Privacy-Preserving, Efficient, and Accurate Dimensionality Reduction

Haonan Yuan, Wenyuan Wu, and Jingwei Chen

***Abstract*—The swift evolution of artificial intelligence and big data has dramatically increased data volume and computational complexity, thereby considerably escalating data storage and processing costs. As a result, dimensionality reduction has become an essential phase in data pre-processing. Moreover, due to the privacy concerns associated with data gathered by various organizations, data sharing is restricted, introducing further challenges in data analysis and the training of machine learning models. Consequently, we introduce an innovative privacy-preserving dimensionality reduction scheme (PP-DR). PP-DR secures participant data using homomorphic encryption and eschews intricate bootstrapping tasks by employing secure interaction protocols, thereby efficiently performing joint dimensionality reduction on data shared among all participants. In contrast to current dimension reduction approaches employing homomorphic encryption, PP-DR achieves superior computational accuracy with an average error of only $1*10^{-8}$. Additionally, it enhances computational efficiency by 30 to 200 times and reduces communication overhead by at least 70%. This study underscores the practical feasibility of secure multi-party collaborative dimension reduction using homomorphic encryption.**

***Index Terms*—Privacy-preserving, efficient, accurate, dimensionality reduction, homomorphic encryption.**

I. Introduction

Big data is crucial in the contemporary societal and economic domains, with notable applications in fields like bio-genetics for purposes such as disease forecasting, clinical decision processes, and genomic studies. Nonetheless, due to the high sensitivity of genetic information and the restriction on public dissemination by data proprietors, many biomedical organizations, labs, and hospitals face challenges of inaccuracy stemming from inadequate data volumes or homogeneous data types when employing their datasets for model training [1]. Directly sharing data for collaborative training poses a risk of data leakage and significantly reduces computational efficiency due to the surge in data dimensions. Therefore, our primary research focus is achieving secure, efficient, and accurate dimension reduction of data under the participation of multiple parties.

Received 26 March 2025; revised 21 August 2025 and 18 January 2026; accepted 3 February 2026. Date of publication 9 February 2026; date of current version 20 February 2026. This work was supported in part by the National Key Research Project of China under Grant 2025YFA1017201, in part by the National Natural Science Foundation of China (NSFC) under Grant 12571553, and in part by the Natural Science Foundation of Chongqing under Grant CSTB2023YSZX-JCX0008. The associate editor coordinating the review of this article and approving it for publication was Prof. Jianting Ning. *(Corresponding author: Wenyuan Wu.)*

The authors are with Chongqing Key Laboratory of Secure Computing for Biology, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences (CAS), Chongqing 400714, China, and also with Chongqing School, University of Chinese Academy of Sciences, Chongqing 400714, China (e-mail: yuanhaonan@cigit.ac.cn; wuwenyuan@cigit.ac.cn; chenjingwei@cigit.ac.cn).

Digital Object Identifier 10.1109/TIFS.2026.3662200

Rank-revealing techniques have garnered sustained interest in research due to their extensive applicability [2], [3], [4]. These methodologies are designed to extract essential information from high-dimensional datasets while maintaining the integrity of the underlying structures and patterns. By elucidating the low-rank structure of data, rank-revealing methods serve as powerful instruments for data analysis, compression, and recovery. Compared to traditional matrix factorization techniques [5], [6], which often entail significant computational costs, rank-revealing methods provide more efficient alternatives. For example, a pioneering low-rank subspace learning algorithm is introduced in [7], and an algorithm to calculate the decomposition of the symmetric rank revealed by a symmetric matrix $n \times n$ is detailed in [8]. These applications underscore the versatility and efficacy of rank-revealing methods in addressing intricate computational challenges.

As discussed previously, the issues of data sharing and security are inherent in institutional collaboration. Among the key privacy-preserving technologies, Differential Privacy (DP) [9], Federated Learning (FL) [10], Secure Multi-Party Computation (SMPC) [11], and Fully Homomorphic Encryption (FHE) [12] have found widespread application in domains such as recommendation systems, cloud computing, and the Internet of Things. However, in the context of dimensionality reduction, the errors introduced by differential privacy may lead to compromised computational accuracy [13]. Furthermore, federated learning necessitates extensive interaction, and data sparsity exacerbates communication overhead and computational complexity [14], [15]. Additionally, neither of these approaches is supported by cryptographic security proofs. The SMPC protocol is frequently employed in collaborative multi-party settings. However, ensuring participant data security makes the protocol design inherently complex. It also necessitates extensive interaction among participants for computation, which notably increases communication overhead, particularly with a large number of participants [16], [17].

A. Related Work

Fully homomorphic encryption (FHE), as an encryption scheme, allows data to remain encrypted during the computation process while ensuring that its computational results are consistent with those of plaintext computations. Therefore, it can effectively ensure the privacy and security of data during storage, transmission, and processing, ensuring that the integrity of computational results is not contingent upon

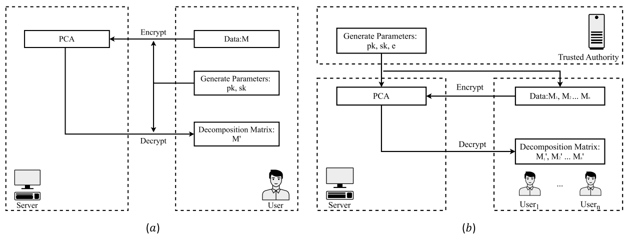

Fig. 1. Two common scenarios. (a) User and Computation Service Provider. (b) Multi-Users, Computation Service Provider, and Trusted Authority.

introducing noise or specific computation methods. In contrast to differential privacy, FHE obviates the necessity for noise-based privacy mechanisms, thereby facilitating computational results that are both accurate and robust against distortions arising from noise.

In recent decades, various methodologies have been developed for data dimensionality reduction using Federated Learning, Secure Multiparty Computation, and Fully Homomorphic Encryption (FHE). FHE schemes are frequently used in Scenario 1, as illustrated in Fig. 1a. The privacy-preserving Singular Value Decomposition (SVD) method proposed by [18] demonstrates limited applicability to real-world datasets. Subsequently, a privacy-preserving matrix factorization technique was implemented in recommendation systems [19]; however, its single iteration duration of approximately three hours poses significant challenges for practical implementation. In response to data leakage risks arising from collusion, [20] introduced a homomorphic encryption-based SVD method tailored for recommendation systems, drawing on insights from [21]. In these frameworks, users encrypt and transmit their data to a computation party, which returns the results for the user to decrypt. However, in practical scenarios, the high degree of data sparsity renders the encryption of the entire dataset computationally intensive and resource-demanding.

Existing Federated Learning often necessitates a trusted party to generate parameters, as depicted in Fig. 1b, ensuring that the computation party cannot access users' data. For example, a distributed SVD method based on federated power was proposed in [22]. The following year, the Federated Singular Vector Decomposition (FedSVD) method was introduced in [23], facilitating rapid federated singular value decomposition of high-dimensional datasets. Additionally, [24] presented a more efficient and secure version of FedSVD, building upon prior work [22], [25], [26]. Also, existing multi-party secure computing schemes require much interaction, resulting in extremely high communication overhead. For example, it is mentioned in the scheme proposed by [27] that for the dataset Lung [28], linear secret sharing-based methods require more than 60GB of communication overhead. Therefore, significant challenges remain, including eliminating reliance on fully trusted servers (or third parties), preventing collusion among multiple entities, and resolving issues related to low computational efficiency and excessive communication overhead.

B. Encrypted SVD: Challenges

Although various FHE-based dimensionality reduction schemes have been proposed, the fundamental computational constraints imposed by encrypted-domain operations remain inadequately addressed. The main algorithmic challenges of implementing Singular Value Decomposition (SVD) under homomorphic encryption include:

- **Computational Inefficiency:** SVD requires iterative procedures such as power iteration or QR decomposition, where each step involves multiple matrix–vector multiplications. In the encrypted setting, these translate into repeated ciphertext multiplications, rescaling, and rotations, which are orders of magnitude slower than plaintext. Bootstrapping can, in principle, reset the accumulated noise budget, but its latency (tens of seconds per call in typical CKKS software [29]) makes it impractical for iterative SVD. Hence, although noise growth can be managed, excessive runtime overhead renders efficiency the dominant challenge.
- **Error Accumulation:** SVD requires repeated normalization and orthogonalization, involving non-linear functions (e.g., square roots, divisions) that must be polynomially approximated in CKKS [30], [31], introducing uncontrolled approximation errors. In addition, CKKS inherently accumulates rounding noise with multiplicative depth [30], and prior encrypted PCA studies show that such errors quickly degrade numerical stability and singular vector orthogonality [32], [33]. Thus, maintaining accuracy is a fundamental difficulty.
- **Limited Scalability:** Encrypted SVD scales poorly with feature dimension. Although recent schemes attempt to reduce cost by formulating updates as matrix–vector multiplications [32], [33], these operations are still carried out between ciphertexts. As the feature dimension grows, the number of ciphertext–ciphertext multiplications and rotations increases proportionally, amplifying both runtime and approximation error. Prior analyses show that when feature counts reach the thousands, the combined overhead in communication and precision loss makes such approaches impractical [31], [34]. Scalability with dimensionality thus remains a central bottleneck.

C. Our Contributions

In response to the algorithmic challenges discussed in the previous subsection, we propose PP-DR, a two-party privacy-preserving protocol explicitly designed to overcome these limitations and achieve efficient, accurate dimensionality reduction. Specifically, our contributions consist of three components that resolve the identified challenges, together with one additional innovation that extends beyond prior approaches:

- **Efficiency and Trusted Third-Party Independence:** We propose PP-DR, a robust dimensionality reduction protocol that operates independently without any trusted third-party intervention. By exclusively performing *plaintext matrix–ciphertext vector* operations and carefully designed masking methods, PP-DR eliminates multiplicative depth growth, avoids bootstrapping, and reduces each iteration to lightweight rotations and plaintext multiplications.

- **Accuracy Preservation:** PP-DR introduces secure two-party protocols to realize nonlinear steps (e.g., reciprocal square root for normalization) without relying on polynomial approximations in the ciphertext domain. These protocols ensure numerical fidelity while keeping all intermediate values private. Furthermore, by shifting such nonlinear operations into interactive rounds, the protocol effectively refreshes ciphertexts and reduces multiplicative depth, thereby mitigating CKKS noise growth and cumulative precision loss.
- **Scalability with High-Dimensional Data:** PP-DR maintains practicality even when the feature dimension reaches several thousand. While prior methods must split high-dimensional vectors across multiple ciphertexts and perform ciphertext–ciphertext multiplications, our design keeps all updates in plaintext–ciphertext form. This reduces both the number of ciphertexts and the communication bandwidth, enabling near-linear runtime scaling with feature dimension.
- **Ciphertext-Based Termination Criterion:** We further propose a novel method to determine the retained rank directly from encrypted values, without decryption or prior negotiation. This ciphertext-only termination criterion avoids information leakage and eliminates the inefficiencies of fixed-rank assumptions, representing a unique advancement beyond existing encrypted SVD frameworks.

Overall, by integrating principles from Secure Multi-Party Computation (SMPC), we refine the Rank-revealing algorithm into a plaintext protocol optimized for multi-party computation scenarios. Subsequently, we apply the CKKS homomorphic encryption scheme to securely encrypt this plaintext protocol. Comprehensive experimental evaluations confirm that PP-DR significantly surpasses existing solutions regarding computational efficiency, reduced communication overhead, and improved accuracy.

This paper is organized as follows: Section II introduces the notation and the plaintext algorithm, while Section III delineates our operational scenario and the potential adversary model. The pivotal Section IV details the designed protocol and establishes its correctness, with a complete proof of the protocol's security provided in Section V. In Section VI, we present extensive experimental data. Finally, Section VII discusses potential improvements, culminating in the conclusion of our study presented in Section VIII.

II. Preliminaries

A. Notations

For all computation protocols, we adopt the following notations:

- Matrices with m rows and n columns are denoted by uppercase letters, e.g., $X^{(m\times n)}$, $Y^{(m\times n)}$.
- Column vectors of m elements are denoted by lowercase boldface letters, e.g., $\boldsymbol{u}^{(m\times 1)}$, $\boldsymbol{v}^{(m\times 1)}$.
- $(\cdot)^T$ denotes the transpose of a matrix or vector.
- $\|\cdot\|$ denotes the ℓ_2 norm of a vector.
- $\lceil x\rceil$ denotes the ceiling function, i.e., the smallest integer greater than or equal to x.
- Since this scheme adopts the CKKS fully homomorphic encryption scheme, pk and sk represent the public and secret keys in CKKS, respectively.
- $c_{(\cdot)}$ denotes the ciphertext of the corresponding plaintext variable, where the subscript identifies the encrypted plaintext; e.g., $c_{\boldsymbol{u}}$ represents the ciphertext of vector $\boldsymbol{u}$.
- For $\boldsymbol{a} = [a_1, a_2, \ldots, a_n]$ and $\boldsymbol{b} = [b_1, b_2, \ldots, b_n]$, the Hadamard product is defined as

$$\boldsymbol{a}\odot\boldsymbol{b} = [a_1\cdot b_1, a_2\cdot b_2, \ldots, a_n\cdot b_n]. \tag{1}$$

- The inverse of $\boldsymbol{b}$ is $\boldsymbol{b}^{-1} = [b_1^{-1}, b_2^{-1}, \ldots, b_n^{-1}]$, and the element-wise division is given by

$$\boldsymbol{a}\odot\boldsymbol{b}^{-1} = \left[\frac{a_1}{b_1}, \frac{a_2}{b_2}, \ldots, \frac{a_n}{b_n}\right]. \tag{2}$$

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

B. SVD and Rank-Revealing

SVD is commonly used in dimensionality reduction to map high-dimensional data into a low-dimensional space. Given a matrix $M \in \mathbb{R}^{m\times n}$ with the assumption of $m \geq n$, for SVD decomposition, we know that

$$M = U\Sigma V^T = \sigma_1\boldsymbol{u}_1\boldsymbol{v}_1^T + \sigma_2\boldsymbol{u}_2\boldsymbol{v}_2^T + \cdots + \sigma_n\boldsymbol{u}_n\boldsymbol{v}_n^T \tag{3}$$

where $U = [\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_n] \in \mathbb{R}^{m\times m}$ and $V = [\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_n] \in \mathbb{R}^{n\times n}$ are left and right orthogonal matrices. Also, $\Sigma = [\sigma_1, \sigma_2, \cdots, \sigma_n] \in \mathbb{R}^{m\times n}$ is a diagonal matrix with non-negative real numbers on the diagonal and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$.

According to the rank-revealing method [35], 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 $k \ll n$, also $\boldsymbol{u}_i^T M = \sigma_i\boldsymbol{v}_i^T$, we have

$$M_i = \boldsymbol{u}_1\boldsymbol{u}_1^T M + \boldsymbol{u}_2\boldsymbol{u}_2^T M + \cdots + \boldsymbol{u}_i\boldsymbol{u}_i^T M \tag{4}$$

First of all, we set $M_1 = M$ and σ_1 as 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 the largest singular value. Thus, we can determine the number of singular values greater than the threshold and the matrix's numerical rank.

As shown in Fig. 2, $U = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_k]$ is left orthogonal matrices, $V = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_k]$ is right orthogonal matrices. It is

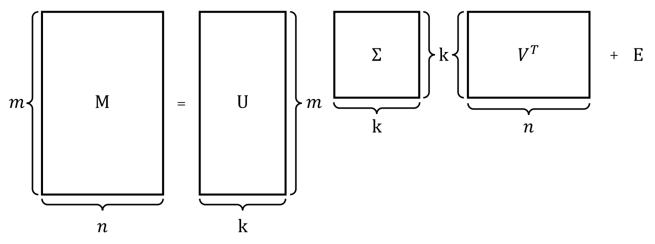

Fig. 2. Approximate singular value decomposition.

worth noting that the values of the matrix Σ on the diagonal $diag\{\sigma_1, \cdots, \sigma_k\}$ are 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, we rewrite Eq. (3)

$$M = M_k + E \tag{5}$$

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 the threshold θ.

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
```

To elucidate the method further, we present the original pseudocode: Larank from [35], as illustrated in Pseudocode 1. The input consists of an $m \times n$ matrix and a threshold θ, with the output being the numerical rank k of matrix M. Specifically, **Steps 7-10** assesses the convergence of the orthogonal vector, while **Steps 12-14** evaluates whether the current result satisfies the specified threshold. Ultimately, the algorithm produces a set of standard orthonormal basis vectors $\{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_k\}$ for matrix M. This orthonormal basis can subsequently be employed to reduce the dimensionality of the original dataset.

In Pseudocode 1, to improve the computational efficiency, it does not apply power iteration of $(M - M_i)(M - M_i)^T$ when calculating the remaining $k - 1$ singular vectors. According to the implicit iteration method mentioned in [35],

$$\begin{aligned}(M - M_i)^T\boldsymbol{y} &= M^T\boldsymbol{y}(1 - \boldsymbol{y}_1\boldsymbol{y}_1^T - \cdots - \boldsymbol{y}_i\boldsymbol{y}_i^T)\\ (M - M_i)\boldsymbol{x} &= M\boldsymbol{x}(1 - \boldsymbol{y}_1\boldsymbol{y}_1^T - \cdots - \boldsymbol{y}_i\boldsymbol{y}_i^T)\end{aligned} \tag{6}$$

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

C. Cheon-Kim-Kim-Song Cryptosystem

CKKS (Cheon-Kim-Kim-Song) is a Fully Homomorphic Encryption scheme explicitly designed for approximate arithmetic on real and complex numbers. It is particularly well-suited for machine learning, signal processing, and other tasks requiring operations on real-valued data, as it allows for efficient computation while managing noise growth in a controlled manner. Also, CKKS is based on Ring Learning with Errors (RLWE), a computational hardness assumption widely used in lattice-based cryptography. The core idea is to encode real numbers into polynomials and perform homomorphic encryption operations on these polynomials, which represent the underlying real data. The operations in some of the CKKS schemes we use are as follows:

- KeyGen(1^λ): It generates a secret key sk and the corresponding public key pk, where λ is security parameter.
- Enc$_{pk}(t)$: It returns a ciphertext c_t by encrypting plaintext t.
- Dec$_{sk}(c_t)$: It returns an approximate plaintext $t' \approx t$ by decrypting ciphertext c_t.
- EvalAdd(c_{t_1}, t_2): c_{t_3} with input as a ciphertext $c_{t_1} = Enc_{pk}(t_1)$ and a plaintext t_2, satisfied $Dec_{sk}(c_{t_3}) \approx t_1 + t_2$. For simplicity, we denote EvalAdd(c_{t_1}, t_2) by $c_{t_1} + t_2$.
- EvalMult(c_{t_1}, t_2): c_{t_3} with input as a ciphertext $c_{t_1} = Enc_{pk}(t_1)$ and a plaintext t_2, satisfied $Dec_{sk}(c_{t_3}) \approx t_1 \cdot t_2$. For simplicity, we denote EvalMult(c_{t_1}, t_2) by $c_{t_1} \cdot t_2$., where $[\cdot]$ is inner product.

III. Scenario and Adversary Model

This section will explain the protocol usage scenarios, data partitioning, and potential adversary models.

A. Scenario

As shown in Fig. 3, our scenario involves two parties, A and B, each possessing $m \times n$ dimensional data matrices, denoted as $X^{(m\times n)}$ and $Y^{(m\times n)}$, respectively. These two data owners aim to improve the precision of dimensionality reduction by jointly integrating their respective data samples.

Within our framework, Party A generates the necessary public-private key pair pk, sk using the CKKS scheme and encrypts the vector that needs to be transmitted to Party B using the public key. Since Party A has the private key sk, to ensure Party B's data security, it is necessary to add another masking to the encrypted result vector passed to Party A. The specific implementation of this masking will be detailed in Section VI-C. During the calculation of each orthogonal basis, both parties only share the final converged orthogonal basis results to ensure data security. After several iterations,

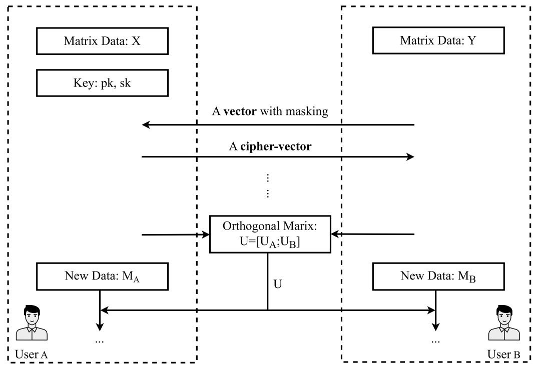

Fig. 3. Two-party Scenario. Each Party holds a $m \times n$ matrix as input.

each party obtains its 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 these matrices, they obtain the final orthogonal matrix $U = [U_a; U_b]$, which is then used for the dimensionality reduction of new data from both parties. Notably, the entire process operates without the involvement of any trusted third party or external computing entity. The security proof of the whole scenario is mainly based on the real/ideal simulation paradigm [16], [36], [37], [38], and the detailed proofs of correctness and security are shown in Section IV and Section V.

B. Adversary Model

Throughout the computational protocol, we operate under the assumption of a semi-honest adversarial model, where Party A, responsible for generating the public key, may act curiously about Party B's original data without deviating from the protocol, attempting to infer information from the received data [16]. While Party B does not transmit its original data, it retains its confidentiality using the public key. Since both parties must share the final computational results, there remains a potential risk of data leakage for Party B. To mitigate this risk, we introduce perturbations to the data that Party B transmits to Party A. The security of this approach is formally proven under the smudge lemma [36] and within the semi-honest model [16].

IV. PP-DR Protocol Design

As detailed in Pseudocode 1, **Steps 4-14** involve numerous normalization and modulus operations to ensure computational accuracy, alongside the evaluation of vector convergence and comparison with a threshold to control algorithm termination. We have restructured the entire protocol to maintain precision while avoiding resource-intensive operations such as "modulus" and "comparison" on the ciphertext. By minimizing the interactions required to bypass these operations under ciphertext, we enhance computational efficiency and significantly reduce the overall communication overhead by streamlining the interaction content. Additionally, we have divided the algorithm into two distinct phases to optimize efficiency further. The first phase focuses exclusively on computing the initial eigenvector using the standard power iteration method, and the second phase modifies the power iteration process based on Eq. (6) to calculate the remaining eigenvectors.

A. Several Utilized Methods

To describe the protocol execution more effectively, we outline several utilized methods: Initial-Vector Method (M1), Secure-Addition Method (M2), Secure-Multiplication Method (M3), and Convergence Method (M4).

1) Initial-Vector Method (M1): Based on the dimension of the data $M^{(m\times n)}$, the given initial vector dimension is m. To ensure the accuracy of the calculation results, the initial vectors of both parties need to satisfy: $\boldsymbol{v}^{(2m\times 1)} = [\boldsymbol{v}_a^{(m\times 1)}; \boldsymbol{v}_b^{(m\times 1)}]$ with the constraint that $\|\boldsymbol{v}\| = 1$, which implies $\|\boldsymbol{v}_a\| = \|\boldsymbol{v}_b\| = \frac{\sqrt{2}}{2}$.

M 1 Initial-Vector Method
```
    Input: $m$
    Output: $\boldsymbol{v}^{(m\times 1)}$
1   generate random vector: $\boldsymbol{v}^{(m\times 1)} \in \mathbb{R}^{m\times 1}$
2   normalization: $\frac{\boldsymbol{v}^{(m\times 1)}}{\|\boldsymbol{v}^{(m\times 1)}\|}$
3   $\boldsymbol{v}^{(m\times 1)} = \frac{\sqrt{2}}{2} \cdot \boldsymbol{v}^{(m\times 1)}$
```

In our model, Party A is assumed to generate a public-private key pair (pk, sk). Therefore, when Party B sends interim computational results to Party A, they can be decrypted with the secret key sk, potentially risking Party B's data confidentiality. To counter this, we propose two masking strategies (M2) and (M3) to protect Party B's data.

2) Secure-Addition Method (M2): This method secures data by masking it before Party B transmits it to Party A and then unmasking it post-decryption to maintain accuracy. We introduce a random perturbation $\boldsymbol{v}^{(m\times 1)} \leftarrow \mathcal{Q}^{m\times 1}$, where $\mathcal{Q} = [-2^l, 2^l]^m$, and the influence of the value of l on the protocol will be discussed in Section VI-B1

M 2 Secure-Addition Method
```
    Input B: ciphertext $c_t \leftarrow Enc_{pk}(\boldsymbol{t})$
    Output B: plaintext $\boldsymbol{t}$
1   B: $c'_t = c_t + \boldsymbol{v}$ / /$\boldsymbol{v}^{(m\times 1)} \leftarrow \mathcal{Q}^{(m\times 1)} = [-2^l, 2^l]^m$
2   B: send $c'_t$ to A
3   A: $\boldsymbol{t}' = Dec_{sk}(c'_t)$
4   A: send $\boldsymbol{t}'$ to B
5   B: $\boldsymbol{t} = \boldsymbol{t}' - \boldsymbol{v}$
```

3) Secure-Multiplication Method (M3): This method introduces masks through the multiplication of corresponding elements, ensuring the security of the data. And through Eq. (2), the masks are removed, thereby ensuring the accuracy of the data.

4) Convergence Method (M4): This method primarily assesses the convergence of calculation results to decide when to terminate the iteration. We assess the convergence of the results by comparing the difference between the outcomes of two successive iterations against the machine-calculated precision (ϵ) or a precision specified ($itol$) by the Party. Let $\boldsymbol{v}_i$ denote the result after the i-th iteration calculation.

M 3 Secure-Multiplication Method

```
Input B: ciphertext $c_t \leftarrow Enc_{pk}(\boldsymbol{t})$
Output B: plaintext $\boldsymbol{t}$
1 B: $c'_t = c_t \odot \boldsymbol{v}$ / / $\boldsymbol{v}^{(m\times 1)} \leftarrow \mathcal{Q}^{(m\times 1)} = [-2^l, 2^l]^m$
2 B: send $c'_t$ to A
3 A: $\boldsymbol{t}' = Dec_{sk}(c'_t)$
4 A: send $\boldsymbol{t}'$ to B
5 B: $\boldsymbol{t} = \boldsymbol{t}' \odot \boldsymbol{v}^{-1}$
```

M 4 Convergence

```
Input  : $\boldsymbol{v}_i, \boldsymbol{v}_{i-1}, \epsilon, itol$
Output: $Bool$
1  $diff = \|\boldsymbol{v}_i - \boldsymbol{v}_{i-1}\|$
2  if $diff < \epsilon$ then
3      $Bool = True$
4  else
5      if $diff < itol$ then
6          $Bool = True$
7      else
8          $Bool = False$
9      end
10 end
```

B. Protocol 1: PP-DR

We designate the entire computational protocol as PP-DR, with detailed specifications provided in Protocol 1. In this context, matrices $X^{(m\times n)}$ and $Y^{(m\times n)}$ represent the original data of both parties, θ denotes the proportion of noise information, and *Ranks* indicates the maximum number of ranks that both parties aim to retain. Upon executing Protocol 1, Party A initializes all requisite encryption parameters, such as *pk* and *sk*. Subsequently, Party A and Party B execute Protocol 2 to derive the first eigenvector $U = [U_a; U_b] = [\bar{\boldsymbol{u}}_{a1}; \bar{\boldsymbol{u}}_{b1}]$. This is followed by comparing the calculated rank count with k to determine whether to proceed with Protocol 3. Finally, Protocol 4 is executed to ascertain the completion of

Protocol 1 PP-DR

```
Input A  : $X^{(m\times n)}$
Input B  : $Y^{(m\times n)}$, $\theta$, $Ranks$
Output B: $U^{(2m\times k)}$
1  A: $pk, sk \leftarrow$ KeyGen$(1^\lambda)$
2  A/B: $\bar{\boldsymbol{u}}_{a1}, \bar{\boldsymbol{u}}_{b1} \leftarrow$ Running Protocol 2: POB
3  B: $k = 1$
4  while $k < Ranks$ do
5      A/B: $\bar{\boldsymbol{u}}_{ak}, \bar{\boldsymbol{u}}_{bk} \leftarrow$ Running Protocol 3: SPOB
6      A/B: Bool $\leftarrow$ Running Protocol 4: SC
7      if $Bool$ = $True$ then
8          break the while loop
9      else
10         B: $U_a = [\bar{\boldsymbol{u}}_{a1}, \cdots, \bar{\boldsymbol{u}}_{ak}]$, $U_b = [\bar{\boldsymbol{u}}_{b1}, \cdots, \bar{\boldsymbol{u}}_{bk}]$
11         B: $U = [U_a; U_b]$
12         $k = k + 1$
13     end
14 end
```

Protocol 2 Principal Orthogonal Basis (POB)

```
Input A  : $X^{(m\times n)}$, $pk$, $sk$
Input B  : $Y^{(m\times n)}$
Output A: $U_a^{(m\times 1)}$
Output B: $U_b^{(m\times 1)}$
1  A/B: $\boldsymbol{u}_{a1}^{(m\times 1)}, \boldsymbol{u}_{b1}^{(m\times 1)} \leftarrow$ M1 $(m)$
2  for $i \leftarrow 1$ to $t$ do        // $t$ is max iterations
3      B: send $\boldsymbol{y}_i = Y^T \boldsymbol{u}_{bi}$ to A
4      A: $\boldsymbol{x}_i = X^T \boldsymbol{u}_{ai}$, $\boldsymbol{u}'_i = \frac{\boldsymbol{x}_i + \boldsymbol{y}_i}{\|\boldsymbol{x}_i + \boldsymbol{y}_i\|}$, $\bar{\boldsymbol{u}}_{ai} = X \cdot \boldsymbol{u}'_i$
5      A: $c_{\boldsymbol{u}'_i} = Enc_{pk}(\boldsymbol{u}'_i)$
6      if $i = 1$ then
7          A: $c_{X\boldsymbol{y}_1} = Enc_{pk}(X \cdot \boldsymbol{y}_1)$
8          A: send $c_{X\boldsymbol{y}_1}$ to B
9      end
10     A: send $\|\bar{\boldsymbol{u}}_{ai}\|, c_{\boldsymbol{u}'_i}$ to B
11     B: $\bar{\boldsymbol{u}}_{bi} \leftarrow$ M2 $(c_{\boldsymbol{u}'_i})$ or M3 $(c_{\boldsymbol{u}'_i})$
12     B: send $\|\bar{\boldsymbol{u}}_{bi}\|$ to A
13     A/B: $\boldsymbol{u}_{ai}, \boldsymbol{u}_{bi} = \frac{\bar{\boldsymbol{u}}_{ai}}{\sqrt{\|\bar{\boldsymbol{u}}_{ai}\|^2 + \|\bar{\boldsymbol{u}}_{bi}\|^2}}, \frac{\bar{\boldsymbol{u}}_{bi}}{\sqrt{\|\bar{\boldsymbol{u}}_{ai}\|^2 + \|\bar{\boldsymbol{u}}_{bi}\|^2}}$
14     A/B: Bool $\leftarrow$ M4 $(\boldsymbol{v}_i, \boldsymbol{v}_{i-1}, \epsilon, itol)$
15     A/B: if $Bool = True$ then
16         A/B: $\bar{\boldsymbol{u}}_{a1} = \boldsymbol{u}_{ai}, \bar{\boldsymbol{u}}_{b1} = \boldsymbol{u}_{bi}$
17         A/B: $U_a = [\bar{\boldsymbol{u}}_{a1}], U_b = [\bar{\boldsymbol{u}}_{b1}]$
18         break the $i$-loop
19     else
20         continue
21     end
22 end
```

Protocol 1 and to provide the final result $U = [U_a; U_b] = [\bar{\boldsymbol{u}}_{a1}; \bar{\boldsymbol{u}}_{b1}, \cdots, \bar{\boldsymbol{u}}_{ak}; \bar{\boldsymbol{u}}_{bk}]$.

Theorem 1: (Correctness and Security of PP-DR) Let $X \in \mathbb{R}^{m\times n}$ be Party A's input, and let $Y \in \mathbb{R}^{m\times n}$ together with parameters θ and k be Party B's input. Protocol 1 outputs $U \in \mathbb{R}^{2m\times k}$ to Party B, and Party B then sends U to Party A. Upon completion of Protocol 1, U is an orthogonal matrix consisting of approximate eigenvectors of the combined matrix $M \in \mathbb{R}^{2m\times n} = [X; Y]$.

Moreover, under the IND-CPA security of CKKS (based on the RLWE assumption), the proposed protocol satisfies semantic security in the semi-honest model, in the sense of the standard real/ideal simulation paradigm (see Definition 1 in Section V).

C. Protocol 2: Principal Orthogonal Basis.(POB)

In this protocol, Party A and Party B each hold data $X^{(m\times n)}$ and $Y^{(m\times n)}$, respectively. Party A generates a public-private key pair: pk, sk. Among them, **Step 1**: Party A and Party B generate initialization vectors through M1. Here, Party A only observes intermediate vectors of the form $Y^\top u_{bi}$ generated during the power iteration used to approximate the dominant singular vector. The max number of iterations t is typically much smaller than the data dimension m. Consequently, Party A's view is restricted to a short sequence of correlated projections that characterize only the top eigen-space, which is information-theoretically insufficient for reconstructing the

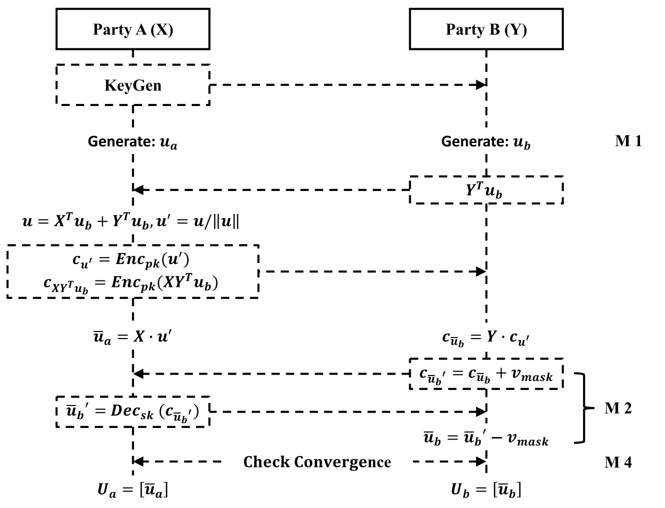
Party A (X)
Party B (Y)
KeyGen
Generate: u_a
Generate: u_b
M 1
$Y^T u_b$
$u = X^T u_b + Y^T u_b, u' = u/\|u\|$
$c_{u'} = Enc_{pk}(u')$
$c_{XY^T u_b} = Enc_{pk}(XY^T u_b)$
$\bar{u}_a = X \cdot u'$
$c_{\bar{u}_b} = Y \cdot c_{u'}$
$c_{\bar{u}_b'} = c_{\bar{u}_b} + v_{mask}$
$\bar{u}_b' = Dec_{sk}(c_{\bar{u}_b'})$
M 2
$\bar{u}_b = \bar{u}_b' - v_{mask}$
Check Convergence
M 4
$U_a = [\bar{u}_a]$
$U_b = [\bar{u}_b]$

Fig. 4. Data exchange between A and B and their respective responsibilities in Protocol. 2.

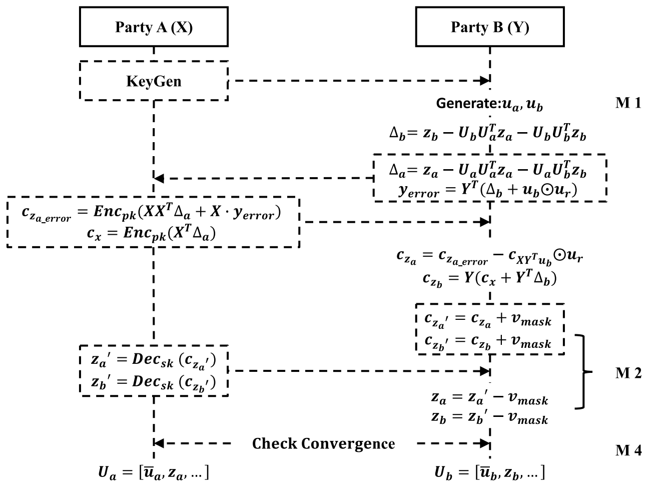
Party A (X)
Party B (Y)
KeyGen
Generate:u_a, u_b
M 1
$\Delta_b = z_b - U_b U_a^T z_a - U_b U_b^T z_b$
$\Delta_a = z_a - U_a U_a^T z_a - U_a U_b^T z_b$
$y_{error} = Y^T(\Delta_b + u_b \odot u_r)$
$c_{z_{a\_error}} = Enc_{pk}(XX^T\Delta_a + X \cdot y_{error})$
$c_x = Enc_{pk}(X^T\Delta_a)$
$c_{z_a} = c_{z_{a\_error}} - c_{XY^T u_b} \odot u_r$
$c_{z_b} = Y(c_x + Y^T\Delta_b)$
$c_{z_a'} = c_{z_a} + v_{mask}$
$c_{z_b'} = c_{z_b} + v_{mask}$
$z_a' = Dec_{sk}(c_{z_a'})$
$z_b' = Dec_{sk}(c_{z_b'})$
M 2
$z_a = z_a' - v_{mask}$
$z_b = z_b' - v_{mask}$
Check Convergence
M 4
$U_a = [\bar{u}_a, z_a, ...]$
$U_b = [\bar{u}_b, z_b, ...]$

Fig. 5. Information exchange and responsibility delegation between Party A and Party B during one iteration of Protocol. 3.

full matrix Y. Under the semi-honest model, where both parties follow the protocol, this ensures that Y remains hidden. **Steps 6-9**: To protect Party B's data in Protocol. 3 with a removable special perturbation. **Step 11**: Party B obtains the accurate plaintext data by M2 and M3, the influence of the two methods on the efficiency and accuracy of the protocol will be discussed in Section VI-C. **Steps 14-21**: Both Party A and Party B judge the convergence of the calculation results through M5. If the results of both parties have converged, the results are output. Furthermore, it is noteworthy that throughout the entire protocol, the multiplication of ciphertexts occurs only once, resulting in a multiplication depth of one. This characteristic significantly enhances computational efficiency, as demonstrated in Section VI.

To clarify the information exchange in Protocol. 2 (POB), we include an interaction flow diagram as Fig. 4. This diagram depicts each protocol step and shows the sequence of data exchanged between Party A and Party B.

1) Correctness: For an arbitrary matrix $M \in \mathbb{R}^{2m\times n}$ and an initial random vector $\boldsymbol{u} \in \mathbb{R}^{2m\times 1}$, the principal eigenvector

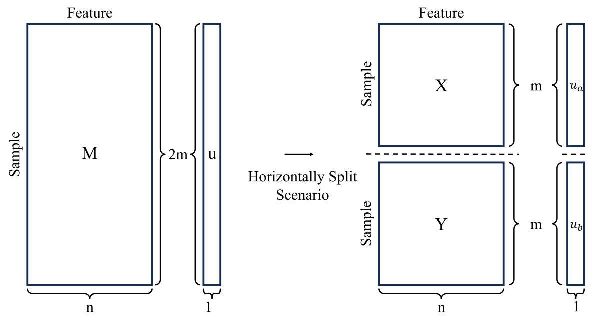

Fig. 6. Experiment data settings.

$\bar{\boldsymbol{u}}$ is finally obtained by using the power method iteration on MM^T. Then, for our protocol, we divide the original matrix M into two matrices $X \in \mathbb{R}^{m\times n}$, $Y \in \mathbb{R}^{m\times n}$ and the initial random vector $\boldsymbol{u}$ into two column vectors $\boldsymbol{u}_a \in \mathbb{R}^{m\times 1}$, $\boldsymbol{u}_b \in \mathbb{R}^{m\times 1}$, which is shown in Fig. 6. Then we have:

$$\begin{pmatrix}\bar{\boldsymbol{u}}_a\\ \bar{\boldsymbol{u}}_b\end{pmatrix} = \begin{pmatrix}X\\ Y\end{pmatrix}\begin{pmatrix}X^T & Y^T\end{pmatrix}\begin{pmatrix}\boldsymbol{u}_a\\ \boldsymbol{u}_b\end{pmatrix} \tag{7}$$

Each party only needs to calculate their parts separately:

$$\begin{aligned}\bar{\boldsymbol{u}}_a &= XX^T\boldsymbol{u}_a + XY^T\boldsymbol{u}_b\\ \bar{\boldsymbol{u}}_b &= YX^T\boldsymbol{u}_a + YY^T\boldsymbol{u}_b\end{aligned} \tag{8}$$

Consequently, the result $\bar{\boldsymbol{u}} = [\bar{\boldsymbol{u}}_a; \bar{\boldsymbol{u}}_b]$ is the principal eigenvector.

D. Protocol 3: Sub-Principal Orthogonal Bases

As previously stated, to enhance computational efficiency, we reformulate the calculation outlined in Protocol 2 following Eq. (6), as presented in Protocol 3. The matrices $X^{(m\times n)}$ and $Y^{(m\times n)}$ represent the datasets of each party, c_{Xy_1} is a removable perturbation used to protect the data security of Party B. **Step 1**: Unlike Protocol 2, to ensure the data security of Party B, the generation of the initial vector can only be done by Party B. **Steps 5-12**: The ciphertext calculates orthogonal vectors. It is worth noting that a random perturbation is introduced here for the purpose of protecting the data security of Party B. At the same time, through the variable c_{Xy_1} generated in Protocol 2, the perturbation is eliminated, thus ensuring both the security of Party B's data and the accuracy of the calculation. **Step 14**: An orthogonal vector is obtained securely and accurately through M2 or M3. **Step 16**: Party B determines the convergence of the obtained orthogonal vector through M4. If converged, the iteration is terminated early; otherwise, the iterative operation continues.

To further clarify the detailed interactions and information flow of the Sub-Principal Orthogonal Basis (SPOB) protocol (Protocol 3), we present a step-by-step interaction diagram as shown in Fig. 5. The figure explicitly depicts the distinct roles and data-handling responsibilities of Parties A and B during each iteration. Compared to the POB protocol, it is clearly illustrated that Party B must generate all the initial vectors. If both parties were to generate initial vectors independently, Party A could potentially deduce Δ_a from Δ_b, consequently compromising Party B's data privacy. Furthermore, the figure distinctly shows that Party A can only

Protocol 3 Sub-Principal Orthogonal Basis (SPOB)

```
Input A  : $X^{(m\times n)}$, $U_a^{(m\times k)}$, $pk$, $sk$
Input B  : $Y^{(m\times n)}$, $U_b^{(m\times k)}$, $c_{X\boldsymbol{y}_1}$
Output B: $U_a^{m\times(k+1)}$, $U_b^{m\times(k+1)}$
B: $\boldsymbol{z}_{a1}, \boldsymbol{z}_{b1} \leftarrow$ M1 (m)
for $j \leftarrow 1$ to $t$ do        // $t$ is max iterations
    B: $\boldsymbol{\Delta}_{aj} = \boldsymbol{z}_{aj} - U_aU_a^T\boldsymbol{z}_{aj} - U_aU_b^T\boldsymbol{z}_{bj}$
    B: $\boldsymbol{\Delta}_{bj} = \boldsymbol{z}_{bj} - U_bU_a^T\boldsymbol{z}_{aj} - U_bU_b^T\boldsymbol{z}_{bj}$
    B: $\boldsymbol{y}_{j\_error} = Y^T(\boldsymbol{\Delta}_{bj} + \boldsymbol{u}_{b1} \odot \boldsymbol{u}_r)$ // $\boldsymbol{u}_r \leftarrow \mathcal{Q}^m$
    B: send $\boldsymbol{\Delta}_{aj}$ and $\boldsymbol{y}_{j\_error}$ to A
    A: $\boldsymbol{x}_j = X^T\boldsymbol{\Delta_{aj}}$
    A: $\boldsymbol{z}_{aj\_error} = X \cdot \boldsymbol{x}_j + X \cdot \boldsymbol{y}_{j\_error}$
    A: $c_{\boldsymbol{z}_{aj\_error}} = Enc_{pk}(\boldsymbol{z}_{aj\_error})$
    A: $c_{\boldsymbol{x}_j} = Enc_{pk}(\boldsymbol{x}_j)$
    A: send $c_{\boldsymbol{z}_{aj\_error}}$, $c_{\boldsymbol{x}_j}$ to B
    B: $c_{\boldsymbol{z}_{aj}} = c_{\boldsymbol{z}_{aj\_error}} - c_{X\boldsymbol{y}_1} \odot \boldsymbol{u}_r$
    B: $c_{\boldsymbol{z}_{bj}} = Y \cdot c_{\boldsymbol{x}_j} + YY^T\boldsymbol{\Delta}_{bj}$
    B: $\boldsymbol{z}_{aj}, \boldsymbol{z}_{bj} \leftarrow$ M2 $(c_{\boldsymbol{z}_{aj}}, c_{\boldsymbol{z}_{bj}})$ or M3 $(c_{\boldsymbol{z}_{aj}}, c_{\boldsymbol{z}_{bj}})$
    B: $\boldsymbol{z}_{aj} = \frac{\boldsymbol{z}_{aj}}{\|\boldsymbol{z}_{aj}\|}$, $\boldsymbol{z}_{bj} = \frac{\boldsymbol{z}_{bj}}{\|\boldsymbol{z}_{bj}\|}$
    B: Bool $\leftarrow$ M4 $(\boldsymbol{z}_{aj}, \boldsymbol{z}_{aj-1}, \boldsymbol{z}_{bj}, \boldsymbol{z}_{bj-1}, , \epsilon, itol)$
    B: if $Bool = True$ then
        break the $j$-loop
    end
end
B: $\bar{\boldsymbol{u}}_a = \frac{\Delta_{aj}}{\|\Delta_{aj}\|}$, $\bar{\boldsymbol{u}}_b = \frac{\Delta_{bj}}{\|\Delta_{bj}\|}$
B: $U_a = [U_a, \bar{\boldsymbol{u}}_a]$, $U_b = [U_b, \bar{\boldsymbol{u}}_b]$
```

access plaintext results containing masking, while Party B is capable of removing the mask to retrieve accurate results. This interaction design ensures both the confidentiality of the participating parties' data and the correctness of the computed results.

1) Correctness: By incorporating the previously discussed concepts of rank-revealing techniques, the subsequent singular vectors are computed as illustrated in Eq. (9).

$$\boldsymbol{u} = MM^T(z - UU^Tz) \tag{9}$$

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

As in Protocol. 2, we horizontally split the matrix and vectors, and we get:

$$\begin{pmatrix}\bar{\boldsymbol{u}}_a\\ \bar{\boldsymbol{u}}_b\end{pmatrix} = \begin{pmatrix}X\\ Y\end{pmatrix}\begin{pmatrix}X^T & Y^T\end{pmatrix}\left[\begin{pmatrix}z_a\\ z_b\end{pmatrix} - \begin{pmatrix}U_a\\ U_b\end{pmatrix}\begin{pmatrix}U_a^T & U_a^T\end{pmatrix}\begin{pmatrix}z_a\\ z_b\end{pmatrix}\right] \tag{10}$$

To simplify the formula, we let

$$\begin{aligned}\boldsymbol{\Delta}_a &= z_a - U_aU_a^Tz_a - U_aU_b^Tz_b\\ \boldsymbol{\Delta}_b &= z_b - U_bU_a^Tz_a - U_bU_b^Tz_b\end{aligned} \tag{11}$$

and then, we get:

$$\begin{aligned}\bar{\boldsymbol{u}}_a &= XX^T\boldsymbol{\Delta}_a + XY^T\boldsymbol{\Delta}_b\\ \bar{\boldsymbol{u}}_b &= YX^T\boldsymbol{\Delta}_a + YY^T\boldsymbol{\Delta}_b\end{aligned} \tag{12}$$

Therefore, as with Protocol 2, $\bar{\boldsymbol{u}} = [\bar{\boldsymbol{u}}_a; \bar{\boldsymbol{u}}_b]$. We assume that the number of ranks is k, then $\bar{\boldsymbol{u}}_k = [\bar{\boldsymbol{u}}_{ak}; \bar{\boldsymbol{u}}_{bk}]$, $U = [\bar{\boldsymbol{u}}_1, \bar{\boldsymbol{u}}_2, \cdots, \bar{\boldsymbol{u}}_k]$.

For two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ with identical dimensions, the commutative nature of the Hadamard product implies $\boldsymbol{a}\odot\boldsymbol{b} = \boldsymbol{b}\odot\boldsymbol{a}$. Thus, $c_{XY^T(\boldsymbol{u}_r\odot\boldsymbol{u}_{b1})} = \boldsymbol{u}_r \odot c_{XY^T\boldsymbol{u}_{b1}}$, and we have

$$c_{z_{aj}} = c_{XX^T\boldsymbol{\Delta}_{aj}} + c_{XY^T\boldsymbol{\Delta}_{bj}} \tag{13}$$

Therefore, according to Eq. (13), we realize the operation of adding and deleting perturbations to the data on the ciphertext, thus ensuring the security and accuracy of the data.

E. Protocol 4: Stop Criteria

Given the inherent complexity of ciphertext comparison in fully homomorphic encryption (FHE) and the substantial computational and communication overhead it entails, traditional approaches to dimensionality reduction typically fall into two categories: 1) complete decomposition of the data. 2) Pre-negotiation of the number of features to retain. However, as the dimensionality of the data increases, the computational efficiency of complete decomposition degrades significantly. Likewise, pre-negotiating the number of retained features often results in inaccurate dimensionality reduction. As a result, it becomes necessary to compute results for varying numbers of eigenvalues, further exacerbating the computational burden. To address these challenges, we introduce the key component of this protocol 4: the stop criterion. In this protocol, the two parties first negotiate the proportion θ of information to be discarded. Following this, they compute the square of the largest eigenvalue, denoted as $c_{\sigma^2_{max}}$(the ciphertext of σ^2_{max}, following Eq. (14), and establish the termination threshold as $\theta^2 \cdot \sigma^2_{max}$. It is important to highlight that, during actual computations, the maximum singular value (σ_{max}) corresponds to the first singular value (σ_1) of the data. Equation (14) outlines how both parties calculate the singular values collaboratively.

$$\begin{aligned}\sigma^2 &= \|X^T\boldsymbol{u}_a + Y^T\boldsymbol{u}_b\|^2\\ &= (X^T\boldsymbol{u}_a + Y^T\boldsymbol{u}_b)^T(X^T\boldsymbol{u}_a + Y^T\boldsymbol{u}_b)\\ &= \boldsymbol{u}_a^TXX^T\boldsymbol{u}_a + 2(\boldsymbol{u}_b^TYX^T\boldsymbol{u}_a) + \boldsymbol{u}_b^TYY^T\boldsymbol{u}_b\end{aligned} \tag{14}$$

Building on the methods outlined in Eq. (14) and Eq. (15), we devise protocol 4 to define the termination condition. To maintain security and prevent leakage of sensitive information, the protocol ensures that Party B shares only the product of the difference between the current singular value and the maximum singular value, masked by a random perturbation $p \leftarrow \mathbb{P}$, where $\mathbb{P} = [0, 2^l]$. This guarantees that no additional information about the individual singular values is disclosed to the other party.

$$\begin{aligned}\sigma &< 0.1 \cdot \sigma_{max}\\ \sigma^2 &< 0.01 \cdot \sigma^2_{max}\\ p \cdot (\sigma^2 - 0.01 \cdot \sigma^2_{max}) &< 0\end{aligned} \tag{15}$$

Protocol 4 Stop Criteria (SC)

```
Input A  : $X^{(m\times n)}$, $pk$, $sk$
Input B  : $Y^{(m\times n)}$, $\bar{\boldsymbol{u}}_a^{(m\times 1)}$, $\bar{\boldsymbol{u}}_b^{(m\times 1)}$
Output A: $Bool$
1  A: $\sigma_a = \bar{\boldsymbol{u}}_a^T X X^T \bar{\boldsymbol{u}}_a$, $\boldsymbol{x} = X^T \bar{\boldsymbol{u}}_a$
2  A: $c_{\sigma_a} = Enc_{pk}(\sigma_a)$
3  A: $c_{\boldsymbol{x}} = Enc_{pk}(\boldsymbol{x})$
4  A: send $c_{\sigma_a}$, $c_{\boldsymbol{x}}$ to B
5  B: $\sigma_b = \bar{\boldsymbol{u}}_b^T Y Y^T \bar{\boldsymbol{u}}_b$
6  B: $c_{\sigma^2} = c_{\sigma_a} + 2(\bar{\boldsymbol{u}}_b^T Y \cdot c_{\boldsymbol{x}}) + \sigma_b$
7  B: $c_{\sigma_{res}} = p \cdot (c_{\sigma^2} - \theta^2 \cdot c_{\sigma^2_{max}})$      // $p \leftarrow \mathbb{P}$
8  B: send $c_{\sigma_{res}}$ to A
9  A: $\sigma_{res} = Dec_{sk}(c_{\sigma_{res}})$
10 A: if $\sigma_{res} < 0$ then
11     return True
12 else
13     return False
14 end
```

Through the method demonstrated in Eq. (15), we can reliably determine whether the singular values have met the specified threshold while preserving the confidentiality of both parties' data. For instance, when the user sets the stop criterion at $\theta = 10\%$, the protocol halts if the most recently computed singular value falls below 10% of the maximum singular value. In this context, p represents a random positive number introduced to ensure data security, with the detailed security analysis provided in Section V-C.

Across Protocols 2-4, the ciphertext multiplication depth does not exceed three. Consequently, when selecting ciphertext parameters, we can set the modulus of the polynomial coefficients (N) to 8192, significantly enhancing computational efficiency. A comprehensive discussion of parameter selection and experimental results is provided in Section VI.

V. Security Analysis

This section conducts a thorough security analysis of the designed computational protocol. Our system involves only two participants: Party A and Party B. All operations are based on the semi-honest adversarial model defined in [16], where participants adhere to the protocol specifications but remain curious about the other party's data. To establish the security of the entire protocol, we first demonstrate the security of Protocols 2-4 using Definition 1 and Lemma 1.

According to the framework for secure protocols under semi-honest assumptions outlined in Definition 7.2.1 in [16], let $f(X, Y)$ represent the computational function executed by both parties($f_{a/b}(X, Y)$ denote the Party A/B of $f(X, Y)$), where X, Y are the respective inputs of Party A and Party B, and Π denotes the protocol they use to compute f jointly. Furthermore, let $view^{\Pi}(X, Y)$ denote the information each party receives while executing the Π protocol, excluding the results they compute.

In this work, we focus solely on deterministic functions f and simplify the analysis by referencing the deterministic case in [16]. When f is a deterministic functionality, we assert that protocol Π is a secure computation if it satisfies the conditions of Definition 1.

Definition 1: (privacy with respect to semi-honest behavior) 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.

$$\begin{aligned} \{S_a(X, f_a(X, Y))\} &\overset{c}{\equiv} \{view_a^{\Pi}(\lambda, X, Y)\} \\ \{S_b(Y, f_b(X, Y))\} &\overset{c}{\equiv} \{view_b^{\Pi}(\lambda, X, Y)\} \end{aligned} \tag{16}$$

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

Furthermore, we introduce the Smudging lemma [36] to ensure the data security of Party B.

Lemma 1: (Smudging) For the distributions $\mathcal{P}_1 = [-B_1, B_1]$ and $\mathcal{P}_2 = [-B_2, B_2]$, where B_1, B_2 are positive integers. Let $e_1 \leftarrow \mathcal{P}_1$ and $e_2 \leftarrow \mathcal{P}_2$. As long as B_1/B_2 is a negligible function, the distribution of e_2 is statistically indistinguishable from that of $e_2 + e_1$.

A. Security of Protocol 2

To demonstrate the protocol's security, we need to construct a simulator first. The protocol is secure if we can prove that the protocol is computationally indistinguishable between the view and the simulator.

A's view is $V_a = (I_a, r, m_a)$, where $I_a = (X^{(m\times n)}, pk, sk)$ are the inputs of A, r represents the outcome of A's internal coin tosses, $m_a = (\boldsymbol{y}_i, c_{\bar{\boldsymbol{u}}_{bi\_error}}, \|\bar{\boldsymbol{u}}_{bi}\|)$ are all the information obtained by A during the execution of the protocol. Also, to better illustrate the security, we assume that the initial random vector $\boldsymbol{u}_{b1}^{(m\times 1)} \leftarrow \mathbb{O}^{m\times 1}$ satisfies $\|\boldsymbol{u}_{b1}\| = \frac{\sqrt{2}}{2}$. Let the original data $X^{(m\times n)}, Y^{(m\times n)} \leftarrow \mathbb{O}^{m\times n}$, where $\mathbb{O}$ is selected according to the actual application scenario.

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

1) $\widetilde{\boldsymbol{y}_i} = \widetilde{Y^T} \cdot \widetilde{\boldsymbol{u}_{bi}} \widetilde{Y^T} \leftarrow \mathbb{O}^{n\times m}, \widetilde{\boldsymbol{u}_{bi}} \leftarrow \mathbb{O}^{m\times 1}$
2) Compute $\|\widetilde{\bar{\boldsymbol{u}}_{bi}}\|$ by normalization
3) Encrypt $\widetilde{\bar{\boldsymbol{u}}_{bi\_error}}$ under CKKS: $c_{\widetilde{\bar{\boldsymbol{u}}_{bi\_error}}}$
4) Let $\widetilde{r}$ be random coins for CKKS encryption
5) Output $(I_a, \widetilde{r}, \widetilde{m_a}, \widetilde{\bar{U}_a})$

Since matrices Y^T, $\widetilde{Y^T} \leftarrow \mathbb{O}^{n\times m}$ and vectors $\boldsymbol{u}_{bi}$, $\widetilde{\boldsymbol{u}_{bi}} \leftarrow \mathbb{O}^{m\times 1}$, then $\boldsymbol{y}_i \overset{c}{\equiv} \widetilde{\boldsymbol{y}_i}$. Similarly,

$$\|\bar{\boldsymbol{u}}_{bi}\| = \|Y \cdot \boldsymbol{u}_i'\| = \|Y \cdot \frac{\boldsymbol{x}_i + \boldsymbol{y}_i}{\|\boldsymbol{x}_i + \boldsymbol{y}_i\|}\| \tag{17}$$

matrices X, $\widetilde{X} \leftarrow \mathbb{O}^{m\times n}$ and vectors $\boldsymbol{u}_{ai}$, $\widetilde{\boldsymbol{u}_{ai}} \leftarrow \mathbb{O}^{m\times 1}$, then $\boldsymbol{x}_i \overset{c}{\equiv} \widetilde{\boldsymbol{x}_i}$ and $\|\bar{\boldsymbol{u}}_{bi}\| \overset{c}{\equiv} \|\widetilde{\bar{\boldsymbol{u}}_{bi}}\|$. For ciphertext $c_{\widetilde{\bar{\boldsymbol{u}}_{bi\_error}}}$, which appears to be encrypted with pk but can be decrypted using sk by Party A, it is necessary to introduce perturbation $\boldsymbol{u}_r$ to ensure that vector $\boldsymbol{u}_{bi}$ remains confidential. Let $\boldsymbol{u}_r \leftarrow \mathcal{Q}^m, \mathcal{Q} = [-2^l, 2^l]$. According to Smudging Lemma [36], when l is sufficiently large, we have $\boldsymbol{u}_{bi\_error} \overset{c}{\equiv} \boldsymbol{u}_r$, leading to $\boldsymbol{u}_{bi\_error} \leftarrow \mathcal{Q}^m$. Thus, we conclude that the distributions $V_a = (I_a, r, m_a)$ and $S_a = (I_a, \widetilde{\bar{\boldsymbol{u}}_a})$ are computationally indistinguishable.

Party B is constructed in a similar way as A. Again, for simplicity of expression, we denote the input of B by $I_b = (Y)$ and all information accepted by Party A during protocol execution by $m_b = (c_{\boldsymbol{u}'_i}, c_{X\boldsymbol{y}_1}, \|\bar{\boldsymbol{u}}_{ai}\|)$. B's view is $V_b = (I_b, r, m_b)$ where r are random internal coin tosses of B. The simulator $S_b = (I_b, \widetilde{\bar{U}_b})$ is shown below:

1) Compute $\|\widetilde{\bar{\boldsymbol{u}}_{ai}}\|$ by normalization
2) Encrypt $\widetilde{\boldsymbol{u}'_i}$ and $\widetilde{X\boldsymbol{y}_1}$ under CKKS: $\widetilde{c_{\boldsymbol{u}'_i}}$ and $\widetilde{c_{X\boldsymbol{y}_1}}$
3) $\widetilde{r}$ is random coins for CKKS encryption
4) Output $(I_b, \widetilde{r}, \widetilde{m_b}, \widetilde{\bar{U}_b})$

For Party A, since $\widetilde{X} \leftarrow \mathbb{O}^{m\times n}$ and $\widetilde{\boldsymbol{u}_{ai}}, \widetilde{\boldsymbol{u}_{bi}} \leftarrow \mathbb{O}^{m\times 1}$, then we have $\boldsymbol{x}_i \overset{c}{\equiv} \widetilde{\boldsymbol{x}_i}$ and $\boldsymbol{y}_i \overset{c}{\equiv} \widetilde{\boldsymbol{y}_i}$, so

$$\|\widetilde{\bar{\boldsymbol{u}}_{ai}}\| = \|\widetilde{X} \cdot \widetilde{\boldsymbol{u}'_i}\| = \|\widetilde{X} \cdot \frac{\widetilde{\boldsymbol{x}_i} + \widetilde{\boldsymbol{y}_i}}{\|\widetilde{\boldsymbol{x}_i} + \widetilde{\boldsymbol{y}_i}\|}\| \overset{c}{\equiv} \|\bar{\boldsymbol{u}}_{ai}\| \tag{18}$$

The remaining ciphertext variables come from the same distribution because of the RLWE assumption. So we have,

$$S_a(I_a, \widetilde{\bar{U}_a}) \overset{c}{\equiv} V_a(I_a, r, m_a)$$
$$S_b(I_b, \widetilde{\bar{U}_b}) \overset{c}{\equiv} V_b(I_b, r, m_b) \tag{19}$$

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

B. Security of Protocol 3

The proof process is similar to that of Protocol 2, A's view is $V_a = (I_a, r, m_a)$ where $I_a = (X, U_a, pk, sk)$, $m_a = (\boldsymbol{\Delta}_{aj}, \boldsymbol{y}_{j\_error}, c'_{z_{aj}}, c'_{z_{bj}})$.

Given the I_a and U_a, the simulator S_a:

1) Computed $\widetilde{\boldsymbol{\Delta}_{aj}} = \widetilde{z_{aj}} - U_a U_a^T \widetilde{z_{aj}} - U_a U_b^T \widetilde{z_{bj}}$pick $\widetilde{z_{aj}}, \widetilde{z_{bj}} \leftarrow \mathbb{O}^{m\times 1}$
2) picked $\boldsymbol{y}\widetilde{_{j\_error}} \leftarrow \mathcal{Q}^m$
3) $\widetilde{r}$, $\widetilde{c_{z'_{aj}}}$ and $\widetilde{c_{z'_{bj}}}$ are generated by CKKS encryption
4) Output $(I_a, \widetilde{r}, \widetilde{m_a})$

The initial random vector $\widetilde{z_{aj}}$ and $\widetilde{z_{bj}}$ are both sampled from $\mathbb{O}^{m\times 1}$, and satisfied $\|\widetilde{z_{aj}}\| = \|\widetilde{z_{bj}}\| = \frac{\sqrt{2}}{2}$. Then, U_a, U_b are known, so we have $\widetilde{\boldsymbol{\Delta}_{aj}} \overset{c}{\equiv} \boldsymbol{\Delta}_{aj}$. Also, Party A have secret key to decrypt the ciphertext $\widetilde{c_{z'_{aj}}}$ and $\widetilde{c_{z'_{bj}}}$, so according to Smudging lemma in [36], when the disturbance $\boldsymbol{u}_r$, $\boldsymbol{u}_{ra}$ and $\boldsymbol{u}_{rb}$ are picked from $\mathcal{Q}^m$ and l is sufficiently large, $\boldsymbol{y}\widetilde{_{j\_error}}$, $\widetilde{z'_{aj}}$, $\widetilde{z'_{bj}}$ are also picked from distribution $\mathcal{Q}^m$. Thus, we can say V_a and S_A are computationally indistinguishable because the corresponding parameters are randomly taken from the same distribution. B's view is $V_b = (I_b, r, m_b)$, where $I_b = (Y, U_b, c_{X\boldsymbol{y}_1})$, $m_b = (c_{z_{aj\_error}}, c_{\boldsymbol{x}_j}, z'_{aj}, z'_{bj})$.

We build simulator S_b by giving I_b:

1) $\widetilde{r}, c\widetilde{_{z_{aj\_error}}}$ and $\widetilde{c_{\boldsymbol{x}_j}}$ are computed from CKKS encryption system
2) Pick $\widetilde{z'_{aj}}, \widetilde{z'_{bj}} \leftarrow \mathcal{Q}$
3) Output $(I_b, \widetilde{r}, \widetilde{U_a}, \widetilde{U_b})$

Since the information related to Party A received by Party B has been encrypted with the public key pk, according to RLWE, $c_{z_{aj\_error}}, c\widetilde{_{z_{aj\_error}}}$ and $c_{\boldsymbol{x}_j}, \widetilde{c_{\boldsymbol{x}_j}}$ are chosen from the same distribution. Moreover, the remaining information is also picked from the same distribution, so V_b and S_b are computationally indistinguishable and we have:

$$S_a(I_a) \overset{c}{\equiv} V_a(I_a, r, m_a)$$
$$S_b(I_b, \widetilde{U_a}, \widetilde{U_b}) \overset{c}{\equiv} V_b(I_b, r, m_b) \tag{20}$$

Thus, we conclude that Protocol 3 is secure in the semi-honest model.

C. Security of Protocol 4

A's view is $V_a = (X, pk, sk, r, \sigma_{res})$, we build A's simulator S_a:

1) Compute $\widetilde{c_{\sigma_{res}}} = \widetilde{p} \cdot (\widetilde{c_{\sigma^2}} - \theta^2 \cdot \widetilde{c_{\sigma^2_{max}}})\widetilde{p} \leftarrow \mathbb{P}$
2) $\widetilde{r}$ is random coins for CKKS encryption
3) Output $(X, pk, sk, \widetilde{r}, \widetilde{Bool})$

Because $p \leftarrow \mathbb{P}, \widetilde{p} \leftarrow \mathbb{P}$, then if both parties follow the protocol and compute according to the semi-honest model, then $c_{\sigma_{res}}$ and $\widetilde{c_{\sigma_{res}}}$ are computationally indistinguishable. We have:

$$(X, \widetilde{r}, \widetilde{\sigma_{res}}) = (X, r, \sigma_{res}) \tag{21}$$

Thus, we conclude that $S_a(X, \widetilde{r}, \widetilde{Bool}) \overset{c}{\equiv} V_a(X, r, \sigma_{res})$.

B's view is $V_b = (Y, \bar{\boldsymbol{u}}_a, \bar{\boldsymbol{u}}_b, r, c_{\sigma_a}, c_{\boldsymbol{x}})$ and B's simulator is $S_b = (Y, \widetilde{r}, \widetilde{\bar{\boldsymbol{u}}_a}, \widetilde{\bar{\boldsymbol{u}}_b})$:

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

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 4 is security in semi-honest model.

$$S_a(X, pk, sk, \widetilde{r}, \widetilde{Bool}) \overset{c}{\equiv} V_a(X, pk, sk, r, \sigma_{res})$$
$$S_b(Y, \widetilde{r}, \widetilde{\bar{\boldsymbol{u}}_a}, \widetilde{\bar{\boldsymbol{u}}_b}) \overset{c}{\equiv} V_b(Y, \bar{\boldsymbol{u}}_a, \bar{\boldsymbol{u}}_b, r, c_{\sigma_a}, c_{\boldsymbol{t}}) \tag{22}$$

D. Proof of Theorem 1

As the algorithm is executed sequentially, where the output of each sub-protocol serves as the input to the subsequent one, the security of each sub-protocol ensures the overall security of the entire process. Following the principles of sequential modular composition outlined in [37], we can confidently assert that the system preserves its security under the semi-honest adversarial model. Therefore, by integrating the results from Eq.(19), Eq.(20), and Eq.(22), we establish that Theorem 1 holds both correctly and securely within the semi-honest model.

Finally, since all exchanged ciphertexts are generated by CKKS, under IND-CPA security the ciphertext transcript is computationally indistinguishable from encryptions of random messages; combined with the simulators constructed for Protocols 2–4 and the sequential modular composition, PP-DR satisfies semantic security in the semi-honest model.

VI. Performance Evaluation

Based on prior protocols, the experimental setup is outlined as follows: Parties A and B, each with data of equal dimensions, seek to cooperatively implement a privacy-preserving dimensionality reduction (PP-DR) protocol to improve the accuracy of data reduction. Since the scenario involves joint dimensionality reduction, data matrices can be horizontally or vertically combined depending on user needs.

It is crucial to note that while our protocol eventually provides only the left orthogonal matrix, the method for achieving various dimensionality reductions depends solely on whether the operation is conducted on A^TA or AA^T. For example, in a dataset containing samples and features, using A^TA yields a left singular matrix helpful for reducing

TABLE I
THEORETICAL OVERHEAD AND COMPUTATION

Protocol	Method	Overhead	Computation
POB	M2	$i \cdot (2 \cdot c_{\boldsymbol{v}} + m + n + 2) + c_v$	$(i+1) \cdot Enc_{pk}(p)$ $+i \cdot [EvalAdd(c_v, p) + Dec_{sk}(c_v)]$
	M3		$(i+1) \cdot Enc_{pk}(p)$ $+i \cdot [EvalMult(c_v, p) + Dec_{sk}(c_v)]$
SPOB	M2	$j \cdot (4 \cdot c_{\boldsymbol{v}} + 3 \cdot m + n)$	$j \cdot [2 \cdot Enc_{pk}(p) + 3 \cdot EvalAdd(c_v, p)$ $+2 \cdot EvalMult(c_v, p) + Dec_{sk}(c_v)]$
	M3		$j \cdot [2 \cdot Enc(p) + 2 \cdot EvalAdd(c_v, p)$ $+3 \cdot EvalMult(c_v, p) + Dec(c_v)]$
SC	/	$3 \cdot c_v$	$2 \cdot Enc_{pk}(p) + 2 \cdot EvalAdd(c_v, p)$ $+3 \cdot EvalMult(c_v, p) + Dec_{sk}(c_v))$

TABLE II
PER-ITERATION COUNTS OF HOMOMORPHIC PRIMITIVES AND CIPHERTEXTS (CKKS WITH N; SLOTS s=$N/2$)

Method	Mult. Depth	EvalMult(c_t,c_t)	EvalMult(c_t,t)	EvalAdd	Comm. (cts)
HPCA [32]	≥ 7	$\approx \lceil m/s \rceil \cdot \lceil n/s \rceil$	0	$\approx \lceil m/s \rceil \cdot \lceil n/s \rceil$	$\approx \lceil m/s \rceil \cdot \lceil n/s \rceil$
PP-PCA [33]	13	$\approx \lceil n/s \rceil^2$	0	$\approx \lceil n/s \rceil^2$	$\approx \lceil n/s \rceil^2$
PP-DR (ours)	≤ 3	0	$\approx \lceil m/s \rceil$	$\approx \lceil m/s \rceil$	$\approx \lceil m/s \rceil$

Notes. Here N is the CKKS polynomial degree and $s{=}N/2$ is the number of complex SIMD slots per ciphertext. **Comm. (cts)** counts ciphertexts sent per iteration.

the dimensionality of features, while applying AA^T achieves dimensionality reduction for the samples. Practically, when two parties share identical feature data, their sample data is often combined. Therefore, in this study, the data matrix $M^{(2m\times n)}$ is split horizontally into two equally sized matrices, representing the datasets for the participants, as shown in Fig. 6.

A. Theoretical Analysis

To concretely quantify the computational and communication overhead of our proposed PP-DR protocol, we provide a detailed breakdown of its core components in Table I. This table summarizes the total computation and communication overhead incurred during a single execution of each sub-protocol—Principal Orthogonal Basis (POB), Sub-principal Orthogonal Basis (SPOB), and Stop Criterion (SC)—under two masking strategies: Secure Addition (M2) and Secure Multiplication (M3). That is, each entry reflects the cost of computing one eigenvector under the corresponding protocol configuration. Here, i and j denote the number of iterations for Protocol 2 and Protocol 3 respectively, c_v represents the ciphertext vector, and m, n are the sample and feature dimensions of the dataset.

In PP-DR, all ciphertext-related operations involve only plaintext matrix and ciphertext vector multiplications, with no ciphertext-ciphertext multiplication or bootstrapping. As a result, the multiplicative depth is strictly bounded: Protocol 2 incurs a depth of 1, while Protocol 3 reaches at most 3. This design allows us to safely operate with a small polynomial modulus degree (N = 8192), significantly improving runtime and reducing memory usage. Moreover, the ciphertexts transmitted are always vectors, not matrices, leading to reduced communication overhead.

1) Protocol-Level Efficiency: Protocol POB executes one secure vector interaction per iteration, followed by a single encryption and homomorphic addition or multiplication. Protocol SPOB adds orthogonal projection and masking steps, resulting in two vector transmissions and two encrypted computations per iteration. Protocol SC is invoked only to check convergence, incurring minimal cost. In all cases, ciphertext size remains linear in m, and operations are limited to low-degree arithmetic circuits.

2) Comparison With Prior Art: We contrast our method with representative prior work in Table II, reporting *per-iteration counts* of the main homomorphic encryption operations, as well as the number of ciphertexts transmitted, under a fixed CKKS slot size ($s = N/2$). Prior schemes such as HPCA [32] and PP-PCA [33] perform a large number of ciphertext–ciphertext multiplications (on the order of $\lceil m/s \rceil \cdot \lceil n/s \rceil$ or $\lceil n/s \rceil^2$ per iteration) and transmit entire ciphertext matrices, leading to high computation and communication costs. In contrast, our PP-DR protocol eliminates ciphertext–ciphertext multiplications, reduces ciphertext–plaintext multiplications and additions to $\lceil m/s \rceil$ per iteration, and minimizes communication to only $\lceil m/s \rceil$ ciphertexts, offering substantial efficiency gains.

3) Theoretical Justification for Accuracy: PP-DR achieves high numerical fidelity by addressing two primary sources of error in encrypted SVD. First, non-linear steps such as normalization typically require square-root or division operations, which in prior HE-based PCA schemes are approximated by polynomials [30], [31]. Such approximations are only accurate within a bounded interval; when inputs fall outside this range, the approximation error grows rapidly and accumulates across iterations. PP-DR instead realizes these operations via secure two-party protocols, ensuring exact results without polynomial approximation error. Second, CKKS itself is an approximate scheme: each multiplication consumes precision and amplifies rounding noise [30]. In prior designs with deep ciphertext–ciphertext circuits, the precision budget is quickly exhausted, leading to instability and loss of orthogonality [32], [33]. By contrast, PP-DR reduces multiplicative depth through interactive protocols. This shallower circuit depth means that, under the same polynomial modulus and security parameters, a larger CKKS scale factor can be selected without exhausting the noise budget, thereby retaining more significant digits throughout the computation and markedly improving accuracy. Together, these mechanisms explain the observed accuracy in our experiments (e.g., orthogonality errors $\leq 10^{-15}$).

PP-DR's design—featuring shallow homomorphic depth, minimal communication, and plaintext-heavy computation—lays a solid theoretical foundation for efficiency and accuracy. The subsequent sections present experimental results that substantiate these claims.

B. Experimental Setup and Baselines

We evaluate the performance of PP-DR in terms of computational efficiency, communication overhead, and numerical accuracy. All implementations were executed in Python using the TenSEAL library [39] on a personal computer equipped with an Intel Core i9-12900KF CPU (3.19GHz) and 64GB RAM. For consistency with prior homomorphic encryption (HE)-based schemes [32], [33], we disabled multi-threading and hardware acceleration. All protocols were tested under the same CKKS encryption parameters.

TABLE III
CKKS Parameter Configurations Used in Our Experiments

Configuration	N	Scale	Slot Capacity	Coeff. Modulus ($\lvert Q \rvert$)
High-Efficiency	8192	2^{35}	4096	205 bits
High-Dimensional	16384	2^{60}	8192	360 bits

Notes. $\lvert Q \rvert$ denotes the total bit-length (sum) of the coefficient modulus primes in the CKKS modulus chain used in our implementation. Both settings are chosen to meet 128-bit **classical** security (tc128) following the HE Standard/SEAL security convention.

TABLE IV
Summary of Performance in Prior HE-Based PCA Methods

Method	Dataset	Runtime	Orthogonality
HPCA [32]	Yale (256×256)	19.86 mins	/
PP-PCA [33]	Yale (165×256)	$\geq$ 0.62 mins	/
K-PCA [42]	Yale (128×128)	$\geq$ 7.36 mins	/
DH-PCA [43]	Yale (165×256)	9.61 mins	/
PP-DR (ours)	Yale (256×256)	$\leq$ 10 sec	$\leq 10^{-15}$

Notes. Runtime values are taken from original papers for reference. Only PP-DR reports orthogonality accuracy.

1) Parameter Selection and Security Level: The CKKS parameter configurations adopted in our PP-DR scheme are summarized in Table III. These settings are chosen to balance *computational efficiency*, *numerical precision*, and *cryptographic security* (measured by the security parameter λ), following the recommendations of the Homomorphic Encryption Standardization Workshop [40], [41]. Throughout this paper, the reported security level λ follows the **128-bit classical** security convention (tc128) under the RLWE assumption.

In most experiments, we use a *high-efficiency configuration* with polynomial modulus degree $N = 8192$, scale 2^{35}, and coefficient modulus size $|Q| = 205$ bits. This setting supports multiplicative depth up to 3 and provides 4096 slots, which is sufficient for medium-dimensional feature vectors, avoiding ciphertext splitting and minimizing computation latency. For high-precision or high-dimensional tasks, we adopt a *high-dimensional configuration* with $N = 16384$ and scale 2^{60}, using coefficient modulus size $|Q| = 360$ bits and slot capacity 8192. This ensures that feature vectors with dimension >4096 fit entirely in one ciphertext, reducing encoding/decoding overhead and preserving evaluation efficiency.

In both configurations, perturbation vectors (u_r, u_{ar}, u_{br}) are uniformly sampled from $\mathcal{Q}^m = [-2^l, 2^l]^m$, where $l = 50$ for $N = 8192$ and $l = 60$ for $N = 16384$. These bounds guarantee correct decryption and statistical indistinguishability under the smudging-based security argument detailed in Section V.

2) Prior Methods: Table IV summarizes the reported runtime and accuracy of representative homomorphic encryption-based PCA schemes. The results are taken directly from the original publications, as no official implementations are publicly available. Consequently, the runtime values should be interpreted as indicative only, given their dependence on implementation optimizations and hardware environments.

Among these methods, HPCA [32] is the most widely adopted baseline and serves as a reference point in multiple follow-up studies. We therefore reimplemented HPCA in our environment to enable direct comparison under a unified setup. Detailed comparisons between PP-DR and HPCA are presented in Sections VI-D and VI-E. While our empirical performance comparison centers on HPCA, we also include the reported R^2 of other schemes, since these measures are largely independent of platform-specific factors and thus support fair cross-study evaluation.

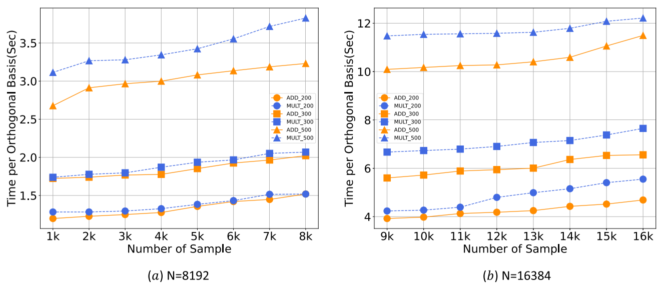

Fig. 7. Computation time per orthogonal basis as the number of samples increases, under fixed feature dimensions ($n \in \{200, 300, 500\}$).

3) Scope of Evaluation: The remainder of this section presents empirical evaluations on both synthetic and real-world datasets. We assess PP-DR in terms of runtime, communication volume, and numerical accuracy under varying dimensions and masking strategies (ADD and MULT). In real-world evaluations, we focus on direct performance comparison with HPCA [32]. Other methods from Table IV are included only in accuracy comparisons using metrics such as orthogonality and subspace distance. These results validate the theoretical advantages discussed in Section VI-A.

C. Synthetic Dataset

To evaluate the scalability and efficiency of PP-DR under varying data dimensions, we conducted two groups of synthetic experiments:

1) Sample Scalability: We fixed the feature size $n \in \{200, 300, 500\}$ and varied the number of samples m from $1k$ to $16k$ in increments of $0.5k$. Each element in the matrix $M \in \mathbb{R}^{m \times n}$ was drawn from $[0, 10]$, with a fixed sparsity of 0.005 and numerical rank of 100.

2) Feature Scalability: We fixed the sample size at $m = 4096$ and increased the number of features $n \in \{256, 512, 1024, 2048, 4096\}$ for $N = 8192$, and $n \in \{512, 1024, 2048, 4096, 8192\}$ for $N = 16384$, to simulate high-dimensional scenarios.

In both experiments, we compared two perturbation methods: **ADD**: perturbation via additive masking; **MULT**: perturbation via element-wise multiplication. As shown in Fig. 7, with increasing sample sizes, the average time to compute a single orthogonal basis remains under 12 seconds (for $n = 500$) with $N = 8192$, and increases moderately under higher dimensions with $N = 16384$. MULT is consistently slower than ADD due to additional ciphertext operations.

In Fig. 8, the feature-scalability experiments demonstrate the impact of CKKS encoding limits on PP-DR. With $N = 8192$, the scheme efficiently handles up to 4096 features, requiring at most 0.87 minutes per orthogonal basis under MULT. With $N = 16384$, the capacity extends to 8192 features with runtime under 5.45 minutes. In both cases, the runtime

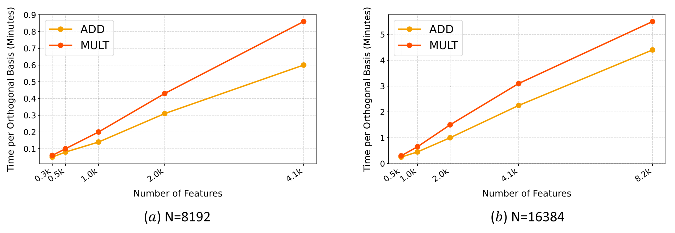

Fig. 8. Computation time per orthogonal basis as the number of features increases, under fixed sample size ($m = 4096$).

TABLE V
PERFORMANCE COMPARISON BETWEEN ADD AND MULT

Datasets	m	n	k	N	Time Taken(Sec)		Overhead(MB)	
					ADD	**MULT**	**ADD**	**MULT**
Breast Cancer	569	30	2	8192	0.92	1.07	21.0	18.3
Wine-quality-red	1599	11	3	8192	1.06	1.09	35.3	30.9
Wine-quality-white	4898	11	3	8192	1.96	1.98	36.3	31.9
Parkinsons	5875	16	4	8192	2.52	2.66	51.3	45.1
UR3 CobotOps	7358	20	2	8192	1.42	1.73	22.3	19.8

gap between ADD and MULT widens as the dimension increases.

The observed linear growth in runtime is consistent with the theoretical complexity of PP-DR. Given a polynomial modulus degree N, at most $N/2$ real slots can be packed into a ciphertext, matching our experimental limits of 4096 features for $N = 8192$ and 8192 for $N = 16384$. Each iteration performs a plaintext–ciphertext matrix–vector multiplication, where the plaintext matrix has dimension $m \times 1$ and the ciphertext vector packs m slots; this results in $\mathcal{O}(m)$ homomorphic multiplications and additions per iteration. Similarly, the communication overhead grows linearly with m, since only $\lceil m/(N/2) \rceil$ ciphertexts are transmitted.

These results confirm that PP-DR scales linearly with the feature dimension up to the CKKS packing limit, enabling efficient deployment in high-dimensional applications such as federated biomedical analysis and large-scale sensor networks.

D. Real Dataset

In addition to utilizing synthetic datasets, we also incorporated several real datasets to evaluate the performance of our scheme in comparison with the Homomorphic PCA (HPCA) scheme [32]. We selected four classification datasets that align with the test datasets used in HPCA: Winequality-white [44], Winequality-red [44], Air Quality [45], Parkinsons [46]. Furthermore, we included additional datasets, namely Breast Cancer [47], UR3 CobotOps [48] datasets. To ensure the consistency of the experiments, the time cost and communication overhead of PP-DR with two different methods are evaluated by counting the same number of eigenvalues, with the results presented in Table V. For each dataset with dimensions $m \times n$, m denotes the number of samples, and n indicates the number of features. Let k denote the number of principal components extracted, and N signify the polynomial modulus degree.

The results in Table V indicate that **PP-DR** has superior performance when dealing with lower latitudes, such as the dataset Breast Cancer [47], with the time for calculating a principal component being approximately 0.5 seconds

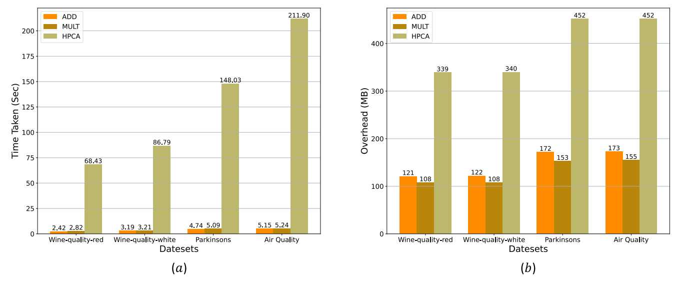

Fig. 9. Performance Comparison between ADD, MULT and HPCA [32] with $N = 16384$. (a) Time Taken (Sec). (b) Overhead (MB).

for both methods. Similarly, for higher-dimensional datasets like UR3 CobotOps [48], it also has good performance, with the calculation of a principal component taking about 0.7 seconds. In terms of communication overhead, the average communication required for calculating a single orthogonal vector for both methods is approximately 10 MB.

Moreover, we compared the HPCA scheme, with the results depicted in Fig. 9. The dataset and the number of principal components used match those in [32]. Due to the greater multiplication depth of HPCA, the minimum polynomial modulus degree (N) is 16384. Data in Fig. 9a indicate that PP-DR increases computational efficiency by 30-40 times, with the communication overhead being roughly one-third of that of HPCA, as illustrated in Fig. 9b.

Finally, as seen in Table V and Fig. 9b, the **MULT** scheme exhibits lower communication overhead than **ADD** because it involves fewer iterations, thus reducing communication frequency.

E. Error Analysis

To demonstrate that the computational results of our ciphertext scheme are as accurate as those of the plaintext scheme, this section thoroughly evaluates the results obtained after matrix decomposition across various datasets for different schemes. We mainly utilize the standard Singular Value Decomposition (SVD) results from the NumPy library as a benchmark and assess the accuracy of our method alongside the HPCA calculation results from two perspectives: orthogonality and the distance between the subspaces. At the same time, we also provide the accuracy of the plaintext scheme Plain-DR that we designed, as detailed in Table VI. Given that recent schemes [33], [42], [43] are all compared to HPCA using the R2 score as a metric of accuracy. Consequently, we have also included the test results in Table VII.

1) Orthogonality: Taking the **ADD** scheme as an example, as shown in Table VI, we observe that when $N = 8192$, the average orthogonality error of our ciphertext protocol is $4.864 * 10^{-15}$, while at $N = 16384$, the average error is $5.655 * 10^{-15}$. The error of the plaintext protocol is $5.498 * 10^{-15}$. Therefore, the calculation results of both our designed encrypted and unencrypted protocols satisfy orthogonality. However, the outcomes derived from HPCA decomposition do not fully satisfy orthogonality. The primary reason for this discrepancy is the substantial cumulative error present in the ciphertext, which only ensures the accuracy of the initial

TABLE VI
ACCURACY(ORTH. AND DIST.) EVALUATION BETWEEN ADD, MULT AND HPCA

Dataset		ADD			MULT			HPCA
		PlainDR	N=8192	N=16384	PlainDR	N=8192	N=16384	
Breast Cancer	$dist(S_1,S_2)$	$2.70*10^{-14}$	$6.70*10^{-8}$	$1.08*10^{-10}$	$2.52*10^{-14}$	$3.99*10^{-7}$	$7.50*10^{-9}$	1.4062
	$\|I-U^TU\|_2$	$1.78*10^{-15}$	$1.67*10^{-15}$	$1.78*10^{-15}$	$2.00*10^{-15}$	$2.00*10^{-15}$	$1.78*10^{-15}$	2.5324
Winequality-red	$dist(S_1,S_2)$	$5.59*10^{-11}$	$7.27*10^{-8}$	$4.07*10^{-9}$	$1.99*10^{-9}$	$4.01*10^{-7}$	$2.51*10^{-9}$	2.6292
	$\|I-U^TU\|_2$	$3.00*10^{-15}$	$2.89*10^{-15}$	$2.92*10^{-15}$	$3.01*10^{-15}$	$3.01*10^{-15}$	$1.52*10^{-15}$	2.4491
Winequality-white	$dist(S_1,S_2)$	$2.80*10^{-10}$	$8.19*10^{-8}$	$3.15*10^{-9}$	$1.77*10^{-8}$	$4.01*10^{-7}$	$1.98*10^{-9}$	2.2861
	$\|I-U^TU\|_2$	$5.44*10^{-15}$	$5.44*10^{-15}$	$5.43*10^{-15}$	$5.11*10^{-15}$	$5.33*10^{-15}$	$5.66*10^{-15}$	2.4495
Parkinsons	$dist(S_1,S_2)$	$8.66*10^{-10}$	$8.68*10^{-8}$	$1.17*10^{-10}$	$1.50*10^{-9}$	$5.23*10^{-7}$	$9.57*10^{-9}$	3.0642
	$\|I-U^TU\|_2$	$6.44*10^{-15}$	$6.33*10^{-15}$	$6.60*10^{-15}$	$6.63*10^{-15}$	$6.45*10^{-15}$	$6.32*10^{-15}$	3.4641
UR3 CobotOps	$dist(S_1,S_2)$	$1.77*10^{-15}$	$6.66*10^{-8}$	$3.05*10^{-13}$	$1.84*10^{-15}$	$3.99*10^{-7}$	$2.81*10^{-11}$	0.9999
	$\|I-U^TU\|_2$	$8.55*10^{-15}$	$7.99*10^{-15}$	$8.22*10^{-15}$	$8.33*10^{-15}$	$8.22*10^{-15}$	$2.22*10^{-15}$	1.4142
Air Quality	$dist(S_1,S_2)$	$1.49*10^{-9}$	/	$2.05*10^{-8}$	$8.16*10^{-7}$	/	$1.51*10^{-6}$	2.3041
	$\|I-U^TU\|_2$	$7.87*10^{-15}$	/	$7.98*10^{-15}$	$8.43*10^{-15}$	/	$9.66*10^{-15}$	2.1399

TABLE VII
R2 SCORE OF DIFFERENT SCHEMES

Datasets	HPCA [32]	K-PCA [42]	DH-PCA [43]	PP-PCA [33]	**PP-DR (ours)**
MNIST	0.141	/	/	0.485	0.505
Yale	0.579	0.581	0.649	0.674	0.711

TABLE VIII
TIME TAKEN (SEC) BETWEEN MULT AND INV

N	Breast Cancer		Winequality-red		Winequality-white		Parkinsons	
	MULT	INV	MULT	INV	MULT	INV	MULT	INV
8192	1.07	6.59	1.09	26.12	1.98	75.75	2.66	130.45
16384	2.79	16.27	2.82	55.23	3.21	185.25	5.09	228.72

calculation results and proves inadequate for the complete decomposition of matrices.

2) Subspace Distance: There exists an infinite number of choices for the unit orthogonal basis of a subspace, necessitating the measurement of result accuracy via the distance between subspaces. A subspace distance approaching zero indicates that the subspaces nearly coincide, thereby demonstrating the accuracy of the results. According to the definition of subspace distance, let S_1, $S_2 \subseteq \mathbb{R}^n$ represent two subspaces of equal dimension, with their subspace distance defined as $dist(S_1,S_2) = \|P_1 - P_2\|_2$. Here, P denotes the orthogonal projection of S and satisfies $P^T = (UU^T)^T = UU^T = P$. We utilize the orthogonal matrix obtained from Singular Value Decomposition (SVD) as a reference standard and compute the distance between the left orthogonal matrices derived from various methods and the subspace spanned by the standard orthogonal matrix. The results presented in Table VI indicate that when $N = 8192$, the average subspace distance for the **ADD** scheme is $7.5 * 10^{-8}$, while for the **MULT** scheme,it is $4.25 * 10^{-7}$. When $N = 16384$, an increase in the scale factor corresponds to enhanced calculation accuracy, resulting in an average subspace distance of $4.65 * 10^{-9}$ for the **ADD** scheme and $2.87 * 10^{-8}$ for the **MULT** scheme. Furthermore, the **MULT** scheme introduces more errors by ciphertext multiplication, resulting in slightly lower accuracy than the **ADD** scheme.

3) R2 Score: To further demonstrate the enhancement in accuracy achieved by our proposed scheme, we conducted a comparative analysis with several similar studies from the past two years. To ensure result consistency, we selected two datasets: MNIST [49] and Yale [50], which have been utilized in the works of HPCA [32], K-PCA [42], PP-PCA [33], and DH-PCA [43]. As illustrated in Table VII, our scheme surpasses the currently highest-accuracy method of the same category, PP-PCA, by improving 0.02 on the MNIST dataset and 0.037 on the Yale dataset.

VII. EXTENSIONS

In this section, we discuss masking strategy enhancements and protocol scalability.

In the previously discussed **MULT** scheme, Party B masks the intermediate vector $\boldsymbol{u}$ using an element-wise random vector $\boldsymbol{u_r}$, such that $\boldsymbol{u} \odot \boldsymbol{u_r}$ is transmitted. The random mask is sampled as $\boldsymbol{u_r} \leftarrow \mathcal{Q}^m$, where $\mathcal{Q}^m = [-2^l, 2^l]^m$. In practice, the value of l is chosen to balance noise growth and encryption parameter efficiency. The **MULT** masking scheme achieves strong security and high computational efficiency with carefully selected parameters, as demonstrated in our experimental evaluations.

For scenarios that may require alternative masking strategies—such as compatibility with other cryptographic components or enhanced robustness against structured leakage—we also explore an optional scheme based on invertible matrix multiplication(**INV**) [23]. In this approach, the masked vector is obtained by computing $\boldsymbol{Mu}$, where $M \in \mathbb{R}^{m\times m}$ is a randomly generated invertible matrix. After Party A decrypts the ciphertext containing the masked vector, Party B applies M^{-1} locally to remove the mask and recover the correct intermediate result. At no point does Party A gain access to unmasked data. This method introduces significantly more computation and communication overhead, as shown in Table VIII, due to the transition from vector-vector to matrix-vector operations and the additional ciphertext size. Moreover, the iterative application of matrix multiplication may accumulate numerical errors, reducing accuracy.

Although our implementation targets the two-party scenario, the proposed PP-DR protocol is naturally extendable to N parties. Since the core computation relies on plaintext matrix and ciphertext vector multiplications, the protocol structure accommodates partitioned data across multiple participants. Each party can locally contribute to the iterative computation using appropriate masking (e.g., **MULT** or **INV**), without revealing raw data. Alternatively, a multi-key homomorphic encryption (MKHE) scheme may be employed to support encrypted computation across parties holding different keys. The integration of MKHE or more scalable masking protocols enables collaborative dimensionality reduction among multiple parties, which we leave as future work.

VIII. Conclusion

We introduced **PP-DR**, a novel CKKS-based homomorphic encryption dimensionality reduction protocol demonstrating enhanced computational efficiency, accuracy, and reduced communication overhead. Our protocol is characterized by the consistent execution of operations on plaintext matrices and ciphertext vectors, thereby avoiding the computational complexities associated with operations between ciphertexts or matrices, as seen in traditional schemes. Additionally, in contrast to prior approaches that necessitated the transmission of ciphertext matrices, our method only requires the transmission of ciphertext vectors, significantly minimizing communication overhead while maintaining data security. From a security standpoint, our scheme eliminates the reliance on a trusted third party and addresses the collusion concerns present in earlier models [21], [23], [24], [25], [26].

References

[1] A. Hartebrodt, R. Nasirigerdeh, D. B. Blumenthal, and R. Röttger, “Federated principal component analysis for genome-wide association studies,” in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Dec. 2021, pp. 1090–1095.

[2] M. W. Berry and M. Browne, *Understanding Search Engines: Mathematical Modeling and Text Retrieval*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1999.

[3] B. H. Dayton and Z. Zeng, “Computing the multiplicity structure in solving polynomial systems,” in *Proc. Int. Symp. Symbolic Algebr. Comput.*, Jul. 2005, pp. 116–123.

[4] E. F. Deprettere, Ed., *SVD and Signal Processing: Algorithms, Applications and Architectures*. Amsterdam, The Netherlands: North-Holland Publishing Co., 1989.

[5] C. Eckart and G. Young, “The approximation of one matrix by another of lower rank,” *Psychometrika*, vol. 1, no. 3, pp. 211–218, Sep. 1936.

[6] E. F. Van de Velde, *QR-Decomposition*. New York, NY, USA: Springer, 1994, pp. 125–140.

[7] P. V. Giampouras and A. A. Rontogiannis, “Online low-rank subspace learning from incomplete data: A Bayesian view,” Tech. Rep., 2016.

[8] N. Mastronardi and P. Van Dooren, “Rank-revealing decomposition of symmetric indefinite matrices via block anti-triangular factorization,” *Linear Algebra Appl.*, vol. 502, pp. 126–139, Aug. 2016.

[9] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in *Theory of Cryptography*. Berlin, Germany: Springer, 2006, pp. 265–284.

[10] H. B. McMahan, “Communication-efficient learning of deep networks from decentralized data,” Tech. Rep., 2023.

[11] R. Cramer, I. B. Damgård, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*. Cambridge, U.K.: Cambridge Univ. Press, 2015.

[12] C. Gentry, “A fully homomorphic encryption scheme,” Tech. Rep., 2009.

[13] J. Park and H. Lim, “Privacy-preserving federated learning using homomorphic encryption,” *Appl. Sci.*, vol. 12, no. 2, p. 734, Jan. 2022.

[14] K. Peter et al., “Advances and open problems in federated learning,” Tech. Rep., 2021.

[15] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[16] O. Goldreich, *Foundations of Cryptography*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[17] Y. Lindell and B. Pinkas, “Secure multiparty computation for privacy-preserving data mining,” Tech. Rep., 2005.

[18] S. Han, W. K. Ng, and P. S. Yu, “Privacy-preserving singular value decomposition,” in *Proc. IEEE 25th Int. Conf. Data Eng.*, Apr. 2009, pp. 1267–1270.

[19] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh, “Privacy-preserving matrix factorization,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 801–812.

[20] S. Chen, R. Lu, and J. Zhang, “A flexible privacy-preserving framework for singular value decomposition under Internet of Things environment,” in *Trust Management XI*. Cham, Switzerland: Springer, 2017.

[21] B. Liu and Q. Tang, “Privacy-preserving decentralised singular value decomposition,” in *Proc. Inf. Commun. Secur., 21st Int. Conf.*, Beijing, China, Dec. 2020, pp. 703–721.

[22] X. Guo, “Fedpower: Privacy-preserving distributed eigenspace estimation,” Tech. Rep., 2023.

[23] D. Chai et al., “Practical lossless federated singular vector decomposition over billion-scale data,” in *Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining*. New York, NY, USA: Association for Computing Machinery, 2022, pp. 46–55.

[24] B. Liu, B. Pejó, and Q. Tang, “Privacy-preserving federated singular value decomposition,” *Appl. Sci.*, vol. 13, no. 13, p. 7373, Jun. 2023.

[25] Q. Guo, C. Zhang, Y. Zhang, and H. Liu, “An efficient SVD-based method for image denoising,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 5, pp. 868–880, May 2016.

[26] K. V. Ravi Kanth, D. Agrawal, and A. Singh, “Dimensionality reduction for similarity searching in dynamic databases,” *ACM SIGMOD Rec.*, vol. 27, no. 2, pp. 166–176, Jun. 1998.

[27] D. Froelicher et al., “Scalable and privacy-preserving federated principal component analysis,” in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2023, pp. 1908–1925.

[28] Q. Lan et al., “Genome-wide association analysis identifies new lung cancer susceptibility loci in never-smoking women in Asia,” *Nature Genetics*, vol. 44, no. 12, pp. 1330–1335, 2012.

[29] W. Jung, S. Kim, J. H. Ahn, J. H. Cheon, and Y. Lee, “Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with GPUs,” in *Cryptographic Hardware and Embedded Systems—CHES 2021*. Cham, Switzerland: Springer, 2021, pp. 3–25. [Online]. Available: https://scale.snu.ac.kr/papers/2021-09-Conference-CHES-100x.pdf

[30] J. H. Cheon, A. Kim, M. Kim, and Y. Song, “Homomorphic encryption for arithmetic of approximate numbers,” in *Advances in Cryptology—ASIACRYPT 2017 (Lecture Notes in Computer Science)*, vol. 10624. Cham, Switzerland: Springer, 2017, pp. 409–437.

[31] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, “Bootstrapping for approximate homomorphic encryption,” in *Advances in Cryptology—EUROCRYPT 2018 (Lecture Notes in Computer Science)*, vol. 11477. Cham, Switzerland: Springer, 2018, pp. 360–384.

[32] S. Panda, “Principal component analysis using CKKS homomorphic scheme,” in *Cyber Security Cryptography and Machine Learning*. Cham, Switzerland: Springer, 2021, pp. 52–70.

[33] X. Ma, C. Ma, Y. Jiang, and C. Ge, “Improved privacy-preserving PCA using optimized homomorphic matrix multiplication,” *Comput. & Secur.*, vol. 138, Mar. 2023, Art. no. 103658.

[34] H. Chen, I. Chillotti, and J. Kilian, “Improved bootstrapping for approximate homomorphic encryption,” in *Advances in Cryptology—EUROCRYPT (Lecture Notes in Computer Science)*, vol. 11477. Cham, Switzerland: Springer, 2019, pp. 34–54.

[35] T.-L. Lee, T.-Y. Li, and Z. Zeng, “A rank-revealing method with updating, downdating, and Applications. Part II,” *SIAM J. Matrix Anal. Appl.*, vol. 31, no. 2, pp. 503–525, Jan. 2009.

[36] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs, “Multiparty computation with low communication, computation and interaction via threshold FHE,” in *Advances in Cryptology—EUROCRYPT 2012*. Berlin, Germany: Springer, 2012, pp. 483–501.

[37] R. Canetti, “Security and composition of multiparty cryptographic protocols,” *J. Cryptol.*, vol. 13, no. 1, pp. 143–202, Jan. 2000.

[38] Y. Lindell, “How to simulate it—A tutorial on the simulation proof technique,” Electron. Colloq. Comput. Complex. (ECCC), Tech. Rep. TR17, 2017.

[39] A. Benaissa and B. Retiat, “TenSEAL: A library for encrypted tensor operations using homomorphic encryption,” 2021, *arXiv:2104.03152*.

[40] HomomorphicEncryption. org. (2020). *Homomorphic Encryption Standardization*. [Online]. Available: https://homomorphicencryption.org/standard/

[41] MR Albrecht et al.(2015). *Estimate All the LWE Security Parameters*. [Online]. Available: https://github.com/malb/lwe-estimator

[42] J. H. Cheon, H. Choe, S. Jung, D. Kim, D. H. Lee, and J. H. Park, "Arithmetic PCA for encrypted data," Int. Assoc. Cryptologic Res. (IACR), USA, Tech. Rep. 2023/1544, 2023.
[43] D. Wang, M. cheng, L. Liu, X. Du, X. Li, and B. Zhu, "Principal component analysis scheme based on homomorphic encryption in a distributed environment," in *Web Information Systems and Applications*. Singapore: Springer Nature, 2024, pp. 415–427.
[44] A. C. Paulo Cortez, F. Almeida, T. Matos, and J. Reis, "Wine quality," UCI Mach. Learn. Repository, Univ. California, Irvine, CA, USA, 2009, doi: 10.24432/C56S3T.
[45] S. Vito, "Air quality," UCI Mach. Learn. Repository, Univ. California, Irvine, CA, USA, 2016, doi: 10.24432/C59K5F.
[46] M. Little, "Parkinsons," UCI Mach. Learn. Repository, Univ. California, Irvine, CA, USA, 2008, doi: 10.24432/C59C74.
[47] N. S. W. Wolberg, O. Mangasarian, and W. Street, "Breast cancer Wisconsin (diagnostic)," UCI Mach. Learn. Repository, Univ. California, Irvine, CA, USA, 1995, doi: 10.24432/C5DW2B.
[48] M. Tyrovolas et al., "UR3 CobotOps," UCI Mach. Learn. Repository, Univ. California, Irvine, CA, USA, 2024, doi: 10.24432/C5J891.
[49] Y. LeCun and C. Cortes, *MNIST Handwritten Digit Database*, vol. 2. ATT Labs: Singapore, 2010.
[50] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

Haonan Yuan is currently pursuing the joint Ph.D. degree in cryptography and privacy-preserving computing with the University of Chinese Academy of Sciences, Beijing, China, and Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences (CAS), Chongqing, China. His research interests include privacy-preserving machine learning and cryptographic protocols, with an emphasis on homomorphic encryption and secure computation for efficient dimensionality reduction and matrix factorization.

Wenyuan Wu received the Ph.D. degree from the Department of Applied Mathematics, The University of Western Ontario, London, ON, Canada, in 2007. He is currently a Professor, a Ph.D. Supervisor, and the Director of the Center for Automated Reasoning and Cognition, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences (CAS), Chongqing, China. His research interests include artificial intelligence, automated reasoning, and privacy-preserving computing.

Jingwei Chen received the Ph.D. degree in computer science from the University of Chinese Academy of Sciences, Beijing, China, in 2013. He is currently an Associate Professor with Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences (CAS), Chongqing, China. His research interests include symbolic computation, lattice-based cryptography, and privacy-preserving computing.