




Privacy-Preserving Federated Singular Value Decomposition

Bowen Liu ¹ , Balázs Pejó ^{2,3}  and Qiang Tang ^{1,*} 

¹ Luxembourg Institute of Science and Technology (LIST), 5, Avenue des Hauts-Fourneaux, L-4362 Esch-sur-Alzette, Luxembourg; bowen.liu@list.lu

² ELKH-BME Information Systems Research Group, Eötvös Loránd Research Network, Hungarian Academy of Sciences, P.O. Box 91, H-1521 Budapest, Hungary; pejo@crysys.hu

³ Laboratory of Cryptography and System Security, Department of Networked Systems and Services, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Műgyetem rkp. 3., H-1111 Budapest, Hungary

* Correspondence: qiang.tang@list.lu

Abstract: Singular value decomposition (SVD) is a fundamental technique widely used in various applications, such as recommendation systems and principal component analyses. In recent years, the need for privacy-preserving computations has been increasing constantly, which concerns SVD as well. Federated SVD has emerged as a promising approach that enables collaborative SVD computation without sharing raw data. However, existing federated approaches still need improvements regarding privacy guarantees and utility preservation. This paper moves a step further towards these directions: we propose two enhanced federated SVD schemes focusing on utility and privacy, respectively. Using a recommendation system use-case with real-world data, we demonstrate that our schemes outperform the state-of-the-art federated SVD solution. Our utility-enhanced scheme (utilizing secure aggregation) improves the final utility and the convergence speed by more than 2.5 times compared with the existing state-of-the-art approach. In contrast, our privacy-enhancing scheme (utilizing differential privacy) provides more robust privacy protection while improving the same aspect by more than 25%.

Keywords: singular value decomposition; federated learning; secure aggregation; differential privacy



Citation: Liu, B.; Pejó, B.; Tang, Q. Privacy-Preserving Federated Singular Value Decomposition. *Appl. Sci.* **2023**, *13*, 7373. <https://doi.org/10.3390/app13137373>

Academic Editor: Luis Javier Garcia Villalba

Received: 11 April 2023

Revised: 18 June 2023

Accepted: 18 June 2023

Published: 21 June 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Advances in networking and hardware technology have led to the rapid proliferation of the Internet of Things (IoTs) and decentralized applications. These advancements, including fog computing and edge computing technologies, enable data processing and analysis to be performed at node devices, avoiding the need for data aggregation. This naturally brings benefits such as efficiency and privacy, but on the other hand, it forces data analysis tasks to be carried out in a distributed manner. To this end, federated learning (FL) has emerged as a promising solution in this context, allowing multiple parties to collaboratively train models without sharing raw data. Instead, only intermediate results are exchanged with an aggregator server, ensuring privacy preservation and decentralized data analysis [1].

With respect to machine learning tasks, research has shown that sensitive information can be leaked from the models [2–5]. For example, in [3], Shokri et al. demonstrated membership inference attacks against machine learning tasks. In such an attack, an attacker can determine whether a data sample has been used in the model training. This will violate privacy if the data sample is sensitive. Regardless of its privacy friendly status, FL suffers similar privacy issues, as demonstrated by Nasr, Shokri and Houmansadr [5]. This makes it necessary to incorporate additional privacy protection mechanisms into FL and to make it rigorously privacy-preserving.

To mitigate information leakages, FL can be aided with other privacy-enhancing technologies, such as secure aggregation (SA) [6] and differential privacy (DP) [7]. SA hides

the individual contributions from the aggregator server in each intermediate step in a way that does not affect the trained model's utility. In other words, the standalone updates are masked such that the masks cancel out during aggregation; therefore, the aggregated results remain intact. The masks could be seen as temporary noise; hence, the privacy protection does not extend to the aggregated data. In contrast, DP adds persistent noise to the model, i.e., it provides broader privacy protection but with an inevitable utility loss (due to the permanent noise). We differentiate between two DP settings depending on where the noise is injected. In local DP (LDP), the participants add noise to their updates, while in central DP (CDP), the server applies noise to the aggregate result. A comparison of LDP, CDP and SA is summarized in Table 1. While there are many privacy protection mechanisms, incorporating them into FL is not a trivial task and remains as open challenges [1].

Table 1. Comparing secure aggregation with local and central differential privacy.

Protects	The Individual Updates	The Aggregate
Secure Aggregation	✓	✗
Central DP	✗	✓
Local DP	✓	✓

Among many data analysis methods, this paper focuses on singular value decomposition (SVD). Plainly, SVD factorizes a matrix into three new matrices. Originating from linear algebra, SVD has several interesting properties and conveys crucial insights about the underlying matrix. Hence, SVD has essential applications in data science, such as in recommendation systems [8,9], principal component analysis [10], latent semantic analysis [11], noise filtering [12,13], dimension reduction [14], clustering [15], matrix completion [16], etc. Existing federated SVD solutions fall into two categories: SVD over horizontally and vertically partitioned datasets [17]. In real-world applications, the former is much more common [18,19]; therefore, in this paper, we choose the horizontal setting and focus on the privacy protection challenges.

1.1. Related Work

The concept of privacy-preserving federated SVD has been studied in several works, which are briefly summarized below.

In the literature, many anonymization techniques have been proposed to enable privacy protection in federated machine learning and other tasks. Ref. [20] proposed substitute vectors and length-based frequent pattern tree (LFP-tree) to achieve the data anonymization. It focuses on what data can be published and how they can be published without associating subjects or identities. With the concept of data anonymization in mind, Ref. [21] proposed a strategy by decreasing the correlation between data and the identities. However, the utility of the data will be affected. And, Ref. [22] focused on high-dimensional dataset, which is divided into different subsets; then, each subset is generalized with a novel heuristic method based on local re-coding. While these works contain interesting techniques, they do not directly offer a solution for privacy-preserving federated SVD. A more detailed analysis can be found in [1].

Technically speaking, the algorithms utilized to compute SVD are mostly iterative, such as the power iteration method [23]. Recently, these algorithms were adopted to a distributed setting to solve large-scale problems [24,25]. While these works tackle important issues and advance the field, they all disregard privacy issues: we are only aware of two federated SVD solutions in the literature explicitly providing a privacy analysis [18,19]. Hartebrodt et al. [19] proposed a federated SVD algorithm with a star-like architecture for high-dimensional data such that the aggregator cannot access the complete eigenvector matrix of SVD results. Instead, each node device has access, but only to its shared part of the eigenvector matrix. In addition to the lack of a rigorous privacy analysis, its aim is different from most other federated SVD solutions where the aim is to jointly compute a global feature space. In contrast, Guo et al. [18] proposed a federated SVD algorithm based

on the distributed power method, where both the server and all the participants learn the entire eigenvector matrix. Their solution incorporated additional privacy-preserving features, such as participant and aggregator server noise injection, but without a rigorous privacy analysis. We improve upon this solution by pointing out an error in its privacy analysis and by providing a tighter privacy protection with less utilized noise. Overall, these existing literature works do not provide a privacy-preserving federated SVD solution with a rigorous analysis in our setting.

1.2. Contribution and Organization

This work focuses on a setting similar to Guo et al. [18], i.e., when the server and all the participants are expected to learn the final eigenvector matrix. As our main contribution, we improve the FedPower algorithm [18] from two perspectives, i.e., both from the privacy and utility points of view. Our detailed contributions are summarized below.

- Firstly, we point out several inefficiencies and shortcomings of FedPower, such as the avoidable double noise injection steps and the unclear and confusing privacy guarantee.
- Secondly, we propose a utility enhanced solution, where the added noise is reduced due to the introduction of SA.
- Thirdly, we propose a privacy enhanced solution, which (in contrast to FedPower) satisfies DP.
- Finally, we empirically validate our proposed algorithms by measuring the privacy-utility trade-off using a real-world recommendation system use-case.

The rest of the paper is organized as follows. In Section 2, we list the fundamental definitions of the relevant techniques used throughout the paper. In Section 3, we recap the scheme proposed by Guo et al. [18], while in Sections 4 and 5, we propose two improved schemes focusing on utility and privacy, respectively. In Section 6, we empirically compare the proposed schemes with the original work. Finally, in Section 7, we conclude the paper.

2. Preliminary

2.1. Singular Value Decomposition

Let \mathbb{M} be a $s \times d$ matrix with assumption of $s \leq d$. As shown in Figure 1, the full SVD of \mathbb{M} is a factorization of the form $\mathbb{U}\Sigma\mathbb{V}^T$, where T means conjugate transpose. The left-singular vectors are $\mathbb{U} = [u_1, u_2, \dots, u_s] \in \mathbb{R}^{s \times s}$, the right-singular vectors are $\mathbb{V} = [v_1, v_2, \dots, v_d] \in \mathbb{R}^{d \times d}$, and the diagonal matrix with the singular values in decreasing order in its diagonal is $\Sigma = \text{diag}\{\sigma_1, \sigma_1, \dots, \sigma_d\} \in \mathbb{R}^{s \times d}$. The partial or truncated SVD [26,27] is used to find the top k ($k \leq d$) singular vectors $\mathbb{U} = [u_1, u_2, \dots, u_k]$, $\mathbb{V} = [v_1, v_2, \dots, v_k]$ and singular values $\Sigma = \text{diag}\{\sigma_1, \sigma_1, \dots, \sigma_k\}$.

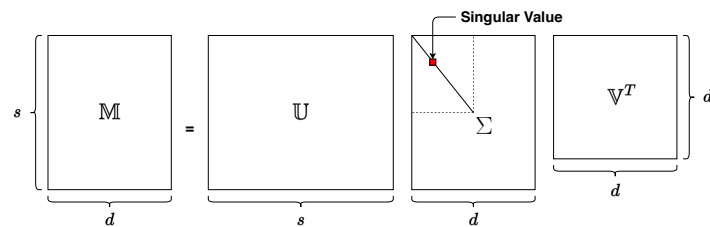


Figure 1. Singular value decomposition.

If $\mathbb{M}' = \frac{1}{s}\mathbb{M}^T\mathbb{M} \in \mathbb{R}^{d \times d}$, then the Power Method [23] could be used to compute the top k right singular vector of \mathbb{M} and the top k eigenvectors of \mathbb{M}' . It works by iterating $\mathbb{Y} = \mathbb{M}'\mathbb{Z}$ and $\mathbb{Z} = \text{orth}(\mathbb{Y})$, where both \mathbb{Y} and \mathbb{Z} are $d \times k$ matrices and $\text{orth}(\cdot)$ is the orthogonalization of the columns with QR factorization.

Moreover, if \mathbb{M} is the composition of n matrices, then computation of the Power Method can be distributed. So, if $\mathbb{M}^T = [\mathbb{M}_1^T, \mathbb{M}_2^T, \dots, \mathbb{M}_n^T] \in \mathbb{R}^{s \times d}$ with $s = \sum_{i=1}^n s_i$, where $\mathbb{M}_i \in \mathbb{R}^{s_i \times d}$ and $\mathbb{M}'_i = \frac{1}{s_i}\mathbb{M}_i^T\mathbb{M}_i$, then Equation (1) holds. Thereby, \mathbb{Y} can be written as

$\mathbb{Y} = \sum_{i=1}^n \frac{s_i}{s} \mathbb{M}'_i \mathbb{Z} \in \mathbb{R}^{d \times k}$, which indicates that the Power Method can be processed in parallel by each data holder [18,28].

$$\mathbb{M}' = \frac{1}{s} \mathbb{M}^T \mathbb{M} = \sum_{i=1}^n \frac{1}{s} \mathbb{M}_i^T \mathbb{M}_i = \sum_{i=1}^n \frac{s_i}{s} \mathbb{M}'_i = \sum_{i=1}^d p_i \mathbb{M}'_i \tag{1}$$

2.2. Secure Aggregation

In simple terms, with SA, the original data of each node device are locally masked in a particular way and shared with the server, so when the masked data are aggregated on the server, the masks are canceled and offset. In contrast, the server does not know all individual node devices' original unmasked intermediate results. In the FL literature, many solutions have widely used the SA protocol of Bonawitz et al. [29]. We recap this protocol in Appendix A and use it in Section 4 to benchmark our enhanced SVD solution.

2.3. Differential Privacy

Besides SA, DP is also exhaustively utilized in the FL literature. DP was introduced by Dwork et al. [30], which ensures that the addition, removal, or modification of a single data point does not substantially affect the outcome of the data-based analysis. One of the core strengths of DP comes from its properties, called composition and post-processing, which we also utilize in this paper. The former ensures that the output of two DP mechanisms still satisfies DP but with a parameter change. The latter ensures that a transformation of the results of a DP mechanism does not affect the corresponding privacy guarantees. Typically, DP is enforced by injecting calibrated noise (e.g., Laplacian or Gaussian) into the computation.

Definition 1 ((ϵ, δ)-Differential Privacy). *A randomized mechanism $\mathcal{M} : \mathcal{X} \rightarrow \mathcal{R}$ with domain \mathcal{X} and range \mathcal{R} satisfies ϵ -differential privacy if for any two adjacent inputs $x, x' \in \mathcal{X}$ and for any subset of output $S \subseteq \mathcal{R}$ it holds that*

$$\Pr(\mathcal{M}(x) \in S) \leq e^\epsilon \cdot \Pr(\mathcal{M}(x') \in S) \tag{2}$$

The variable ϵ is called the privacy budget, which measures the privacy loss. It captures the trade-off between privacy and utility: the lower its value, the more noise is required to satisfy Equation (2), resulting in higher utility loss. Another widely used DP notion is approximate DP, where a small additive term δ is added to the right side of Equation (2). Typically, we are interested in values of δ that are smaller than the inverse of the database size. Although DP has been adopted to many domains [7] such as recommendation systems [31], we are not aware of any work besides [18] which adopts DP for SVD computation. Thus, as we later show a flaw in that work, we are the first to provide a distributed SVD computation with DP guarantees.

3. The FedPower Algorithm

Following Guo et al. [18], we assume there are n node devices, and each device i holds an independent dataset, an s_i -by- d matrix \mathbb{M}_i . Each row represents a record item, while the columns of each matrix correspond to the same feature space. Moreover, \mathbb{M} denotes the composition of matrices \mathbb{M}_i such that $\mathbb{M}^T = [\mathbb{M}_1^T, \mathbb{M}_2^T, \dots, \mathbb{M}_n^T] \in \mathbb{R}^{s \times d}$, with $s = \sum_{i=1}^n s_i$. The solution proposed by Guo et al. [18] is presented in Algorithm 1 with the following parameters.

- T : the number of local computations performed by each node device.
- \mathcal{I}_T^p : the rounds where the node devices and the server communicate, i.e., $\mathcal{I}_T^p = \{0, p, 2p, \dots, p \lfloor T/p \rfloor\}$.
- (ϵ, δ) : the privacy budget.
- (σ, σ') : the variance of noises added by the clients and the server, respectively:

$$\sigma = \frac{\lfloor T/p \rfloor}{\varepsilon \cdot \min_i(s_i)} \sqrt{2 \log\left(\frac{1.25 \lfloor T/p \rfloor}{\delta}\right)} \quad \sigma' = \frac{\lfloor T/p \rfloor \max_i(p_i)}{\varepsilon \cdot \min_i(s_i)} \sqrt{2 \log\left(\frac{1.25 \lfloor T/p \rfloor}{\delta}\right)}$$

Algorithm 1 Full participation protocol of FedPower by Guo et al. [18].

Input: Datasets $\{\mathbb{M}_i\}_{i=1}^n$, target rank k , iteration rank $r \geq k$, number of iteration T , synchronous set \mathcal{I}_T^p , and the variance of noises (σ, σ')

Output: Approximated eigenspace $\bar{\mathbb{Z}}_T$

- 1: initialise $\mathbb{Z}_0^{(i)} = \mathbb{Z}_0 \in \mathbb{R}^{d \times r} \sim \mathcal{N}(0, 1)^{d \times r}$
- 2: **for** $t = 1$ to T **do**
- 3: each node device i computes $\mathbb{Y}_t^{(i)} = \mathbb{M}'_i \mathbb{Z}_{t-1}^{(i)}$, where $\mathbb{M}'_i = \frac{1}{s_i} \mathbb{M}_i^T \mathbb{M}_i$
- 4: **if** $t \in \mathcal{I}_T^p$ **then**
- 5: each node device i computes $\hat{\mathbb{Y}}_t^{(i)} = \mathbb{Y}_t^{(i)} \mathbb{D}_t^{(i)}$ (orthogonal transformation)
- 6: each node device i adds the Gaussian noise:
 $\mathbb{Y}'_t^{(i)} = \hat{\mathbb{Y}}_t^{(i)} + \mathbb{N}^{(i)} \sim \mathcal{N}(0, \|\mathbb{Z}_{t-1}^{(i)}\|_{\max}^2 \sigma^2)^{d \times r}$
- 7: each node device i sends $\mathbb{Y}'_t^{(i)}$ to the server
- 8: the server performs perturbed aggregation with an extra Gaussian noise:
 $\mathbb{Y}_t = \sum_{i=1}^n \frac{s_i}{s} \mathbb{Y}'_t^{(i)} + \mathbb{N} \sim \mathcal{N}(0, \max_i \|\mathbb{Z}_{t-1}^{(i)} \mathbb{D}_t^{(i)}\|_{\max}^2 \sigma'^2)^{d \times r}$
- 9: the server broadcasts \mathbb{Y}_t to all node devices
- 10: each node device i sets $\mathbb{Y}_t^{(i)} = \mathbb{Y}_t$
- 11: **end if**
- 12: each node device i performs orthogonalization: $\mathbb{Z}_t^{(i)} = \text{orth}(\mathbb{Y}_t^{(i)})$
- 13: **end for**
- 14: **return** approximated eigenspace

$$\bar{\mathbb{Z}}_T = \begin{cases} \sum_{i=1}^n \frac{s_i}{s} \mathbb{Z}_T^{(i)} \mathbb{D}_{T+1}^{(i)} & \text{if } T \notin \mathcal{I}_T^p \\ \sum_{i=1}^n \frac{s_i}{s} \mathbb{Z}_T^{(i)} & \text{otherwise.} \end{cases}$$

In the proposed solution, each node device holds its raw data and processes the SVD locally, its eigenvectors are aggregated on the server by the orthogonal procrustes transformation (OPT) mechanism. The basic idea behind this is to find an orthogonal transformation matrix that maps one set onto another while preserving their relative characteristics. And, the aggregation result is sent back for further iterations. More details (e.g., the computation of $\mathbb{D}_t^{(i)}$) are given in [18].

4. Enhancing the Utility of FedPower

Adversary Model. Throughout this paper, we consider a semi-honest setup, i.e., where the clients and the server are honest but curious. This means that they follow the protocol truthfully, but in the meantime, they try to learn as much as possible about the dataset of other participants. We also assume that the server and the clients cannot collude, so the server cannot control node devices.

Utility Analysis of FedPower. It is not a surprise that adding Gaussian noise twice (i.e., the local and the central noise in Step 6 and 8 in Algorithm 1) severely affects the accuracy of the final result. A straightforward way to increase the utility is to eliminate some of this noise. As highlighted in Table 1, the local noise protects the individual clients from the server. Moreover, it also protects the aggregate from other clients and from external attackers. On the other hand, the central noise merely covers the aggregate. Hence, if the protection level against the server is sufficient against other clients and external attackers, the central noise becomes obsolete.

Moreover, all the locally added noise accumulates during aggregation, which also negatively affects the utility of the final result. Loosely speaking, as shown in Table 1, CDP combined with SA could provide the same protection as LDP. Consequently, by utilizing

cryptographic techniques with a single local noise, we can hide the individual updates and protect the aggregate as well.

Utility Enhanced FedPower. We improve on FedPower [18] from two aspects: (1) we apply an SA protocol to hide the individual intermediate results of the node devices from the server, and (2) we use a secure multi-party computation (SMPC) protocol to enforce the CDP in an oblivious manner to the server. In SMPC, multiple parties can jointly compute a function over their private inputs without revealing those inputs to each other or to the server. More details of this topic can be found in the book [32]. We supplement the assumptions and the setup of Guo et al. [18] with a homomorphic encryption key pair generated by the server. The server holds the private key and shares the public key with all node devices. The remaining part of our solution is shown in Algorithm 2. To ease understanding, the pseudo code is simplified. The actual implementation is more optimized, e.g., the encrypted results are aggregated before decryption in Step 11, and in Step 7, the ciphertexts are re-randomized rather than generated from scratch. We describe all these tricks in Section 6.

By performing SA in Step 7, the server obtains the aggregated result with Gaussian noises from all node devices. With the simple SMPC procedure (Steps 8–12), the server receives all Gaussian noises apart from the one (i.e., node device j) is randomly selected (which is hidden from the node devices). Then, in Step 13, it removes them from the output of the SA protocol. Compared with FedPower [18], our intermediate aggregation result only contains a single instance of Gaussian noise from the randomly chosen node device instead of n . Consequently, via SA and SMPC, the proposed utility-enhancing protocol reduced the locally added noise n -fold and completely eliminated the central noise.

Computational Complexity. Regarding computational complexity, we compare the proposed scheme with the original solution in Table 2. The major difference is that we have integrated SA to facilitate our new privacy protection strategy. Let SA_e and SA_s be the asymptotic computational complexities of SA on each node device and server side, respectively.

Table 2. Complexity comparison between FedPower [18] and Algorithm 2.

		Addition	Multiplication	Noise Gen.	Encryption	Decryption	Secure Agg.
[18]	Node	$T \times (k^2 - k) + \lfloor T/p \rfloor \times k^2$	$T \times k^2$	$\lfloor T/p \rfloor \times k^2$			
	Server	$(\lfloor T/p \rfloor + 1) \times k^2 \times (d - 1) + \lfloor T/p \rfloor$	$(\lfloor T/p \rfloor + 1) \times d \times k^2 + \lfloor T/p \rfloor \times d + 1$	$\lfloor T/p \rfloor \times k^2$			
Ours	Node	$T \times (k^2 - k) + \lfloor T/p \rfloor \times k^2$	$T \times k^2 + \lfloor T/p \rfloor \times k^2$	$\lfloor T/p \rfloor \times k^2$			$\lfloor T/p \rfloor \times SA_e$
	Server	$\lfloor T/p \rfloor \times k^2 \times d + k^2 \times (d - 1)$	$d \times (k^2 + 1)$		$\lfloor T/p \rfloor \times k^2 \times m$	$\lfloor T/p \rfloor \times k^2 \times m$	$\lfloor T/p \rfloor \times SA_s$

Although we have added more operations, as seen in Table 2, we have distributed some computations to individual node devices. Most importantly, we no longer add secondary server-side Gaussian noise to the final aggregation result and only retain the Gaussian noise from one node device.

Analysis. As we mentioned in our adversarial model, the semi-honest server cannot collude with any of the node devices, which are also semi-honest. Thus, the server cannot eliminate the remaining noise from the final result. In terms of the node device, since no one except the server is aware of the random index in Step 8, apart from its data, a node device only knows the aggregation result with the added noise, even if the retained noise comes from itself.

Algorithm 2 Utility-enhanced FedPower.

Input: Datasets $\{\mathbb{M}_i\}_{i=1}^n$, target rank k , iteration rank r , number of iteration T , synchronous trigger p , the variance of noise σ , and key pair (sk_{hm}, pk_{hm})

Output: Approximated eigenspace $\bar{\mathbb{Z}}_T$

- 1: initialise $\mathbb{Z}_0^{(i)} = \mathbb{Z}_0 \in \mathbb{R}^{d \times r} \sim \mathcal{N}(0, 1)^{d \times r}$ with orthonormal columns and generate an $r \times r$ zero matrix \mathbb{P} and another all-ones matrix \mathbb{P}' of the same size
- 2: **for** $t = 1$ to T **do**
- 3: each node device i computes $\mathbb{Y}_t^{(i)} = \mathbb{M}'_i \mathbb{Z}_{t-1}^{(i)}$, where $\mathbb{M}'_i = \frac{1}{s_i} \mathbb{M}_i^T \mathbb{M}_i$
- 4: **if** $t \equiv 0 \pmod{p}$ **then**
- 5: each node device i computes $\hat{\mathbb{Y}}_t^{(i)} = \mathbb{Y}_t^{(i)} \mathbb{D}_t^{(i)}$ (orthogonal transformation)
- 6: each node device i adds Gaussian noise: $\mathbb{Y}'_t^{(i)} = \hat{\mathbb{Y}}_t^{(i)} + \mathbb{N}^{(i)} \sim \mathcal{N}(0, \sigma)^{d \times r}$
- 7: SA protocol is executed among the server and all node devices, with inputs $\mathbb{Y}'_t^{(i)}$ and output \mathbb{Y}_t
- 8: the server chooses one random index $j \in [1, n]$ and encrypts \mathbb{P} and \mathbb{P}' : $\mathbb{C}^{(j)} = \text{Enc}_{pk_h}(\mathbb{P})$ and $\mathbb{C}^{(j')} = \text{Enc}_{pk_h}(\mathbb{P}')$ for $j' \in [1, n] \setminus \{j\}$
- 9: the server sends value $\mathbb{C}^{(j)}$ and $\mathbb{C}^{(j')}$ to the appropriate node devices
- 10: each node device i computes $\mathbb{C}'^{(i)} = \mathbb{N}^{(i)} \cdot \mathbb{C}^{(i)}$ which is $\text{Enc}_{pk_h}(\mathbb{N}^{(i)} \cdot \mathbb{P}')$ if $i = j$ and $\text{Enc}_{pk_h}(\mathbb{N}^{(i)} \cdot \mathbb{P})$ otherwise
- 11: each node device i sends $\mathbb{C}'^{(i)}$ back to the server
- 12: for all $i \in [1, n] \setminus \{j\}$, the server decrypts the receiving messages $\mathbb{C}'^{(i)}$ to obtain $\mathbb{N}^{(i)} \equiv \mathbb{N}^{(i)} \cdot \mathbb{P}' = \text{Dec}_{sk_h}(\mathbb{C}'^{(i)})$
- 13: the server updates aggregation result as $\mathbb{Y}''_t = \mathbb{Y}'_t - \sum_{i \in [1, n] \setminus \{j\}} \mathbb{N}^{(i)}$
- 14: the server performs orthogonalization $\mathbb{Z}_t = \text{orth}(\mathbb{Y}''_t)$
- 15: the server broadcasts \mathbb{Z}_t to all node devices
- 16: each node device i sets $\mathbb{Z}_t^{(i)} = \mathbb{Z}_t$
- 17: **else**
- 18: each node device i calculates the latest $\mathbb{Z}_t^{(i)} = \text{orth}(\mathbb{Y}_t^{(i)})$
- 19: **end if**
- 20: **end for**
- 21: **return** approximated eigenspace

$$\bar{\mathbb{Z}}_T = \begin{cases} \sum_{i=1}^n \frac{s_i}{s} \mathbb{Z}_T^{(i)} \mathbb{D}_{T+1}^{(i)} & \text{if } T \notin \mathcal{I}_T^p \\ \sum_{i=1}^n \frac{s_i}{s} \mathbb{Z}_T^{(i)} & \text{otherwise.} \end{cases}$$

Compared with the original solution by Guo et al. [18], we have improved the utility of the aggregation result by keeping the added noise from one single node device. As a side effect, the complexity has grown due to the SA protocol. This is a trade-off between result accuracy and solution efficiency.

5. Differentially Private Federated SVD Solution

Privacy Analysis of FedPower. Algorithm 1 injects noise both at the local (Step 6) and the global (Step 8) levels. Consequently, the claimed privacy protection of Algorithm 1 is $(2\epsilon, 2\delta)$ -DP, which originates from (ϵ, δ) -LDP and (ϵ, δ) -CDP [18]. Firstly, as we highlighted in Table 1, LDP and CDP provide different privacy protections; hence, merely combining them is inappropriate, so the claim must be more precise. Instead, Algorithm 1 seems to provide (ϵ, δ) -DP for the clients from the server and stronger protection (due to the additional central noise) from other clients and external attackers.

Yet, this is still not entirely sound, as not all computations were included in the sensitivity calculation; hence, the noise scaling is incorrect. Indeed, the authors only considered the sensitivity of the multiplication with \mathbb{Z} in Step 3 when determining the variance of the

Gaussian noise in Step 6; however, the noise is only added after the multiplication with \mathbb{D} in Step 5. Thus, the sensitivity of the orthogonalization is discarded.

Privacy-Enhanced FedPower. We improve on FedPower [18] from two aspects: (1) we incorporate clipping in the protocol to bound the sensitivity of the local operations performed by the clients and (2) we use SA with DP to obtain a strong privacy guarantee. For this reason, similar to FedPower [18], we assume that for all i the elements of $\mathbb{M}'_i = \frac{1}{s_i} \mathbb{M}'_i{}^T \mathbb{M}_i$ are bounded with \hat{m} . In Algorithm 1, the computations the nodes undertake (besides noise injection at Step 6) are in Steps 3, 5 and 12, where the last two could be either discarded for the sensitivity computation or completely removed, as explained below.

- Step 12: Orthogonalization is intricate, so its sensitivity is not necessarily traceable. To tackle this, we propose applying the noise before, in which case it would not affect the privacy guarantee, as it would count as post-processing.
- Step 5: We remove this client-side operation from our privacy-enhanced solution, as it is not essential; only the convergence speed would be affected slightly.

The FedPower protocol with enhanced privacy is present in Algorithm 3, where besides the orthogonalization, clipping is also performed with \hat{z} . The only client operation which must be considered for the sensitivity computation (i.e., before noise injection) is Step 3. We calculate its sensitivity in Theorem 1.

Theorem 1. *If we assume $|m'_{ij}| \leq \hat{m}$ for all $i, j \in [1, d]$, then the sensitivity (calculated via the Euclidean distance) of the client-side operations (i.e., Step 3 in Algorithm 3 is bounded by $2 \cdot \sqrt{r} \cdot \hat{m} \cdot \hat{z}$.*

Proof. To make the proof easier to follow, we remove the subscript round counter from the notation. Let us define \mathbb{M}' and $\tilde{\mathbb{M}}'$ such that they are equal except at position $1 \leq i, j \leq d$. Now, multiply these with \mathbb{Z} from the left results in \mathbb{Y} and $\tilde{\mathbb{Y}}$, respectively, which are the same except in row i :

$$\begin{aligned}
 & [m'_{i1} \cdot z_{11} + \cdots + \overbrace{m'_{ij} \cdot z_{j1} + \cdots + m'_{id} \cdot z_{d1}}^{abs(\cdot) \leq \hat{m} \cdot \hat{z}}, \dots, m'_{i1} \cdot z_{1r} + \cdots + \overbrace{m'_{ij} \cdot z_{jr} + \cdots + m'_{id} \cdot z_{dr}}^{abs(\cdot) \leq \hat{m} \cdot \hat{z}}] \text{ for } \mathbb{Y}' \\
 & [m'_{i1} \cdot z_{11} + \cdots + \overbrace{\tilde{m}'_{ij} \cdot z_{j1} + \cdots + m'_{id} \cdot z_{d1}}^{abs(\cdot) \leq \hat{m} \cdot \hat{z}}, \dots, m'_{i1} \cdot z_{1r} + \cdots + \overbrace{\tilde{m}'_{ij} \cdot z_{jr} + \cdots + m'_{id} \cdot z_{dr}}^{abs(\cdot) \leq \hat{m} \cdot \hat{z}}] \text{ for } \tilde{\mathbb{Y}}'
 \end{aligned}$$

Hence, the Euclidean distance of \mathbb{Y} and $\tilde{\mathbb{Y}}$ boils down to this row i :

$$dist(\mathbb{Y}, \tilde{\mathbb{Y}}) = \sqrt{\sum_{k=1}^d \sum_{l=1}^r (y_{kl} - \tilde{y}_{kl})^2} = \sqrt{\sum_{l=1}^r (y_{il} - \tilde{y}_{il})^2} = \sqrt{\sum_{l=1}^r (m'_{ij} \cdot z_{jl} - \tilde{m}'_{ij} \cdot z_{jl})^2}$$

As a direct corollary of $abs(m \cdot z) \leq \hat{m} \cdot \hat{z}$, we know that each of the r squared elements is bounded by $2 \cdot \hat{m} \cdot \hat{z}$. Therefore, $dist(\mathbb{Y}, \tilde{\mathbb{Y}}) \leq \sqrt{r \cdot 4 \cdot \hat{m}^2 \cdot \hat{z}^2}$. \square

It is known that adding Gaussian noise with $\sigma^2 = \frac{2 \cdot s^2 \log(1.25/\delta)}{\epsilon^2}$ (where s is the sensitivity) results in (ϵ, δ) -DP. As a corollary, we can state in Theorem 2 that a single round in Algorithm 3 is differentially private. An even tighter result was presented in [33]; we leave the exploration of this as future work. The best practice is to set δ as the inverse of the size of the underlying dataset, so there is a direct connection between the variance σ and the privacy parameter ϵ .

Theorem 2. *If $T = 1$, then Algorithm 3 provides (ϵ, δ) -DP, where*

$$\epsilon = \frac{\sqrt{8 \cdot r \cdot \log(1.25/\delta)} \cdot \hat{m} \cdot \hat{z}}{\sigma}$$

Proof. Can be verified by combining the provided formula with the appropriate sensitivity. \square

Algorithm 3 Privacy-enhanced FedPower.

Input: Datasets $\{\mathbb{M}_i\}_{i=1}^n$, target rank k , iteration rank r , number of iteration T , the clipping bound \hat{z} , the variance of noise σ

Output: Approximated eigenspace $\bar{\mathbb{Z}}_T$

- 1: initialise $\mathbb{Z}_0^{(i)} = \mathbb{Z}_0 \in \mathbb{R}^{d \times r} \sim \mathcal{N}(0, 1)^{d \times r}$ with orthonormal columns
- 2: **for** $t = 1$ to T **do**
- 3: each node device i computes $\mathbb{Y}_t^{(i)} = \mathbb{M}'_i \mathbb{Z}_{t-1}^{(i)}$, where $\mathbb{M}'_i = \frac{1}{s_i} \mathbb{M}_i^T \mathbb{M}_i$
- 4: each node device i adds Gaussian noise: $\mathbb{Y}_t^{(i)} = \mathbb{Y}_t^{(i)} + \mathbb{N}^{(i)} \sim \mathcal{N}(0, \sigma)^{d \times r}$
- 5: **if** $t \equiv 0 \pmod{p}$ **then**
- 6: SA protocol is executed among the server and all node devices, with inputs $\mathbb{Y}_t^{(i)}$ and output \mathbb{Y}_t
- 7: the server performs orthogonalization and clipping $\mathbb{Z}_t = \text{clip}(\text{orth}(\mathbb{Y}_t'), \hat{z})$
- 8: the server broadcasts \mathbb{Z}_t to all node devices
- 9: each node device i sets $\mathbb{Z}_t^{(i)} = \mathbb{Z}_t$
- 10: **else**
- 11: each node device i calculates the latest $\mathbb{Z}_t^{(i)} = \text{clip}(\text{orth}(\mathbb{Y}_t^{(i)}), \hat{z})$
- 12: **end if**
- 13: **end for**
- 14: **return** approximated eigenspace

$$\bar{\mathbb{Z}}_T = \begin{cases} \sum_{i=1}^n \frac{s_i}{s} \mathbb{Z}_T^{(i)} \mathbb{D}_{T+1}^{(i)} & \text{if } T \notin \mathcal{I}_T^p \\ \sum_{i=1}^n \frac{s_i}{s} \mathbb{Z}_T^{(i)} & \text{otherwise.} \end{cases}$$

One can easily extend this result for $T \geq 1$ with the composition property of DP: Algorithm 3 satisfies $(T \cdot \epsilon, T \cdot \delta)$ -DP. Besides this basic loose composition, one can obtain better results by utilizing more involved composition theorems such as in [34]. We leave this for future work.

Analysis. Similarly to Section 4, we protect the individual intermediate results with SA. On the other hand, it is equivalent to generate n Gaussian noise with variance σ and select one, or to generate n Gaussian noise with variance $\frac{\sigma}{n}$ and sum them all up. Consequently, instead of relying on an SMPC protocol to eliminate most of the local noise, we could merely scale them down. combining SA with such a downsized local noise is, in fact, a common practice in FL: this is what distributed differential privacy (DDP) [35] does, i.e., DDP combined with SA provides LDP but with n times smaller noise, where n is the number of participants.

6. Empirical Comparison

In order to compare our proposed schemes with FedPower, we implement the schemes in Python [36]. As we only encrypt 0 and 1 in Section 4, we optimize the performance and take advantage of the utilized Paillier cryptosystem. More specifically, we re-randomize the corresponding ciphertexts to obtain new ciphertexts. In addition, we also exploit the homomorphic property, and instead of decrypting each value ($d \times r \times |\text{number of node devices}|$ times), we first calculate the product of all the ciphertexts (elementary matrix multiplication) and then perform the decryption on a signal matrix. In this way, we obtain the sum of all Gaussian noises more efficiently. The decryption result is the sum of noise which will be canceled in Algorithm 2. Furthermore, we prepare the $\mathbb{M}'_i = \frac{1}{s_i} \mathbb{M}_i^T \mathbb{M}_i$, $\mathbb{Z}_0^{(i)}$ and all keys of SA offline for each node device i .

Metric. We use Euclidean distance to represent the similarity of two $m \times n$ matrix $\mathbb{A} = (a_{ij})$ and $\mathbb{B} = (b_{ij})$, i.e., $\text{dist}(\mathbb{A}, \mathbb{B}) = \sqrt{\sum_{i=1}^m \sum_{j=1}^n (a_{ij} - b_{ij})^2}$. Let \mathbb{Z} denote the true eigenspace computed without any noise, let $\mathbb{Z}_g(\sigma, \sigma')$ denote the eigenspace generated with Algorithm 1, let $\mathbb{Z}_u(\sigma)$ denote the eigenspace generated with Algorithm 2, and let $\mathbb{Z}_p(\sigma)$ denote the eigenspace generated with Algorithm 3.

Setup. For our experiments, we used the well-known NETFLIX rating dataset [37], and we pre-process it similarly to [38] (instead of 10, we removed users and movies with less than 50 ratings). It consists of 96.310.835 ratings corresponding to 17.711 movies from 324.468 users. We split them horizontally into 100 random blocks to simulate node devices. Moreover, we set the security parameter to 128; thus, we adopt 3072 bits for N in Paillier cryptosystem (this is equivalent to RSA-3072, which provides a 128-bit security level [39]). The number of iteration rank and top eigenvectors is set to $r = k = 10$, and we keep the same synchronous trigger $p = 4$ as [18]. To compare FedPower with our enhanced solutions, we set the noise size for these algorithms as $\sigma = \sigma' = 0.1$. Moreover, for Algorithm 3 we bounded \mathbb{M}_i^t with 0.05 and $\mathbb{Z}_t^{(i)}$ with 0.2 for all possible i and t . Using Theorem 2, we can calculate that a single round corresponds to privacy budget $\epsilon = 30.6$ with $\delta = 10^{-5}$.

In order to determine the number of global rounds T , we set up a small experiment. We built a data matrix \mathbb{M} of size 3000×100 filled with integers in $[0, 5]$, and randomly divided it for 100 node devices (each has at least 10 rows). We executed Algorithm 1 for 200 rounds and compared the distance between the aggregation result \mathbb{Z} and the real singular values of \mathbb{M} . From the result in Figure 2, we can see that convergence occurs around the round 92, since the subsequent results vary only slightly ($<1\%$). Thus, we set $T = 92$ for our experiments.

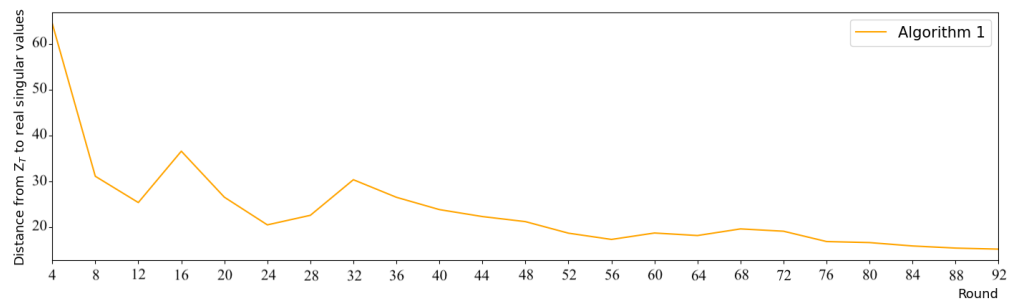


Figure 2. Experiment to determining T (with Algorithm 1).

The experiment is implemented in a Docker container of 40-core Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz and 755G RAM. We run our experiments 10-fold and take the average execution time.

Results. Firstly, we compare the efficiency of our enhanced schemes and the original algorithm. The computation times are presented in Table 3. Compared with FedPower, the overall computation burden of the devices increased by a factor of $\times 39.68$ for the utility-enhanced solution in Section 4 and only $\times 1.74$ and the privacy-enhanced solution in Section 5. Concerning the server, the increase is $\times 6.97$ and $\times 1.17$, respectively.

Table 3. Runtime comparison of Algorithms 1–3 in milliseconds.

Name	Device	Computation Time			Running Time ¹
		Aggr.	SMPC	Rest	
FedPower	Node	164	-	18,768	2.698×10^6
	Server	37	-	127,449	
Utility Enhanced	Node	15,829	7.182×10^5	17,259	9.203×10^7
	Server	20,647	7.247×10^5	143,145	
Privacy Enhanced	Node	15,742	-	17,291	4.477×10^6
	Server	20,581	-	128,903	

¹ Due to the large volume of memory required for matrix calculations, we had to access data by reading and writing local files, which caused the longer overall execution time.

The rise in computational demand comes with benefits. Concerning Algorithm 2, significant progress is achieved in the utility while it offers a similar privacy guarantee as FedPower. Concerning Algorithm 3, the privacy guarantee is more robust, as it provides a formal DDP protection (while FedPower fails to satisfy DP). Moreover, it obtains a higher utility, which could make this solution preferable despite its computational appeal. We compare the distance between the results of each algorithm and the real eigenvalues, as shown in Figure 3, and the utility is improved (i.e., the distances are lower) with both Algorithms 2 and 3.

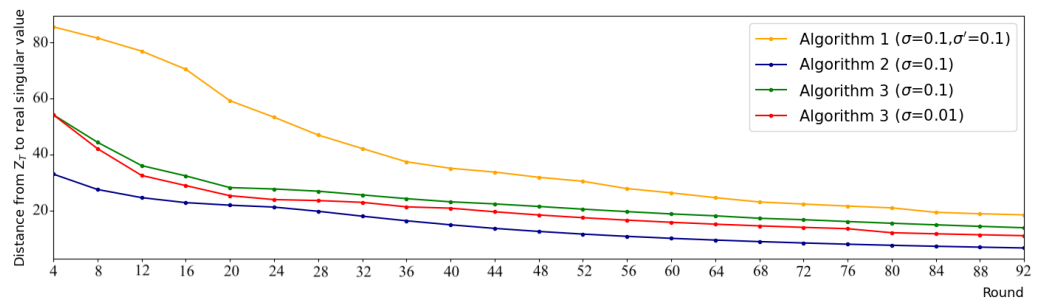


Figure 3. Comparison of Eigenspaces calculated using Algorithms 1–3 with different settings.

Our utility-enhanced solution significantly outperforms FedPower: after 92 rounds, the obtained error of our scheme is almost three times ($2.74\times$) smaller than that for FedPower. The final error of Algorithm 2 is $dist(\mathbb{Z}, \mathbb{Z}_u(\sigma)) = 6.72$, while this value for Algorithm 1 is $dist(\mathbb{Z}, \mathbb{Z}_g(\sigma, \sigma')) = 18.42$. Note that this level of accuracy (~ 18.5) was obtained using our method in the 32nd round, i.e., almost three times ($2.88\times$) faster. Hence, the superior convergence speed can compensate for most of the computational increase caused by SA and SMPC.

Let us shift our attention to our privacy-enhanced solution. In that case, we can see that besides more robust privacy protection, our solution offers better utility: Algorithms 1 and 3 obtains $dist(\mathbb{Z}, \mathbb{Z}_g(\sigma, \sigma')) = 18.42$ and $dist(\mathbb{Z}, \mathbb{Z}_p(\sigma)) = 13.94$ RMSE values, respectively, i.e., we acquired a 24% error reduction. Our method (with actual DP guarantees) achieved the same level of accuracy (~ 18.5) only after 65 rounds, which is a 29% convergence speed increase.

We also compare our two proposed schemes, in a way, that the size of the accumulated noises is equal. Besides the nature of noise injection (many small vs. one large), the only factor that differentiates the results is the clipping bounds. As expected, the error is $1.65\times$ larger with clipping, i.e., $dist(\mathbb{Z}, \mathbb{Z}_p(\frac{\sigma}{10})) = 11.11$ compared with $dist(\mathbb{Z}, \mathbb{Z}_u(\sigma)) = 6.72$. Concerning the convergence speed, the utility enhanced solution is $1.7\times$ faster, reaching similar accuracy (~ 11) in round 54. Note though that this result still vastly outperforms FedPower: the accuracy and the convergence speed are increased by 40% and 43%, respectively.

Finally, we study the effect of different levels of privacy protection on the accuracy of each algorithm. As we noticed in Figure 3, after the 60th round, the error ratios of the algorithms are reasonably stable, so for this experiment, we set $T = 60$. Since the clipping rate \hat{z} and the noise variance σ both contributed to the privacy parameter ϵ (as seen in Theorem 2), we varied each independently. Our results are presented in Figure 4. It is visible that the previously seen trends hold with other levels of privacy protection, making our proposed schemes favorable for a wide range of settings.

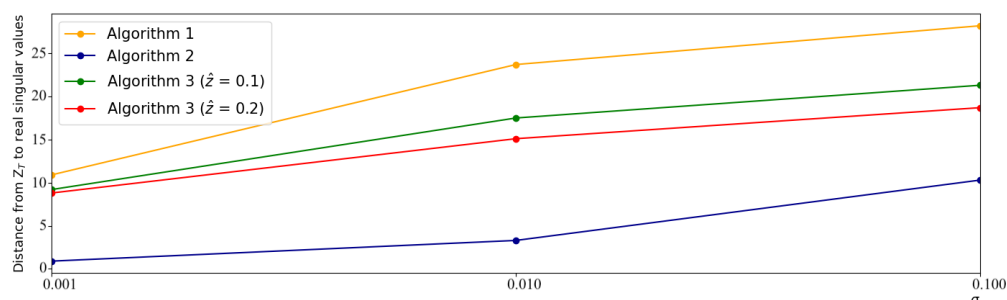


Figure 4. The effect of various privacy parameters on the accuracy for Algorithms 1–3.

7. Conclusions

Motivated by Guo et al.’s distributed privacy-preserving SVD algorithm based on the federated power method [18], we have proposed two enhanced federated SVD schemes, focusing on utility and privacy, respectively. Both use secure aggregation to reduce the added noise, which reverts to the initial design intent and interest. Yet, the added cryptographic operations trade efficiency for superior performance ($\times 10$ better results) while providing either similar or superior privacy guarantee. Our work leaves several future research topics. One is to further investigate the computational complexity, particularly the secure aggregation, to achieve more efficient solutions. Another is to investigate the scalability of the proposed solutions, regarding larger datasets and different datasets in applications other than recommendation systems. In addition, scalability also concerns the number of node devices. Yet, another topic is to look further into the security assumptions. For example, the security assumptions can be weaker so that the server can be allowed to collude with one or more node devices.

Author Contributions: Conceptualization, Q.T.; methodology, Q.T., B.L. and B.P.; software, B.L.; validation, Q.T., B.L. and B.P.; formal analysis, Q.T., B.L. and B.P.; investigation, Q.T., B.L. and B.P.; resources, Q.T.; data curation, B.L.; writing—original draft preparation, Q.T. and B.L.; writing—review and editing, Q.T., B.L. and B.P.; visualization, Q.T., B.L. and B.P.; supervision, Q.T.; project administration, Q.T.; funding acquisition, Q.T., B.L. and B.P. All authors have read and agreed to the published version of the manuscript.

Funding: Bowen Liu and Qiang Tang are supported by the 5G-INSIGHT bi-lateral project (ANR-20-CE25-0015-16) funded by the Luxembourg National Research Fund (FNR) and by the French National Research Agency (ANR). Balázs Pejó is supported by Project no. 138903, which has been implemented with the support provided by the Ministry of Innovation and Technology from the NRDI Fund and financed under the FK_21 funding scheme.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Practical Secure Aggregation

The practical secure aggregation by Bonawitz et al. [29] is summarized below. First and foremost, the following parameters are generated during the setup phase and sent to relevant node devices.

- Pseudorandom Generator (PRG) [40,41]: PRG takes a fixed length seed as the input and outputs in space $[0, R)$, where R is a prefixed value.
- Secret Sharing [42]: $SS.share(s, t, \mathcal{U}) \rightarrow \{(u, s_u)\}_{u \in \mathcal{U}}$ takes a secret s , a set of user IDs (e.g. integers), and a threshold $s \leq |\mathcal{U}|$ as the input and outputs a set of shares s_u associated with the user $u \in \mathcal{U}$; a reconstruction algorithm $SS.recon(\{(u, s_u)\}_{u \in \mathcal{V}}, t) \rightarrow$

s takes threshold t as an input and shares the corresponding inputs to a user subset $\mathcal{V} \subseteq \mathcal{U}$ such that $|\mathcal{V}| \geq t$, and outputs a field element s .

- Key Agreement [43]: $\text{KA.param}(k) \rightarrow pp$ takes a security parameter k and returns some public parameters; $\text{KA.gen}(pp) \rightarrow (s^{SK}, s^{PK})$ generates a secret/public key pair; $\text{KA.agree}(s_u^{SK}, s_v^{PK}) \rightarrow s_{u,v}$ allows a user u to combine its private key with the public key of another user v into a private shared key between them.
- Authenticated Encryption [44]: AE.enc and AE.dec are algorithms for encrypting a plaintext with a public key and for decrypting a ciphertext with a secret key.
- Signature Scheme [45]: SIG.gen takes a security parameter k and outputs a secret/public key pair; SIG.sign signs a message with a secret key and returns the relevant signature; and SIG.ver verifies the signature of the relevant message and returns a boolean bit indicating whether the signature is valid.
- Number of node devices m .
- Security parameter k .
- Public parameter of key agreement $pp \leftarrow \text{KA.param}(k)$.
- Threshold value t , where $t < n$ and n is the number of node devices.
- Input space \mathbb{Z}_R .
- Secrets sharing field \mathbb{F} .
- Signature key pairs (d_u^{SK}, d_u^{PK}) of each node device, where $u \in [1, m]$.

The complete execution of the protocol between node devices and the server is provided in the following.

- Round 0 (AdvertiseKeys):
 - 0.1. Each node device u generates secret/public key pairs of encryption and sharing algorithms (c_u^{SK}, c_u^{PK}) and (s_u^{SK}, s_u^{PK}) .
 - 0.2. Each node device u signs c_u^{PK} and s_u^{PK} into $\sigma_u \leftarrow \text{SIG.sign}(d_u^{SK}, c_u^{PK} || s_u^{PK})$.
 - 0.3. The two public keys and all n signatures $(c_u^{PK} || s_u^{PK} || \sigma_u)$ are sent to the server.
 - 0.4. If the server receives at least t messages from individual node devices (denote by \mathcal{U}_1 this set of node devices), then broadcast $\{(v, c_v^{PK}, s_v^{PK}, \sigma_v)\}_{v \in \mathcal{U}_1}$ to all node devices in \mathcal{U}_1 .
- Round 1 (ShareKeys):
 - 1.1. Once a node device u in \mathcal{U}_1 receives the messages from the server, it verifies if all signatures are valid with $\text{SIG.ver}(d_u^{PK}, c_u^{PK} || s_u^{PK}, \sigma_u)$, where $u \in \mathcal{U}_1$.
 - 1.2. The node device u samples a random element $b_u \leftarrow \mathbb{F}$ as a seed for a PRG.
 - 1.3. The node device u generates two t -out-of- $|\mathcal{U}_1|$ shares of $s_u^{SK} : \{(v, s_{u,v}^{SK})\}_{v \in \mathcal{U}_1} \leftarrow \text{SS.share}(s_u^{SK}, t, \mathcal{U}_1)$ and $b_u : \{(v, b_{u,v})\}_{v \in \mathcal{U}_1} \leftarrow \text{SS.share}(b_u, t, \mathcal{U}_1)$.
 - 1.4. For each node device $v \in \mathcal{U}_1 \setminus \{u\}$, u computes $e_{u,v} \leftarrow \text{AE.enc}(\text{KA.agree}(c_u^{SK}, c_v^{PK}), u || v || s_{u,v}^{SK} || b_{u,v})$ and sends them to the server.
 - 1.5. If the server receives at least t messages from individual node devices (denoted by $\mathcal{U}_2 \subseteq \mathcal{U}_1$ this set of node devices), then it shares to each node device $u \in \mathcal{U}_2$ all ciphertexts for it $\{e_{u,v}\}_{v \in \mathcal{U}_2}$.
- Round 2 (MaskedInputCollection):
 - 2.1. For the node device $u \in \mathcal{U}_2$, once the ciphertexts are received, it computes $s_{u,v} \leftarrow \text{KA.agree}(s_u^{SK}, s_v^{PK})$, where $v \in \mathcal{U}_2 \setminus \{u\}$.
 - 2.2. $s_{u,v}$ is expanded using PRG into a random vector $p_{u,v} = \Delta_{u,v} \cdot \text{PRG}(s_{u,v})$, where $\Delta_{u,v} = 1$ when $u > v$ and $\Delta_{u,v} = -1$ when $u < v$; moreover, define $p_{u,u} = 0$.
 - 2.3. The node device u computes its own private mask vector $p_u = \text{PRG}(b_u)$ and the masked input vector x_u into $y_u \leftarrow x_u + p_u + \sum_{v \in \mathcal{U}_2} p_{u,v} \pmod R$; then, y_u is sent to the server.
 - 2.4. If the server receives at least t messages (denote with $\mathcal{U}_3 \subseteq \mathcal{U}_2$ this set of node devices), share the node device set \mathcal{U}_3 with all node devices in \mathcal{U}_3 .
- Round 3 (ConsistencyCheck):

- 3.1. Once the node device $u \in \mathcal{U}_3$ receives the message, it returns the signature $\sigma'_u \leftarrow \text{SIG.sign}(d_u^{SK}, \mathcal{U}_3)$.
- 3.2. If the server receives at least t messages (denote by $\mathcal{U}_4 \subseteq \mathcal{U}_3$ this set of node devices), share the set $\{u', \sigma'_{u'}\}_{u' \in \mathcal{U}_4}$.
- Round 4 (Unmasking):
 - 4.1. Each node device u verifies $\text{SIG.ver}(d_v^{PK}, \mathcal{U}_3, \sigma'_v)$ for all $v \in \mathcal{U}_4$
 - 4.2. For each node device $v \in \mathcal{U}_2 \setminus \{u\}$, u decrypts the ciphertext (received in the MaskedInputCollection round) $v' || u' ||_{s_{v,u}} || b_{v,u} \leftarrow \text{AE.dec}(\text{KA.agree}(c_u^{SK}, c_v^{PK}), e_{v,u})$ and asserts that $u' = u \wedge v' = v$.
 - 4.3. Each node device u sends the shares $s_{v,u}^{SK}$ for node devices $v \in \mathcal{U}_2 \setminus \mathcal{U}_3$ and $b_{v,u}$ for node devices in $v \in \mathcal{U}_3$ to the server.
 - 4.4. If the server receives at least t messages (denote with \mathcal{U}_5 this set of node devices), it re-constructs, for each node device $u \in \mathcal{U}_2 \setminus \mathcal{U}_3$, $s_u^{SK} \leftarrow \text{SS.recon}(\{s_{u,v}^{SK}\}_{v \in \mathcal{U}_5}, t)$ and re-computes $p_{v,u}$ using PRG for all $v \in \mathcal{U}_3$.
 - 4.5. The server also re-constructs, for all node devices $u \in \mathcal{U}_3$, $b_u \leftarrow \text{SS.recon}(\{b_{u,v}\}_{v \in \mathcal{U}_5}, t)$ and re-computes $p_{v,u}$ using the PRG.
 - 4.6. Finally, the server outputs $z = \sum_{u \in \mathcal{U}_3} x_u = \sum_{u \in \mathcal{U}_3} y_u - \sum_{u \in \mathcal{U}_3} p_u + \sum_{u \in \mathcal{U}_3, v \in \mathcal{U}_2 \setminus \mathcal{U}_3} p_{v,u}$.

We summarize the asymptotic computational complexity of each node device and the server in Table A1. For simplicity of description, we assume that all devices participate in the protocol, that is, $t = m$. Since some operations can be considered as offline pre-configuration, we focus on online operations starting from masking messages in Step 2.3.

Table A1. Asymptotic computational complexity of online operations.

	Vector Add	SIG.sign	SIG.vef	KA.agree	AE.dec	SS.recon	PRG
Node	$m + 1$	1	$m - 1$	$m - 1$	$m - 1$		1
Server	$2m - 1$					m	m

References

1. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.* **2021**, *14*, 1–210. [CrossRef]
2. Fredrikson, M.; Jha, S.; Ristenpart, T. Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015.
3. Shokri, R.; Stronati, M.; Song, C.; Shmatikov, V. Membership inference attacks against machine learning models. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–24 May 2017.
4. Zhu, L.; Liu, Z.; Han, S. Deep leakage from gradients. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019.
5. Nasr, M.; Shokri, R.; Houmansadr, A. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In Proceedings of the 2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, 19–23 May 2019; pp. 739–753.
6. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
7. Pejó, B.; Desfontaines, D. *Guide to Differential Privacy Modifications: A Taxonomy of Variants and Extensions*; Springer Nature: Berlin/Heidelberg, Germany, 2022.
8. Polat, H.; Du, W. SVD-based collaborative filtering with privacy. In Proceedings of the 2005 ACM Symposium on Applied computing, Santa Fe, NM, USA, 13–17 March 2005; pp. 791–795.
9. Zhang, S.; Wang, W.; Ford, J.; Makedon, F.; Pearlman, J. Using singular value decomposition approximation for collaborative filtering. In Proceedings of the Seventh IEEE International Conference on E-Commerce Technology (CEC'05), Munich, Germany, 19–22 July 2005; pp. 257–264.
10. Wold, S.; Esbensen, K.; Geladi, P. Principal component analysis. *Chemom. Intell. Lab. Syst.* **1987**, *2*, 37–52. [CrossRef]
11. Dumais, S. T. Latent semantic analysis. *Annu. Rev. Inf. Sci. Technol.* **2004**, *38*, 188–230. [CrossRef]

12. Guo, Q.; Zhang, C.; Zhang, Y.; Liu, H. An efficient SVD-based method for image denoising. *IEEE Trans. Circuits Syst. Video Technol.* **2015**, *26*, 868–880. [CrossRef]
13. Rajwade, A.; Rangarajan, A.; Banerjee, A. Image denoising using the higher order singular value decomposition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *35*, 849–862. [CrossRef] [PubMed]
14. Ravi Kanth, K.; Agrawal, D.; Singh, A. Dimensionality reduction for similarity searching in dynamic databases. *ACM SIGMOD Rec.* **1998**, *27*, 166–176. [CrossRef]
15. Von Luxburg, U. A tutorial on spectral clustering. *Stat. Comput.* **2007**, *17*, 395–416. [CrossRef]
16. Candès, E.J.; Recht, B. Exact matrix completion via convex optimization. *Found. Comput. Math.* **2009**, *9*, 717–772. [CrossRef]
17. Yakut, I.; Polat, H. Privacy-preserving SVD-based collaborative filtering on partitioned data. *Int. J. Inf. Technol. Decis. Mak.* **2010**, *9*, 473–502. [CrossRef]
18. Guo, X.; Li, X.; Chang, X.; Wang, S.; Zhang, Z. Privacy-preserving distributed SVD via federated power. *arXiv* **2021**, arXiv:2103.00704.
19. Hartebrodt, A.; Röttger, R.; Blumenthal, D.B. Federated singular value decomposition for high dimensional data. *arXiv* **2022**, arXiv:2205.12109.
20. Eom, C.S.H.; Lee, C.C.; Lee, W.; Leung, C.K. Effective privacy preserving data publishing by vectorization. *Inf. Sci.* **2020**, *527*, 311–328. [CrossRef]
21. Caruccio, L.; Desiato, D.; Polese, G.; Tortora, G.; Zannone, N. A decision-support framework for data anonymization with application to machine learning processes. *Inf. Sci.* **2022**, *613*, 1–32. [CrossRef]
22. Wang, R.; Zhu, Y.; Chang, C.C.; Peng, Q. Privacy-preserving high-dimensional data publishing for classification. *Comput. Secur.* **2020**, *93*, 101785. [CrossRef]
23. Golub, G.H.; Van Loan, C.F. *Matrix Computations*; JHU Press: Baltimore, MD, USA, 2013.
24. Fan, J.; Wang, D.; Wang, K.; Zhu, Z. Distributed estimation of principal eigenspaces. *Ann. Stat.* **2019**, *47*, 3009. [CrossRef]
25. Chen, X.; Lee, J.D.; Li, H.; Yang, Y. Distributed estimation for principal component analysis: An enlarged eigenspace analysis. *J. Am. Stat. Assoc.* **2022**, *117*, 1775–1786. [CrossRef]
26. Eckart, C.; Young, G. The approximation of one matrix by another of lower rank. *Psychometrika* **1936**, *1*, 211–218. [CrossRef]
27. Stewart, G.W. On the early history of the singular value decomposition. *SIAM Rev.* **1993**, *35*, 551–566. [CrossRef]
28. Arbenz, P.; Kressner, D.; Zürich, D. Lecture notes on solving large scale eigenvalue problems. *D-MATH EHT Zur.* **2012**, *2*, 3.
29. Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H.B.; Patel, S.; Ramage, D.; Segal, A.; Seth, K. Practical secure aggregation for privacy-preserving machine learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1175–1191.
30. Dwork, C.; McSherry, F.; Nissim, K.; Smith, A. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference, Proceedings of the Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, 4–7 March 2006*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 265–284.
31. Guerraoui, R.; Kermarrec, A.M.; Patra, R.; Taziki, M. D 2 p: Distance-based differential privacy in recommenders. *Proc. VLDB Endow.* **2015**, *8*, 862–873. [CrossRef]
32. Ronald Cramer, Ivan Bjerre Damgård, J.B.N. *Secure Multiparty Computation and Secret Sharing*; Cambridge University Press: Cambridge, UK, 2015.
33. Balle, B.; Wang, Y.X. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 394–403.
34. Kairouz, P.; Oh, S.; Viswanath, P. The composition theorem for differential privacy. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 1376–1385.
35. Truex, S.; Baracaldo, N.; Anwar, A.; Steinke, T.; Ludwig, H.; Zhang, R.; Zhou, Y. A hybrid approach to privacy-preserving federated learning. In Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, London, UK, 15 November 2019; pp. 1–11.
36. LIU, B. Implementation Source Code. Available online: <https://github.com/MoienBowen/Privacy-preserving-Federated-Singular-Value-Decomposition> (accessed on 10 April 2023).
37. Bennett, J.; Lanning, S. The netflix prize. In *Proceedings of the KDD Cup and Workshop*; Association for Computing Machinery: New York, NY, USA, 2007; p. 35.
38. Pejo, B.; Tang, Q.; Biczok, G. Together or alone: The price of privacy in collaborative learning. *Proc. Priv. Enhancing Technol.* **2019**, *2019*, 47–65. [CrossRef]
39. Barker, E. *Recommendation for Key Management: Part 1—General*; Technical Report NIST Special Publication (SP) 800-57, Rev. 5; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2020. [CrossRef]
40. Blum, M.; Micali, S. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM J. Comput.* **1984**, *13*, 850–864. [CrossRef]
41. Yao, A.C. Theory and application of trapdoor functions. In Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (SFOCS 1982), Chicago, IL, USA, 3–5 November 1982; pp. 80–91.
42. Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 612–613. [CrossRef]
43. Diffie, W.; Hellman, M. New directions in cryptography. *IEEE Trans. Inf. Theory* **1976**, *22*, 644–654. [CrossRef]

44. McGrew, D.; Viega, J. The Galois/counter mode of operation (GCM). *Submiss. NIST Modes Oper. Process* **2004**, *20*, 0278–0070.
45. Bellare, M.; Neven, G. Transitive signatures: New schemes and proofs. *IEEE Trans. Inf. Theory* **2005**, *51*, 2133–2151. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.