding precision of 30 bits is enough.

[5] The Decryption algorithm largely depends on the Smudging Lemma [24]. The Smudging Lemma claims that $e_1 + e_2$ is statistically indistinguishable e_2 when $B_1/B_2 = \mathrm{negl}(\kappa)$, where $e_1 \in [-B_1, B_1]$ is a fixed integer, $e_2 \in [-B_2, B_2]$ is according to an uniform distribution, and $\mathrm{negl}(\kappa)$ is negligible function about κ.

$$\gamma_{i_j} = \mathbf{c}_{i_j} \cdot \mathbf{s}_{L_i,i_j} + e_{i_j}^{sm} \left(\mathrm{mod}\ q_{L_i}\right) (\mathrm{mod}\ p),$$

where $e_{i_j}^{sm} \in \left[-B_{smdg}, B_{smdg}\right]$ is chosen uniformly to ensure the IND-CPA + security against the weakness of the CKKS scheme[6]. Then $m(X)$ can be recovered by

$$\mathrm{Dec1}(ct, sk_S) = m(X) = \sum_{i_j \in S} \gamma_{i_j} (\mathrm{mod}\ p).$$

2) After obtaining the plaintext polynomial $m(X)$, the underlying message can be recovered with a high precision by

$$\mathrm{Dec2}(ct, sk_S) = \mathbf{m} = \lfloor \triangle^{-1} \cdot \sigma(m(X)) \rceil.$$

The correctness and homomorphism of our improved CKKS scheme under multi keys holds since the expanded ciphertexts are just the concentration of all the ciphertexts of involved parties. The transformation of ciphertext evaluations can be performed via Key Switching algorithm, Modulus Switching algorithm and RGSW algorithm. Refer to [8,10–12,21] for more details.

In terms of the security of our improved CKKS scheme under multi keys, we utilize the Smudging Lemma to ensure that the decryption algorithm will not reveal more information about the plaintext for passive adversaries as [21] emphasized, even any involved party are faced with many times decryption. In other words, our improved CKKS scheme achieves the security of IND-CPA+, following from the IND-CPA security of CKKS scheme and Smudging Lemma. We prefer to add some noises rather than restrict the power of the adversaries to avoid the passive adversaries, since the latter is uncontrollable in real life.

Remark. Now we intuitively express the differences between our multi-key CKKS and the original CKKS in [12] as follows.

(1) Our multi-key CKKS is in the setting of multi keys, and supports computations over ciphertexts under different keys, while the original CKKS is in the setting of single key, whose evaluations over ciphertexts are under the same key.
(2) Compared with the original CKKS, there is an additional ciphertext expanded algorithm for our multi-key CKKS. This algorithm is utilized to generate the ciphertexts under the public keys of all involved parties from the ciphertexts under any involved party's public key. As a result, the ciphertext size of our multi-key CKKS is larger than the one of original CKKS.
(3) The evaluation algorithm is different, especially the relinearization for ciphertext multiplications. First, the ciphertext addition of our multi-key CKKS is performed as a component-wise addition, where every component can be regraded as a ciphertext of original CKKS. Second, the relinearization for ciphertext multiplications of our multi-key CKKS invokes the RGSW algorithm in [10], while big modulus is applied in the original CKKS.
(4) Our multi-key CKKS supports the threshold decryption protocol for every involved party while the threshold decryption protocol is useless for the original CKKS. As a result, the communication of our multi-key CKKS in decryption is more than the one in the original CKKS. Nevertheless, every involved party decrypts the ciphertext partially with his own secret key, without leaking the information of his secret key.

3. Giacomelli et al.'s Linear Regression

In this section, we mainly describe the linear regression algorithm proposed by Giacomelli et al..

3.1. System model

As shown in Fig. 1, the system model in [17] involves three entities. The descriptions of these entities are given below.

- Data Owners (DO): there are m Data Owners DO_1, DO_2, ..., DO_m; every DO_j owns k_j items data $\left\{(\mathbf{x}_{j1}, y_{j1}), (\mathbf{x}_{j2}, y_{j2}), \ldots, \left(\mathbf{x}_{jk_j}, y_{jk_j}\right)\right\}$ for $j \in [m]$. The data instances of every data owner are encrypted with the public key generated by Crypto Service Provider (CSP).
- Machine-Learning Engine (MLE): it wants to run a linear regression algorithm over the data instances from data owners DO_1, DO_2, ..., DO_m, but only has access to the encrypted data. Then he interacts with CSP and attain the linear regression model with the help of CSP.
- Crypto Service Provider (CSP): he randomly generates a pair of public key and secret key. From the construction of the protocol, CSP has no access to the encryption of every DO's data, otherwise CSP obtains the data and compute the linear regression model by himself. The public key is utilized by every data owner to encrypt their data without leaking the information. The secret key plays his role by CSP in the interaction with MLE.

[6] Recently, Li et al. [21] presented passive attacks against CKKS. General security IND-CPA (indistinguishable under chosen plaintext attack) for FHE is not suitable for approximate encryption. They claimed the security requirements of FHE for approximate encryption are IND-CPA+ (or SIM-CPA+) rather than IND-CPA (or SIM-CPA). After that, Cheon et al. [11] investigated some strategies of corresponding homomorphic libraries including HEAAN, HElib, SEAL, PALISADE and Lattigo against the query decryption on valid ciphertexts of adversary.

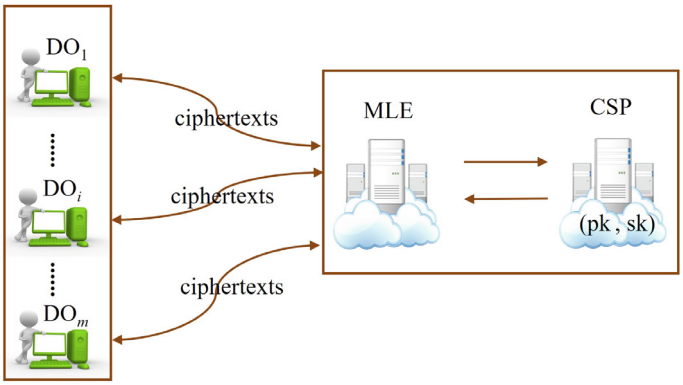


Fig. 1. The System Model in [17].

The MLE and CSP are required to be honest-but-curious. They strictly follow the algorithm to perform the corresponding operations. But they are still curious and try to obtain some information during the whole linear regression algorithm. One more requirement for MLE and CSP is necessary: they do not collude with each other.

The linear regression algorithm aims to help MLE to obtain the model without leakage of the information of the data instances. The information of the data instances of every DO are privacy-preserving. Meanwhile, neither data owners nor CSP can obtain the information of the linear regression model.

3.2. Basic Scheme

In this subsection, we briefly describe the main algorithms and parameters setting to obtain the linear regression model in [17]. The data instances are over the real interval $[-\delta, \delta]$ with l digits in the fractional part.

(1) As Section 2.1 illustrated, the model $\mathbf{w}^T$ can be obtained from $\mathbf{w}^T = \mathbf{A}^{-1}\mathbf{b}^T$, only if the matrix $\mathbf{X}$ is column full-rank, where $\mathbf{A} = \mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ and $\mathbf{b}^T = \mathbf{X}^T\mathbf{y}^T$.
(2) Firstly they skip the verification whether the matrix $\mathbf{X}$ is column full-rank and directly assumed it was. Then $\mathbf{A}$ is invertible. Consequently, in order to protect the information of the data instances over the real interval $[-\delta, \delta]$ of every data owner, they first encode the data instances of real numbers to be integers; then a relatively efficient labeled homomorphic encryption [6] is chosen to encrypt the plaintexts of integers, since the labeled homomorphic encryption is composed of offline phase and online phase. MLE obtains the linear regression model over the integers as Fig. 2. Then the Lagrange-Gauss algorithm in [15] is applied to achieve the transformations from the integers to real numbers, in order to get the linear regression model over $\mathbb{Q}$.
(3) As shown in Fig. 2, MLE randomly chooses an invertible matrix $\mathbf{M} \in \mathbb{Z}_N^{d\times d}$, and a vector $\mathbf{r}^T \in \mathbb{Z}_N^{d\times 1}$. After obtaining $\mathrm{Enc}(\mathbf{A})$ and $\mathrm{Enc}\left(\mathbf{b}^T\right)$ from data owners, MLE computes $\mathbf{C} = \mathrm{Enc}(\mathbf{AM})$ and $\mathbf{t} = \mathrm{Enc}\left(\mathbf{b}^T + \mathbf{Ar}^T\right)$ using the homomorphism of labeled homomorphic encryption over the plaintext space $\mathbb{Z}_N$. CSP provides the decryption after receiving $\mathbf{C}, \mathbf{t}$ from MLE, and computes $\mathbf{w}_1^T = (\mathbf{AM})^{-1} \cdot \left(\mathbf{b}^T + \mathbf{Ar}^T\right)$. Upon the receipt of $\mathbf{w}_1^T$ from CSP, MLE can attain the linear regression model $\mathbf{w}^T$ by $\mathbf{w}^T = \mathbf{Mw}_1^T - \mathbf{r}^T$.
(4) Benefit from the Lagrange-Gauss algorithm, the final linear regression model over $\mathbb{Q}$ can be obtained by MLE. However, the parameters setting have stronger restrictions:

$$2d(d-1)^{(d-1)/2}10^{4ld}\left(n\delta^2 + \lambda\right)^{2d} < N. \tag{1}$$

The item (1) largely depends on $\det(\mathbf{A})$ and $||\mathrm{adj}(\mathbf{A})\mathbf{b}^T||_\infty$, where the determinant of $\mathbf{A}$ is

$$0 < \det(\mathbf{A}) \leqslant 10^{2ld}\left(n\delta^2 + \lambda\right)^d$$

and the adjoint matrix of $\mathbf{A}$ is

$$||\mathrm{adj}(\mathbf{A})\mathbf{b}^T||_\infty \leqslant d(d-1)^{(d-1)/2}10^{2ld}\left(n\delta^2 + \lambda\right)^d.$$

When

$$2\det(\mathbf{A})||\mathrm{adj}(\mathbf{A})\mathbf{b}^T||_\infty < N$$

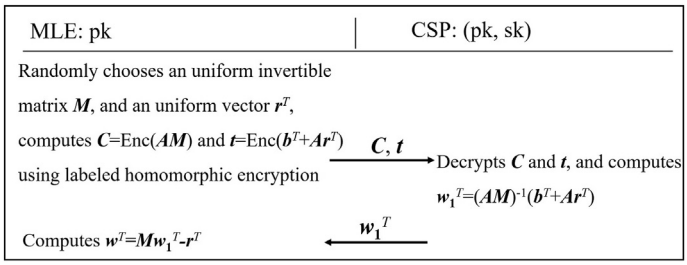


Fig. 2. Linear Regression Algorithm in [17].

holds, i.e., the Eq. (1), the corresponding rational numbers can be recovered from integers by the Lagrange-Gauss algorithm. Refer to [17] for further details of the linear regression algorithm.

4. Analysis of Giacomelli et al.'s

In this section, we present some analysis of Giacomelli et al.'s linear regression algorithm over the system model and parameters setting.

4.1. System Model Drawbacks

For the sake of accuracy of the linear regression model, MLE prefers to attain the result from various data owners, as Giacomelli et al. claimed there are m data owners DO_1, $DO_{2,\ldots}$, DO_m. However, the data instances of every data owner are encrypted by the same public key generated by a third party CSP. There are two main drawbacks for the linear regression model in the horizontally-partitioned setting. One is that CSP controls the secret key not the data owners themselves and may lead to security threat to all the data owners, even though CSP is required to have no access to the encryptions of all the data instances. This is due to that CSP has the ability to decrypt the encrypted data instances. The other is that every data owner shares the same pair of public key and secret key, which is not practical. Generally, every data owner can be regarded as an authority storing his own data instances and tends not to share common pair of public key and secret key in real life, such as a hospital for medical data.

In our opinion, we suggest to amend the system model of linear regression algorithm as Fig. 3. Every data owner generates his own pair of public key and secret key out of the control of CSP. Otherwise, every data owner may interact with each other from a key agreement protocol to generate a pair of keys, which produces more communication and computation costs. Any data owner can retrieve the data instances of other involved data owners. When every data owner possesses his own keys, the privacy of the data sets of every data owner can be protected effectively. As a result, the role of CSP is unnecessary since MLE can directly interact with data owners. Details can refer to Section 5.1 in this paper.

4.2. Parameters Setting

Parameters setting are embraced with the following two aspects.

(i) The assumption of data instances matrix **X**;
Giacomelli et al. directly assumed the matrix **X** was column full-rank. Note that every data owner uploads the ciphertexts of data instances, and the labeled homomorphic encryption based on Paillier cryptosystem is not homomorphic over the multiplication of ciphertexts. How to generate a column full-rank **X** comes into an essential issue.
(ii) The condition of parameter N satisfying

$$2d(d-1)^{(d-1)/2}10^{4ld}\left(n\delta^2+\lambda\right)^{2d} < N.$$

The above setting aims to $\mathbf{w}^T = \mathbf{A}^{-1}\mathbf{b}^T$ and $\det(\mathbf{A}), ||\mathrm{adj}(\mathbf{A})\mathbf{b}^T||_\infty$. Even Akavia et al. also followed the similar parameters setting to design a packed linear-regression algorithm. Now we consider the situation from the perspective of plaintexts instead of ciphertexts, in the original linear regression algorithm with the two-server model as Fig. 2. The rational matrices $\mathbf{A}, \mathbf{b}^T$ will be masked with the invertible integer matrix $\mathbf{M} \in \mathbb{Z}_N^{d\times d}$ and the integer vector $\mathbf{r}^T \in \mathbb{Z}_N^{d\times 1}$ by MLE. CSP will perform some computations $\mathbf{w}_1^T = (\mathbf{AM})^{-1}\cdot\left(\mathbf{b}^T + \mathbf{Ar}^T\right)$, to help MLE get the linear regression model over $\mathbb{Q}$ by $\mathbf{w}^T = \mathbf{Mw}_1^T - \mathbf{r}^T$. Therefore, in the sense of integers representing the rational numbers, the operations with integers to mask the plaintexts $\mathbf{AM}$ and $\mathbf{b}^T + \mathbf{Ar}^T$ should also be taken into consideration. In terms of $\mathbf{w}^T = \mathbf{Mw}_1^T - \mathbf{r}^T$, it is an exact recovery from $\mathbf{w}_1^T$ to $\mathbf{w}^T$. Hence with respect

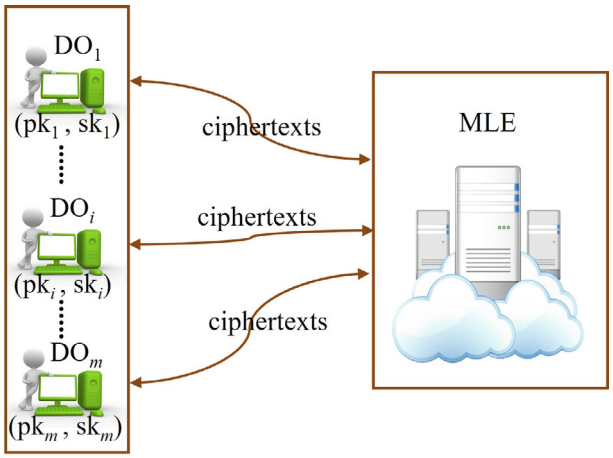


Fig. 3. Our System Model.

to the parameter N for the linear homomorphic encryption, taking $\det(\mathbf{AM})$ and $||\text{adj}(\mathbf{AM})\left(\mathbf{b}^T + \mathbf{Ar}^T\right)||_\infty$ into consideration is more appropriate to recover the model from $\mathbb{Z}$ to $\mathbb{Q}$.

Now we provide corresponding remedies of the above parameters setting.

(i) Drop the assumption that $\mathbf{X}$ is column full-rank;
We provide a preprocessing phase to generate a column full-rank $\mathbf{X}$, rather than any $\mathbf{X}$ composed of data instances from the data owners. The number of data instances in $\mathbf{X}$ from DO_j is n_j, which is less than or much less than the total data instances of DO_j. We pick different combinations of all the k_j data instances of DO_j for $j \in [m]$ to ensure a column full-rank $\mathbf{X}$, which are evaluated on the data instances in the encryption form without giving up the requirement of privacy preserving. Undoubtedly, the improvement comes into being under the improved CKKS scheme of multi keys in Section 2.2 with the system model as Fig. 3. Refer to Section 5.2 for details.
(ii) Change the condition of N into

$$2d^{2d}(d-1)^{\frac{d-1}{2}}(d+1)10^{4ld}\left(n\delta^2+\lambda\right)^{2d} < N. \tag{2}$$

Note that $\mathbf{M}, \mathbf{r}^T$ are randomly chosen from the $\mathbb{Z}_N$, we can reduce it into a small range, i.e., $\mathbb{Z}_2$. The "small" space of $\mathbb{Z}_2$ is much better for linear regression model over $\mathbb{Q}$ and without leaking the information of $\mathbf{A}, \mathbf{b}^T$. Recall that $||\mathbf{A}||_\infty, ||\mathbf{b}^T||_\infty \leqslant 10^{2l}(n\delta^2+\lambda)$, we obtain

$$||\mathbf{AM}||_\infty \leqslant 10^{2l}d\left(n\delta^2+\lambda\right)$$

and

$$||\mathbf{b}^T + \mathbf{Ar}^T||_\infty \leqslant 10^{2l}\left(n\delta^2+\lambda\right)(d+1).$$

Using the Hadamard's inequality, we have

$$0 < \det(\mathbf{AM}) \leqslant 10^{2ld}d^d\left(n\delta^2+\lambda\right)^d$$

and

$$\begin{aligned}&||\text{adj}(\mathbf{AM})\left(\mathbf{b}^T + \mathbf{Ar}^T\right)||_\infty \\ &\leqslant d^d(d-1)^{\frac{d-1}{2}}(d+1)10^{2ld}\left(n\delta^2+\lambda\right)^d.\end{aligned}$$

If

$$N > 2\det(\mathbf{AM}) \cdot ||\text{adj}(\mathbf{AM})\left(\mathbf{b}^T + \mathbf{Ar}^T\right)||_\infty,$$

always holds, i.e.,

$$2d^{2d}(d-1)^{\frac{d-1}{2}}(d+1)10^{4ld}\left(n\delta^2+\lambda\right)^{2d} < N,$$

Giacomelli et al. and Akavia et al. can obtain $\mathbf{w}_1^T$ correctly following the Lagrange-Gauss algorithm and the linear regression model over $\mathbb{Q}$ follows from $\mathbf{w}^T = \mathbf{M}\mathbf{w}_1^T - \mathbf{r}^T$.

5. Our Proposal

In this section, we first describe our system model, then put forward a preprocessing phase to generate a column full-rank $\mathbf{X}$. After that we use the improved CKKS scheme under multi keys in Section 2.2 as a linearly homomorphic encryption to design a more practical and efficient linear regression algorithm over $\mathbb{Q}$.

5.1. Our System Model

As Fig. 3 depicts, our system model consists of two entities Data Owners and MLE. Every data owner uploads his encrypted data instances with his own public key. Then MLE utilizes the ciphertexts to obtain the underlying linear model. During the process, MLE needs the help of every data owner to preform the threshold decryption protocol. In other words, the data owners are allowed to possess his own pair of public key and secret key, without getting out of control of their data instances. When needed, data owners perform corresponding operations for MLE.

Concretely, first rearrange the representations of the data instances with additional tags to distinguish the data instances that will be used to compute the model by MLE: the data instances of the j-th for $j \in [m]$ are denoted by $\left\{(\tau_{j1}, \mathbf{x}_{j1}, y_{j1}), (\tau_{j2}, \mathbf{x}_{j2}, y_{j2}), \ldots, \left(\tau_{jk_j}, \mathbf{x}_{jk_j}, y_{jk_j}\right)\right\}$, where τ_{jk_j} is an index representing the data instance $\left(\mathbf{x}_{jk_j}, y_{jk_j}\right)$ of the j-th data owner. Then every data owner uploads the encryption of all data instances with his own public key under our improved CKKS scheme with multi keys. The encryption of $(\tau_{j1}, \mathbf{x}_{j1}, y_{j1})$ is denoted by $(\tau_{j1}, \mathrm{Enc}(\mathbf{x}_{j1}[1]), \mathrm{Enc}(\mathbf{x}_{j1}[2]), \ldots, \mathrm{Enc}(\mathbf{x}_{j1}[d]), \mathrm{Enc}(y_{j1}))$. MLE randomly chooses $n_j (n_j < k_j)$ items from the j-th data owner satisfying that $\sum_{j=1}^{m} n_j = n$ and the encryption matrix is column full-rank, where $\mathrm{Enc}(\mathbf{x}_i)$ is the i-th row vector of $\mathrm{Enc}(\mathbf{X})$ for $i \in [n]$. After that MLE interacts with all the data owners as Fig. 2. All the data owners play the role of the initial CSP in [17]. But they performs a threshold decryption protocol without leaking the information of the secret keys of every data owner. As a result, MLE can attain the linear regression model.

With respect to the security model, every data owner is required to be honest-but-curious. None of the data owners colludes with MLE and each other, and attains something else beyond his own data instances. Meanwhile, the information of the data instances of every data owners are privacy-preserving, and MLE cannot obtain anything else beyond the linear regression model over $\mathbb{Q}$.

5.2. Preprocessing Phase

In this subsection, we present how MLE generates a column full-rank matrix $\mathbf{X}$ with only access to k_j encryptions from the j-th data owner for $j \in [m]$, before our linear regression algorithm.

It is apparent that the ciphertexts under our improved CKKS scheme of multi keys are polynomials with integer coefficients, supporting homomorphic additions and multiplications. Then MLE can directly pick n items from m data owners and may try many times to generate a column full-rank matrix $\mathrm{Enc}(\mathbf{X})$, which implies $\mathbf{X}$ is column full-rank. There are some important points we must emphasize.

(i) Our improved CKKS scheme under multi keys first encodes the message into the plaintext space, as the bijection σ^{-1} transfers $\mathbb{C}^{\mathcal{N}/2}$ into $\mathbb{Q}_p[X]/\left(X^{\mathcal{N}}+1\right)$. In this paper, data instances of rational numbers can be padded with 0 into $\mathcal{N}/2$ dimensions before encryption.

(ii) Every n_j encryptions for the j-th data owner are chosen as a part of $\mathrm{Enc}(\mathbf{X})$[7] satisfying that $\sum_{j=1}^{m} n_j = n$. For MLE, a failure comes into being, drop it, and a new attempt follows up. There are $\binom{k_j}{n_j}$ combinations for the j-th data owner, let alone the value of n_j, which depends on $\sum_{j=1}^{m} n_j = n$.

(iii) Verifying a matrix composed of polynomials with integer coefficients is column full-rank or not, is similar to a matrix over integers. After receving $\mathrm{Enc}(\mathbf{X})$ under m different public keys, MLE generates the expanded ciphertext of $\mathrm{Enc}(\mathbf{X})$ and verifies whether the expanded ciphertext is column full-rank by elementary transformation of matrix over $\mathbb{R}_p$. For ease, MLE perhaps judge whether any two columns of the expanded ciphertext of $\mathrm{Enc}(\mathbf{X})$ are linear to verify $\mathrm{Rank}(\mathrm{Enc}(\mathbf{X})) \stackrel{?}{=} d$.

(iv) One most important point is that the elementary transformation over the expanded ciphertexts is from $\mathbb{R}_p$ instead of the ciphertext space $\mathbb{R}_q$, which is exactly corresponding to the space of plaintext polynomial. Only in this way, from the homomorphism of our improved CKKS scheme, a column full-rank ciphertext matrix implies the corresponding plaintext

[7] For simplicity, $\mathrm{Enc}(\mathbf{X})$ and the expanded ciphertext of $\mathrm{Enc}(\mathbf{X})$ in this paper are denoted by the ciphertext generated by data owners and its ciphertext under m public keys, respectively.

matrix is column full-rank. Specifically, for any $r(X) \in \mathbb{R}_p$, and expanded ciphertext $\overline{ct}$ with underlying plaintext polynomial $\overline{m(X)}$ and message $\overline{\mathbf{m}}$ respectively, there is always

$$\mathrm{Dec1}\left(r(X)\cdot\overline{ct}\right) = r(X)\cdot\overline{m(X)}\left(\mathrm{mod}\ p, X^{\mathcal{N}}+1\right),$$

$$\mathrm{Dec2}\left(r(X)\cdot\overline{ct}\right) = \lfloor\triangle^{-1}\cdot\sigma(r(X))\rceil\cdot\overline{\mathbf{m}}.$$

The equation $\mathrm{Dec1}\left(r(X)\cdot\overline{ct}\right)$ holds largely depending on the linear homomorphism of our improved CKKS scheme, and the latter equation $\mathrm{Dec2}\left(r(X)\cdot\overline{ct}\right)$ holds largely depending on $\mathrm{Dec1}\left(r(X)\cdot\overline{ct}\right)$ and the bijection of σ.

After verifying some expanded ciphertexts of $\mathrm{Enc}(\mathbf{X})$ of column full-rank, MLE sends corresponding tags to every data owner. In turn, every data owner computes corresponding $\mathrm{Enc}\left(\mathbf{X}^T\mathbf{X}\right), \mathrm{Enc}\left(\mathbf{X}^T\mathbf{y}^T\right)$ and uploads them to MLE, where $\mathrm{Enc}\left(\mathbf{X}^T\mathbf{X}\right), \mathrm{Enc}\left(\mathbf{X}^T\mathbf{y}^T\right)$ are represented as

$$\mathrm{Enc}\left(\mathbf{X}^T\mathbf{X}\right) = \sum_{j=1}^{m}\sum_{i_j=1}^{n_j}\mathrm{Enc}\left(\mathbf{x}_{ji_j}^T\mathbf{x}_{ji_j}\right),$$

and

$$\mathrm{Enc}\left(\mathbf{X}^T\mathbf{y}^T\right) = \sum_{j=1}^{m}\sum_{i_j=1}^{n_j}\mathrm{Enc}\left(\mathbf{x}_{ji_j}^T y_{ji_j}\right)$$

respectively. Concretely, MLE sends $\left\{\tau_{1i_{11}},\ldots,\tau_{1i_{1n_1}}\right\}, \left\{\tau_{2i_{21}},\ldots,\tau_{2i_{2n_2}}\right\}, \ldots, \left\{\tau_{mi_{m1}},\ldots,\tau_{mi_{mn_m}}\right\}$ to DO_1, DO_2, …, DO_m respectively, where $\left\{i_{j1},\ldots,i_{jn_j}\right\} \subseteq \{1,\ldots,k_j\}$ for $j \in [m]$. After receiving the tags, DO_j computes and encrypts $\mathbf{x}_{ji_{j1}}^T\mathbf{x}_{ji_{j1}},\ldots,\mathbf{x}_{ji_{jn_j}}^T\mathbf{x}_{ji_{jn_j}}, \mathbf{x}_{ji_{j1}}^T y_{ji_{j1}},\ldots,\mathbf{x}_{ji_{jn_j}}^T y_{ji_{jn_j}}$.

Then we summarize a preprocessing phase of our linear regression algorithm as follows.

Step 0. This step includes a preprocessing phase for MLE. Every data owners first uploads the encryptions of all his data instances under his own public key with the improved CKKS scheme under multi keys. After MLE receives $\mathrm{Enc}(\mathbf{X})$, he generates a column full-rank expanded ciphertext of $\mathrm{Enc}(\mathbf{X})$ and sends corresponding tags to every data owner. Then $\mathrm{Enc}\left(\mathbf{X}^T\mathbf{X}\right)$ and a corresponding $\mathrm{Enc}\left(\mathbf{X}^T\mathbf{y}^T\right)$ are computed and uploaded by all the data owners to MLE.

5.3. Our Linear Regression Algorithm

Our improved CKKS scheme under multi keys is utilized as a linearly homomorphic encryption avoiding the complicated tensor product of ciphertext multiplications, since the tensor product operations of ciphertexts produce a large number of noises. Our linear regression algorithm is described as Fig. 4. Details are as follows.

Step 1. After obtaining $\mathrm{Enc}\left(\mathbf{X}^T\mathbf{X}\right), \mathrm{Enc}\left(\mathbf{X}^T\mathbf{y}^T\right)$, MLE utilizes the public keys of all the data owners $(pk_1,\ldots,pk_m)$ to generate $\mathrm{Enc}(\lambda\mathbf{I})$ and computes the expanded ciphertext of $\mathrm{Enc}(\mathbf{A}) = \mathrm{Enc}\left(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I}\right)$ by the linear homomorphism of the improved CKKS scheme under multi keys. Note that the bijection $\sigma : \mathbb{Q}_p[X]/\left(X^{\mathcal{N}}+1\right) \to \mathbb{C}^{\mathcal{N}/2}$, the invertible matrix $\mathbf{M}$ and the vector $\mathbf{r}^T$ in Fig. 4 are randomly chosen from $\mathbb{Q}^{d\times d}$ and $\mathbb{Q}^{d\times 1}$ by MLE. By linear homomorphism, MLE also attains the expanded ciphertexts of $\mathbf{C} = \mathrm{Enc}(\mathbf{AM}), \mathbf{t} = \mathrm{Enc}\left(\mathbf{b}^T+\mathbf{Ar}^T\right)$, where $\mathbf{b}^T = \mathbf{X}^T\mathbf{y}^T$.

Step 2. After MLE broadcasts the expanded ciphertexts of $\mathbf{C}, \mathbf{t}$ to $DO_{1,\ldots}$, DO_m, they collaboratively perform a threshold decryption protocol to get $\mathbf{AM}, \mathbf{b}^T+\mathbf{Ar}^T$. Any data owner can help MLE execute the following computations $\mathbf{w}_1 = (\mathbf{AM})^{-1}\cdot\left(\mathbf{b}^T+\mathbf{Ar}^T\right)$ since every data owner is assumed to be honest-but-curious.

Step 3. Thus MLE acquires the linear regression model $\mathbf{w} = \mathbf{Mw}_1 - \mathbf{r}^T$.

Correctness. We discuss the correctness of Step 1–3 of our linear regression algorithm here.

In Step 1, following the linear homomorphism of the improved CKKS scheme under multi keys, we find that $\mathrm{Enc}\left(\mathbf{X}^T\mathbf{X}\right)+\mathrm{Enc}(\lambda\mathbf{I}) = \mathrm{Enc}\left(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I}\right) = \mathrm{Enc}(\mathbf{A}), \mathrm{Enc}(\mathbf{A})\cdot\mathbf{M} = \mathrm{Enc}(\mathbf{AM}) = \mathbf{C}, \mathrm{Enc}\left(\mathbf{b}^T\right)+\mathrm{Enc}(\mathbf{A})\cdot\mathbf{r}^T = \mathrm{Enc}\left(\mathbf{b}^T+\mathbf{A}\cdot\mathbf{r}^T\right) = \mathbf{t}$, likewise for the expanded ciphertexts of $\mathbf{C}, \mathbf{t}$.

The correctness of Step 2 follows from the threshold decryption protocol of the improved CKKS under multi keys.

With respect to Step 3, consider that

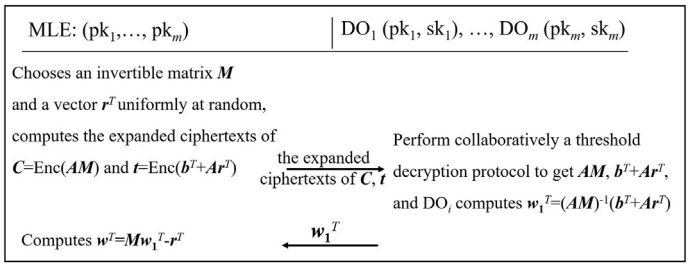


Fig. 4. Our Linear Regression Algorithm.

$$\begin{aligned}\mathbf{w}_1 &= (\mathbf{AM})^{-1}\cdot\left(\mathbf{b}^T+\mathbf{Ar}^T\right)\\ &= \mathbf{M}^{-1}\mathbf{A}^{-1}\left(\mathbf{b}^T+\mathbf{Ar}^T\right)\\ &= \mathbf{M}^{-1}\mathbf{A}^{-1}\mathbf{b}^T+\mathbf{M}^{-1}\mathbf{r}^T,\end{aligned}$$

then we can obtain $\mathbf{Mw}_1-\mathbf{r}^T=\mathbf{A}^{-1}\mathbf{b}^T=\mathbf{w}$.

Security Analysis. Now we present the security of our linear regression algorithm by the following theorems. The proof that $\mathbf{AM}$ and $\mathbf{b}^T+\mathbf{Ar}^T$ can efficiently protect the information of $\mathbf{A}$ and $\mathbf{b}^T$ respectively is first elaborated. Some necessary definition and lemma are presented here.

Definition 1 (*Universal hash function [29]*). Let R,S,T be finite and non-empty sets. Suppose that for each $r\in R$, we have a hash function $\mathcal{H}_r: S\to T$. The hash function $\mathcal{H}_r$ is called universal if the probability $\Pr[\mathcal{H}_r=\mathcal{H}_{r'}|r\neq r']<\epsilon$ holds for all $r,r'\in S$, where ϵ is negligible.

Lemma 1 (*Leftover Hash Lemma [13]*). Let $\mathcal{H}$ be a family of 2-universal hash functions from $\mathcal{X}$ to $\mathcal{Y}$. Suppose that $h\leftarrow\mathcal{H}$ and $x\leftarrow\mathcal{X}$ are chosen uniformly and independently. Then $(h,h(x))$ is $\frac{1}{2}\sqrt{|\mathcal{Y}|/|\mathcal{X}|}$-uniform over $\mathcal{H}\times\mathcal{Y}$.

Now we describe the security of $\mathbf{AM}$ and $\mathbf{b}^T+\mathbf{Ar}^T$ as Theorem 1.

Theorem 1. When $\mathbf{M}\in\mathbb{Q}^{d\times d},\mathbf{r}^T\in\mathbb{Q}^{d\times 1},\mathbf{AM}$ and $\mathbf{b}^T+\mathbf{Ar}^T$ can still protect the information of $\mathbf{A}$ and $\mathbf{b}^T$ respectively.

Proof. The security of $\mathbf{b}^T+\mathbf{Ar}^T$ follows from the one of $\mathbf{AM}$, since $\mathbf{r}^T$ can be seen as a column of $\mathbf{M}$. Now we illustrate the uniform distribution of $\mathbf{AM}$.

Assume a hash function

$$\mathcal{H}:\mathbb{Q}^{d\times d}\to\mathbb{Q}^{d\times d},\mathcal{H}(\mathbf{M})=\mathbf{AM}.$$

Recall the matrix $\mathbf{A}=\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I}$ where $\mathbf{X}$ is chosen as a matrix of column full-rank, we can claim that $\mathbf{A}$ of order d is invertible. Assume two matrices $\mathbf{M}_1\neq\mathbf{M}_2$ from $\mathbb{Q}^{d\times d}$, the probability of $\mathbf{AM}_1=\mathbf{AM}_2$ is negligible. Otherwise, if $\mathbf{AM}_1=\mathbf{AM}_2$, i.e., $\mathbf{A}(\mathbf{M}_1-\mathbf{M}_2)=0$. We have $\mathbf{M}_1-\mathbf{M}_2=0$ due to the invertible square matrix $\mathbf{A}$ of order d over $\mathbb{Q}$, which leads to a contradiction to the assumption of $\mathbf{M}_1\neq\mathbf{M}_2$.

Under this circumstance, we know the hash function $\mathcal{H}$ is universal. Hence we claim the matrix $\mathbf{AM}$ is according to a uniform distribution, which effectively protects the information of $\mathbf{A}$.

To sum up, the proof is finished.

Theorem 2. Our linear regression algorithm is secure against the adversaries $\mathcal{A}=\{\mathrm{DO}_1,\ldots,\mathrm{DO}_m\}$.

Proof. This proof shows that there are two probabilistic polynomial time algorithms such that the real view and the ideal one are computationally indistinguishable.

First, construct the simulator $\mathcal{S}^{\mathrm{DO}}$ which simulate the views of the entities in $\mathcal{A}$.

- MLE interacts with $\mathcal{S}^{\mathrm{DO}_i}$ for $i\in[m]$ by the following steps:

1) Following Step 0, after MLE sends tags to every data owner to generate a column full-rank expanded ciphertext of $\mathrm{Enc}(\mathbf{X})$, all data owners except the i-th DO for $i \in [m]$ send $\mathrm{Enc}\left(\mathbf{X}^T\mathbf{X}\right)$ and a corresponding $\mathrm{Enc}\left(\mathbf{X}^T\mathbf{y}^T\right)$ to MLE, while the i-th DO sends the encryptions of zero matrix and zero vector respectively.
2) Sample $\mathbf{M}, \mathbf{r}^T$ uniformly as our linear regression algorithm, then MLE performs Step 1 and broadcasts the expanded ciphertexts of $\mathbf{C}, \mathbf{t}$ to all data owners.
3) All data owners collaboratively decrypt to obtain $\mathbf{AM}, \mathbf{b}^T + \mathbf{Ar}^T$, and the i-th data owner sends $\mathbf{w}_1$ to MLE.
4) MLE computes $\mathbf{Mw}_1 - \mathbf{r}^T = \mathbf{w}$.

It follows from the security of our improved CKKS scheme under multi keys, i.e., IND-CPA+, that the simulated $\mathcal{S}^{\mathrm{DO}_i}$ for $i \in [m]$ has the same distribution with the view of $\mathrm{DO_i}$ in our linear regression algorithm.

- $\mathcal{S}^{\mathrm{DO}_i}$ against $\mathrm{DO_j}$ for $i, j \in [m]$ and $i \neq j$ is defined to perform as Fig. 4 except the following steps. Without loss of generality, assume $\mathrm{DO_i}$ computes $\mathbf{w}_1 = (\mathbf{AM})^{-1} \cdot \left(\mathbf{b}^T + \mathbf{Ar}^T\right)$.

(1) $\mathrm{DO_i}$ performs Step 0 as our linear regression algorithm.
(2) After receiving the expanded ciphertexts of $\mathbf{C}, \mathbf{t}$ from MLE, $\mathrm{DO_i}$ performs his own decryption as $\gamma_i(\mathbf{C}) = \mathbf{AM} - \sum_{u=1, u\neq i}^{m} \gamma_u(\mathbf{C})$ and $\gamma_i(\mathbf{t}) = \mathbf{t} - \sum_{u=1, u\neq i}^{m} \gamma_u(\mathbf{t})$, where $\gamma_i(\cdot)$ is the partial decryption of the i-th DO as the decryption algorithm in Section 2.2.

It follows from the security of the threshold decryption of our improved CKKS scheme under multi keys, that $\mathcal{S}^{\mathrm{DO}_i}$ has the same distribution of the view of $\mathrm{DO_i}$ against $\mathrm{DO_j}$ in our linear regression algorithm.

To sum up, from Theorem 1 and Theorem 2, our linear regression algorithm efficiently helps MLE to attain the model, and protects the information of all the involved entities from exposing the privacy.

6. Performance Evaluation

We elaborate the performance of our linear regression algorithm from the perspective of theoretical analysis and experimental evaluation, compared with [2,17].

6.1. Theoretical Analysis

Our goal is to evaluate the performance on the error rate, computation complexity and communication complexity to test the efficiency of linear regression algorithm. In the description of this paper, we follow [17] to construct linear regression algorithm surrounding with the data instances from the real interval $[-\delta, \delta]$ with l digits in the fractional part, just for comparison of efficiency under the same conditions.

Error Rate. As Giacomelli et al. claimed, the parameter l has effects on the evaluating accuracy. With the digits of fractional part l increasing, the linear regression model in [17] is more accurate, compared with the data instances in the clear under the same regularization parameter λ. For example, when $l = 3$, the error rate of the linear regression model is 10^{-4}. On the other hand, [2,17] directly assumed the matrix $\mathbf{X}$ is column full-rank without filtrating the data instances from data owners. Note that the system in [2,17] are same other than the encryption algorithm, then we just think the error rate in [2] is also 10^{-4}.

Our algorithm essentially supports any data instances from complexity number space owing to the bijection $\sigma : \mathbb{Q}_p[X]/\left(X^{\mathcal{N}} + 1\right) \rightarrow \mathbb{C}^{\mathcal{N}/2}$. The bijection can map any complex vectors to polynomials with rational numbers, in order to be encoded into the plaintext space. Additional preprocessing phase is performed to choose data instances to generate a column full-rank $\mathbf{X}$. The decryption algorithm outputs the approximate message under the corresponding precision. As shown in Table 2, our algorithm achieves an approximate result of the linear regression model even without the help of CSP, while [2,17] cannot. The errors are produced from the decryption precision of our improved CKKS scheme under multi keys, i.e., 30 bits. That is, the error rate of our linear regression model is 2^{-30}, which is much smaller than [2,17].

Computational Complexity. Table 3 summarizes the computational complexity in terms of number of elementary operations. We capitalize the first letter and use the lowercase of the first letter to distinguish the operations among our algorithm and [2,17], where the capitals represent ours. The notation "Mul, mul". (resp. "Add, add") represent the multiplication (resp. addition) on plaintext messages, "Enc-Add, enc-add" represents for the operation on ciphertexts. "Inversion" denotes finding the inverse matrix over $\mathbb{R}_p$ of order d while "inversion" denotes finding the inverse matrix with integers of order d.

Table 2
The basic comparisons of error rate.

	Entities			
	CSP	Data Owners	MLE	Error Rate
Giacomelli et al.[17]	✓	✓	✓	10^{-4}
Akavia et al. [2]	✓	✓	✓	10^{-4}
Ours	×	✓	✓	2^{-30}

Table 3
Comparisons of Computational Complexity.

Step	Proposed Algorithm		Existing Algorithm [2,17]		
	Entities		Entities		
	DO_j	MLE	CSP	DO_j	MLE
Step 0	1 execution of KeyGen $+n_j\left[\frac{d(d+1)}{2}+d\right]$ Mul $+n_j\left[\frac{d(d+1)}{2}+d\right]$ Add $+\left(k_j+\frac{d(d+1)}{2}+d\right)$ Enc	Judgements $+m\left[\frac{d(d+1)}{2}+d\right]$ Enc-Add	1 execution of keygen	$n_j\left[\frac{d(d+1)}{2}+d\right]$ mul $+n_j\left[\frac{d(d+1)}{2}+d\right]$ add $+\frac{d(d+1)}{2}+d$ enc	$m\left[\frac{d(d+1)}{2}+d\right]$ enc-add
Step 1	–	$\left(d^3+d^2+d\right)$ Enc-Add $+\left(d^3+d^2\right)$ cMul	–	–	$\left(d^3+d^2+d\right)$ enc-add $+\left(d^3+d^2\right)$ cmul
Step 2	$\left(d^2+d\right)$ Dec $+1$ Inversion $+d^2$ Mul $+d^2$ Add	–	$\left(d^2+d\right)$ dec $+1$ inversion $+d^2$ mul $+d^2$ add	–	–
Step 3	–	$\left(d^2+d\right)$ Add $+d^2$ Mul	–	–	$\left(d^2+d\right)$ add $+d^2$ mul $+d$ rational reconstructions

Actually, during the preprocessing phase in our linear regression algorithm, DO_j uploads k_j encryptions additionally, and MLE performs the decision whether the matrix $\mathbf{X}$ is column full-rank. Hence, the computational complexity of our algorithm and [17] is subequal in the view of the number of operations, except that our algorithm performs a preprocessing phase and without d rational reconstructions.

In the view of the total computation complexity, we must consider the costs of detailed operations, such as "Mul, mul, Add, add, Enc-Add, enc-add, Inversion, inversion" and so on. All these depend on our improved CKKS scheme under multi keys and labeled homomorphic encryption respectively. Basic costs of conventional operations are same, for example, the complexity of calculating $ab \bmod q$ is $O\left(\log^2 q\right)$, and the complexity of calculating $a^b \bmod q$ is $O\left(\log^3 q\right)$. Only the corresponding plaintext and ciphertext spaces are different: they are $\mathbb{R}_p, \mathbb{R}_q^m$ in our improved CKKS scheme under multi keys of m data owners while they are $\mathbb{Z}_N, \mathbb{Z}_N$ in labeled homomorphic encryption in [17], and $\mathbb{Z}_N, \mathbb{R}_q$ in [2].

Totally, it seems that our linear regression algorithm is more expensive than [17]. However, our improved CKKS scheme under multi keys supports batching, Chinese Reminder Theorem and Single Instruction Multiple Data which can significantly improve the total computation overheads. Moreover, the technique of matrix (of order d) multiplications in [19], rotations and Single Instruction Multiple Data can also be utilized. The detailed performance of running time can refer to experimental analysis in Section 6.2.

Communication Complexity. Table 4 presents the communication complexity with the number of the plaintexts and ciphertexts. It seems almost equal except the k_j additional ciphertexts from DO_j to MLE in Step 0. Then there are mk_j ciphertexts more than [2,17] in terms of the communication complexity. Moreover, for the consideration of security, every data owner cooperatively performs the threshold decryption protocol with his own secret key. Note that the ciphertexts are broadcasted to data owners, then the communication for MLE is the ciphertext size. When performing the threshold decryption protocol, the communication for data owners is $2dm$ times of the plaintext size, which is unnecessary in [2,17]. Finally, some data owner sends d plaintexts to MLE. Nevertheless, the plaintext size is much smaller than the ciphertext size, hence the ciphertext size has a bigger effect on the communication of threshold decryption protocol. In other words, the communication of threshold decryption protocol relies on the multi-key CKKS scheme. The magnitude of the communication complexity represented with d is identical to $\Theta\left(d^2 m\mathcal{N} \log q\right)$.

Considering the plaintext and ciphertext spaces of the two encryption algorithm $\mathbb{R}_p, \mathbb{R}_q^m, \mathbb{Z}_p, \mathbb{Z}_N, \mathbb{Z}_N, \mathbb{R}_q$, the communication complexity of ours and [2,17] can also be represented by $O\left(d^2 m\mathcal{N} \log q\right), O\left(d^2\mathcal{N} \log q\right)$ and $O\left(d^3 \log (Nnd)\right)$ respectively.

Table 4
Comparisons of Communication Complexity.

Step	Our Algorithm	Giacomelli et al. [17], Akavia et al. [2]
Step 0	– DO_j sends k_j ciphertexts to MLE to generate a full-rank matrix – DO_j sends $\frac{d(d+1)}{2}+d$ ciphertexts to MLE	– CSP sends pk to every data owner – DO_j sends $\frac{d(d+1)}{2}+d$ ciphertexts to MLE
Step 1	– MLE broadcasts d^2+d expanded ciphertexts to $DO_{1,\ldots}$, DO_m	– MLE sends d^2+d ciphertexts to CSP
Step 2	– All data owners cooperatively perform the threshold decryption protocol, and send d plaintexts to MLE	– CSP sends d plaintexts to MLE

6.2. Experimental Analysis

From the computation complexity and communication complexity in the view of theoretical analysis, it seems to cost a lot for our linear regression algorithm compared with [17] owning to a larger ciphertext space. Now we elaborate the performance from experiments.

Before performing the experiment, we first present several techniques to accelerate our linear regression algorithm.

- Batching. By using Chinese Reminder Theorem and Single Instruction Multiple Data, we can divide the plaintext space into many slots, and encrypt them in parallel. Meanwhile, $\frac{\mathcal{N}}{2}$ number of messages are batched into a ciphertext, largely reducing the computational complexity. Moreover, the parameters setting make it possible to adapt fast Number Theory Transformation to the polynomial operations with computation complexity reduced from $O\left(\mathcal{N}^2\right)$ to $O(\mathcal{N}\log\mathcal{N})$.
- Speedup. The technique of matrix multiplication in [19] can be utilized, i.e., rotations, the component-wise product and Single Instruction Multiple Data. For matrix–vector multiplication, the vector can be padded with $\mathbf{0}$ or $\mathbf{0}^T$ to perform matrix multiplication.

The experiment environments are with Intel R CoreTM i7-7700HQ CPU@.2.80 GHz/16 GB Ram, we use PyCharm 2020.1.1 x64 under Python 3.6 to encrypt the data under the Windows 10 operating system. The data are over $[0,1)$ with precision of $l=3$ decimal digits. The entities are of a server and $m=10$ data owners with horizontally-partitioned data $n=1000$ instances, $d=10,20,30,40$ features. We call the TenSEAL library [7] based on our improved CKKS scheme and BFV scheme while call the Original-paillier [32] which is based on Paillier scheme, NTL and GMP libraries, to evaluate the linear regression algorithm in comparison with ours and [2,17][8].

Correspondingly, the parameters in our linear regression algorithm are set as $\mathcal{N}=2^{13}, |q|=200$ bits ($|\cdot|$ denotes the length) for a security level of 128 bits. The modulus q will be divided into 4 coprime numbers of length 60, 40, 40, 60 bits and the rescaling factor is $\Delta=2^{40}$. Then the precision is 30 bits for homomorphic operations and decryption results, which is sufficient to output the linear regression model for the data instances over $[0,1)$ with 3 decimal digits. For comparison, the parameter in [17] is $|N|=3584$ bits satisfying Eq. (2) for $d\leqslant 40$, with a security level of at least 128 bits. The dimension is $\mathcal{N}=2^{13}$ and the modulus is a prime congruent 1 modulo $2\mathcal{N}$ in [2]. We set $q=2277377$. The plaintext modulus is of length 3584 bits. By Chinese Reminder Theorem, the plaintext will be divided into nearly 60 slots with 60 bits. The performance comparisons are shown in Table 5 and Fig. 5.

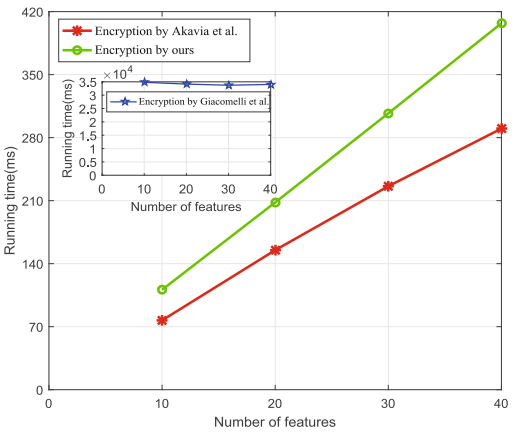
In Table 5, we present the average time in second for several steps. The third column "Encrypt" denotes the running time for every data owners to encrypt his data instances. "LHE.op" represents the linear homomorphism time in Step 1 for MLE. The "Total" time is including the running time of Encrypt, Step 1, Step 2 and Step 3. For Step 3, [2,17] also perform an operation to recover the linear regression model from integers to rational numbers while ours does not need. In order to improve the efficiency of [17], we perform the Paillier cryptosystem in parallel. In other words, the running times of encryption and decryption for [17] is d^2 times more than the ones in Table 5. This is mainly because that our evaluations over Paillier exploit the batching technique for the matrix $\mathbf{A}$. The running time of encryption in [17] is clearly to encrypt an element in $\mathbf{A}$. The time depends on the parameter N, which is the same $|N|=3584$ bits satisfying Eq. (2) for all $d\leqslant 40$.

From Fig. 5, it is apparent to see that our linear regression algorithm costs less time during the "Encrypt" phase compared with [17], even more than [2]. Relatively speaking, every data owner can be offline after uploading conditioned ciphertexts of data instances, and MLE trains the encrypted data instances to learn the linear regression model. Hence the Step 1, Step 2 and Step 3 are more important than "Encrypt". And the running time of ours during the LHE operation and the total procedure in Fig. 5 is much less than [17]. It takes 6 ms with $d=40$ for our algorithm in LHE.op and almost 1932 and 4 times less than the

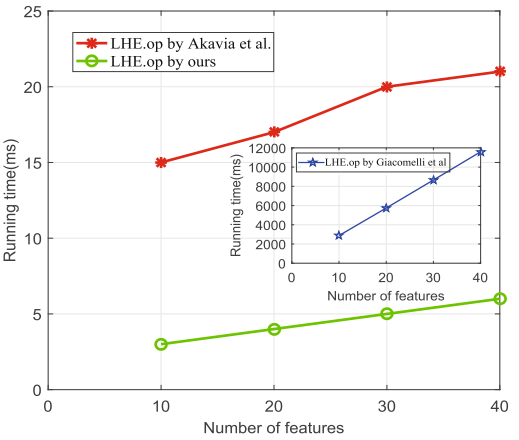
[8] Considering that [7] is simply a wrapper for the C++ Microsoft SEAL library, a C++ library is also employed to implement the paillier cryptosystem. On the other hand, we directly perform the BFV scheme instead of BGV scheme to evaluate the efficiency of [2], since the BFV and BGV are pretty much the same, and TenSEAL library includs CKKS scheme and BFV scheme.

Table 5
Comparisons of Running time in Second.

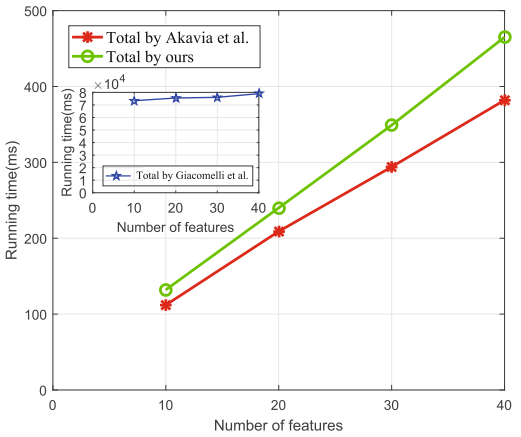
d	Algorithms	Encrypt	Step 1	Step 2	Step 3	LHE.op.	Total
10	Giacomelli et al.[17]	34.829	2.877	35.667	0.005	2.877	73.378
	Akavia et al.[2]	0.077	0.015	0.015	0.005	0.015	0.112
	Ours	0.111	0.003	0.015	0	0.003	0.132
20	Giacomelli et al.[17]	34.148	5.742	35.632	0.010	5.742	75.532
	Akavia et al.[2]	0.155	0.017	0.027	0.010	0.017	0.209
	Ours	0.208	0.004	0.027	0.001	0.004	0.240
30	Giacomelli et al.[17]	33.732	8.654	33.675	0.010	8.654	76.062
	Akavia et al.[2]	0.226	0.020	0.038	0.010	0.020	0.294
	Ours	0.307	0.005	0.038	0	0.005	0.349
40	Giacomelli et al.[17]	34.022	11.591	33.376	0.018	11.591	79.007
	Akavia et al.[2]	0.290	0.021	0.053	0.018	0.021	0.382
	Ours	0.406	0.006	0.053	0	0.006	0.465



(a) Running time of encryption.



(b) Running time of LHE operation.



(c) Running time of total operation.

Fig. 5. Comparisons of Running Time with Suggested Parameters.

one in [17,2] respectively. The total running time for a 1000×40 linear regression task is almost 170 times less than [17]. Therefore, our linear regression algorithm is more efficient and practical.

7. Conclusion

In this paper, we focus on the linear regression algorithm over rational numbers proposed by Giacomelli et al. using only LHE. Even Akavia et al. designed a packed privacy-preserving linear regression algorithm over rational numbers, following the same transformation between rational numbers and integers as Giacomelli et al.. However, we propose a more practical linear regression algorithm over rational numbers. First, every data owner generates his own public key and secret key, independent on a third authority (i.e., CSP). Second, a preprocessing phase is taken into consideration to produce an invertible data sets **A** instead of an assumption. Third, we design an improved CKKS scheme under multi keys, and apply it to encrypt the data sets. This leads to a situation where our linear regression scheme avoids the rational reconstruction technique between rational numbers and integers.

CRediT authorship contribution statement

Wenju Xu: Validation, Formal analysis, Writing - original draft. **Baocang Wang:** Conceptualization, Methodology, Supervision. **Jiasen Liu:** Software. **Pu Duan:** Writing - review & editing. **Zhiyong Hong:** Writing - review & editing.

Declaration of Competing Interest

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

Acknowledgements

This research was funded by the National Natural Science Foundation of China under Grant Nos. U19B2021, and Key Research and Development Program of Shaanxi under Grant No. 2020ZDLGY08-04.

References

[1] Agrawal, R., Srikant, R., 2000. Privacy-preserving data mining. In: Chen, W., Naughton, J.F., Bernstein, P.A. (Eds.), Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, Texas, USA. ACM, pp. 439–450. .
[2] Akavia, A., Shaul, H., Weiss, M., Yakhini, Z., 2019. Linear-regression on packed encrypted data in the two-server model. In: Brenner, M., Lepoint, T., Rohloff, K. (Eds.), Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography, WAHC@CCS 2019, London, UK. ACM, pp. 21–32. .
[3] Y. Aono, T. Hayashi, L.T. Phong, L. Wang, Fast and secure linear regression and biometric authentication with security update, IACR Cryptol. ePrint Arch. 2015 (2015) 692.
[4] K. Arimitsu, K. Otsuka, Privacy-preserving fast and exact linear equations solver with fully homomorphic encryption, IACR Cryptol. ePrint Arch. 2020 (2020) 272.
[5] Baidu,???? url:https://baike.baidu.com/item/linear regression. .
[6] M. Barbosa, D. Catalano, D. Fiore, Labeled homomorphic encryption - scalable and privacy-preserving processing of outsourced data, in: S.N. Foley, D. Gollmann, E. Snekkenes (Eds.), Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Vol. 10492, Springer, Oslo, Norway, 2017, pp. 146–166.
[7] Bcebere, 2021. Tenseal: A library for encrypted tensor operations using homomorphic encryption. url:https://github.com/OpenMined/TenSEAL. .
[8] Z. Brakerski, C. Gentry, V. Vaikuntanathan, (leveled) fully homomorphic encryption without bootstrapping, in: S. Goldwasser (Ed.), Innovations in Theoretical Computer Science 2012, ACM, Cambridge, MA, USA, 2012, pp. 309–325.
[9] F. Chen, T. Xiang, X. Lei, J. Chen, Highly efficient linear regression outsourcing to a cloud, IEEE Trans. Cloud Comput. 2 (4) (2014) 499–508.
[10] Chen, L., Zhang, Z., Wang, X., 2017. Batched multi-hop multi-key FHE from ring-lwe with compact ciphertext extension. In: Kalai, Y., Reyzin, L. (Eds.), Theory of Cryptography Conference-TCC 2017, Baltimore, MD, USA. Vol. 10678 of Lecture Notes in Computer Science. Springer, pp. 597–627. .
[11] J.H. Cheon, S. Hong, D. Kim, Remark on the security of CKKS scheme in practice, IACR Cryptol. ePrint Arch. 2020 (2020) 1581.
[12] Cheon, J.H., Kim, A., Kim, M., Song, Y.S., 2017. Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (Eds.), Advances in Cryptology - ASIACRYPT 2017, Hong Kong, China. Vol. 10624 of Lecture Notes in Computer Science. Springer, pp. 409–437. .
[13] M. Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan, Fully homomorphic encryption over the integers, in: Advances in Cryptology - EUROCRYPT 2010, Monaco, French Riviera, 2010, pp. 24–43.
[14] Esperança, P.M., Aslett, L.J.M., Holmes, C.C., 2017. Encrypted accelerated least squares regression. In: Singh, A., Zhu, X.J. (Eds.), Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, Fort Lauderdale, FL, USA. Vol. 54 of Proceedings of Machine Learning Research. PMLR, pp. 334–343. .
[15] Fouque, P., Stern, J., Wackers, J., 2002. Cryptocomputing with rationals. In: Blaze, M. (Ed.), Financial Cryptography, 6th International Conference, FC 2002, Southampton, Bermuda. Vol. 2357 of Lecture Notes in Computer Science. Springer, pp. 136–146. .
[16] Gentry, C.,???? Fully homomorphic encryption scheme using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing. Springer, Bethesda, USA, pp. 169–178. .
[17] Giacomelli, I., Jha, S., Joye, M., Page, C.D., Yoon, K., 2018. Privacy-preserving ridge regression with only linearly-homomorphic encryption. In: Preneel, B., Vercauteren, F. (Eds.), Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium. Vol. 10892 of Lecture Notes in Computer Science. Springer, pp. 243–261. .
[18] Halevi, S., Shoup, V., 2014. Algorithms in helib. In: Garay, J.A., Gennaro, R. (Eds.), Advances in Cryptology - CRYPTO 2014–34th Annual Cryptology Conference, Santa Barbara, CA, USA. Vol. 8616 of Lecture Notes in Computer Science. Springer, pp. 554–571. .

[19] Jiang, X., Kim, M., Lauter, K.E., Song, Y., 2018. Secure outsourced matrix computation and application to neural networks. In: Lie, D., Mannan, M., Backes, M., Wang, X. (Eds.), Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada. ACM, pp. 1209–1222. .
[20] Kenton, W., 2021. Capital asset pricing model (capm). url:https://www.investopedia.com/terms/c/capm.asp. .
[21] B. Li, D. Micciancio, On the security of homomorphic encryption on approximate numbers, IACR Cryptol. ePrint Arch. 2020 (2020) 1533.
[22] Lindell, Y., Pinkas, B., 2000. Privacy preserving data mining. In: Bellare, M. (Ed.), Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA. Vol. 1880 of Lecture Notes in Computer Science. Springer, pp. 36–54. .
[23] A. Machanavajjhala, D. Kifer, J. Gehrke, M. Venkitasubramaniam, L-diversity: Privacy beyond k-anonymity, ACM Trans. Knowl. Discov. Data 1 (1) (2007) 3.
[24] P. Mukherjee, D. Wichs, Two round multiparty computation via multi-key FHE, in: M. Fischlin, J. Coron (Eds.), Advances in Cryptology - EUROCRYPT 2016–35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, 9666 of Lecture Notes in Computer Science. Springer, Vienna, Austria. Vol, 2016, pp. 735–763.
[25] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, N. Taft, Privacy-preserving ridge regression on hundreds of millions of records, in: 2013 IEEE Symposium on Security and Privacy, SP 2013, USA. IEEE Computer Society, Berkeley, CA, 2013, pp. 334–348.
[26] Paillier, P., 1999. Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (Ed.), Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic. Vol. 1592 of Lecture Notes in Computer Science. Springer, pp. 223–238. .
[27] L.T. Phong, Y. Aono, T. Hayashi, L. Wang, S. Moriai, Privacy-preserving deep learning via additively homomorphic encryption, IEEE Trans. Inf. Forensics Secur. 13 (5) (2018) 1333–1345.
[28] Samarati, P., Sweeney, L., 1998. Generalizing data to provide anonymity when disclosing information (abstract). In: Mendelzon, A.O., Paredaens, J. (Eds.), Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Seattle, Washington, USA. ACM Press, p. 188. .
[29] V. Shoup, A computational introduction to number theory and algebra, The 2nd Edition., Cambridge University Press, 2008.
[30] N.P. Smart, F. Vercauteren, Fully homomorphic SIMD operations, Des. Codes Cryptogr. 71 (1) (2014) 57–81.
[31] J. Wu, H. Yang, Linear regression-based efficient SVM learning for large-scale classification, IEEE Trans. Neural Networks Learn. Syst. 26 (10) (2015) 2357–2369.
[32] Ziyao002, 2019. url:https://github.com/ziyao002/Original-Paillier. .