Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

AMA,TIMES1COL

# Improved Secure PCA and LDA algorithms for Intelligent Computing in IoT-to-cloud Setting

**Liu Jiasen**[1] | **Wang Xu An**[2] | **Li Guofeng**[1] | **Yu Dan**[1] | **Zhang Jindan**[3]

[1]The Second Mobile Corps under Chinese Armed Police Force, Guangzhou, China

[2]Key Laboratory for Network and Information Security of the PAP, Engineering University of the PAP, Xian, China

[3]VocationalTechnical College, Xianyang, China

**Correspondence**
Wang Xu An
Email: 1261510059@qq.com

**Present address**
Huadu District, Guangzhou City, Guangdong Province, China.

## Abstract

The rapid development of new technologies such as artificial intelligence and big data analysis requires the simultaneous development of cloud computing technology. The application of IoT-to-cloud setting has been fully applied in various industry sectors, such as Sensor-Cloud System which is composed of wireless sensor network and cloud computing technology. With the increasing amount and types of collected data, companies need to reduce the dimension of massive data in cloud servers for obtaining data analysis reports rapidly. Due to frequent cloud server data leaks, companies must adequately protect the privacy of some confidential data. To this end, we designed a dimension reduction method for ciphertext data in the Sensor-Cloud System based on the CKKS encryption scheme, PCA and LDA dimension reduction algorithm. As data cannot be directly calculated using traditional PCA and LDA algorithm after encryption, we add some interactive operations and iterative calculations to replace some steps in traditional algorithms. Finally, we select the classification dataset IRIS which is commonly used in machine learning, and screen out the best encryption and calculation parameters, and efficiently realize the dimension reduction method of ciphertext data through a large number of experiments.

**KEYWORDS**

IoT-to-cloud setting, Sensor-Cloud System, Homomorphic encryption, Big data analysis, Dimension reduction method

## 1 | INTRODUCTION

Wireless sensor networks is a brand-new information acquisition and processing technology, combing the logical information world with the objective physical world. Wireless sensor networks shows broad application prospects in more and more fields. Cloud computing is a flexible way of organization and provision of IT resources. It supports distributed storage and parallel processing, and its data processing framework can process most of the data in a local computing manner instead of distance transmission. Sensor cloud [1,2,3,4] is composed of wireless sensor network and cloud computing, integrating measurement perception, network transmission and cloud components, as shown in Figure 1. By integrating the sensor components and networking components, the data collected by the sensors are directly uploaded to the cloud platform, and users directly use the data from the cloud platform.

As shown in Figure 1, Sensors convert a large number of physical parameters into control information and upload it to the cloud server, which lead to a dramatic increase in the data objects and index variables in the cloud server. In order to generate data analysis reports for decision makers, the company needs to reduce the dimension of large-scale data in a large number of dimensions in the cloud server for digging deeper. PCA [5,6] and LDA [7,8] are two classic algorithms in statistics, and they have been widely used in various fields. They convert multiple indicators into fewer comprehensive indicators by looking for a linear combination of the same or different categories of things to achieve the purpose of dimensionality reduction or classification.

In some confidentiality scenarios, such as chip manufacturing, military enterprises, and nuclear power plants, companies encrypt important data collected by sensors in the production process, as well as data analysis results, and save them locally,

**Abbreviations:** PCA, principal component analysis; LDA, linear discriminant analysis; CSU, cloud server user; CSP, cloud server; SNP, sensor service provider.

  

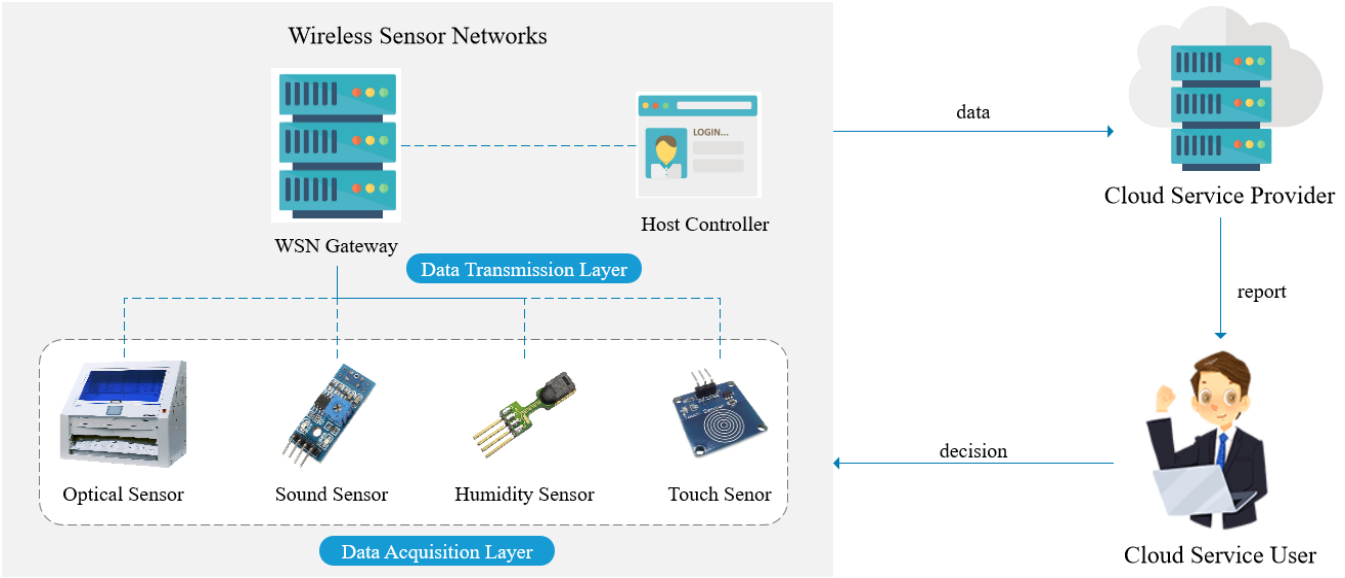


**FIGURE 1** Sensor-Cloud System of Industrial Process Control.

but it costs a lot of storage and computing at the same time. With the rapid development of machine learning and data analysis algorithms, the requirements for computing resources are getting higher and higher. Due to the hardware limitations of wireless sensor networks, companies have to outsource computing tasks to cloud servers. However, such applications have been greatly restricted for lack of trust in cloud servers. Therefore, how to solve the problem of cloud data leakage has become the focus of recent cloud computing research.

Homomorphic encryption can make direct operations on ciphertext, and the operation effect is equivalent to plaintext operations. Users can perform homomorphic encryption on files, entrust the cloud server to directly perform homomorphic operations on the ciphertext on the premise of ensuring data privacy, which is equivalent to the operation on the plaintext. This makes it possible for us to do machine learning and data analysis on ciphertexts, and this kind of research also has important practical significance. Chen et al.[9] proposed an approximate homomorphic encryption scheme (CKKS encryption scheme) in 2016, and has been widespread concerned recently.

There are two main schemes for data processing on cryptographic domains, one is to do some simple statistical analysis algorithms using the traditional Paillier homologous encryption scheme, the other is to do some classical machine learning algorithms using the new homologous encryption scheme. Because Paillier is a semi-homologous encryption scheme, its computing and encoding methods are limited, so the previous oneComputing and communicating efficiency of the scheme is low and the scheme is complex. Because self-bootstrapping and division of ciphertext are not efficient at present, the latter scheme is limited to simple machine learning algorithms with low multiplication depth, such as training model on plain text, making simple prediction of ciphertext through encryption model. Because the CKKS homologous encryption scheme supports floating-point encryption.The number is encrypted, and has considerable computational efficiency, and can realize the batch operation of ciphertext. Therefore, this paper chooses CKKS homologous encryption scheme as the basis of the scheme.

In view of the importance of dimensionality reduction for massive data in sensor cloud system while protecting data privacy, this paper designs PCA and LDA algorithms in security sensor cloud system based on CKS homologous encryption scheme.

## 1.1 | Related Work

Rivest et al.[10] originally proposed the concept of homomorphic encryption in 1978, which allows people to directly operate on the ciphertext, and the result of the operation after decryption is the same as that on the plaintext. RSA algorithm[11] and EIGamel algorithm[12] were proposed in 1978 and 1985, supporting homomorphic multiplication. Benalo algorithm[13] and Paillier algorithm[14] were proposed in 1988 and 1999 respectively, and they both support homomorphic addition. The Goldwasser Micali algorithm[15] was proposed in 1982, which supports homomorphic bit XOR. These early classic partial homomorphic encryption schemes have been fully applied in some specific scenarios.

**TABLE 1** Functionality Comparison with Existing Related Schemes.

| Literature | Encryption scheme | application |
|---|---|---|
| 22 | BGV11 [19] | calculate mean and standard deviation |
| 23 | Bra12 [20] | calculate covariance |
| 24 | Paillier [14] | fit Logistic Functions |
| 25,26,27 | CKKS [9] | logistic regression |
| 31 | CKKS [9] | nGraph-HE2 |
| 35 | CKKS [9] | CNN building blocks |

In 2009, Gentry [16], the employer of IBM proposed the first fully homomorphic encryption scheme based on ideal lattice, which triggered the hot spot of fully homomorphic encryption. In the past few years, the computational efficiency of fully homomorphic encryption has been a big problem. After that, Gentry proposed the DGHV scheme [17] and GSW13 scheme [18] in 2010 and 2013 respectively. The former one implemented a fully homomorphic encryption refering to integers based on the approximate greatest common divisor problem, and the latter one proposed the first homomorphic encryption scheme refering to identity based on the error learning problem. Identity-based homomorphic encryption scheme. At the same time, Brakerski et al. proposed the BV11 scheme [19] based on the difficult problem of the standard lattice, which effectively shortened the ciphertext and reduced the decryption complexity of the scheme. In 2012 and 2014, they respectively proposed the Bra12 scheme [20] and the BGV12 scheme [21]. The former one establishes a hierarchical homomorphic encryption scheme by avoiding analog-to-digital conversion, and the latter one can establish hierarchical homomorphic encryption without bootstrapping. Recently, Cheon [9] et al. proposed the CKKS scheme based on the error learning problem, which has been widely used because it supports the approximate operation of the ciphertext. We have sorted out the articles related to data processing using classical homologous encryption schemes, as shown in Table 1.

With the gradual maturity of fully homomorphic encryption technology, researchers focus more on its application. As early as 2011 and 2012, researchers applied fully homomorphic encryption to statistical analysis. Literature [22] gave a description and evaluation of the mean and standard deviation of the ciphertext, literature [23] also showd how to calculate the covariance of the ciphertext. With the advent of big data era and the development of artificial intelligence, homomorphic encryption technology is gradually combined with machine learning. Wu [24] et al. used the Paillier encryption scheme and used a polynomial to fit the Logistic function, but the calculation component of the polynomial fit increased exponentially. Kim et al. [25,26] successively proposed two homomorphic schemes on Logistic regression based on the CKKS encryption scheme, but they only considered small-scale data sets, and the multiplication depth of their schemes was limited. Han [27] et al. improved the scheme of Kim et al., which can support multiplication iterations for any number of times. At the same time, there are also many studies on predictions using machine learning algorithms including neural networks on homomorphic encrypted data [28,29,30,31,32], but the amount of calculation using models to predict is much easier compared to training models. With the gradual maturity of the combination of homomorphic encryption and machine learning, many studies have begun to train models on encrypted data [33,34,35]. While researching data processing on ciphertext, people have also done a lot of research on security authentication scheme in cloud server [39-43], which has a key practical significance [36,37,38,39,40]. Nevertheless, in terms of statistical analysis, people have to use some iterative algorithms to replace some of the steps in the traditional calculation model for the limitation of the calculations of the ciphertext data. How to perform large-scale iterative operations while ensuring computational efficiency and accuracy is a tricky problem for lack of ciphertext space.

## 1.2 | Our Contribution

In order to efficiently and safely implement the dimensionality reduction algorithm of the ciphertext in the sensor cloud system, and to protect the privacy of the real data and processing results in the process of data transmission and processing, this paper designs the dimension method of the ciphertext data in two sensor cloud systems algorithm. Our contributions are as follows:

1. This paper encrypts data based on the CKKS homomorphic encryption scheme, and describes in detail the PCA and LDA dimensionreduction methods in ciphertext data.
2. We use an iterative algorithm to replace the division operation, and use the power multiplication method to iteratively find the eigenvalues of the ciphertext matrix to solve the limitations of the ciphertext data operation method. And by

appropriately increasing the interactive transmission between the cloud server and the user, the limitation of the ciphertext operation efficiency and the number of times in the ciphertext space is solved.

3. The algorithm in this paper applies an iterative algorithm, which only involves matrix addition and multiplication. Therefore, we can evaluate these algorithms directly on the ciphertext. We select the classification dataset IRIS which is commonly used in machine learning and call the TenSEAL[41] library, and through a large number of experimental tests on the CKKS encryption scheme, ciphertext domain basic operations, and iterative algorithms, we screened out the best encryption parameters and iterative parameters, and realized the dimension reduction method of ciphertext data in the two sensor cloud systems efficiently and safely.

The rest of this article is introduced as follows. We introduce the system model and security goals of our solution in Chapter 2, and introduce several basic algorithms used in this article in Chapter 3. We respectively introduce the PCA algorithm and LDA algorithm of the ciphertext domain as well as the security analysis in Chapter 4 and Chapter 5. In Chapter 6, we conduct experimental tests on the basic algorithms, and the PCA algorithm and LDA algorithm of the ciphertext domain to prove its effectiveness and efficiency. Finally, we conclude the article in Chapter 7.

## 2 | MODELS AND GOALS

### 2.1 | System and Security Model

The system model of this article is shown in Figure 2, which consists of three parts, CSU, CSP, and SNP. We perform dimension reduction analysis on the ciphertext data stored in the CSU and the encrypted data collected in the SNP. In the system model, CSP is assumed to be "honest and curious", with strong computing power and storage capacity. SNP is responsible for processing and transmitting the collected data, and has a small amount of computing power. CSU is responsible for generating keys, encrypting and uploading ciphertext data, and can bear part of the calculation overhead. The functions of each part are as follows:

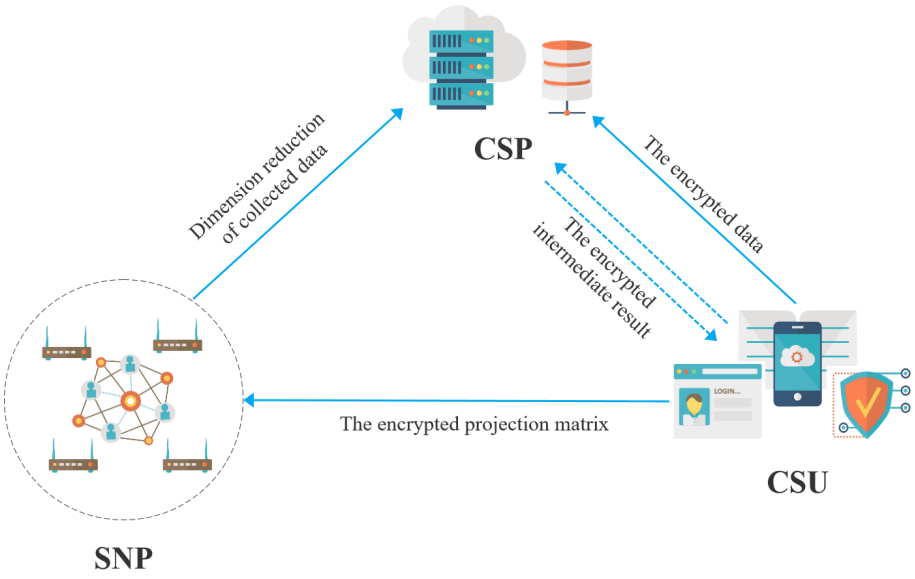


**FIGURE 2** PCA and LDA Algorithms in Securing Sensor-Cloud System.

1. CSU: Generates public and private keys locally, encrypts and uploads cryptographic data, interacts with CSP to compute cryptographic results, and sends an encrypted projection matrix to SNP.

2. CSP: Provides remote storage and computing services for CSUs and SNPs, but can only obtain and compute ciphertext data.
3. SNP: Installed near the CSU, it is part of the company or a key asset responsible for transmitting the collected data to the CSU. It can also process the data in real time based on the projection matrix sent by the CSU, and send the ciphertext processing results to the CSP for storage.

In the system model, CSU outsources local encrypted data to CSP, and data confidentiality is protected by basic encryption primitives. In the communication process, the CSU and the CSP transmit intermediate calculation results to each other, and the CSU sends the final calculation results to the SNP. Because both the original data, the intermediate and final calculation results may contain the users' sensitive information, we should implement the privacy protection of the data throughout the communication process. CSP can perform any operations on ciphertext data, but cannot obtain any plaintext and key information through ciphertext. SNP can use the ciphertext result to perform simple processing on the collected plaintext data, but it must transmit the data to other parts in ciphertext form. Since CSP is assumed to be "honest and curious", we list the security issues that security sensor cloud systems may encounter during the cryptographic dimension reduction process:

1. CSP works strictly according to the protocol designed, but extrapolates other information based on information received legally.
2. CSP uses stored cryptographic data and intermediate calculation results to attempt to crack or steal the clear data and private key of the CSU.
3. In the process of CSU and SNP transferring data to CSP, hackers may maliciously intercept data to crack private keys and attack CSP to recover CSU plaintext data.

## 2.2 | Notations

In order to show the algorithm in this article intuitively, we briefly introduce the related symbols used in this article, as shown in Table 2. Lowercase bold letters represent vectors and uppercase bold letters represent matrices. Add *enc* in front to indicate the ciphertext form of the data.

# 3 | PRELIMINARIES

## 3.1 | CKKS homomorphic encryption algorithm

The homomorphic operation of the homomorphic algorithm on the ciphertext is equivalent to the operation on the plaintext: $D(E(m_1) + E(m_2)) = m_1 + m_2$, $D(E(m_1) \cdot E(m_2)) = m_1 \cdot m_2$. Users can perform homomorphic encryption on files, and entrust the cloud server to directly perform homomorphic operations on the ciphertext on the premise of ensuring data privacy, and the result is equivalent to the operation on the plaintext. At present, the more mature applications are generally based on the Paillier encryption algorithm [14], but since the algorithm was proposed very early, its security and efficiency need to be improved. Chen

**TABLE 2** Notations.

| Symbols | Description |
|---|---|
| $\mathbf{x_i}, i \in [1, m]$ | Indicator variables |
| $\mathbf{y_i}, i \in [1, n]$ | Data object |
| $\mathbf{Y}^{n \times m}$ | Sample data |
| $\mathbf{U}$ | Projection Matrix |
| $\mathbf{Z}$ | feature space |
| $\mathbf{R}$ | Correlation matrix |
| $\lambda_i$ | Characteristic value |
| $\mathbf{u}_i$ | Feature vector |
| $\mathbf{S_W}$ | Intra-class Walking Matrix |
| $\mathbf{S_B}$ | Interclass Walking Matrix |
| $\mathbf{a}$ | Initial vector |

et al.[9] proposed an approximate arithmetic homomorphic encryption (CKKS) algorithm in 2016. Recently, a large number of theoretical and application researches have been conducted on the CKKS algorithm. In this part, this article mainly analyzes the CKKS homomorphic encryption algorithm. The following describes the main algorithm flow of CKKS:

1. **Setup**$(1^\lambda) \rightarrow (N, \chi_{key}, \chi_{err}, \chi_{enc}, L, q_l)$
   Given the security parameter $\lambda$, choose the power N of two integers.Set distributions $\chi_{key}$, $\chi_{err}$, $\chi_{enc}$ for keys, error learning, and encryption on $R = \mathbb{Z}[X]/(X^N + 1)$.For a base integer $p$ and the number of levels L, set the modulus of the ciphertext $q_l = p^l (1 \leq l \leq L)$, and then randomly generate an integer $p$ to output $pp = (N, \chi_{key}, \chi_{err}, \chi_{enc}, L, q_l)$.
2. **KeyGen**(**params**) $\rightarrow$ (**pk**, **sk**, **ks**, $\mathbf{rk}_r$, **ck**)
   a.**PSKeyGen**(**params**) $\rightarrow$ (**pk**, **sk**): Randomly generate $s \leftarrow \chi_{key}$, and set the private key $sk \leftarrow (1, s)$. Randomly generate $a \leftarrow U\left(R_{q_L}\right)$ and $e \leftarrow \chi_{err}$, and set the public key $pk \leftarrow (b, a) \in R_{q_L}^2$, where $b = -as + e \pmod{q_L}$ are.
   b.**KSGen**$(sk, s')$: Randomly generate $s' \in R$, $a' \leftarrow R_{P \cdot q_L}$, and $e' \leftarrow \chi_{err}$ and set the evaluation key $evk \leftarrow \left(b', a'\right) \in R_{P \cdot q_L}^2$, where $b' = -a's + e' + Ps' \pmod{P \cdot q_L}$. Then generate $\mathbf{sk} \leftarrow \mathbf{KSGen}_{\mathbf{sk}}(s^2)$, $\mathbf{rk}_r \leftarrow \mathbf{KSGen}_{\mathbf{sk}}(\kappa_{5^r}(s))$, $\mathbf{ck} \leftarrow \mathbf{KSGen}_{\mathbf{sk}}(\kappa_{-1}(s))$ respectively.
3. **Encrypt**$(m, pk) \rightarrow$ **ct**
   Randomly generate $r \leftarrow \chi_{enc}$ and $e_0, e_1 \leftarrow \chi_{err}$. Output ciphertext $\mathbf{ct} = r \cdot pk + (m + e_0, e_1) \pmod{q_L}$.
4. **Decrypt**$(\mathbf{ct}, sk) \rightarrow m$
   For the ciphertext of level $l$, calculate and output the plaintext $m' =< \mathbf{ct}, \mathbf{sk} > \pmod{q_l}$ .
5. **Add**$(\mathbf{ct}, \mathbf{ct}') \rightarrow \mathbf{ct}_{\text{add}}$
   For the ciphertext $\mathbf{ct}, \mathbf{ct}'$ of the same level, calculate and output the addition result $\mathbf{ct}_{\text{add}} = \mathbf{ct} + \mathbf{ct}' \pmod{q_l}$.
6. $\mathbf{CMult}_{\mathbf{ks}}(a, \mathbf{ct}) \rightarrow \mathbf{ct}_{\text{cmult}}$
   For a ciphertext **ct** with a constant $a \in R$ and $l$ level, calculate and output the multiplication result: $\mathbf{ct}_{\text{cmult}} = a \cdot \mathbf{ct} \pmod{q_l}$.
7. $\mathbf{Mult}_{\mathbf{ks}}(\mathbf{ct}, \mathbf{ct}') \rightarrow \mathbf{ct}_{\text{mult}}$
   $\mathbf{ct} = (c_0, c_1)$, $\mathbf{ct}' = (c'_0, c'_1) \in R_{q_l}^2$, calculates $(d_0, d_1, d_2) = (c_0 c'_0, c_0 c'_1 + c'_0 c_1, c_1 c'_1) \pmod{q_l}$. Output the result of ciphertext multiplication $\mathbf{ct}_{\text{mult}} = (d_0, d_1) + P^{-1} \cdot d_2 \cdot \mathbf{sk} \pmod{q_l}$.
8. $\mathbf{Rescale}_{l \rightarrow l'} \rightarrow (\mathbf{ct})$
   For the ciphertext of the level $l$, calculate and output the ciphertext $\mathbf{ct}' = p^{l'-l} \cdot \mathbf{ct} \in \pmod{q_{l'}}$ of the level$l'$.

Due to the homomorphic nature of the CKKS encryption scheme, the calculation of the ciphertext by the cloud server is equivalent to the calculation on the plaintext, which protect users privacy and improve the efficiency of encryption.

## 3.2 | Principal Component Analysis

Principal component analysis (PCA) is a dimension reduction algorithm[42], which was introduced by Pearson in 1901 for non-random variables. In 1933, Hotelling extended this method to the case of random vectors. Principal component analysis is different from the cluster analysis for its strict mathematical theory.

The main purpose of principal component analysis is to use fewer variables to explain most of the variation in the original data, and to transform many highly correlated variables we have into mutually independent or uncorrelated variables. Usually, less several new variables so-called principal components are selected, which can explain most of the variation in the data, especially the comprehensive indicators of the data.

Assume that the data $\mathbf{Y}^{n \times m}$ stored by users in the cloud server has a total of $n$ data objects, each data object has $m$ index variables, $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m$ respectively, and the $j$-th index variable of the $i$-th data object is $a_{ij}$.For the $j$-th indicator variable $\mathbf{x}_j$, calculate the standardized indicator variable $\mathbf{x}_j$ and the correlation coefficient matrix $\mathbf{R} = (r_{ij})_{m \times m}$.

Calculate the eigenvalues $\lambda_1, \lambda_2, ..., \lambda_m$ of the correlation coefficient matrix $\mathbf{R}$ and the corresponding eigenvectors $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_m$, form the projection matrix by the eigenvectors $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_k]$, and transform the data samples into the new eigenspace $\mathbf{Z} = \mathbf{U}^T \mathbf{Y}$.

Finally, calculate the information contribution rate $b_i = \lambda_i / \sum_{j=1}^{m} \lambda_j, i \in [1, m]$ of each eigenvalue and principal component and the cumulative contribution rate $\alpha_k = \sum_{j=1}^{k} \lambda_j / \sum_{j=1}^{m} \lambda_j$, $k \in [1, m]$ of the first $k$ eigenvalues. Select the first $k$ principal components close to 1 to replace the original $m$ index variables for subsequent analysis.

## 3.3 | Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is also a dimension reduction algorithm. It was first proposed by Fisher in 1936 focused on two classification problems[43], also known as Fisher linear discrimination. PCA is an unsupervised dimension reduction technique, learning the output of sample categories alone, whlie LDA is a supervised data dimension reduction algorithm. On the training samples, there is no need to use the category label by using the PCA algorithm, while it's necessary to provide the category label of the data by using the LDA algorithm. The central idea of the LDA algorithm is "the smallest within-class variance, and the largest between-class variance after projection", which means that the data is projected in low dimensions. After projection, it is hoped that the projection points of the same category of data are as close as possible, and the different distance between the center of category is as far as possible.

Assume that the data $\mathbf{Y}^{n \times m}$ stored in the cloud server has $n$ data object $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n$, which are divided into into $d$ categories $\chi_i, i = 1, 2, ..., d$ within $N_i$. The number of samples in each category is $N_i$.Each data object has $m$ index variables, which are $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m$ respectively.

First, calculate the average value of each index variable of data object in each category, and obtain the m-dimensional mean value vector $\mathbf{m}_i, i = 1, 2, ..., d$. Then calculate the mean vector $\mathbf{m}$ of all data objects.

Calculate the $m \times m$ within-dimension matrix $\mathbf{S_W} = \sum_{i=1}^{d} \sum_{\mathbf{y}_j \in \chi_i} \left(\mathbf{y}_j - \mathbf{m}_i\right) \left(\mathbf{y}_j - \mathbf{m}_i\right)^T$ and the $m \times m$ inter-dimension matrix $\mathbf{S_B} = \sum_{i=1}^{d} N_i (\mathbf{m}_i - \mathbf{m}) (\mathbf{m}_i - \mathbf{m})^T$. And calculate the eigenvalues and eigenvectors of $\mathbf{S_W^{-1}S_B}$ are, and sort the eigenvectors according to the eigenvalue size. Then select the eigenvectors corresponding to the first $k$ largest eigenvalues to form a $m \times k$ dimensional projection matrix $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_k]$. Finally, the samples are transformed into the new feature space $\mathbf{Z} = \mathbf{U}^T\mathbf{Y}$.

## 3.4 | Ciphertext Basic Operations

Because ciphertext data cannot be directly used for inversion and radical operations, it is necessary to use the functions $Inv(), Sqrt()$ raised by Cheon[43] et al. before performing eigenvalue operations. Because ciphertext has a limited number of operations, we use an iterative algorithm to replace the traditional division and square operations. The calculation steps are shown in algorithm 1 and 2 respectively.

**Algorithm 1** *Inv*($x$;$d$)

**Input:** $0 < x < 2, d \in N$
**Output:** $1/x$

$a_0 = 2 - x$
$b_0 = 1 - x$
**for** $n$ **in** $(0, d-1)$ **do**
  $b_{n+1} = b_n^2$
  $a_{n+1} = a_n \cdot (1 + b_{n+1})$
**end for**
**return** $a_d$

**Algorithm 2** *Sqrt*$(x;d)$

**Input:** $0 \leq x \leq 2, d \in N$

**Output:** $\sqrt{x}$

$a_0 = x$

$b_0 = x - 1$

**for** $n$ **in** $(0, d-1)$ **do**

$a_{n+1} = a_n \cdot \left(1 - \frac{b_n}{2}\right)$

$b_{n+1} = b_n^2 \cdot \left(\frac{b_n - 3}{4}\right)$

**end for**

**return** $a_d$

# 4 | PCA FOR ENCRYPTED DATA

## 4.1 | Ideal Case

First, assume that under ideal circumstances, the CKKS encryption scheme is absolutely secure, the ciphertext space is large enough to support sufficient modulus reduction in multiplication, and the computational efficiency in the ciphertext is considerable, with small error during the operation. The ideal system model is shown in Figure 3.

According to the system model, the system algorithm is listed as follows:

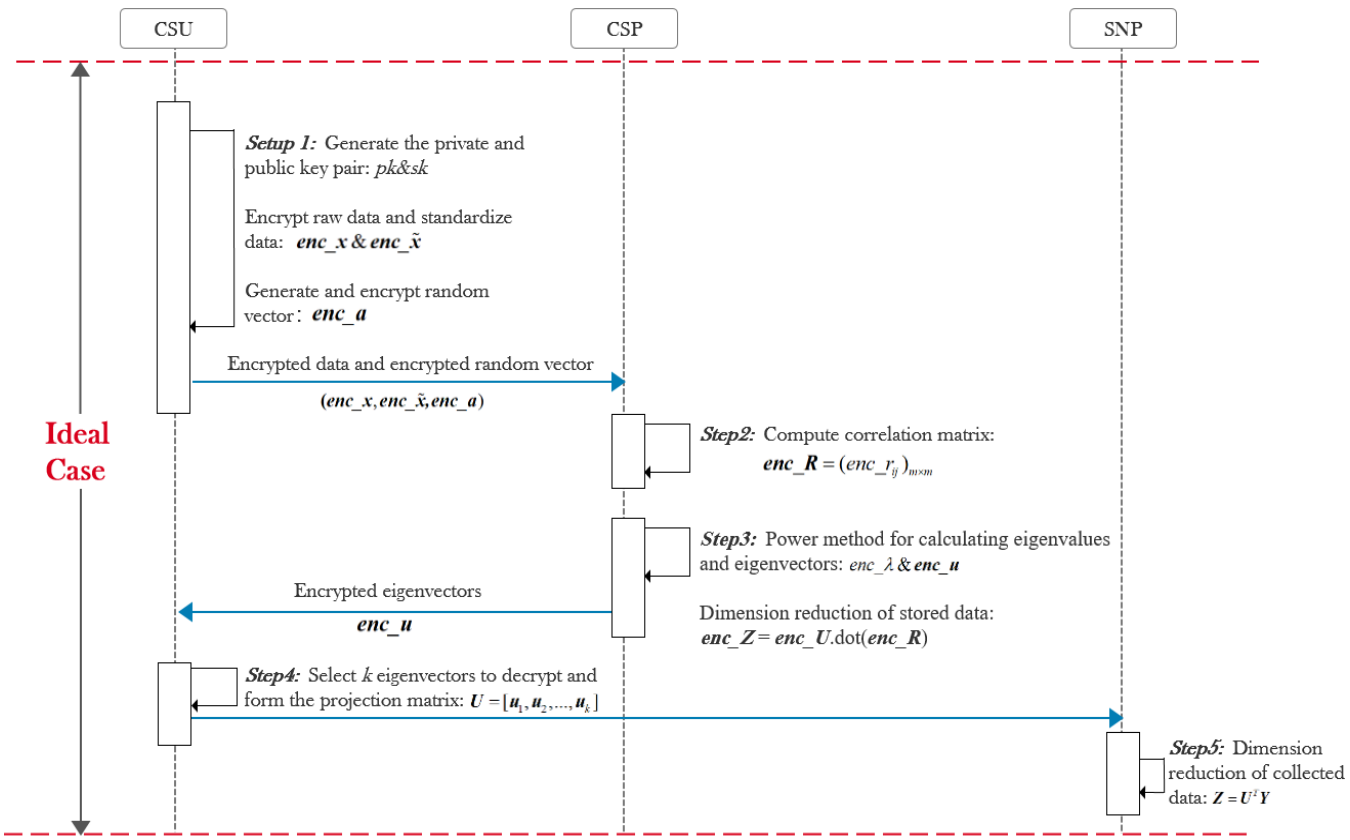


**FIGURE 3** PCA for encrypted data in ideal case.

**Step1**: Before uploading data, users generate public and private keys locally, and respectively encrypt the uploaded original data and standardized data by column (indicator variables), to obtain $\mathbf{enc\underline{c}x}_1, \mathbf{enc\underline{c}x}_2, ..., \mathbf{enc\underline{c}x}_m$ and $\mathbf{enc\underline{c}x}_1, \mathbf{enc\underline{c}x}_2, ..., \mathbf{enc\underline{c}x}_m$, where $\mathbf{enc\_x}_i = \left(\mathbf{enc\underline{c}a_{ij}}\right), \mathbf{enc\_x}_i = \left(\mathbf{enc\underline{c}a_{ij}}\right), i \in [1, m], j \in [1, n]$.

Initialize an m-dimensional vector **a** randomly and encrypt it. In order to simplify subsequent calculations, users need to upload the encrypted original data, encrypted standardized data, and encrypted vector ($\mathbf{enc\underline{c}x}, \mathbf{enc\underline{c}x}, \mathbf{enc\underline{c}a}$) to the cloud server at the same time.

**Step2**: The cloud server calculates the correlation coefficient matrix $\mathbf{enc\_R} = (enc\_r_{ij})_{m\times m}$, where:

$$enc\_r_{ij} = \frac{\sum_{k=1}^{n} enc\_\tilde{a}_{ki} enc\_\tilde{a}_{kj}}{n-1} = \frac{\mathbf{enc\_x_i}.dot\left(\mathbf{enc\_x_j}\right)}{n-1}, i, j \in [1, m]$$

$\mathbf{enc\_R}$ is a symmetric matrix, $enc\_r_{ij} = enc\_r_{ji}$, $enc\_r_{ii} = enc\_r_{jj}$.

**Step3**: Power method to calculate eigenvalues and eigenvectors $enc\_\lambda$ & $\mathbf{enc\underline{c}u}$. Since all calculations are performed on standardized data, so $enc\_r_{ij} \leq 1$, should be all within the calculation range of algorithm 1 and 2. After that, the power method is used for the correlation coefficient matrix $\mathbf{enc\_R}$ to calculate each eigenvalue and eigenvector according to the size. The calculation steps are shown in algorithm 3.

**Algorithm 3** *PowerMethod*($\mathbf{enc\_R}$;$\mathbf{enc\underline{c}a}$;$d$)

**Input:** $\mathbf{enc\_R}, \mathbf{enc\underline{c}a}, d \in N$
**Output:** $enc\_\lambda_k, \mathbf{enc\underline{c}u_k}$

```
for k in (0, m) do
    enc_R^(k) = enc_R − Σ_{i=1}^{k−1} enc_λ_i · enc_u_i · enc_u_i^T
    for n in (0, d) do
        enc_a^(n+1) = enc_R^(k) · enc_a^(n)
    end for
    l = Sqrt(enc_a^(d).dot(enc_a^(d)))
end for
return enc_λ_k = enc_a^(d)[1] / enc_a^(d−1)[1], enc_u_k = enc_a^(d) / l
```

According to the preset, the cloud server selects the eigenvector $\mathbf{enc\underline{c}u}_k, i = 1, 2, ..., k$ ccorresponding to the first $k$-th eigenvalue to form the projection matrix $\mathbf{enc\underline{c}U}$, realizes the dimension reduction of the original data by calculating $\mathbf{enc\_Z} = \mathbf{enc\underline{c}U}.dot(\mathbf{enc\_R})$. And send the encrypted projection matrix to the user.

**Step4**: After receives the projection matrix sent by the cloud server, CSU decrypts it for $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_k]$ and sends it to the SNP.

**Step5**: SNP can process dimension reduction on data $\mathbf{Y}'$ collected in real time $\mathbf{Z}' = \mathbf{U}^T\mathbf{Y}'$ according to the projection matrix.

## 4.2 | Real Case

However, in real situations, the size and space of the ciphertext are limited, and the size of the ciphertext on the operation efficiency, the limitation of the ciphertext space on the number of multiplications, and the accuracy of the number of iteration operations should be considered. Therefore, the CSU needs to bear part of the calculation overhead, and increase the number of communication interactions between the CSU and the CSP. At the same time, due to the security of the CKKS encryption scheme itself, the ciphertext space needs to be further restricted. The real system model is shown in Figure 4.

Since Step3 is the only difference between the actual situation and the ideal situation, here is only Step3's algorithm:

**Step3**: The CSP obtains the maximum eigenvalue $enc\_\lambda_1$ of the current matrix and the vector $\mathbf{enc\underline{c}a}_1^{(d)}$ after iterative calculation, by using the power method to $\mathbf{enc\_R}$. CSU decrypts ($enc\_\lambda_1, \mathbf{enc\underline{c}a}_1^{(d)}$) sent by CSP, calculates and saves the current

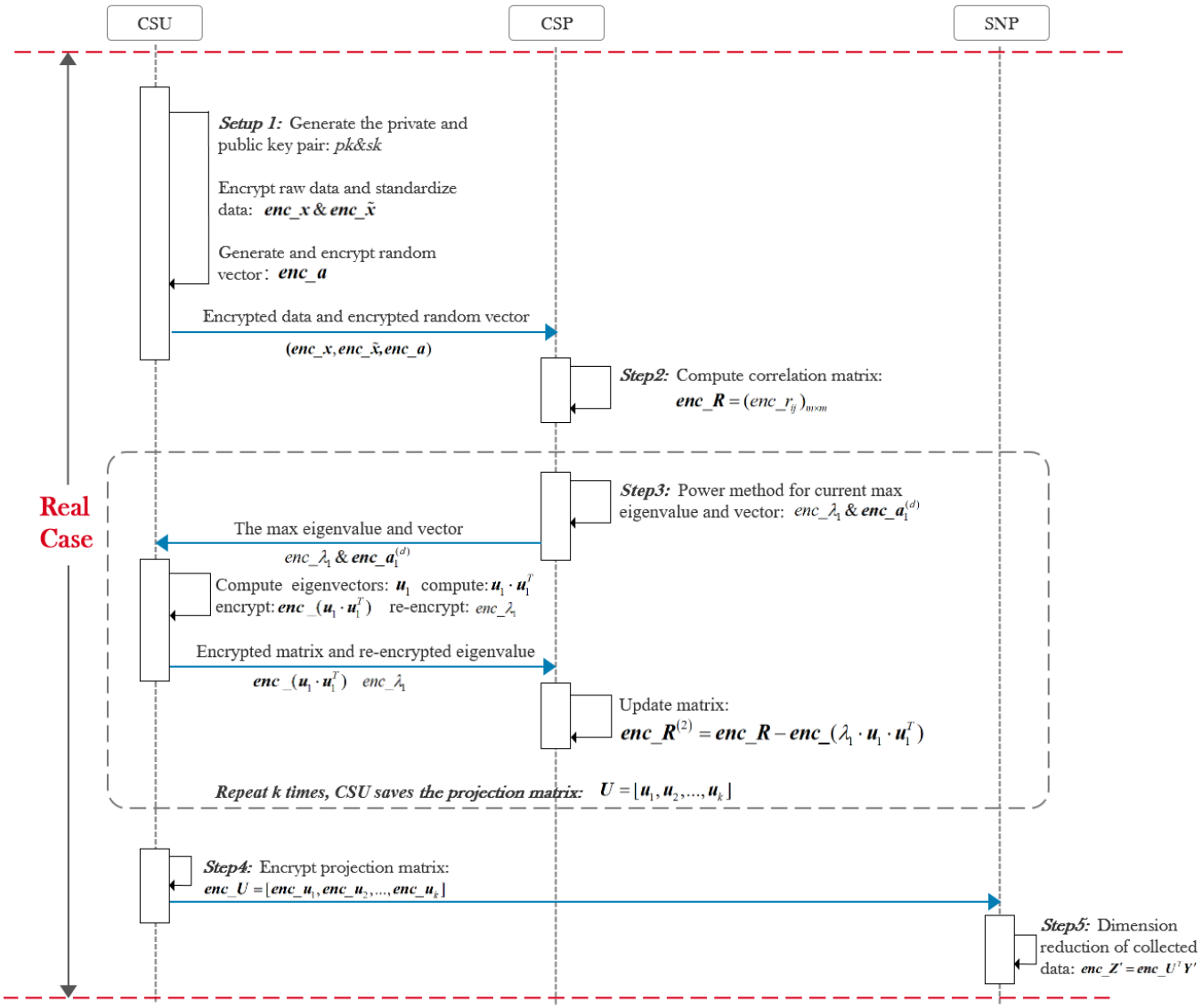


**FIGURE 4** PCA for encrypted data in real case.

characteristic value $\mathbf{u}_1 = \frac{\mathbf{a}^{(d)}}{l=Sqrt(\mathbf{a}^{(d)}.dot(\mathbf{a}^{(d)}))}$. The CSU calculates and encrypts the matrix $\mathbf{enc\_}(\lambda_1 \cdot \mathbf{u_1} \cdot \mathbf{u_1^T})$ and sends it to the CSP.CSP updates Matrix $\mathbf{enc\_R}^{(2)} = \mathbf{enc\_R} - \mathbf{enc\_}(\lambda_1 \cdot \mathbf{u}_1 \cdot \mathbf{u}_1^T)$. According to the pre-arranged agreement, the two parties circulate the above operations $k$ times. The CSU forms a projection matrix $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_k]$ according to the stored $k$ eigenvalues.

## 4.3 | security model

According to the system algorithm, CSU's data set $\mathbf{Y}^{n\times m}$, random vector $\mathbf{a}$ and eigenvalue $\mathbf{U}$ are all encrypted using the CKKS scheme. Therefore, the security of the data set $\mathbf{Y}^{n\times m}$, the random vector $\mathbf{a}$, and the feature value $\mathbf{U}$ can be attributed to the security of the CKKS encryption scheme. The CSP's correlation coefficient matrix $\mathbf{R}$ and the iterative result of the random vector are all calculated on the ciphertext, so its security can also be attributed to the security of the CKKS encryption scheme. The security of the CKKS scheme is determined by its own algorithm, and the CSP cannot recover the key and original data from the data it obtains. The CSU sends the projection matrix to the SNP and CSP in the form of ciphertext, so the SNP and CSP cannot obtain the eigenvalues of the plaintext and recover the key and the original data through the dimension reduction result.

In addition, supposing in the step of calculating eigenvalues in **Step3**, CSP performs $p$ iterations on the power method and $q$ iterations on the division, then the size of the ciphertext must satisfy the support of $p + q + 3$ multiplications, including 1 multiplication in calculating the correlation matrix and the eigenvalue respectively, $p$ times power iteration and $q + 1$ times division iteration.

# 5 | LDA FOR ENCRYPTED DATA

## 5.1 | Real Case

Since the LDA system model is similar to the PCA system model, the ideal situation and the real situation are only different in the steps of repeated calculation of the characteristic value, so we only discuss the LDA model in real situation, as shown in Figure 5.

The system algorithm is as follows:

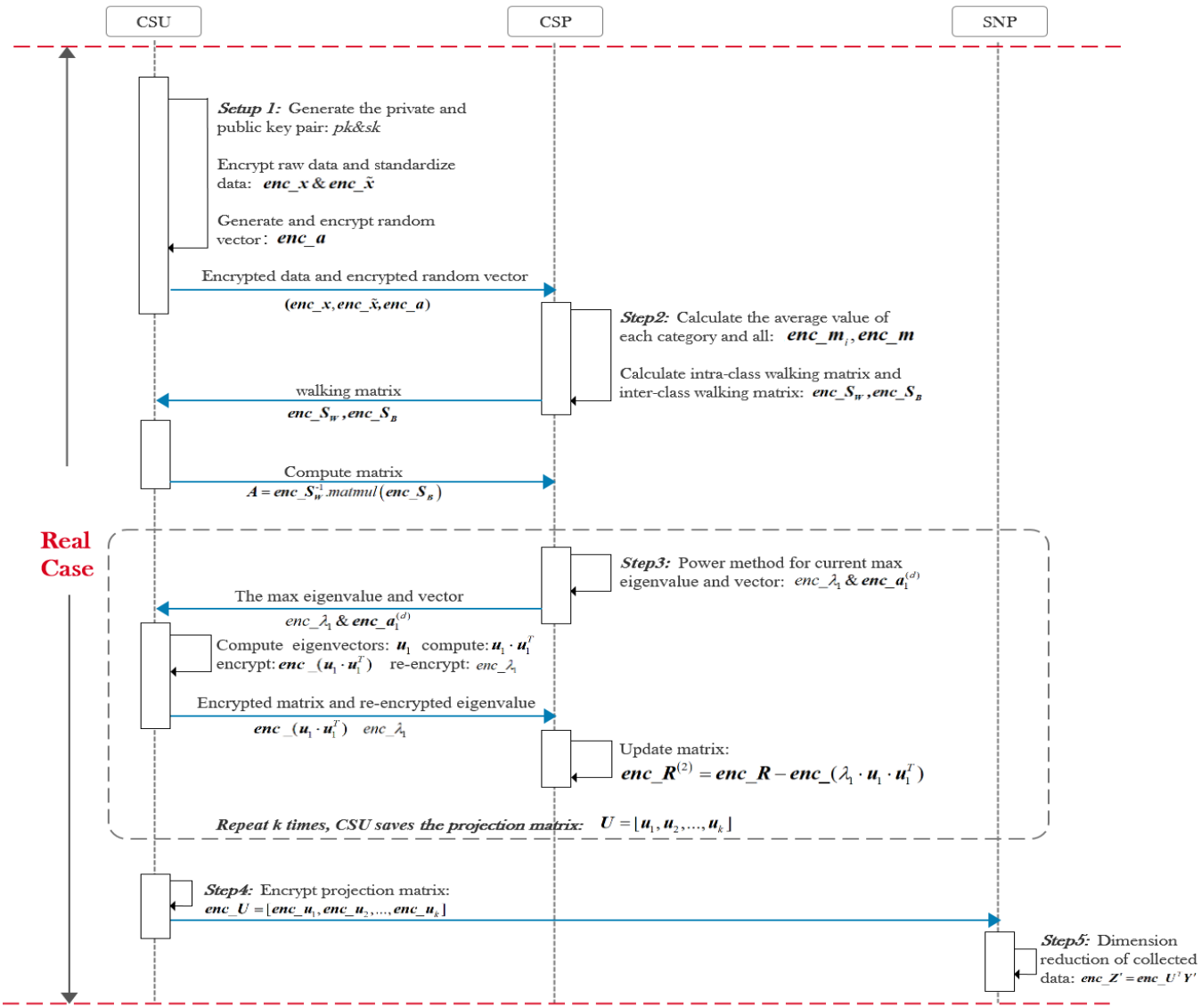


**FIGURE 5** LDA for encrypted data in real case.

**Step1**: Locally encrypted data $\underline{\mathbf{enc}}\mathbf{y}$,$\underline{\mathbf{enc}}\mathbf{\_m}_i$, $\underline{\mathbf{enc}}\mathbf{m}$. Assuming that the original data is standardized data, before uploading the data, the CSU generates public and private key locally, encrypts the uploaded original data line (data object), and obtains $\underline{\mathbf{enc}}\mathbf{y}_1, \underline{\mathbf{enc}}\mathbf{y}_2, ..., \underline{\mathbf{enc}}\mathbf{y}_n$. We initialize an m-dimensional vector $\mathbf{a}$ and encrypt it to get $\underline{\mathbf{enc}}\mathbf{a}$. Finally, the user uploads the encrypted original data and the encrypted mean vector ($\underline{\mathbf{enc}}\mathbf{y}$, $\underline{\mathbf{enc}}\mathbf{m}_i$, $\underline{\mathbf{enc}}\mathbf{m}$, $\underline{\mathbf{enc}}\mathbf{a}$) to the CSP at the same time.

**Step2**: Calculate the dispersion matrix $\underline{\mathbf{enc}}\mathbf{S_W}, \underline{\mathbf{enc}}\mathbf{S_B}$.

CSP first calculates the average value $\mathbf{enc\_m}_i, i = 1, 2, ..., d$ f each index variable of each category data object and the average value vector $\mathbf{enc\_m}$ of all data objects. Since the original data is standardized data, all subsequent calculation results are within the calculation range of algorithm 1 and 2. After the cloud server receives the data, it calculates the intra-class dispersion matrix and the inter-class dispersion matrix $\underline{\mathbf{enc}}\mathbf{S_W}, \underline{\mathbf{enc}}\mathbf{S_B}$ according to algorithm 4. After that, the CSP sends the dispersion matrix $\underline{\mathbf{enc}}\mathbf{S_W}, \underline{\mathbf{enc}}\mathbf{S_B}$ to the CSU, and the CSU calculates and encrypts the matrix $\mathbf{enc\_A} = \mathbf{S_W^{-1}}.matmul\,(\mathbf{S_B})$.

**Algorithm 4** *CatterMatrix*($\underline{\mathbf{enc}}\mathbf{y}$;$\underline{\mathbf{enc}}\mathbf{m}_i$;$\underline{\mathbf{enc}}\mathbf{m}$)

**Input:** $\underline{\mathbf{enc}}\mathbf{y}$,$\underline{\mathbf{enc}}\mathbf{m}_i$,$\underline{\mathbf{enc}}\mathbf{m}$
**Output:** $\mathbf{enc\_S_W}, \mathbf{enc\_S_B}$

```
for i in (1, d) do
    for y_j in χ_i do
        S_j = (y_j − m_i) .matmul(y_j − m_i)^T
        enc_S_W+ = S_i
        enc_S_B+ = N_i · (m_i − m) .matmul(m_i − m)^T
    end for
end for
return enc_S_W, enc_S_B
```

**Step3**: CSP calculates the eigenvectors corresponding to the first $k$ largest eigenvalues through the interaction between *PowerMethod*($\underline{\mathbf{enc}}\mathbf{\_A}$;$\underline{\mathbf{enc}}\mathbf{a}$;$d$) and CSU. The CSU forms the projection matrix $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_k]$ according to the stored $k$ eigenvalues.

**Step4**: CSU encrypts projection matrix $\mathbf{enc\_U} = [\mathbf{enc\_u}_1, \mathbf{enc\_u}_2, ..., \mathbf{enc\_u}_k]$ and sends it to SNP.

**Step5**: SNP processes dimension reduction on the real-time collected data $\mathbf{Y}'$ according to the projection matrix $\mathbf{enc\_U}$, $\underline{\mathbf{enc}}\mathbf{Z}' = \underline{\mathbf{enc}}\mathbf{U}^T\mathbf{Y}'$ and stores or transmits the dimension reduction results in ciphertext form.

## 5.2 | security model

Similar to the security of the PCA algorithm, the security of CSU's data set $\mathbf{Y}^{n\times m}$, random vector $\mathbf{a}$ and feature value $\mathbf{U}$ can be attributed to the security of the CKKS encryption scheme. The security of the inter-class and intra-class walking matrix $\mathbf{S_W}$*and*$\mathbf{S_B}$, mean vector $\mathbf{m}_i$*and*$\mathbf{m}$, and random vector of CSP can also be attributed to the security of the CKKS encryption scheme. CSP cannot recover the key and original data from the data it obtains. SNP and CSP cannot recover the key and the original data through the dimensionality reduction result.

In addition, supposing that in the step of calculating eigenvalues in **Step3**, CSP performs $p$ iterations on the power method and $q$ iterations on the division, then the size of the ciphertext must satisfy the support of $p + q + 2$ multiplications, including 1 multiplication in calculating eigenvalue, $p$ times power iteration and $q + 1$ times division iteration. Since in **Step2**, CSU re-encrypts the intermediate matrix, leaving the modulus reduction before **Step3** alone.

# 6 | PERFORMANCE ANALYSIS

In this section, we evaluate the efficiency and accuracy of the algorithms presented in this paper. Under the environment of Intel? Core? i7-7700HQ CPU @ 2.80GHz/16 GB Ram, we use PyCharm 2020.1.1 x64 to call the TenSEAL library to encrypt

the data under the Windows 10 operating system, and the sklearn library to evaluate the plaintext, and The scheme is tested on the IRIS data set.

## 6.1 | Encryption efficiency test

In this part, we test the encryption efficiency of the CKKS encryption scheme under different encryption parameters. We call the TenSEAL library, use the CKKS encryption scheme, and record the encryption time of the IRIS data set by changing the values of the three encryption parameters poly-modulus-degree, coeff-mod-bit-sizes, and scale.

Among them, according to the TenSEAL library manual[41], poly-modulus-degree must be a power of 2. The larger the parameter's value, the larger the ciphertext size and the slower the ciphertext calculation efficiency, but it can support more complex ciphertext operations. The encoder scales the floating-point number by using the scale parameter. After each multiplication, the scale of the ciphertext will be doubled. Therefore, it is necessary to perform a rescaling operation to reduce it. The bit length of the large prime number of the reduction modulus is determined by the parameter in coeff-modules. The number of parameters in coeff-mod-bit-sizes determines the number of scalings and multiplications. The maximum number of digits of the value is directly related to poly-modulus-degree, as shown in Table 3.

First, we fix the value of poly-modulus-degree and the number of parameters in coeff-mod-bit-sizes, encrypt the data set under different scales, and record the average encryption time of 10 experiments as the final result, as shown in Figure 6.

After that, we fix the value of poly-modulus-degree, encrypt the data set under different parameters in coeff-mod-bit-sizes, and record the average encryption time of 10 experiments as the final result, as shown in Figure 7.

**TABLE 3** Encryption parameter correspondence.

| poly-modulus-degree | max coeff-modulus bit-length |
|---|---|
| 1024 | 27 |
| 2048 | 54 |
| 4096 | 109 |
| 8192 | 218 |
| 16384 | 438 |
| 32768 | 881 |

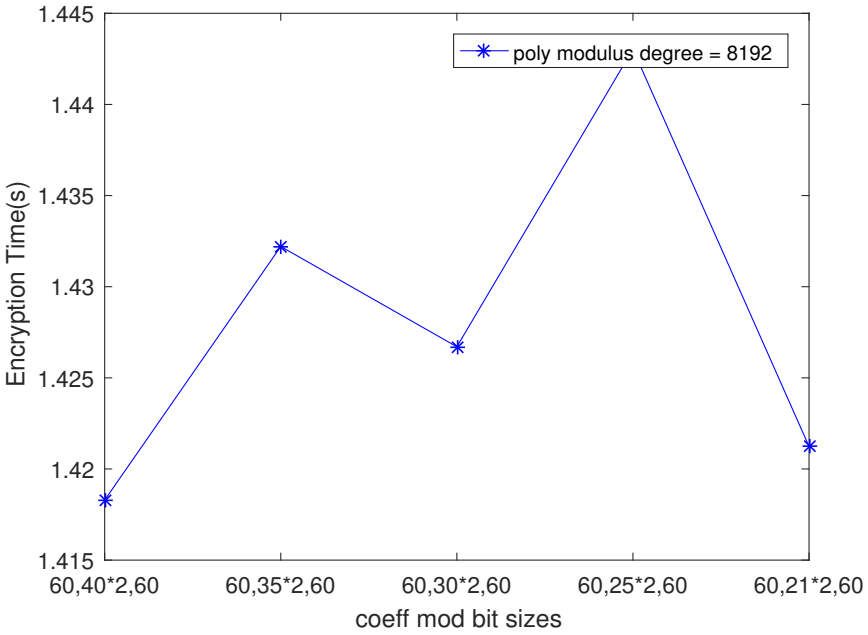


**FIGURE 6** Encryption time under different scale when poly-modulus-degree = 8192 and the number of coeff-mod-bit-sizes is 4.

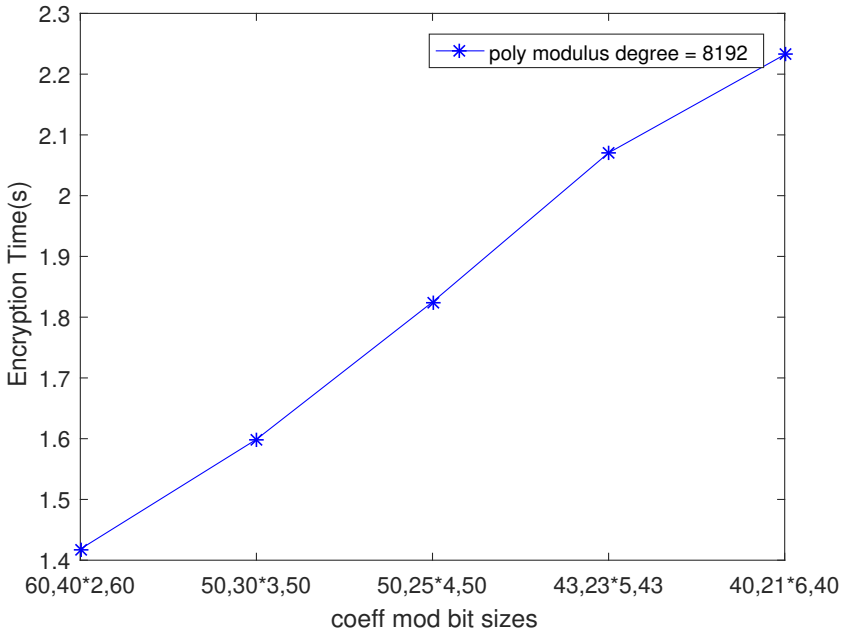


**FIGURE 7** Encryption time under different numbers of coeff-mod-bit-sizes when poly-modulus-degree = 8192.

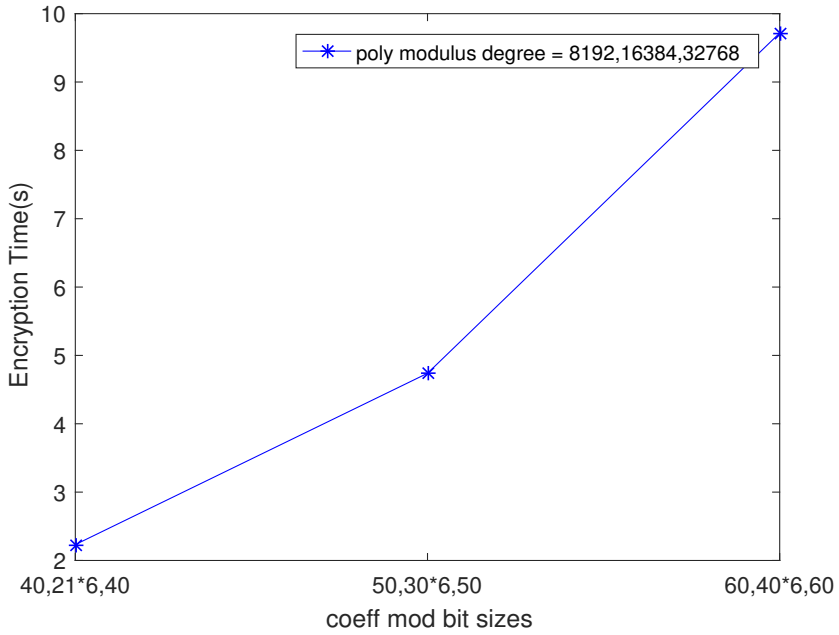


**FIGURE 8** Encryption time under different poly-modulus-degree when the number of coeff-mod-bit-sizes is 8.

Finally, we fix the number of parameters in coeff-mod-bit-sizes, encrypt the data set under different poly-modulus-degrees, and record the average encryption time of 10 experiments as the final result, as shown in Figure 8.

It can be seen that in case of the same value of poly-modulus-degree and the number of parameters in coeff-mod-bit-sizes, the encryption time of the data set has nothing to do with the scale size. In case of the same value of poly-modulus-degree, the data and encryption time increase linearly with the increasing of the number of parameters in coeff-mod-bit-sizes. In case of

the same number of parameters in coeff-mod-bit-sizes, the encryption time of the data set increases with the increasing value of poly-modulus-degree. These experimental results laid a good foundation for the next.

## 6.2 | Basic Operational Efficiency Test in Cryptographic Domain

In this section, we evaluate the division and square root operations of the CKKS encryption scheme, and look for a number of iterations with high calculation accuracy and high calculation rate for the parameter selection of subsequent experiments. Although in theory, the ciphertext of the CKKS encryption scheme can achieve homomorphic division by inverting the lattice, in fact there is no public code to achieve it. Therefore, the division and square root operations between ciphertexts need to be calculated iteratively.

First, we evaluate the ciphertext division. According to previous experiments, the encryption time of the data set increases with the number of parameters in coeff-mod-bit-sizes, instead of the size of the scale. The number of parameters in coeff-mod-bit-sizes determines the number of multiplications for ciphertext. So here we fix the value of poly-modulus-degree, and choose a set of random numbers between 0 and 2 [0.3459971, 0.7761305, 1.1590192, 1.4529988, 1.9371326] in 4 different sets of coeff-mod-bit-sizes parameters for calculation, as shown in Table 4.

**TABLE 4** Encryption parameter settings.

| nmber | poly-modulus-degree | coeff-mod-bit-sizes | scale |
|---|---|---|---|
| 1 | 8192 | 60,40,40,60 | 40 |
| 2 | 8192 | 50,30,30,30,50 | 30 |
| 3 | 8192 | 50,25,25,25,25,50 | 25 |
| 4 | 8192 | 40,23,23,23,23,23,40 | 23 |

For each random value under each set of encryption parameters, we record the average encryption time of 10 experiments and calculate the average result as the final result, as shown in Figure 9. The ciphertext calculation time is shown in Figure 10.It can be seen that when we are set poly-modulus-degree = 8192, coeff-mod-bit-sizes = [50,30,30,30,50], scale = 30, the number of iteration operations is 2, and the calculation accuracy and efficiency of ciphertext division are the best.

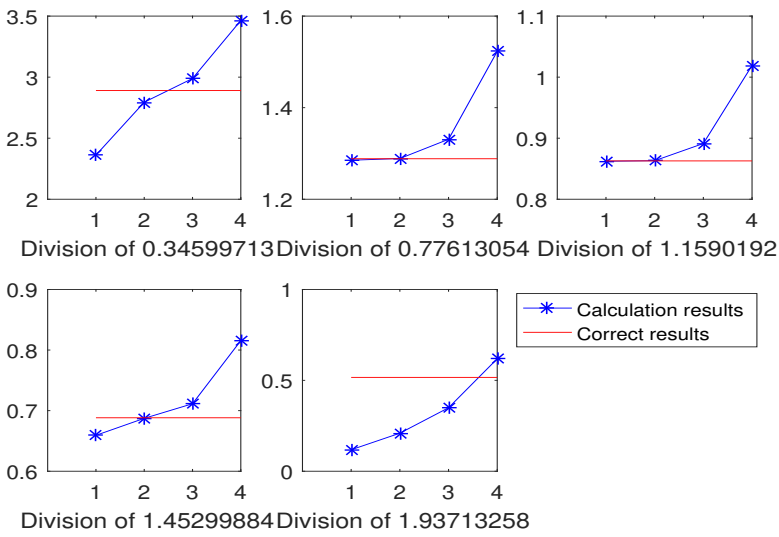


**FIGURE 9** Division evaluation of random numbers.

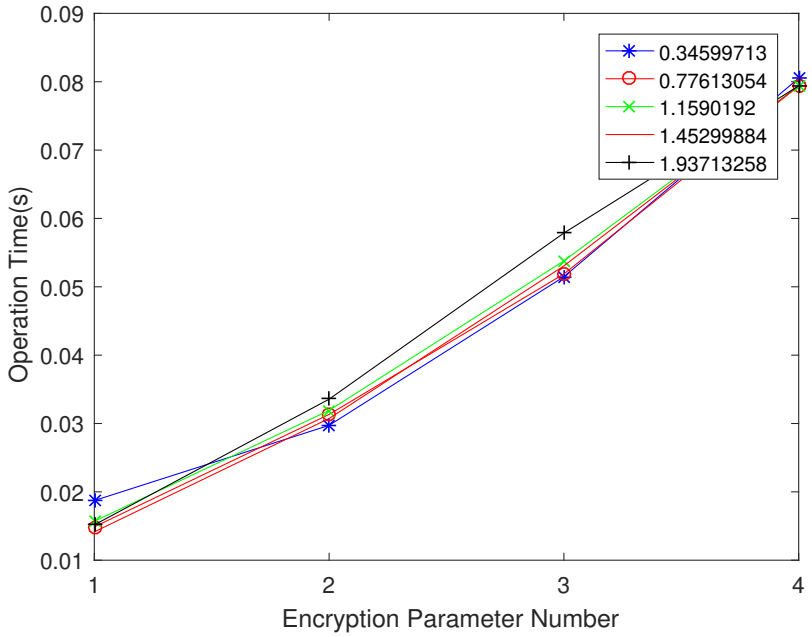


**FIGURE 10** Operation time of each ciphertext under different parameters.

After that, we use the same method to evaluate the square root of the ciphertext, choose random numbers [0.3459971, 0.7761305, 1.1590192, 1.4529988, 1.9371326] for calculation. The parameter configuration is shown in Table 5.

**TABLE 5** Encryption parameter settings.

| nmber | poly-modulus-degree | coeff-mod-bit-sizes | scale |
|---|---|---|---|
| 1 | 8192 | 60,40,40,60 | 40 |
| 2 | 8192 | 50,25,25,25,25,50 | 25 |

For each random value under each set of encryption parameters, we record the average encryption time of 10 experiments and calculate the average result as the final result, as shown in Figure 11. The ciphertext calculation time is shown in Figure 12.

What needs to be pointed out here is that the range of the square root of CKKS ciphertext in the original text is [0,1], and the experiment shows that the actual calculation range is [0,2]. When the plaintext is close to 1,the square root operation of the ciphertext has the maximum precision. But in general, the precision and efficiency of the square root of the ciphertext is not high, so the algorithm in this paper avoids the square root of the ciphertext and transfers it to the computational cost of the plaintext of the CSU.

## 6.3 | PCA algorithm test in Ciphertext

In this section, we test the PCA algorithm in the ciphertext domain. According to the previous experimental results, here we set the encryption parameters as poly-modulus-degree = 16384, coeff-mod-bit-sizes = [50,30,30,30,30,30,30,30,30,30,50], scale = 30, the dimensionality reduction process reduce the original 4-dimensional features of the IRIS data set to 2-dimensional. We use the algorithm proposed in this article and the PCA function in the sklearn machine learning library to reduce the dimension of the ciphertext data set and the plaintext data set respectively. The dimension reduction results of the two algorithms are shown in Figure 13 (The left figure figure shows CKKS-PCA, and the right figure shows sklearn-PCA), the three colors represent three different categories of data.

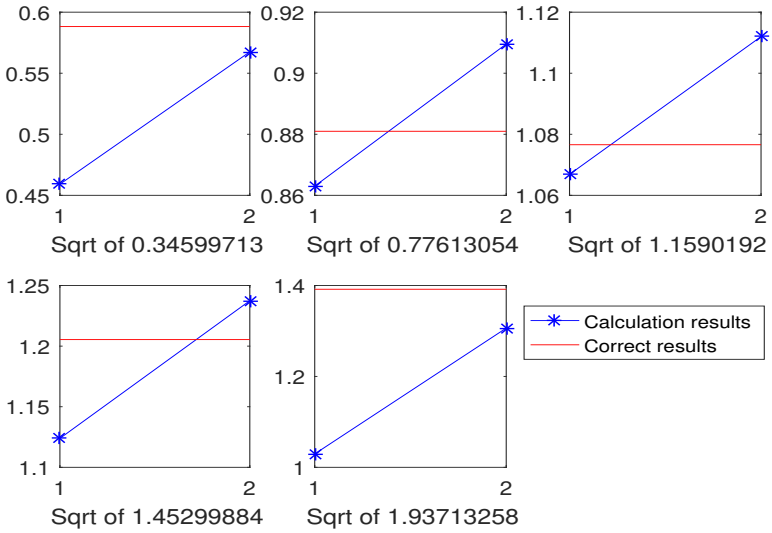


**FIGURE 11** Square root evaluation of random numbers.

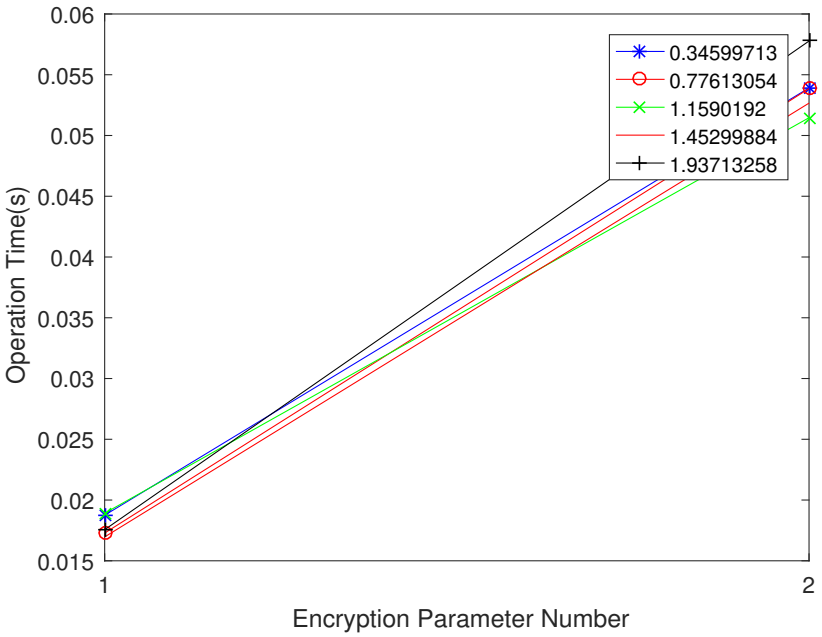


**FIGURE 12** Operation time of each ciphertext under different parameters.

The calculation results of the ciphertext data set using the algorithm proposed in this paper are shown in Table 6. There are simple data encryption and ciphertext multiplication in other steps besides **Step3**, so we only record the running results of **Step3**.

Among them, we carried out 4 iterations in the power method to find the eigenvalues, and 2 iterations in the ciphertext division. It can be seen that the results of the PCA dimensionality reduction algorithm in the ciphertext domain are effective. However, the errors caused by ciphertext calculations and iterative calculations lead to certain unavoidable errors in the dimension reduction results of the data set. Although processing data in the ciphertext domain guarantees the privacy and security of the data, the computational efficiency is reduced compared to the plaintext, but it is still feasible from the experimental data.

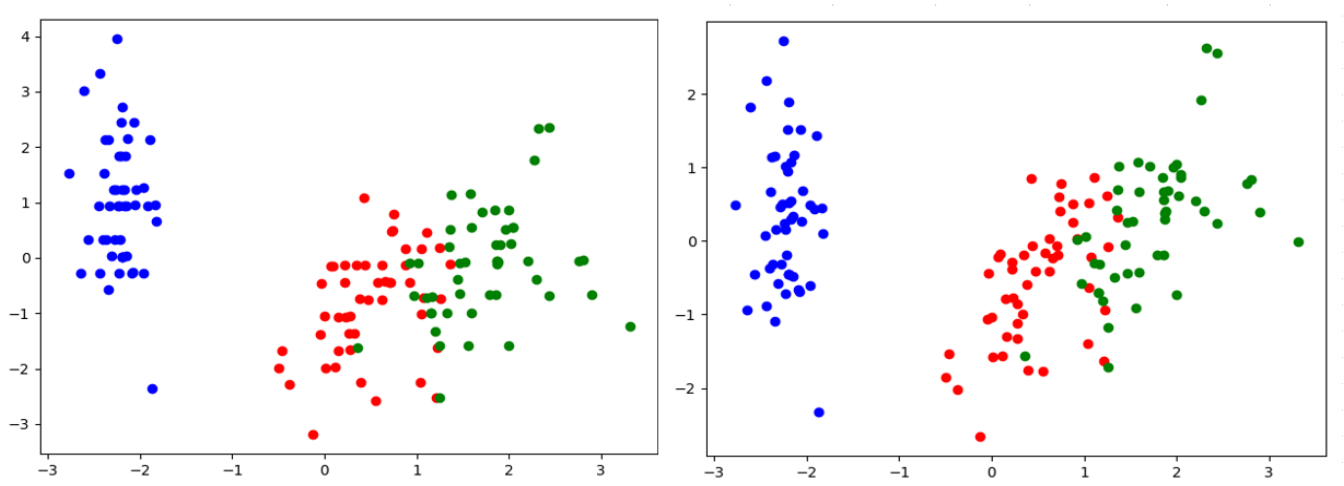

**FIGURE 13** PCA on ciphertext data (left) and plaintext data (right).

**TABLE 6** Results of Step3 in PCA.

| eigenvalues | feature vectors | time(s) |
|---|---|---|
| 0.7277 | 0.5227,-0.2624,0.5812,0.5658 | 3.6628 |
| 0.2302 | 0.3706,0.9263,0.0192,0.0636 | 3.4589 |

Based on the IRIS dataset and its related studies[44,45], and referring to the clear dimension reduction results, the scheme in cryptography is effective, which proves the reliability of the scheme.

## 6.4 | LDA algorithm test in Ciphertext

Similar to the previous experiment, in this part we test the LDA algorithm in the ciphertext domain. According to the previous experimental results, here we set the encryption parameters as poly-modulus-degree=16384, coeff-mod-bit-sizes=[50,30,30,30,30,30,30,30,30,30,50], scale=30, set the dimension reduction process reduces the original 4-dimensional features of the IRIS data set to 2-dimensional. We use the algorithm proposed in this article and the LDA function in the sklearn machine learning library to reduce the dimension of the ciphertext data set and the plaintext data set respectively. The dimension reduction results of the two algorithms are shown in Figure 14 (The left figure shows CKKS-LDA, and the right figure shows sklearn-LDA), and three colors represent three different computation of data.

The result of cryptographic dataset calculation using the algorithm presented in this paper is shown in Table 7.The run time is the calculation time of **Step3**.

**TABLE 7** Results of Step3 in LDA.

| eigenvalues | feature vectors | time(s) |
|---|---|---|
| 32.2719 | -0.1497,-0.1481,0.8511,0.4808 | 3.2711 |
| 0.2775 | 0.0071,0.3754,-0.4877,0.6848 | 3.1654 |

Among them, the number of iterations in the power method to find the eigenvalues and the ciphertext division is the same as before, 4 times and 2 times respectively. Excluding the difference between the algorithm in this paper and sklearn in processing the data set, the result of the LDA dimension reduction algorithm proposed in this paper is effective.

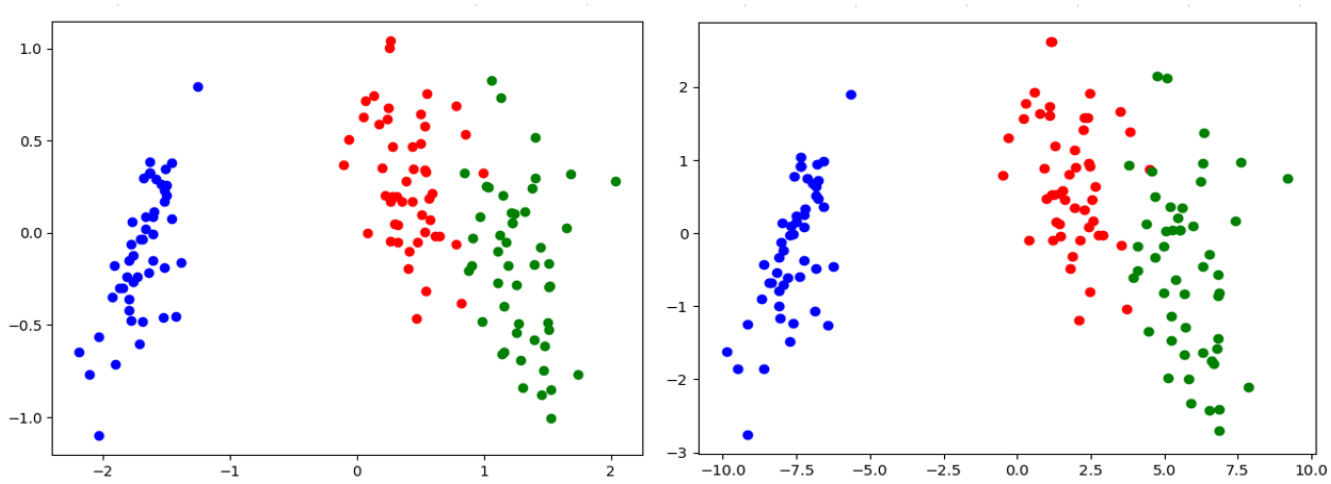


**FIGURE 14** LDA on ciphertext data (left) and plaintext data (right).

## 7 | CONCLUSION

In order to protect the privacy of the data in the sensor cloud system, and at the same time meet the demand for dimension reduction processing in many different dimensions, this paper proposes two ciphertext domains in the sensor cloud system based on the CKKS homomorphic encryption scheme and the PCA and LDA algorithms, and implement PCA algorithm and LDA algorithm on ciphertext data. Among them, due to the limitation of the ciphertext data on the operation rules and the ciphertext space on the multiplication depth, we use iterative operations in the system algorithm to replace some steps in the traditional algorithm, and appropriately increase the number of interactions between CSP and CSU. We implement the CKKS homomorphic encryption scheme by calling the TenSEAL homomorphic encryption library, and test the encryption efficiency with different parameters and the computational efficiency of basic operations in the ciphertext domain. And on this basis, we realize PCA and LDA in the ciphertext domain. algorithm. Through a large number of experiments, we finally screened out the best encryption parameters and iterative parameters, and realize the PCA and LDA dimension reduction algorithms of ciphertext data in the sensor cloud system efficiently and safely.

However, because CKKS ciphertext can not be bootstrapped at present, the calculation method that ciphertext can do is limited, and only a limited number of times of multiplication can be done. Therefore, this paper chooses some iterative algorithms to replace the traditional calculation method, which will result in large computation and communication costs. In the next step, we will test the bootstrapping process of CKKS homomorphic encryption to improve the dimension reduction algorithm of the two ciphertext domains designed in this article, and test our algorithm on more and larger data sets.

### ACKNOWLEDGMENTS

This work was supported by the National Cryptography Development Fund of China (grant no. MMJJ20170112), Natural Science Basic Research Plan in Shaanxi Province of China (grant no. 2018JM6028), National Natural Science Foundation of China (grant no. 61772550, U1636114, and 61572521), and National Key Research and Development Program of China (grant no. 2017YFB0802000). This work is also supported by Engineering University of PAP's Funding for Scientific Research Innovation Team (grant no. KYTD201805).

### CONFLICT OF 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.

### REFERENCES

1. Wang T, Mei Y, Jia W, Zheng X, Wang G, Xie M. Edge-based differential privacy computing for sensor–cloud systems. *Journal of Parallel and Distributed computing.* 2020;136:75–85.
2. Chaudhry SA, Yahya K, Al-Turjman F, Yang MH. A secure and reliable device access control scheme for IoT based sensor cloud systems. *IEEE Access.* 2020;8:139244–139254.

3. Dwivedi RK, Kumar R, Buyya R. Gaussian distribution-based machine learning scheme for anomaly detection in healthcare sensor cloud. *International Journal of Cloud Applications and Computing (IJCAC).* 2021;11(1):52–72.
4. Ali I. Data Collection in Studies on Internet of Things (IoT), Wireless Sensor Networks (WSNs), and Sensor Cloud (SC): Similarities and Differences. 2021.
5. Smith LI. A tutorial on principal components analysis. 2002.
6. Bryant FB, Yarnold PR. Principal-components analysis and exploratory and confirmatory factor analysis.. 1995.
7. Izenman AJ. Linear discriminant analysis. In: , , Springer, 2013:237–280.
8. Balakrishnama S, Ganapathiraju A. Linear discriminant analysis-a brief tutorial. *Institute for Signal and information Processing.* 1998;18(1998):1–8.
9. Cheon JH, Kim A, Kim M, Song Y. Homomorphic encryption for arithmetic of approximate numbers. In: Springer. 2017:409–437.
10. Rivest RL, Adleman L, Dertouzos ML, others . On data banks and privacy homomorphisms. *Foundations of secure computation.* 1978;4(11):169–180.
11. Rivest RL, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM.* 1978;21(2):120–126.
12. ElGamal T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory.* 1985;31(4):469–472.
13. Benaloh JDC. Verifiable secret-ballot elections.. 1989.
14. Paillier P. Public-key cryptosystems based on composite degree residuosity classes. In: Springer. 1999:223–238.
15. Goldwasser S, Micali S. Probabilistic encryption & how to play mental poker keeping secret all partial information. In: , , , 2019:173–201.
16. Gentry C, others . *A fully homomorphic encryption scheme*. 20. Stanford university Stanford, 2009.
17. Van Dijk M, Gentry C, Halevi S, Vaikuntanathan V. Fully homomorphic encryption over the integers. In: Springer. 2010:24–43.
18. Gentry C, Sahai A, Waters B. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Springer. 2013:75–92.
19. Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (standard) LWE. *SIAM Journal on Computing.* 2014;43(2):831–871.
20. Brakerski Z. Fully homomorphic encryption without modulus switching from classical GapSVP. In: Springer. 2012:868–886.
21. Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT).* 2014;6(3):1–36.
22. Naehrig M, Lauter K, Vaikuntanathan V. Can homomorphic encryption be practical?. In: 2011:113–124.
23. Wu D, Haven J. Using homomorphic encryption for large scale statistical analysis. *FHE-SI-Report, Univ. Stanford, Stanford, CA, USA, Tech. Rep. TR-dwu4.* 2012.
24. Wu S, Teruya T, Kawamoto J, Sakuma J, Kikuchi H. Privacy-preservation for stochastic gradient descent application to secure logistic regression. In: . 27. 2013:1–4.
25. Kim A, Song Y, Kim M, Lee K, Cheon JH. Logistic regression model training based on the approximate homomorphic encryption. *BMC medical genomics.* 2018;11(4):23–31.
26. Kim M, Song Y, Wang S, Xia Y, Jiang X. Secure logistic regression based on homomorphic encryption: Design and evaluation. *JMIR medical informatics.* 2018;6(2):e19.
27. Han K, Hong S, Cheon JH, Park D. Efficient Logistic Regression on Large Encrypted Data.. *IACR Cryptol. ePrint Arch..* 2018;2018:662.
28. Li P, Li J, Huang Z, Gao CZ, Chen WB, Chen K. Privacy-preserving outsourced classification in cloud computing. *Cluster Computing.* 2018;21(1):277–286.
29. Juvekar C, Vaikuntanathan V, Chandrakasan A. {GAZELLE}: A low latency framework for secure neural network inference. In: 2018:1651–1669.
30. Li M, Yan Y, Wang Q, Du M, Qin Z, Wang C. Secure Prediction of Neural Network in the Cloud. *IEEE Network.* 2020.
31. Boemer F, Costache A, Cammarota R, Wierzynski C. ngraph-he2: A high-throughput framework for neural network inference on encrypted data. In: 2019:45–56.
32. Morampudi MK, Prasad MV, Verma M, Raju U. Secure and verifiable iris authentication system using fully homomorphic encryption. *Computers & Electrical Engineering.* 2021;89:106924.
33. Sirichotedumrong W, Maekawa T, Kinoshita Y, Kiya H. Privacy-preserving deep neural networks with pixel-based image encryption considering data augmentation in the encrypted domain. In: IEEE. 2019:674–678.
34. Xu R, Joshi JB, Li C. Cryptonn: Training neural networks over encrypted data. In: IEEE. 2019:1199–1209.
35. Jain N, Nandakumar K, Ratha N, Pankanti S, Kumar U. Efficient CNN Building Blocks for Encrypted Data. *arXiv preprint arXiv:2102.00319.* 2021.
36. Kumari S, Karuppiah M, Das AK, Li X, Wu F, Kumar N. A secure authentication scheme based on elliptic curve cryptography for IoT and cloud servers. *The Journal of Supercomputing.* 2018;74(12):6428–6453.
37. He D, Kumar N, Khan MK, Wang L, Shen J. Efficient privacy-aware authentication scheme for mobile cloud computing services. *IEEE Systems Journal.* 2016;12(2):1621–1631.
38. Wazid M, Das AK, Kumar N, Vasilakos AV. Design of secure key management and user authentication scheme for fog computing services. *Future Generation Computer Systems.* 2019;91:475–492.
39. Gope P, Das AK, Kumar N, Cheng Y. Lightweight and physically secure anonymous mutual authentication protocol for real-time data access in industrial wireless sensor networks. *IEEE transactions on industrial informatics.* 2019;15(9):4957–4968.
40. Roy S, Chatterjee S, Das AK, Chattopadhyay S, Kumar N, Vasilakos AV. On the design of provably secure lightweight remote user authentication scheme for mobile cloud computing services. *IEEE Access.* 2017;5:25808–25825.
41. Benaissa A, Retiat B, Cebere B, Belfedhal AE. TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption. *arXiv preprint arXiv:2104.03152.* 2021.
42. Zhifa Y, shide S. *Multivariate statistical analysis*. Science Press of China, 2009.
43. Cheon JH, Kim D, Kim D, Lee HH, Lee K. Numerical method for comparison on homomorphically encrypted numbers. In: Springer. 2019:415–445.
44. Oliveira S. Data transformation for privacy-preserving data mining.. 2005.
45. Fisher R. Iris Data Set. http://archive.ics.uci.edu/ml/datasets/Iris; .