Fast, Lagre Scale Dimensionality Reduction Schemes Based on CKKS

Haonan Yuan*
yuanhaonan@cigit.ac.cn
Chongqing Key Laboratory of Secure Computing for Biology, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences

Wenyuan Wu
wuwenyuan@cigit.ac.cn
Chongqing Key Laboratory of Secure Computing for Biology, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences

Jingwei Chen
chenjingwei@cigit.ac.cn
Chongqing Key Laboratory of Secure Computing for Biology, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences

ABSTRACT

The proliferation of artificial intelligence and big data has resulted in a surge in data demand and increased data dimensionality. This escalation has consequently heightened the costs associated with storage and processing. Concurrently, the confidential nature of data collected by various institutions, which cannot be disclosed due to personal privacy concerns, has exacerbated the challenges associated with data analysis and machine learning model training. Therefore, designing a secure and efficient high-dimensional data reduction method that supports multi-party joint participation becomes critical to solving these problems.

This paper proposes a novel homomorphic encryption dimensionality reduction scheme (HE-DR) based on CKKS[9], which modifies the Rank-Revealing (RR) method to make it more applicable to fully homomorphic encryption, thereby achieving fast and secure dimension reduction for high-dimensional data. Compared to traditional homomorphic encryption dimensionality reduction schemes, our approach does not transmit the user's original data to other participants in any format (Ciphertext or Plaintext). Moreover, our method's computational efficiency is nearly $60 - 200$ times faster than similar algorithms, and the communication overhead is only $1/3$ of theirs. Finally, we have shown that our proposed scheme can preserve its computational efficiency and accuracy even when dealing with high-dimensional data. As dimensionality escalates, the ratio of ciphertext to plaintext computational efficiency plateaus at approximately 5 times, while the computational error (distance between subspaces) remains around $1e^{-11}$

KEYWORDS

high-dimensional, efficient, dimension reduction, homomorphic encryption, CKKS scheme

1 INTRODUCTION

Big data plays a pivotal role in contemporary social and economic sectors, with applications spanning numerous domains, especially biological genetics for disease prediction, clinical decision support, and genomic research. However, genetic data's high sensitivity prohibits its owners' public publication [18]. Certain biological institutions, laboratories, or hospitals possess only a limited amount of this data. Furthermore, the inherent high dimensionality of gene data, even when subjected to rudimentary filtering techniques, results in elevated residual dimensions and exhibits significant correlation. Concurrently, the non-public nature of this data limits the number of samples per owner, rendering data dimension reduction challenging. Consequently, our primary research objective is to explore how multi-party data owners can collaboratively achieve precise data dimension reduction safely and efficiently.

Historically, rank-revealing techniques have garnered significant attention and research across diverse application domains. These methods endeavor to distill pivotal information from high-dimensional datasets while preserving the majority of their inherent structure and patterns. By adeptly unveiling the data's low-rank architecture, rank-revealing approaches furnish robust tools for data analysis, compression, and recovery tasks. Unlike conventional matrix factorization techniques that often demand substantial computational resources and time, rank-revealing methods offer efficient solutions. For instance, a pioneering low-rank subspace learning algorithm is introduced in [14]. Also, an algorithm for computing symmetric rank revealing factorizations of symmetric $n \times n$ matrices M is delineated in [26]. Such applications underscore the broad applicability of rank-revealing methods and attest to their efficacy in addressing intricate problems.

As previously noted, the exponential surge in data volume necessitates urgent attention to data privacy and security assurance. Differential Privacy(DP), Federated Learning(FL), Secure Multi-Party Computation(SMPC), and Fully Homomorphic Encryption(FHE) are the most prevalent privacy protection technologies, extensively utilized across diverse sectors such as recommendation systems, cloud computing, and Internet of Things (IoT).

The concept of DP is first proposed in [11], which ensures the privacy of individual data points by adding noise while still providing useful statistical analysis results. For tasks like dimensionality reduction, however, the addition of noise can significantly affect the accuracy of the results [30]. Moreover, the calculation process usually involves multiple iterative computations, each involving updates and recalculations of matrices. Under the differential privacy mechanism, noise needs to be added in each iteration, leading to a cumulative effect of the noise. As the number of iterations increases, the impact of the noise becomes increasingly significant, resulting in biases in the computational results and affecting the correctness and effectiveness of the decomposition outcomes. In recent years, federated learning has been proposed in [27]. As a distributed machine learning framework, it is applicable in scenarios where data is stored on multiple devices, and each device needs to preserve the privacy of the data. However, high-dimensional matrix decomposition requires frequent exchanges of model parameters, leading to a surge in communication overhead. Also, handling sparse data further increases communication costs and computational complexity [19, 21]. SMPC [10] protocol involves many encryption and

decryption operations, especially floating-point operations in high-dimensional matrix factorization, resulting in huge computational overhead. In addition, high-dimensional data processing requires frequent data exchanges and complex protocol coordination, leading to high communication overhead [15]. FHE [13] offers superior privacy protection than other privacy protection technologies. This is because data remains encrypted throughout the entire computation process. Data remains encrypted to protect privacy during storage, transmission, or processing. Furthermore, the precision of computational outcomes is not contingent upon adding noise to the data or the distribution of computations, thereby entirely mitigating the risk of data exposure. Unlike differential privacy, fully homomorphic encryption obviates the need for noise-induced privacy protection. This implies that computational results can be rendered completely accurate and immune to distortion caused by noise.

In recent decades, numerous solutions utilizing Federated Learning, Secure Multi-Party Computation, and Fully Homomorphic Encryption have been proposed to reduce data dimensionality. This process is a crucial component in every application scenario and has consequently become a significant area of research in privacy preservation. Initial work by [17] introduced a privacy-preserving Singular Value Decomposition (SVD) method; however, its applicability to real data is limited. Following this, a privacy-preserving matrix factorization method [28] is applied to recommendation systems, but its single iteration time is approximately 3 hours, making it still difficult to apply in practical scenarios. Recent studies by [8] propose a homomorphic encryption-based SVD method and apply it to recommendation systems, and their work is further explored in subsequent work by [24], thereby mitigating the collusion-induced data leakage problem to some extent. Federated Singular Vector Decomposition (FedSVD) [7] can execute rapid federated singular value decomposition of high-dimensional data. Concurrently, a more efficient and secure FedSVD was proposed in [23], contrasting with previous works by [16, 32]. Despite significant advancements in privacy-preserving techniques for matrix operations, numerous challenges persist. These include eliminating the involvement of entirely trusted servers (or trusted third parties) within the system, preventing collusion among multiple entities, and addressing issues related to low computational efficiency and excessive communication overhead.

We use fully homomorphic encryption to avoid trusted third parties and guarantee accuracy and dimensionality reduction. Several fully homomorphic encryption schemes exist, such as BGV [5], BFV [12], and CKKS. BGV and BFV are based on LWE [33] and RLWE [25] problems, respectively. BGV is more complex in implementation but has better performance than BFV. CKKS is more efficient in real number operations and has slower noise growth, which is suitable for tasks requiring many iterative computations. In addition, after multiple homomorphic operations, the expansion of the ciphertext of CKKS is relatively small, which is beneficial to maintaining computational efficiency and resource usage.

In the past, dimensionality reduction schemes utilizing fully homomorphic encryption technology have been user-centric. In these models, users encrypt and transmit their data to the computation party. The computation party returns the results, which the user decrypts to ascertain the outcome. However, in practical scenarios, the high data sparsity makes encrypting the entire dataset computationally intensive and resource-consuming. Additionally, whether it involves data decomposition or dimensionality reduction, the computational process necessitates iteration. Adhering to traditional schemes, we must select a larger polynomial modulus degree (N) to guarantee computational depth. Moreover, in real-world applications, multiple communications with the computation party are still required to re-encrypt certain parameters and eliminate noise within them [29].

Our study presents a rapid dimensionality reduction scheme utilizing CKKS fully homomorphic encryption(HE-DR) for two data owners. It aims to facilitate rapid dimensionality reduction of high-dimensional data without compromising the confidentiality of the participants' original data. Our primary contributions and strengths lie in these areas:

- The system is entirely independent of the involvement of a trusted third party. Neither the initialization nor the computation process requires the assistance of a trusted third party, thereby ensuring that only two parties are involved in the entire system.
- Our scheme guarantees that user data is not transmitted to other participants in any form, thereby eliminating the necessity for encrypting the original high-dimensional data. This approach prevents computation between high-dimensional ciphertext data. Furthermore, it eliminates the need for users to transmit large volumes of encrypted data, thereby reducing communication overhead.
- We have proposed a new termination method for ciphertext algorithms, which effectively enhances the precision of data reduction under ciphertext. The scheme avoids the additional communication overhead of sending the data back to the owner for decryption and then deciding whether it meets the requirement and the computational inaccuracy of another method by which the desired number of dimension reduction is agreed upon beforehand.

This paper is structured as follows: Section 2 introduces the notation and the plaintext algorithm. Section 3 delineates our ciphertext computation protocol, with a comprehensive security analysis provided in Section 4. The experimental results on computational efficiency and communication overhead are presented in Section 5, while Section 6 concludes this study.

2 NOTATION AND PRELIMINARIES

2.1 Notations

For all computation protocols, we denote matrices by capital letters, e.g., X, Y, U, and column vectors by lowercase boldface letters, e.g., $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{t}$. The notation $(\cdot)^T$ denotes the transpose of a matrix or vector. Since this scheme only adopts CKKS fully homomorphic encryption scheme, pk and sk represent the public and private keys in the CKKS encryption scheme, and we denote the ciphertext by $c_{(\cdot)}$, where the lower subscript denotes the corresponding plaintext variable for encryption. For example, $c_{\boldsymbol{u}}$ represents the ciphertext of vector $\boldsymbol{u}$.

For a concrete value m, $i \leftarrow m$ means that the value of m is assigned to i. And for an arbitrary distribution $\mathcal{P}$, $\boldsymbol{u} \leftarrow \mathcal{P}$ means that $\boldsymbol{u}$ is uniformly sampled from $\mathcal{P}$.

2.2 Singular Value Decomposition

SVD is commonly used in dimensionality reduction to map high-dimensional data into a low-dimensional space for ease of subsequent processing or visualization. However, there are some difficulties and challenges in practical applications:

- **High computational complexity**: The computational complexity of Singular Value Decomposition (SVD) is relatively high, primarily due to the extensive matrix operations involved and the requirements for numerical stability. This can be particularly resource-intensive when calculating high-dimensional, sparse, low-rank matrices. Furthermore, the SVD algorithm requires multiple iterations and error corrections during computation to ensure the accuracy and stability of the results, thereby increasing computational complexity.
- **Large storage space requirement**: When the matrix scale is large (i.e., m and n are large), computing SVD becomes time-consuming and requires a lot of memory resources to store intermediate matrices and computational results. This poses additional challenges for large-scale data processing.
- **Inappropriate data matrix**: In practical applications, the data matrix is typically sparse, as observed in recommendation systems and biological gene data matrices. Nonetheless, Singular Value Decomposition (SVD) is more adept at handling dense matrices.

Therefore, it is unsuitable for reducing the dimension of a high-dimensional sparse matrix. Another method, rank-revealing, can compensate for SVD's shortcomings in this area.

2.3 Rank-revealing

This section briefly introduces the rank-revealing solution proposed in [20]. Given a $m \times n$ matrix M, where $m \geq n$ and $rank(M) = k$, for SVD decomposition, we know that

$$M = \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T + \sigma_2 \boldsymbol{u}_2 \boldsymbol{v}_2^T + \cdots + \sigma_n \boldsymbol{u}_n \boldsymbol{v}_n^T \tag{1}$$

also, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. According to the rank-revealing method, if there is a threshold $\theta > 0$, the numerical rank of M can be determined, as well as its numerical range, without calculating the rank-revealing decomposition. Assuming $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > \theta > \sigma_{k+1} \geq \cdots \geq \sigma_n \geq 0$ are nonzero singular values of M, $\boldsymbol{u}_i$ is the unit left singular vector and $\boldsymbol{v}_i$ is the unit right singular vector associated with singular value σ_i. For $i = 1, \cdots, k$ and $k \ll n$, we have

$$M_i = \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T + \sigma_2 \boldsymbol{u}_2 \boldsymbol{v}_2^T + \cdots + \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T \tag{2}$$

Since $\boldsymbol{u}_i^T M = \sigma_i \boldsymbol{v}_i^T$, we rewrite

$$M_i = \boldsymbol{u}_1 \boldsymbol{u}_1^T M + \boldsymbol{u}_2 \boldsymbol{u}_2^T M + \cdots + \boldsymbol{u}_i \boldsymbol{u}_i^T M \tag{3}$$

First of all, we set $M^{(1)} = M$ and σ_1 is the largest singular value of $M^{(1)}$, then for $M^{(2)} = M - \boldsymbol{u}_1 \boldsymbol{u}_1^T M$, the largest singular value σ_1 of $M^{(1)}$ is replaced by zero. The second largest singular value σ_2 of $M^{(1)}$ becomes $M^{(2)}$ largest singular value, and the rank of $M^{(2)}$ is less than $M^{(1)}$ by one. In the same way, through this method, the rank of $M^{(3)} = M - \boldsymbol{u}_1 \boldsymbol{u}_1^T M - \boldsymbol{u}_2 \boldsymbol{u}_2^T M$ is equal to $k - 2$. The third largest singular value σ_3 of $M^{(1)}$ becomes $M^{(3)}$ largest singular value. Thus, we can determine the number of singular values greater than the threshold and the matrix's numerical rank.

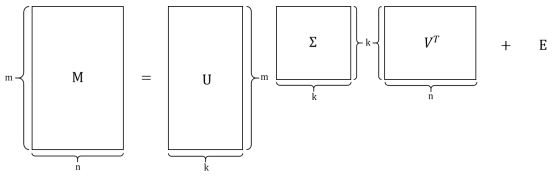

Figure 1: Approximate Singular Value Decomposition

Pseudocode 1: Larank

Input : Matrix $M \in \mathbb{R}^{m \times n}$, numerical rank threshold $\theta > 0$
Output: $k = rank_\theta(M)$

1 Initialize $\epsilon_m = \|M\|_\infty \epsilon_{machine}$ along with empty matrices U and V
2 **for** $k = 0, 1, \cdots, n$ **do**
3 generate a random unit vector $\boldsymbol{y}_0$, set $\zeta_0 = 0$
4 **for** $j = 1, 2, \cdots$ **do**
5 set $\boldsymbol{u} = M^T[\boldsymbol{y}_{j-1} - U(U^T \boldsymbol{y}_{j-1})]$, $\boldsymbol{x}_j = \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|_2}$
6 set $\boldsymbol{p} = M\boldsymbol{x}_j$, $\boldsymbol{v} = \boldsymbol{p} - U(U^T \boldsymbol{p})$
7 calculate $\zeta_j = \|\boldsymbol{v}\|_2$, $\boldsymbol{y}_j = \frac{\boldsymbol{v}}{\zeta_j}$
8 **if** $(\frac{\theta}{\zeta_j})^{2j} < \epsilon_m$ *or* $\frac{|\zeta_j - \zeta_{j-1}|^2}{|\zeta_{j-1} - \zeta_{j-2}| - |\zeta_j - \zeta_{j-1}|} < \theta$ **then**
9 break the j-loop
10 **end**
11 **end**
12 **if** $\zeta_j < \theta$ **then**
13 break the k-loop
14 **end**
15 update $U = [U, \boldsymbol{y}_j]$
16 **end**

As shown in Fig. 1, $U = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_k]$ is left orthogonal matrices, $V = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_k]$ is right orthogonal matrices. It is worth noting that the values of the matrix Σ on the diagonal $diag\{\sigma_1, \cdots, \sigma_k\}$ is still satisfying $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > \theta > \sigma_{k+1} \geq \cdots \geq \sigma_n \geq 0$. Thus, for a given threshold θ of the user, we rewrite Eq. 1

$$M = M_k + E \tag{4}$$

where $E = \sigma_{k+1} \boldsymbol{u}_{k+1} \boldsymbol{v}_{k+1}^T + \cdots + \sigma_n \boldsymbol{u}_n \boldsymbol{v}_n^T$. We shall call M_k the dominant part and E the noise part of M within threshold θ.

To further elucidate the method, we present the original pseudocode from [20] as shown in Pseudocode. 1. The input is an $m \times n$ matrix and a threshold θ, and the output of the **larank** algorithm is the numerical rank k of matrix M. In addition, step 8 determines whether the feature vector converges, and step 12 judges whether the current result has reached the set threshold. Finally, the algorithm allows for the acquisition of a set of standard orthonormal basis $\{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_k\}$ for matrix M. This orthonormal basis can be utilized solely to achieve dimension reduction of the original data.

In Pseudocode. 1, to improve the computational efficiency, it does not apply power iteration of $(M - M_i)(M - M_i)$ when calculating the remaining $k - 1$ singular vectors. According to the implicit

iteration method mentioned in [20],

$$\begin{aligned}(M-M_i)^T\boldsymbol{y} &= M^T(\boldsymbol{y}-\boldsymbol{y}_1\boldsymbol{y}_1^T\boldsymbol{y}-\cdots-\boldsymbol{y}_i\boldsymbol{y}_i^T\boldsymbol{y})\\ (M-M_i)\boldsymbol{x} &= M\boldsymbol{x}-\boldsymbol{y}_1\boldsymbol{y}_1^T(M\boldsymbol{x})-\cdots-\boldsymbol{y}_i\boldsymbol{y}_i^T(M\boldsymbol{x})\end{aligned} \tag{5}$$

where $\boldsymbol{y}$ is unit vector and $\boldsymbol{x}$ is $(M-M_i)^T\boldsymbol{y}$. Therefore, computing and storing M_i is unnecessary.

2.4 Cheon-Kim-Kim-Song Cryptosystem

Cheon-Kim-Kim-Song (CKKS) is a scheme for fully homomorphic encryption, which means that addition and multiplication over ciphertexts can be performed without decryption. Moreover, it belongs to the class of lattice-based fully homomorphic encryption schemes whose security is established on hard mathematical problems such as the Learning With Errors (LWE) problem and the Ring Learning With Errors (RLWE) problem. Finally, since matrix operations are essentially carried out with floating point numbers, CKKS supports the addition and multiplication of floating point numbers in a homomorphic encryption environment, and the property that multiple rounds of computation can be performed without losing too much precision can be well adapted to our scheme.

2.5 Relative work of Security proofs

According to the description of secure protocols under semi-honest behavior in Definition 7.2.1 in [15], let $f(x,y)$ be a computational formula for both parties, where x, y are the inputs of both parties respectively, and Π is the protocol that both parties compute f. Also, let $view^{\Pi}(x,y)$ denote the information that both parties receive while re-executing the Π protocol, and it does not contain the results calculated by itself. Finally, the output of both parties is denoted as $output^{\Pi}(x,y)$.

Definition 2.1. (privacy with respect to semi-honest behavior[15]) If there exist probabilistic polynomial-time algorithms S_a and S_b for any input X, Y of functionality f, we can say the two-part protocol Π privately computes f.

$$\{S_a(1^\lambda, X, f_a(X,Y), f(X,Y))\} \overset{c}{\equiv} \{view_a^{\Pi}(\lambda,X,Y), output^{\Pi}(\lambda,X,Y)\} \tag{6}$$

$$\{S_b(1^\lambda, Y, f_b(X,Y), f(X,Y))\} \overset{c}{\equiv} \{view_b^{\Pi}(\lambda,X,Y), output^{\Pi}(\lambda,X,Y)\} \tag{7}$$

where $\overset{c}{\equiv}$ denotes computational indistinguishability against probabilistic polynomial time adversaries with negligible advantage in the security parameter λ.

Definition. 2.1 is satisfied for the general case, X, Y, and $f_a(X,Y)$, $f_b(X,Y)$ represent the inputs and outputs of A and B, respectively. At the same time, in our scheme, we only consider the deterministic equation f, so we refer to the deterministic case in [15] to simplify the expression. When f is a deterministic functionality, we can say that protocol Π is secure computation, and it only needs to satisfy the following relationship (omit the security parameter Eq. 8:

$$\begin{aligned}\{S_a(X, f_a(X,Y))\} &\overset{c}{\equiv} \{view_a^{\Pi}(\lambda,X,Y)\}\\ \{S_b(Y, f_b(X,Y))\} &\overset{c}{\equiv} \{view_b^{\Pi}(\lambda,X,Y)\}\end{aligned} \tag{8}$$

Furthermore, we introduce the Lindeberg–Lévy central limit theorem to justify the consistency of the simulator and the subsequently constructed actual observation distribution.

Theorem 2.2. *(Lindeberg–Lévy CLT) Let the random variable sequence X_k be independent and identically distributed, and $E(X_x) = \mu, D(X_k) = \sigma^2 > 0, k = 1, 2, \cdots$. Donate $Y_n = \frac{\sum_{k=1}^{n} X_k - n\mu}{\sqrt{n}\sigma}$, then for any $x \in \mathbb{R}$,*

$$\lim_{n\to\infty} F_n(x) = \lim_{n\to\infty} P(Y_n \le x) = \Phi(x) \tag{9}$$

It is also known as the central limit theorem for i.i.d., and the application of this theorem is that when n is sufficiently large, we approximately have:

$$\sum_{k=1}^{n} X_k \overset{\cdot}{\sim} N(n\mu, n\sigma^2) \tag{10}$$

where $\overset{\cdot}{\sim}$ represents the approximation of distribution.

3 SCENARIO AND PROTOCOL

3.1 Scenario

As shown in Fig. 2, there are only two parties, A and B, in our scenario, and they have $m \times n$ dimensional data matrices, denoted as X and Y, respectively. Two data proprietors aim to enhance the precision of dimension reduction by amalgamating the data samples from each other. In practical life, data usually have horizontal and vertical combinations, as shown in Fig. 3. We use $M = [X; Y]$ to represent the vertical combination of data, and $M = [X, Y]$ to represent the horizontal combination of data. Our scheme mainly demonstrates the results of the vertical combination form, as shown in Fig. 4. Secondly, user A generates the required public-private key pair pk, sk based on the CKKS scheme and uses the public key to encrypt the vector that needs to be transmitted to User B. Meanwhile, to ensure data security, User B will add a mask to the vector transmitted to User A. During the calculation of each orthogonal basis, to guarantee the safety of both parties' data, they

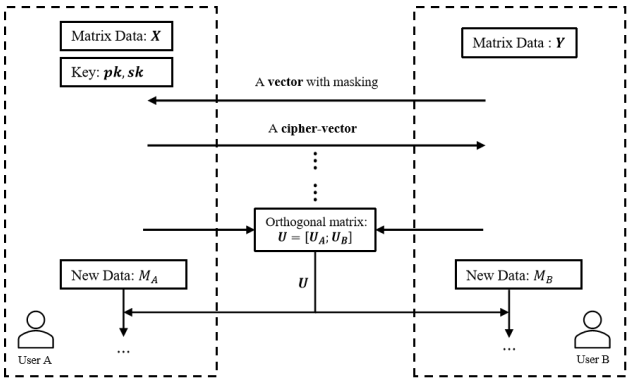

Figure 2: Two-party Framework

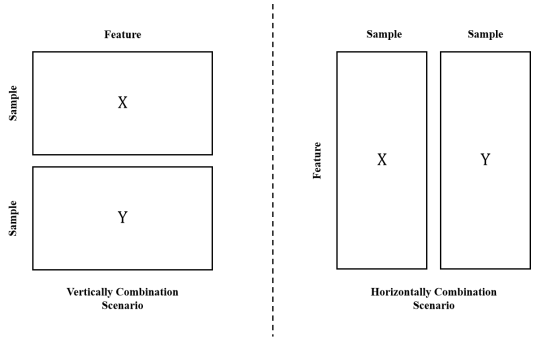

Figure 3: Different Data Partitioning

only transmit some intermediate results and only share the final converged orthogonal basis. Finally, after several iterations, both parties obtain their corresponding left orthogonal matrices $U_a = [\boldsymbol{u}_{a1}, \cdots, \boldsymbol{u}_{ak}]$ and $U_b = [\boldsymbol{u}_{b1}, \cdots, \boldsymbol{u}_{bk}]$ respectively. By vertically combining them, they get the final orthogonal matrix $U = [U_a; U_b]$, which is shared for subsequent dimension reduction of new data from both sides. Moreover, the entire system operation process does not require any trusted third party or computing entity's participation.

It is important to highlight that the security proof for our comprehensive system is predicated on the semi-honest model. As delineated in [6, 15], our task is merely to demonstrate that the outcomes observed by both parties within this model align with the actual results derived from input and output. This alignment serves as evidence of the system's secure privacy computation. The comprehensive proof detailing each protocol will be presented in Section 4.

Protocol 1: Principal orthogonal basis

Input A : X, $\boldsymbol{u}_a$, pk, sk
Input B : Y, $\boldsymbol{u}_b$
Output A : $\bar{\boldsymbol{u}}_a$
Output B : $\bar{\boldsymbol{u}}_b$

1 **for** $i \leftarrow 1$ **to** m **do** `// m is max number of iterations`
2 B: send $Y^T \boldsymbol{u}_b$ to A
3 A: $\boldsymbol{t} = \frac{X^T \boldsymbol{u}_a + Y^T \boldsymbol{u}_b}{\|X^T \boldsymbol{u}_a + Y^T \boldsymbol{u}_b\|}$, $\bar{\boldsymbol{u}}_a = X \cdot \boldsymbol{t}$
4 A: $c_{\boldsymbol{t}}$, $c_{XY^T \boldsymbol{u}_b} = Enc(\boldsymbol{t}, XY^T \boldsymbol{u}_b;\ pk)$
5 A: send $c_{\boldsymbol{t}}$, $c_{XY^T \boldsymbol{u}_b}$ to B
6 B: $c_{\bar{\boldsymbol{u}}_{be}} = Y \cdot c_{\boldsymbol{t}} + \boldsymbol{u}_r$ `// u_r ← Q`
7 B: send $c_{\bar{\boldsymbol{u}}_{be}}$ to A
8 A: $\bar{\boldsymbol{u}}_{be} = Dec(c_{\bar{\boldsymbol{u}}_{be}};\ sk)$
9 A: send $\|\bar{\boldsymbol{u}}_a\|^2$, $\bar{\boldsymbol{u}}_{be}$ to B
10 B: $\bar{\boldsymbol{u}}_b = \bar{\boldsymbol{u}}_{be} - \boldsymbol{u}_r$
11 B: send $\|\bar{\boldsymbol{u}}_b\|^2$ to A
12 B: $\boldsymbol{u}_b = \frac{\bar{\boldsymbol{u}}_b}{\sqrt{\|\bar{\boldsymbol{u}}_a\|^2 + \|\bar{\boldsymbol{u}}_b\|^2}}$
13 A: $\boldsymbol{u}_a = \frac{\bar{\boldsymbol{u}}_a}{\sqrt{\|\bar{\boldsymbol{u}}_a\|^2 + \|\bar{\boldsymbol{u}}_b\|^2}}$
14 A: **if** $\boldsymbol{u}_a$ *are converge* **then**
15 A: $\bar{\boldsymbol{u}}_a = \boldsymbol{u}_a$
16 B: **if** $\boldsymbol{u}_b$ *are converge* **then**
17 B: $\bar{\boldsymbol{u}}_b = \boldsymbol{u}_b$
18 break the i-loop
19 **end**
20 **end**
21 **end**

3.2 Protocol

As illustrated in the preceding plaintext Pseudocode. 1, the plaintext scheme is designed for a single-user and computing party model unsuitable for multiparty joint computation. Moreover, as shown in steps 5 to 12 of the Pseudocode. 1, this scheme requires many normalization and modulo operations to ensure computational accuracy, and it also needs to determine whether vectors converge and compare with a threshold to control the termination of the algorithm. However, the computational complexity of implementing these operations using fully homomorphic encryption is substantial, leading to suboptimal efficiency in practical applications. Therefore, we redesign a judgment method by increasing the number of communications, ensuring security and accuracy while maintaining high computational efficiency. We also segment the algorithm into two parts to enhance computational efficiency. The initial part only computes the first orthogonal basis using the conventional power iteration method, and the second part modifies the power iteration method following Eq. 5 and determines the remaining orthogonal bases.

In Protocol. 1, each party's data is denoted by X and Y, respectively. Concurrently, the initial unit vectors are represented by $\boldsymbol{u}_a$ and $\boldsymbol{u}_b$. The output singular vectors post each iterative cycle are signified as $\bar{\boldsymbol{u}}_a$ and $\bar{\boldsymbol{u}}_b$. In this protocol, the primary challenge lies in safeguarding user B's data from potential breaches while maintaining computational accuracy. We introduce a random perturbation $\boldsymbol{u}_r$ to the ciphertext in step 6, ensuring that user A cannot access user B's original data and eliminating this perturbation from the calculation outcomes in step 10. This approach avoids data leakage and maintains computation precision. As for steps 14 and 16, we judge whether the results have converged by comparing the difference between the results of two successive iterations with the machine calculation precision or user-specified precision.

As mention before, to improve the efficiency of calculation, we rewrite the calculation Protocol. 1 according to Eq. 5, which is shown in Protocol. 2. X and Y still denote each party's data, and the initial random vector $\boldsymbol{z}_{ai}$, $\boldsymbol{z}_{bi}$ must be re-generated when computing each singular vector. It is worth noting that to guarantee the security of user B and the accuracy of the calculation results, a method similar to Protocol. 1 was used, where the perturbation added by user B in Protocol. 2 steps 5 and 12 makes it impossible for user A to obtain the data of user B. However, the perturbation at this time will contain ciphertext data that user B cannot decrypt during the calculation process. Therefore, to solve this problem, the perturbation added by user B in Protocol. 2 step 5 is a random number r multiplied by the initial vector $\boldsymbol{u}_b$ of Protocol. 1. Since $c_{XY^T \boldsymbol{u}_b}$ is shared, so User B can eliminate it in Protocol. 2 step 10.

Protocol 2: Sub-principal orthogonal basis

Input A : X, U_a, pk, sk
Input B : $Y, U_b, \boldsymbol{z}_a, \boldsymbol{z}_b, c_{XY^T\boldsymbol{u}_b}$
Output B : $\bar{\boldsymbol{u}}_a, \bar{\boldsymbol{u}}_b$

for $i \leftarrow 1$ **to** m **do** // m is max number of iterations
  B: $\boldsymbol{\Delta_{ai}} = \boldsymbol{z}_a - U_a U_a^T \boldsymbol{z}_a - U_a U_b^T \boldsymbol{z}_b$
  B: $\boldsymbol{\Delta_{bi}} = \boldsymbol{z}_b - U_b U_a^T \boldsymbol{z}_a - U_b U_b^T \boldsymbol{z}_b$
  B: $\boldsymbol{t} = Y^T(\boldsymbol{\Delta_{bj}} + r \cdot \boldsymbol{u}_b)$ // $r \leftarrow Q$
  B: send $\boldsymbol{\Delta_{aj}}$ and $\boldsymbol{t}$ to A
  A: $\boldsymbol{z_{ae}} = XX^T\boldsymbol{\Delta_{aj}} + X \cdot \boldsymbol{t}$
  A: $c_{\boldsymbol{z_{ae}}}, c_{X^T\boldsymbol{\Delta_{aj}}} = Enc(\boldsymbol{z}_{ae}, X^T\boldsymbol{\Delta_{aj}};\ pk)$
  A: send $c_{\boldsymbol{z_{ae}}}, c_{X^T\boldsymbol{\Delta_{aj}}}$ to B
  B: $c_{\boldsymbol{z_a}} = c_{\boldsymbol{z}_{ae}} - r \cdot c_{XY^T\boldsymbol{u}_b}$
  B: $c_{\boldsymbol{z_b}} = Y \cdot c_{X^T\boldsymbol{\Delta}_{aj}} + YY^T\boldsymbol{\Delta_{bj}}$
  B: $(c_{\boldsymbol{z_{aee}}}, c_{\boldsymbol{z_{bee}}}) = (c_{\boldsymbol{z_a}}, c_{\boldsymbol{z}_b}) + \boldsymbol{u}_r$ // $\boldsymbol{u}_r \leftarrow Q$
  B: send $c_{z_{aee}}$ and $c_{z_{bee}}$ to A
  A: $z_{aee}, z_{bee} = Dec(c_{z_{aee}}, c_{z_{bee}};\ sk)$
  A: send $\boldsymbol{z}_{aee}, \boldsymbol{z}_{bee}$ to B
  B: $\boldsymbol{z}_a = \frac{\boldsymbol{z}_{aee} - \boldsymbol{u}_r}{\|\boldsymbol{z}_{aee} - \boldsymbol{u}_r\|}, \boldsymbol{z}_b = \frac{\boldsymbol{z}_{bee} - \boldsymbol{u}_r}{\|\boldsymbol{z}_{bee} - \boldsymbol{u}_r\|}$
  B: **if** $\boldsymbol{z}_a$ *and* $\boldsymbol{z}_b$ *are converge* **then**
    break the j-loop
  end
end
B: $\bar{\boldsymbol{u}}_a = \frac{\Delta_{ai}}{\|\Delta_{ai}\|},\ \bar{\boldsymbol{u}}_b = \frac{\Delta_{bi}}{\|\Delta_{bi}\|}$
B: share $\bar{\boldsymbol{u}}_a$ and $\bar{\boldsymbol{u}}_b$ to A

Its correctness is shown in Eq. 11.

$$\begin{aligned} c_{\boldsymbol{z}_{ai}} &= c_{\boldsymbol{z}_{aie}} - r \cdot c_{XY^T\boldsymbol{u}_b} \\ &= c_{(XX^T\boldsymbol{\Delta_{aj}} + X\cdot\boldsymbol{t})} - r \cdot c_{XY^T\boldsymbol{u}_b} \\ &= c_{XX^T\boldsymbol{\Delta_{aj}}} + c_{XY^T(\boldsymbol{\Delta_{bj}} + r\cdot\boldsymbol{u}_b)} - r \cdot c_{XY^T\boldsymbol{u}_b} \\ &= c_{XX^T\boldsymbol{\Delta_{aj}}} + c_{XY^T\boldsymbol{\Delta_{bj}}} + c_{XY^T(r\cdot\boldsymbol{u}_b)} - r \cdot c_{XY^T\boldsymbol{u}_b} \\ &= c_{XX^T\boldsymbol{\Delta_{aj}}} + c_{XY^T\boldsymbol{\Delta_{bj}}} \end{aligned} \tag{11}$$

Also, The convergence conditions are the same as those in Protocol.1, and the initial vector $\boldsymbol{z}_{ai},\ \boldsymbol{z}_{bi}$ can only be generated by B in the execution of Protocol. 2.

Furthermore, due to the complexity of ciphertext comparison under fully homomorphic encryption and its significant computational and communication overhead, traditional solutions are generally divided into two types: 1. Completely decompose the data. 2. Negotiate in advance the number of features to be retained. Nevertheless, when the data's dimensionality is elevated, its complete decomposition's computational efficiency becomes significantly compromised. Moreover, negotiating in advance the number of features to retain can lead to inaccurate dimension reduction results. The accuracy can only be improved by guessing or continuously trying to retain different numbers of features, but the computational efficiency remains suboptimal. Therefore, we present this scheme's most critical protocol (Stop criteria). Firstly, two parties will negotiate the proportion θ of information to be eliminated, then calculate the square of maximum eigenvalue $c_{\sigma^2_{max}}$(the ciphertext of σ^2_{max}) according

Protocol 3: Stop criteria

Input A : X, pk, sk
Input B : $Y, \bar{\boldsymbol{u}}_a, \bar{\boldsymbol{u}}_b$
Output A : *Bool*

A: $\sigma_a = \bar{\boldsymbol{u}}_a^T XX^T\bar{\boldsymbol{u}}_a, \boldsymbol{t} = X^T\bar{\boldsymbol{u}}_a$
A: $c_{\sigma_a}, c_{\boldsymbol{t}} = Enc(\sigma_a, \boldsymbol{t};\ pk)$
A: send $c_{\sigma_a}, c_{\boldsymbol{t}}$ to B
B: $\sigma_b = \bar{\boldsymbol{u}}_b^T YY^T\bar{\boldsymbol{u}}_b$
B: $c_{\sigma^2} = c_{\sigma_a} + 2(\bar{\boldsymbol{u}}_b^T Y \cdot c_{\boldsymbol{t}}) + \sigma_b$
B: $c_{\sigma_{res}} = r \cdot (c_{\sigma^2} - \theta^2 \cdot c_{\sigma^2_{max}})$ // $r \leftarrow Q$
B: send $c_{\sigma_{res}}$ to A
A: $\sigma_{res} = Dec(c_{\sigma_{res}};\ sk)$
A: **if** $\sigma_{res} < 0$ **then**
  return True
else
  return False
end

to Eq. 12, and calculate the termination threshold $\theta^2 \cdot \sigma^2_{max}$. It is noteworthy that in the actual operation process, the max singular value(σ_{max}) is the first singular value(σ_1) of the data. Equation Eq. 12 shows how both parties calculate the singular value.

$$\begin{aligned} \sigma^2 &= \|X^T\boldsymbol{u}_a + Y^T\boldsymbol{u}_b\|^2 \\ &= (X^T\boldsymbol{u}_a + Y^T\boldsymbol{u}_b)^T(X^T\boldsymbol{u}_a + Y^T\boldsymbol{u}_b) \\ &= (\boldsymbol{u}_a^T X + \boldsymbol{u}_b^T Y)(X^T\boldsymbol{u}_a + Y^T\boldsymbol{u}_b) \\ &= \boldsymbol{u}_a^T XX^T\boldsymbol{u}_a + 2(\boldsymbol{u}_b^T YX^T\boldsymbol{u}_a) + \boldsymbol{u}_b^T YY^T\boldsymbol{u}_b \end{aligned} \tag{12}$$

Through the method demonstrated in Eq. 13, we can accurately determine whether the singular values have reached the threshold, ensuring the confidentiality of data from both parties. We give an example when the user sets the calculation stop condition($\theta = 10\%$): when the latest calculated singular value result is less than 10% of the maximum singular value, where r is a random number required to ensure information security and the specific security analysis will be introduced in Section 4.

$$\begin{aligned} \sigma &< 0.1 \cdot \sigma_{max} \\ \sigma^2 &< 0.01 \cdot \sigma^2_{max} \\ \sigma^2 - 0.01 \cdot \sigma^2_{max} &< 0 \\ r(\sigma^2 - 0.01 \cdot \sigma^2_{max}) &< 0, r \textit{ is a random number} \end{aligned} \tag{13}$$

Following the methods mentioned in Eq. 12 and Eq. 13, we construct protocol. 3 for the stop condition. To ensure security, the protocol stipulates that user B only shares the product of the difference between the current and maximum singular values multiplied by the perturbation r.

Following the protocol described previously, we now provide the protocol for the complete system, as outlined in Protocol. 4. During the actual execution of Protocol.4, we can not only obtain relatively accurate results through the termination conditions in Protocol.2, but also calculate according to the number of features

Protocol 4: HE-DR

Input : X, Y, pk, sk, θ, n
Output: U

A: Initializing the encryption parameters
A/B: Running Protocol. 1
A/B: $i = 1$
while $i < n$ **do** `// n is max number of rank`
  A/B: Running Protocol. 2
  `// θ is the proportion of noise information`
  if *Protocol. 3 with* θ **then**
    break the **while** loop
  else
    continue
  end
  B: $U_a = [\bar{\boldsymbol{u}}_{a1}, \cdots, \bar{\boldsymbol{u}}_{ai}]$, $U_b = [\bar{\boldsymbol{u}}_{b1}, \cdots, \bar{\boldsymbol{u}}_{bi}]$
  B: $U = [U_a; U_b]$
  $i++$
end

that users desire. Moreover, in solving each orthogonal basis, we found that most of the time, only 3-5 iterations are needed to reach the convergence condition.

4 SECURITY ANALYSIS

In this section, we perform a security analysis on the previously designed computation protocol. We only have two parties in the computation system: user A and user B. Secondly, all our work is established based on the semi-honest model stated in [15]: computation parties strictly follow the protocol but remain curious about each other's data.

4.1 Proof of Protocol. 1

Correctness For arbitrary matrix M and an initial random vector $\boldsymbol{u}$, the principal eigenvector $\bar{\boldsymbol{u}}$ is finally obtained by using power method iteration on MM^T, and the specific convergence analysis can refer to [20].

$$\bar{\boldsymbol{u}} = MM^T\boldsymbol{u} \tag{14}$$

Then, for our scheme, we divide the original matrix M into two matrices M_a, M_b with consistent dimensions in the horizontal direction. The initial random vector $\boldsymbol{u}$ is also divided into two column vectors $\boldsymbol{u}_a$, $\boldsymbol{u}_b$ with the same dimension in the horizontal direction, which is shown in Fig. 4. Then we have:

$$\begin{pmatrix}\bar{\boldsymbol{u}}_a\\ \bar{\boldsymbol{u}}_b\end{pmatrix} = \begin{pmatrix}M_a\\ M_b\end{pmatrix}\begin{pmatrix}M_a^T & M_b^T\end{pmatrix}\begin{pmatrix}\boldsymbol{u}_a\\ \boldsymbol{u}_b\end{pmatrix} \tag{15}$$

Each party only needs to calculate their parts separately:

$$\begin{aligned}\bar{\boldsymbol{u}}_a &= M_aM_a^T\boldsymbol{u}_a + M_aM_b^T\boldsymbol{u}_b\\ \bar{\boldsymbol{u}}_b &= M_bM_a^T\boldsymbol{u}_a + M_bM_b^T\boldsymbol{u}_b\end{aligned} \tag{16}$$

Then, integrate the results to obtain those consistent with those obtained by directly performing power iteration in Eq. 14.

Security According to the proof idea in [4], we need to construct a simulator first. The protocol is secure if we can prove that the protocol is computationally indistinguishable between the real and the simulator.

A's view is $V_a(I_a, coins, m_a)$, where $I_a = (X, \boldsymbol{u}_a, pk, sk)$ are the input of A, *coins* represents the outcome of the A's internal coin tosses, $m_a = (Y^T\boldsymbol{u}_b, c_{\bar{\boldsymbol{u}}_{be}}, \|\bar{\boldsymbol{u}}_b\|^2)$ are all information obtained by A during the execution of the protocol. Also, we assume that the initial vector is sampled from $\mathcal{P}$ distribution. Since the data and initial vector are two independent variables and come from two independent distributions, the distribution of the product of two variables is equal to the product of two distributions. We denote this new distribution as $\mathcal{P}'$, $Y^T\boldsymbol{u}_b \leftarrow \mathcal{P}'$. The above assumptions and notations for distributions also apply to other protocols.

Given I_a, $\widetilde{\bar{\boldsymbol{u}}_a}$, we build the simulator S_a:

(1) Pick $\widetilde{Y^T\boldsymbol{u}_b} \leftarrow \mathcal{P}'$
(2) Pick $\widetilde{\|\bar{\boldsymbol{u}}_b\|^2} \leftarrow \widetilde{\mathcal{R}}$
(3) Encrypt $\widetilde{\bar{\boldsymbol{u}}_{be}}$ under CKKS: $\widetilde{c_{\bar{\boldsymbol{u}}_{be}}}$
(4) Let $\widetilde{r}$ be random coins for CKKS encryption
(5) Compute $\widetilde{\bar{\boldsymbol{u}}_a}$ by normalization
(6) Output $(I_a, \widetilde{coins}, \widetilde{m_a}, \widetilde{\bar{\boldsymbol{u}}_a})$

For vectors $Y^T\boldsymbol{u}_b$ and $\widetilde{Y^T\boldsymbol{u}_b}$, both are sampled from distribution $\mathcal{P}'$. The variable $\bar{\boldsymbol{u}}_b$ is taken from the $\mathcal{R}$ distribution, but after taking the normalization and square operation, the distribution of the new variable $\widetilde{\|\bar{\boldsymbol{u}}_b\|^2}$ changes, we donate it as distribution $\widetilde{\mathcal{R}}$. For distribution $\widetilde{\mathcal{R}}$, according to the Theorem. 2.2 and Eq. 10, when the dimension of $\bar{\boldsymbol{u}}_b$ is large enough($\geq$ 50), we can say $\mathcal{R} \stackrel{\cdot}{\sim} \widetilde{\mathcal{R}}$, $\|\bar{\boldsymbol{u}}_b\|^2$ and $\widetilde{\|\bar{\boldsymbol{u}}_b\|^2}$ are sampled from distribution $\mathcal{R}$. According to Smudging Lemma [2], when Q is larger enough, $(\boldsymbol{u}_r, \bar{\boldsymbol{u}}_{be}) \leftarrow Q$ and $(\bar{\boldsymbol{u}}_{be}, \widetilde{\bar{\boldsymbol{u}}_{be}}) \leftarrow Q$. Thus, for ciphertext $c_{\bar{\boldsymbol{u}}_{be}}$, its distribution is the same as that of $\widetilde{c_{\bar{\boldsymbol{u}}_{be}}}$, and we conclude that the distributions $V_a = (I_a, coins, m_a)$ and $S_a = (I_a, \widetilde{\bar{\boldsymbol{u}}_a})$ are the same.

User B is constructed in a similar way as user A. Again, for simplicity of expression, we denote the input of user B by $I_b = (B, \boldsymbol{u}_b, pk, sk)$ and all information accepted by user B during protocol execution by $m_b = (c_{\boldsymbol{t}}, c_{AB^T\boldsymbol{u}_b}, \|\bar{\boldsymbol{u}}_a\|^2)$.

B's view is $V_b = (I_b, coins, m_b)$ where r are random internal coin tosses of B. The simulator $S_b = (I_b, \widetilde{\bar{\boldsymbol{u}}_b})$ is shown below:

(1) Pick $\widetilde{\|\bar{\boldsymbol{u}}_a\|^2} \leftarrow \widetilde{\mathcal{R}}$
(2) Encrypt $\widetilde{\boldsymbol{t}}$ and $\widetilde{AB^T\boldsymbol{u}_b}$ under CKKS: $\widetilde{c_{\boldsymbol{t}}}$ and $\widetilde{c_{AB^T\boldsymbol{u}_b}}$
(3) $\widetilde{coins}$ is random coins for CKKS encryption
(4) Compute $\widetilde{\bar{\boldsymbol{u}}_b}$ by normalization
(5) Output $(I_b, \widetilde{coins}, \widetilde{m_b}, \widetilde{\bar{\boldsymbol{u}}_b})$

For B, the distribution of $\widetilde{\|\bar{\boldsymbol{u}}_a\|^2}$ can also be approximated by $\mathcal{R}$ and the remaining ciphertext variables come from the same distribution because of the RLWE assumption. So we have,

$$\begin{aligned}S_a(I_a, \widetilde{\bar{\boldsymbol{u}}_a}) &\stackrel{c}{\equiv} V_a(I_a, coins, m_a)\\ S_b(I_b, \widetilde{\bar{\boldsymbol{u}}_b}) &\stackrel{c}{\equiv} V_b(I_b, coins, m_b)\end{aligned} \tag{17}$$

Thus, we conclude that protocol. 1 is secure in the semi-honest model.

4.2 Proof of Protocol. 2

Correctness Combining the knowledge of Rank-Revealing mentioned before, the subsequent singular vectors are calculated as shown in Eq. 18.

$$\boldsymbol{u} = MM^T(\boldsymbol{z} - UU^T\boldsymbol{z}) \tag{18}$$

where, $\boldsymbol{z}$ is initial random vector, $U = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_k]$ is orthogonal matrix.

As in Protocal. 1, we horizontally split the matrix and vectors, and we get:

$$\begin{pmatrix} \bar{\boldsymbol{u}}_a \\ \bar{\boldsymbol{u}}_b \end{pmatrix} = \begin{pmatrix} M_a \\ M_b \end{pmatrix} \left(M_a^T \; M_b^T \right) \left[\begin{pmatrix} \boldsymbol{z_a} \\ \boldsymbol{z_b} \end{pmatrix} - \begin{pmatrix} U_a \\ U_b \end{pmatrix} \left(U_a^T \; U_a^T \right) \begin{pmatrix} \boldsymbol{z_a} \\ \boldsymbol{z_b} \end{pmatrix} \right] \tag{19}$$

To simplify the formula, we let

$$\begin{aligned} \boldsymbol{\Delta_a} &= \boldsymbol{z_a} - U_a U_a^T \boldsymbol{z_a} - U_a U_b^T \boldsymbol{z_b} \\ \boldsymbol{\Delta_b} &= \boldsymbol{z_b} - U_b U_a^T \boldsymbol{z_a} - U_b U_b^T \boldsymbol{z_b} \end{aligned} \tag{20}$$

and then, we get:

$$\begin{aligned} \bar{\boldsymbol{u}}_a &= M_a M_a^T \boldsymbol{\Delta_a} + M_a M_b^T \boldsymbol{\Delta_b} \\ \bar{\boldsymbol{u}}_b &= M_b M_a^T \boldsymbol{\Delta_a} + M_b M_b^T \boldsymbol{\Delta_b} \end{aligned} \tag{21}$$

Therefore, as with Protocol. 1, the left singular vector of matrix M can be obtained after re-concatenating the results of both parties.

Security The proof process is similar to that of Protocol. 1, A's view is $V_a = (I_a, coins, m_a)$ where $I_a = (X, U_a, pk, sk)$, $m_a = (\boldsymbol{\Delta_{aj}}, \boldsymbol{t}, c_{\boldsymbol{z_{aee}}}, c_{\boldsymbol{z_{bee}}})$.

Given the I_a, the simulator S_a:

(1) $\widetilde{r}$, $\widetilde{c_{\boldsymbol{z_{aee}}}}$ and $\widetilde{c_{\boldsymbol{z_{bee}}}}$ are generated by CKKS encryption
(2) Pick $\widetilde{\boldsymbol{\Delta_{aj}}} \leftarrow \mathcal{P}'$
(3) Pick $\widetilde{\boldsymbol{t}} \leftarrow \mathcal{Q}$
(4) Output $(I_a, \widetilde{coins}, \widetilde{m_a})$

As shown above, we assume the initial vector $(\boldsymbol{z_a}, \boldsymbol{z_b}) \leftarrow \mathcal{P}$, and $\boldsymbol{\Delta_{aj}} \leftarrow \mathcal{P}'$. For the simulator S_a, $\widetilde{\boldsymbol{\Delta_{aj}}}$ is directly sampled from $\mathcal{P}'$ so that $\boldsymbol{\Delta_{aj}}$ and $\widetilde{\boldsymbol{\Delta_{aj}}}$ is picked from the same distribution. Also, the distribution of $\boldsymbol{t}$ and $\widetilde{\boldsymbol{t}}$ are the same because we use a larger distribution $\mathcal{Q}$ to replace the original distribution [2]. Thus, we can say the view and simulator of A have followed the same distribution. B's view is $V_b = (I_b, coins, m_b)$, where $I_b = (Y, U_b, c_{XY^T\boldsymbol{u}_b}, pk)$, $m_b = (c_{\boldsymbol{z_{ae}}}, c_{X^T\boldsymbol{\Delta_a}}, \boldsymbol{z_{aee}}, \boldsymbol{z_{bee}})$.

We build simulator S_b by giving I_b:

(1) $\widetilde{coins}$, $\widetilde{c_{\boldsymbol{z_{ae}}}}$ and $\widetilde{c_{A^T\boldsymbol{\Delta_a}}}$ are computed from CKKS encryption system
(2) Pick $\widetilde{\boldsymbol{z_{aee}}}, \widetilde{\boldsymbol{z_{bee}}} \leftarrow \mathcal{Q}$
(3) Output $(I_b, \widetilde{coins}, \widetilde{U_a}, \widetilde{U_b})$

Since the distribution $\mathcal{Q}$ is large enough, $\widetilde{\boldsymbol{z_{aee}}}$, $\widetilde{\boldsymbol{z_{bee}}}$ and $\boldsymbol{z_{aee}}, \boldsymbol{z_{bee}}$ are chosen from the same distribution [2]. Thus, we can say the B's view V_b and simulator S_b are computed from the same distribution and we have:

$$\begin{aligned} S_a(I_a) &\overset{c}{\equiv} V_a(I_a, coins, m_a) \\ S_b(I_b, \widetilde{U_a}, \widetilde{U_b}) &\overset{c}{\equiv} V_b(I_b, coins, m_b) \end{aligned} \tag{22}$$

Thus, we conclude that Protocol. 2 is secure in the semi-honest model.

4.3 Proof of Protocol. 3

Correctness it is obvious according to Eq. 12. **Security** A's view is $V_a = (X, pk, sk, coins, \sigma_{res})$, we build A's simulator S_a:

(1) $\widetilde{\sigma_{res}} \leftarrow \mathcal{Q}$
(2) $\widetilde{coins}$ is random coins for CKKS encryption
(3) Output $(X, pk, sk, \widetilde{coins}, \widetilde{\sigma_{res}})$

Since A has the private key, the accepted information is σ_{res}. Also, Q is large enough that σ_{res} and $\widetilde{\sigma_{res}}$ are sampled from the same distribution [2], we can say that the distributions of V_a and S_a coincide. B's view is $V_b = (Y, \bar{\boldsymbol{u}}_a, \bar{\boldsymbol{u}}_b, coins, c_{\sigma_a}, c_{\boldsymbol{t}})$ and B's simulator is $S_b = (Y, \widetilde{coins}, \widetilde{\bar{\boldsymbol{u}}_a}, \widetilde{\bar{\boldsymbol{u}}_b})$:

(1) Encrypt $\widetilde{\sigma_a}$, $\widetilde{t}$ by CKKS encrytion: $\widetilde{c_{\sigma_a}}$, $\widetilde{c_{\boldsymbol{t}}}$
(2) $\widetilde{coins}$ is random coins for CKKS encryption
(3) Output $(Y, \widetilde{coins}, \widetilde{c_{\sigma_a}}, \widetilde{c_{\boldsymbol{t}}})$

Based on RLWE assumption, c_{σ_a}, c_t and $\widetilde{c_{\sigma_a}}$, $\widetilde{c_{\boldsymbol{t}}}$ choose from the same distribution, we conclude that Protocol. 3 is security in semi-honest model.

$$\begin{aligned} S_a(X, pk, sk, \widetilde{\sigma_{res}}) &\overset{c}{\equiv} V_a(X, pk, sk, coins, \sigma_{res}) \\ S_b(Y, \widetilde{coins}, \widetilde{\bar{\boldsymbol{u}}_a}, \widetilde{\bar{\boldsymbol{u}}_b}) &\overset{c}{\equiv} V_b(Y, \bar{\boldsymbol{u}}_a, \bar{\boldsymbol{u}}_b, coins, c_{\sigma_a}, c_{\boldsymbol{t}}) \end{aligned} \tag{23}$$

Therefore, by combining Eq. 17, Eq. 22 and Eq. 23, our designed computation protocol is secure in the semi-honest model.

4.4 Proof of Protocol. 4

Since the entire algorithm is executed sequentially according to each sub-protocol, the output of one sub-protocol becomes the input for the next. Therefore, the security of each sub-protocol is guaranteed. Based on the sequential modular composition mentioned in [6], we can conclude that the system still maintains its security under the semi-honest model.

5 APPLICATION AND EXPERIMENT

Based on the previously designed protocols, we design the experimental scenario as follows: there are two parties, A and B, involved, both of which have data with the same dimension, and they want to realize the joint Dimensionality Reduction(DR) of data through privacy computation, to achieve more accurate data dimensionality reduction. Since the scheme is a joint DR for two-party user data, the matrix data are horizontally or vertically combinations according to users' needs, as shown in Fig. 3.

It is worth noting that although our scheme only obtains the left orthogonal matrix at last, it is only necessary to change whether it is A^TA or AA^T to achieve dimensionality reduction of different parameters. For example, assume that the data consists of samples and features, then when we iterate over A^TA, the final output left singular matrix can achieve dimensionality reduction of features. Conversely, when we exchange samples and features, the result is a dimensionality reduction of samples. In real life, most of the time, both sides have the same characteristics, and the data of different samples are so that the data of both parties in this experiment is divided into horizontal cuts through the data matrix $M(m \times n)$, as shown in Fig. 4.

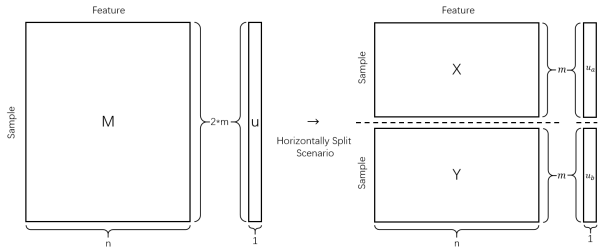

Figure 4: Experiment data settings

5.1 Experiment Setting

This section gives the scheme's computation efficiency, communication overhead, and accuracy test. Meanwhile, a comparison with other similar homomorphic encryption-based schemes is also given. Regarding the security factor of CKKS, to meet the executable multiplication depth, we set the polynomial modulus degree as 16384, and at the same time, to meet the requirements of accuracy, the scale factor is set as 2^{60}. Furthermore, in the previous protocol, we introduce a perturbation sampled from Q, where the range of Q is $[0, 2^{\lambda}]$ and λ is represented as security parameter, according to the Smudging Lemma [2]. In practical experiments, due to the limitations of the CKKS scheme and Tenseal library [3], when the polynomial modulus degree equals 16384, the max coefficient modulus bit-length equals 438. With the scaling factor is set as 2^{60} and the multiplication depth as 5, if $\lambda = 128$, the max coefficient modulus bit-length required will exceed 438. Therefore, to better illustrate our scheme's feasibility, computational efficiency, and accuracy, we set $\lambda = 40$.

All the computations are carried out in PyCharm on a personal computer with Intel Core i9-12900KF CPU (3.19GHz) and 64GB RAM. In addition, to compare with the previous experiments, we did not use multithreading to implement parallel computation in this experiment. All experiments have identical testing settings with the same security parameters.

5.2 Previous experiment

Our scheme has higher computational efficiency and supports higher data dimensions than previous privacy-preserving dimensionality reduction schemes. For example, The scheme in [1] takes more than 100 minutes to process a 50×50 data, while our scheme only needs 45 seconds to decompose a 128×128 data. Schemes in [24] takes several years to decompose an $80k \times 1k$ matrix, while our scheme only takes several hours.

5.3 Synthetic dataset

First, we use synthetic data to test the homomorphic encryption dimensionality reduction scheme(HE-DR) performance. To test the performance of the HE-DR method with different dimensions ($m \times n = sample \times feature$), we divide the dataset into three groups with feature numbers 2048, 4096, and 8192. The number of samples in each group gradually increases from 8192 to 40k. In addition, we set the value of the matrix to be randomly selected within the interval $[0, 10]$, which obeys a uniform distribution.

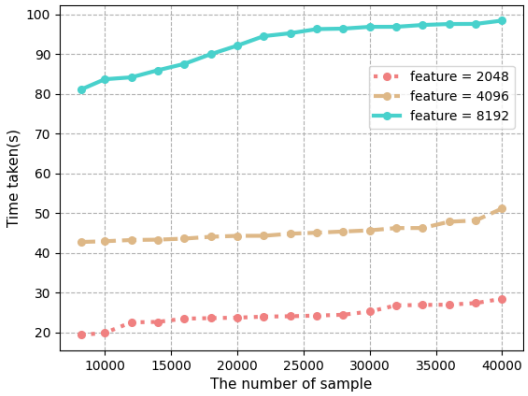

Figure 5: Time Consumption

We tested the computational efficiency of the HE-DR method with three groups of different feature numbers as the sample number increases, and its results are shown in Fig. 5. It can be known from the figure that when the number of features is 2048, as the sample number increases and when the sample number reaches 40k, which is to say that both sides cooperate to calculate one of the orthogonal bases of a matrix with more than 80 million elements in no more than the 30s, then it is extrapolated to the actual scene, the time to extract ten orthogonal bases, which is only several minutes. In addition, since the whole algorithm is the multiplication of plaintext matrix and ciphertext vector, the influence of the number of features on the calculation time is greater than that of the growth of the number of samples. Therefore, our scheme will be advantageous in dealing with data with many samples.

Meanwhile, to better reflect that the HE-DR method still has high computational efficiency in high-dimensional data, we also compared the computational efficiency of ciphertext and plaintext, as shown in Fig. 6, by calculating the ratio of computation time to illustrate that this scheme still has good computational efficiency performance under high-dimensional data. Also, with the increase in sample size, the ratio of ciphertext time to plaintext time gradually decreases, and finally, both stabilize around 5 times.

5.4 Real world dataset

In addition to testing HE-DR with synthetic datasets, we also use some real datasets. To compare the performance of our scheme and Homomorphic PCA(HPCA) scheme[29], we select four classification datasets which are the same as the test dataset of HPCA, Winequailty-white [31], Winequailty-red [31], Air Quality [34], and Parkinson's telemonitoring [22]. We additionally add one Breast Cancer Wisconsin dataset [35]. To guarantee the consistency of experiments, we only test the time cost of computing the same number of singular values and communication overhead when N=16384 and the results are shown in Table 1. For each data set with dimension $m \times n$, m denotes the number of samples, n denotes the number of features, and l refers to the number of principal components extracted.

Table 1: Performance of HPCA and HE-DR algorithm

Dataset	m	n	l	Computation(Sec)		Communication(MB)	
				HPCA	HE-DR	HPCA	HE-DR
Winequailty-white	4898	11	3	183.912	3.3262	424	150
Winequailty-red	1599	11	3	118.626	2.6627	424	150
Air Quality	9357	13	4	290.094	5.2901	563	198
Parkinsons	5875	16	4	209.946	4.8089	563	198
Breast Cancer Wisconsin	569	30	2	125.574	2.0822	286	103

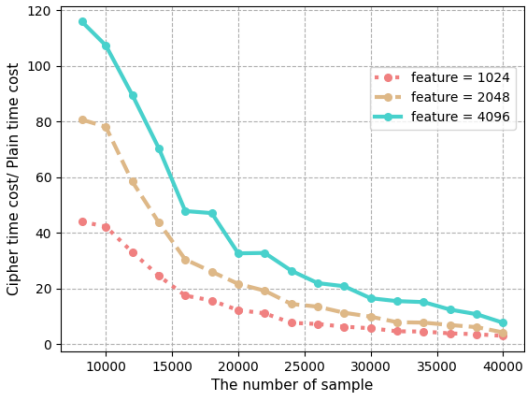

Figure 6: The ratio of cipher and plain text time consumption

Then, it can be seen from Table 1 that when the dimensions are the same, and the same number of principal components are extracted, our scheme is nearly 60 times faster than HPCA regarding computational efficiency. Furthermore, we conduct a comparative analysis of the communication overhead. Given that both are computed iteratively, the magnitude of the communication overhead is primarily contingent upon the total number of principal components calculated and the associated communication overhead for each extraction. As Table 1 shows that our proposed scheme requires approximately 50MB of communication overhead to extract each principal component. In contrast, HPCA requires roughly 150MB. Consequently, our scheme only demands one-third of the communication overhead required by HPCA.

5.5 Error analysis

To demonstrate that the computational results of our ciphertext scheme maintain a high level of accuracy as the plaintext scheme, we present in this section the error results obtained when different schemes perform matrix decomposition on the same dataset. Firstly, we choose the CRANFIELD dataset as the test set, with dimensions of 3000×1400. According to previous data partitioning methods, the dimensions of datasets for User A and User B are both 1500×1400. We then use the SVD, RR, DR(Plain algorithm), HE-DR(cipher algorithm), and HPCA schemes to decompose the dataset $M(3000 \times 1400)$ into orthogonal matrices U. Finally, we measure the accuracy of the results from two aspects: orthogonality and the distance between two subspaces, as shown in Table 2.

According to the definition of orthogonal matrices, when the column vectors or row vectors of a matrix are pairwise orthogonal, and each vector has a length of 1, $U^TU = UU^T = I$. The matrix U we seek is a left orthogonal matrix, so we use $\|I - U^TU\|_2$ to verify the results' orthogonality. Table 2 shows that the errors in orthogonality of our plaintext and ciphertext algorithms are $2.61e^{-15}$ and $2.43e^{-15}$, respectively. The results can be approximated to 0, indicating that the outputs of our algorithm satisfy orthogonality. It is important to note that all error results are calculated under the scale factor of 2^{60}. This choice is primarily due to two reasons: Firstly since the final solution is derived through iterative processes, the accumulation of errors in the ciphertext occurs relatively rapidly. Secondly, the nature of the task demands high precision, and each computational outcome can significantly influence subsequent calculations. Consequently, if the retained precision is insufficient, even minor errors introduced will impact future calculation results. In practical testing scenarios, if the value of the scale factor is less than 2^{60}, then only the first two feature vectors are correct.

Secondly, there are infinitely many choices for the unit orthogonal basis of a subspace, so we measure the accuracy of the results by the distance between the subspace. When the subspace distance approaches 0, it indicates that the subspaces almost coincide, demonstrating the results' accuracy. Based on the definition of subspace distance, let $S_1,\ S_2 \subseteq \mathbb{R}^n$ as two subspaces with the same dimension, and their subspace distance is $dist(S_1, S_2) = \|P_1 - P_2\|_2$. Here, P is the orthogonal projection of S and satisfies $P^T = (UU^T)^T = UU^T = P$. We use the orthogonal matrix obtained from QR decomposition as a standard and calculate the distance between the left orthogonal matrices obtained by different methods and the subspace spanned by the standard orthogonal matrix. The results are shown in Table 2. According to the results, it can be seen that the distance between the subspaces obtained by plaintext(DR) and ciphertext(HE-DR) algorithms and the subspace of the standard algorithm is very small, almost equal to 0, which means that the subspaces overlap. Therefore, we can assert the correctness of the results obtained by our method.

6 CONCLUSION AND FUTURE WORK

This paper introduces a new CKKS-based homomorphic encryption dimensionality reduction scheme that offers superior computational

Table 2: The error of each method

	SVD	Rank-Revealing	DR	HE-DR	HPCA
$\|I - U^TU\|_2$	1.79e-14	2.54e-15	2.61e-15	2.43e-15	1.34
$dist(S_1, S_2)$	6.42e-15	8.68e-15	5.52e-13	1.46e-11	/

efficiency and reduced communication overhead. In our scheme, we consistently perform operations on the plaintext matrix and ciphertext vector, avoiding computations between ciphertexts or matrices as in traditional schemes. Moreover, unlike previous schemes that required transmission of the ciphertext matrix, our approach only requires transmission of the ciphertext vector, significantly reducing communication overhead while ensuring data security. From a security perspective, our scheme obviates the need for a trusted third party and eliminates collusion concerns inherent in previous models[7, 16, 23, 24, 32].

Since the current scheme is developed based on Tenseal, some ciphertext calculations and parameter selection are not optimal, such as packing ciphertext data, parallel computing, scaling factors, λ, etc. In subsequent work, we will further optimize the ciphertext calculation to improve computational efficiency.

REFERENCES

[1] Mohammad Al-Rubaie, Pei-yuan Wu, J. Morris Chang, and Sun-Yuan Kung. 2017. Privacy-preserving PCA on horizontally-partitioned data. In *2017 IEEE Conference on Dependable and Secure Computing*. 280–287. https://doi.org/10.1109/DESEC.2017.8073817

[2] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. 2012. Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE. In *Advances in Cryptology – EUROCRYPT 2012*, David Pointcheval and Thomas Johansson (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 483–501.

[3] Ayoub Benaissa, Bilal Retiat, Bogdan Cebere, and Alaa Eddine Belfedhal. 2021. TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption. arXiv:2104.03152 [cs.CR]

[4] Raphael Bost, Raluca A. Popa, Stephen Tu, and Shafi Goldwasser. 2015. Machine Learning Classification over Encrypted Data. *IACR Cryptol. ePrint Arch.* 2014 (2015), 331. https://api.semanticscholar.org/CorpusID:2259147

[5] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2014. (Leveled) Fully Homomorphic Encryption without Bootstrapping. *ACM Trans. Comput. Theory* 6, 3, Article 13 (jul 2014), 36 pages. https://doi.org/10.1145/2633600

[6] Ran Canetti. 1998. Security and Composition of Multi-party Cryptographic Protocols. https://api.semanticscholar.org/CorpusID:264203571

[7] Di Chai, Leye Wang, Junxue Zhang, Liu Yang, Shuowei Cai, Kai Chen, and Qiang Yang. 2022. Practical Lossless Federated Singular Vector Decomposition over Billion-Scale Data. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 46–55. https://doi.org/10.1145/3534678.3539402

[8] Shuo Chen, Rongxing Lu, and Jie Zhang. 2017. A Flexible Privacy-Preserving Framework for Singular Value Decomposition Under Internet of Things Environment. In *Trust Management XI*, Jan-Philipp Steghöfer and Babak Esfandiari (Eds.). Springer International Publishing, Cham, 21–37.

[9] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic Encryption for Arithmetic of Approximate Numbers. In *Advances in Cryptology – ASIACRYPT 2017*, Tsuyoshi Takagi and Thomas Peyrin (Eds.). Springer International Publishing, Cham, 409–437.

[10] Ronald Cramer, Ivan Bjerre Damgård, and Jesper Buus Nielsen. 2015. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press.

[11] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. [n. d.]. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography* (Berlin, Heidelberg, 2006), Shai Halevi and Tal Rabin (Eds.). Springer, 265–284. https://doi.org/10.1007/11681878_14

[12] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat Practical Fully Homomorphic Encryption. *IACR Cryptol. ePrint Arch.* 2012 (2012), 144. https://api.semanticscholar.org/CorpusID:1467571

[13] Craig Gentry. 2009. *A fully homomorphic encryption scheme*. Ph. D. Dissertation. Stanford, CA, USA. Advisor(s) Boneh, Dan. AAI3382729.

[14] Paris V. Giampouras, Athanasios A. Rontogiannis, Konstantinos E. Themelis, and Konstantinos D. Koutroumbas. 2016. Online Low-Rank Subspace Learning from Incomplete Data: A Bayesian View. arXiv:1602.03670 [stat.ML]

[15] Oded Goldreich. 2004. *Foundations of Cryptography*. Cambridge University Press.

[16] Qiang Guo, Caiming Zhang, Yunfeng Zhang, and Hui Liu. 2016. An Efficient SVD-Based Method for Image Denoising. *IEEE Transactions on Circuits and Systems for Video Technology* 26, 5 (2016), 868–880. https://doi.org/10.1109/TCSVT.2015.2416631

[17] Shuguo Han, Wee Keong Ng, and Philip S. Yu. 2009. Privacy-Preserving Singular Value Decomposition. In *2009 IEEE 25th International Conference on Data Engineering*. 1267–1270. https://doi.org/10.1109/ICDE.2009.217

[18] Anne Hartebrodt, Reza Nasirigerdeh, David B. Blumenthal, and Richard Röttger. 2021. Federated Principal Component Analysis for Genome-Wide Association Studies. In *2021 IEEE International Conference on Data Mining (ICDM)*. 1090–1095. https://doi.org/10.1109/ICDM51629.2021.00127

[19] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2021. Advances and Open Problems in Federated Learning. arXiv:1912.04977 [cs.LG]

[20] Tsung-Lin Lee, Tien-Yien Li, and Zhonggang Zeng. 2009. A Rank-Revealing Method with Updating, Downdating, and Applications. Part II. *SIAM J. Matrix Anal. Appl.* 31 (2009), 503–525. https://api.semanticscholar.org/CorpusID:184024

[21] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine* 37, 3 (May 2020), 50–60. https://doi.org/10.1109/msp.2020.2975749

[22] Max Little. 2008. Parkinsons. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C59C74.

[23] Bowen Liu, Balázs Pejó, and Qiang Tang. 2023. Privacy-Preserving Federated Singular Value Decomposition. *Applied Sciences* 13, 13 (2023). https://doi.org/10.3390/app13137373

[24] Bowen Liu and Qiang Tang. 2019. Privacy-Preserving Decentralised Singular Value Decomposition. In *Information and Communications Security: 21st International Conference, ICICS 2019, Beijing, China, December 15–17, 2019, Revised Selected Papers* (Beijing, China). Springer-Verlag, Berlin, Heidelberg, 703–721. https://doi.org/10.1007/978-3-030-41579-2_41

[25] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2010. On Ideal Lattices and Learning with Errors over Rings. *J. ACM* 60 (2010), 43:1–43:35. https://api.semanticscholar.org/CorpusID:1606347

[26] Nicola Mastronardi and Paul Van Dooren. 2016. Rank-revealing decomposition of symmetric indefinite matrices via block anti-triangular factorization. *Linear Algebra Appl.* 502 (2016), 126–139. https://doi.org/10.1016/j.laa.2015.09.037 Structured Matrices: Theory and Applications.

[27] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2023. Communication-Efficient Learning of Deep Networks from Decentralized Data. arXiv:1602.05629 [cs.LG]

[28] Valeria Nikolaenko, Stratis Ioannidis, Udi Weinsberg, Marc Joye, Nina Taft, and Dan Boneh. 2013. Privacy-preserving matrix factorization. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security* (Berlin, Germany) *(CCS '13)*. Association for Computing Machinery, New York, NY, USA, 801–812. https://doi.org/10.1145/2508859.2516751

[29] Samanvaya Panda. 2021. Principal Component Analysis Using CKKS Homomorphic Scheme. In *International Conference on Cyber Security Cryptography and Machine Learning*. https://api.semanticscholar.org/CorpusID:235803242

[30] Jaehyoung Park and Hyuk Lim. [n. d.]. Privacy-Preserving Federated Learning Using Homomorphic Encryption. 12, 2 ([n. d.]), 734. https://doi.org/10.3390/app12020734

[31] A. Cerdeira Paulo Cortez, F. Almeida, T. Matos, and J. Reis. 2009. Wine Quality. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C56S3T.

[32] K. V. Ravi Kanth, Divyakant Agrawal, and Ambuj Singh. 1998. Dimensionality reduction for similarity searching in dynamic databases. *SIGMOD Rec.* 27, 2 (jun 1998), 166–176. https://doi.org/10.1145/276305.276320

[33] Oded Regev. 2009. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* 56, 6, Article 34 (sep 2009), 40 pages. https://doi.org/10.1145/1568318.1568324

[34] Saverio Vito. 2016. Air Quality. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C59K5F.

[35] Nick Street William Wolberg, Olvi Mangasarian and W. Street. 1995. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5DW2B.