Distributed Differentially-Private Algorithms for Matrix and Tensor Factorization

Hafiz Imtiaz, *Student Member, IEEE,* and Anand D. Sarwate, *Senior Member, IEEE*

***Abstract*—In many signal processing and machine learning applications, datasets containing private information are held at different locations, requiring the development of distributed privacy-preserving algorithms. Tensor and matrix factorizations are key components of many processing pipelines. In the distributed setting, differentially private algorithms suffer because they introduce noise to guarantee privacy. This paper designs new and improved distributed and differentially private algorithms for two popular matrix and tensor factorization methods: principal component analysis (PCA) and orthogonal tensor decomposition (OTD). The new algorithms employ a correlated noise design scheme to alleviate the effects of noise and can achieve the same noise level as the centralized scenario. Experiments on synthetic and real data illustrate the regimes in which the correlated noise allows performance matching with the centralized setting, outperforming previous methods and demonstrating that meaningful utility is possible while guaranteeing differential privacy.**

***Index Terms*—Differential privacy, distributed orthogonal tensor decomposition, latent variable model, distributed principal component analysis**

I. Introduction

MANY signal processing and machine learning algorithms involve analyzing private or sensitive data. The outcomes of such algorithms may leak information about individuals present in the dataset. A strong and cryptographically-motivated framework for protection against such information leaks is differential privacy [1]. Differential privacy measures privacy risk in terms of the probability of identifying individual data points in a dataset from the results of computations (algorithms) performed on that data.

In several modern applications the data is distributed over different locations or sites, with each site holding a smaller number of samples. For example, consider neuro-imaging analyses for mental health disorders, in which there are many individual research groups, each with a modest number of subjects. Learning meaningful population properties or efficient feature representations from high-dimensional functional magnetic resonance imaging (fMRI) data requires a large sample size. Pooling the data at a central location may enable efficient feature learning, but privacy concerns and high communication overhead often prevent such sharing. Therefore, it is desirable to have efficient distributed privacy-preserving algorithms that provide utility close to centralized case [2].

This paper focuses on the Singular Value Decomposition (SVD), or Principal Component Analysis (PCA), and orthogonal tensor decompositions. Despite some limitations, PCA/SVD is one of the most widely-used preprocessing stages in any machine learning algorithm: it projects data onto a lower dimensional subspace spanned by the singular vectors of the sample second-moment matrix. Tensor decomposition is a powerful tool for inference algorithms because it can be used to infer complex dependencies (higher order moments) beyond second-moment methods such as PCA. This is particularly useful in latent variable models [3] such as mixtures of Gaussians and topic modeling.

Related Works. For a complete introduction to the history of tensor decompositions, see the comprehensive survey of Kolda and Bader [4]. The CANDECOMP/PARAFAC, or CP decomposition [5], [6] and Tucker decomposition [7] are generalizations of the matrix SVD to multi-way arrays. While finding the decomposition of arbitrary tensors is computationally intractable, specially structured tensors appear in some latent variable models. Such tensors can be decomposed efficiently [3], [4] using a variety of approaches such as generalizations of the power iteration [8]. Exploiting such structures in higher-order moments to estimate the parameters of latent variable models has been studied extensively using the so-called orthogonal tensor decomposition (OTD) [3], [9]–[11]. To our knowledge, these decompositions have not been studied in the setting of distributed data.

Several distributed PCA algorithms [12]–[17] have been proposed. Liang et al. [12] proposed a distributed PCA scheme where it is necessary to send both the left and right singular vectors along with corresponding singular values from each site to the aggregator. Feldman et al. [18] proposed an improvement upon this, where each site sends a $D \times R$ matrix to the aggregator. Balcan et al. [13] proposed a further improved version using fast sparse subspace embedding [19] and randomized SVD [20].

This paper proposes new privacy-preserving algorithms for distributed PCA and OTD and builds upon our earlier work on distributed differentially private eigenvector calculations [17] and centralized differentially private OTD [21]. It improves on our preliminary works on distributed private PCA [17], [22] in terms of efficiency and fault-tolerance. Wang and Anandkumar [23] recently proposed an algorithm for differentially private tensor decomposition using a noisy version of the tensor power iteration [3], [8]. Their algorithm adds noise at each step of the iteration and the noise variance grows with the predetermined number of iterations. They also make

H. Imtiaz and A. D. Sarwate are with the Department of Electrical and Computer Engineering, Rutgers, The State University of New Jersey, Piscataway, NJ 08854 USA. e-mail: hafiz.imtiaz@rutgers.edu, anand.sarwate@rutgers.edu.

This work was supported in part by the United States National Institutes of Health under award 1R01DA040487-01A1, the United States National Science Foundation under award CCF-1453432, and DARPA and SSC Pacific under contract N66001-15-C-4070.

Manuscript received April xx, 2018; revised xxxx xx, 20xx.

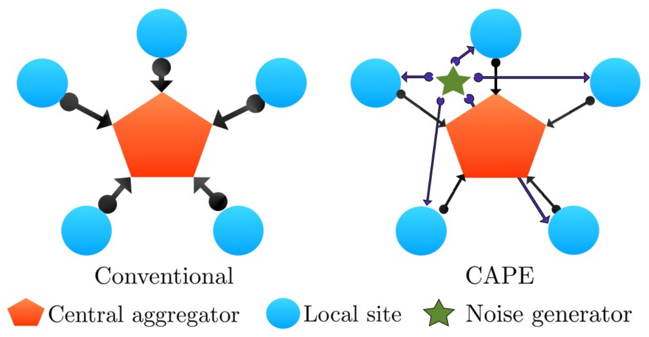

Fig. 1. The structure of the network: left – conventional, right – CAPE

the restrictive assumption that the input to their algorithm is orthogonally decomposable. Our centralized OTD algorithms [21] avoid these assumptions and achieve better empirical performance (although without theoretical guarantees). To our knowledge, this paper proposes the first differentially private OTD algorithm for distributed settings.

Our Contribution. In this paper, we propose two new (ϵ, δ)-differentially private algorithms, capePCA and capeAGN, for distributed differentially private PCA and OTD, respectively. The algorithms are inspired by the recently proposed correlation assisted private estimation (CAPE) protocol [24] and input perturbation methods for differentially-private PCA [25], [26]. The CAPE protocol improves upon conventional approaches, which suffer from excessive noise, at the expense of requiring a trusted "helper" node that can generate correlated noise samples for privacy. We extend the CAPE framework to handle site-dependent sample sizes and privacy requirements. In capePCA, the sites share noisy second-moment matrix estimates to a central aggregator, whereas in capeAGN the sites use a distributed protocol to compute a projection subspace used to enable efficient private OTD. This paper is about algorithms with provable privacy guarantees and experimental validation. While asymptotic sample complexity guarantees are of theoretical interest, proving performance bounds for distributed subspace estimation is quite challenging. To validate our approach we show that our new methods outperform previously proposed approaches, even under strong privacy constraints. For weaker privacy requirements they can achieve the same performance as a pooled-data scenario.

II. Problems Using Distributed Private Data

Notation. We denote tensors with calligraphic scripts $(\mathcal{X})$, vectors with bold lower case letters $(\mathbf{x})$, and matrices with bold upper case letters $(\mathbf{X})$. Scalars are denoted with regular letters (M). Indices are denoted with lower case letters and they typically run from 1 to their upper-case versions $(m = 1, 2, \ldots, M)$. We sometimes denote the set $\{1, 2, \ldots, M\}$ as $[M]$. The n-th column of the matrix $\mathbf{X}$ is denoted as $\mathbf{x}_n$. $\|\cdot\|_2$, $\|\cdot\|_F$ and $\mathrm{tr}(\cdot)$ denote the Euclidean (or $\mathcal{L}_2$) norm of a vector and the spectral norm of a matrix, the Frobenius norm and the trace operation, respectively.

Distributed Data Model. We assume that the data is distributed in S sites, where each site $s \in [S]$ has a data matrix $\mathbf{X}_s \in \mathbb{R}^{D \times N_s}$. The data samples in the local sites are assumed to be disjoint. There is a central node that acts as an aggregator (see Figure 1). We denote $N = \sum_{s=1}^{S} N_s$ as the total number of samples over all sites. The data matrix $\mathbf{X}_s = [\mathbf{x}_{s,1} \ \ldots \ \mathbf{x}_{s,N_s}]$ at site s is considered to contain the D-dimensional features of N_s individuals. Without loss of generality, we assume that $\|\mathbf{x}_{s,n}\|_2 \leq 1 \ \forall s \in [S]$ and $\forall n \in [N_s]$. If we had all the data in the aggregator (pooled data scenario), then the data matrix would be $\mathbf{X} = [\mathbf{X}_1 \ \ldots \ \mathbf{X}_S] \in \mathbb{R}^{D \times N}$. Our goal is to approximate the performance of the pooled data scenario using distributed differentially private algorithms.

Matrix and Tensor Factorizations. We first formulate the problem of distributed PCA. For simplicity, we assume that the observed samples are mean-centered. The $D \times D$ sample second-moment matrix at site s is $\mathbf{A}_s = \frac{1}{N_s}\mathbf{X}_s\mathbf{X}_s^\top$. In the pooled data scenario, the $D \times D$ positive semi-definite second-moment matrix is $\mathbf{A} = \frac{1}{N}\mathbf{X}\mathbf{X}^\top$. According to the Schmidt approximation theorem [27], the rank-K matrix $\mathbf{A}_K$ that minimizes the difference $\|\mathbf{A} - \mathbf{A}_K\|_F$ can be found by taking the SVD of $\mathbf{A}$ as $\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$, where without loss of generality we assume $\mathbf{\Lambda}$ is a diagonal matrix with entries $\{\lambda_d(\mathbf{A})\}$ and $\lambda_1(\mathbf{A}) \geq \ldots \geq \lambda_D(\mathbf{A}) \geq 0$. Additionally, $\mathbf{V}$ is a matrix of eigenvectors corresponding to the eigenvalues. The top-K PCA subspace of $\mathbf{A}$ is the matrix $\mathbf{V}_K(\mathbf{A}) = [\mathbf{v}_1 \ldots \mathbf{v}_K]$. Given $\mathbf{V}_K(\mathbf{A})$ and the eigenvalue matrix $\mathbf{\Lambda}$, we can form an approximation $\mathbf{A}_K = \mathbf{V}_K(\mathbf{A})\mathbf{\Lambda}_K\mathbf{V}_K(\mathbf{A})^\top$ to $\mathbf{A}$, where $\mathbf{\Lambda}_K$ contains the K largest eigenvalues in $\mathbf{\Lambda}$. For a $D \times K$ matrix $\hat{\mathbf{V}}$ with orthonormal columns, the quality of $\hat{\mathbf{V}}$ in approximating $\mathbf{V}_K(\mathbf{A})$ can be measured by the *captured energy* of $\mathbf{A}$ as $q(\hat{\mathbf{V}}) = \mathrm{tr}(\hat{\mathbf{V}}^\top\mathbf{A}\hat{\mathbf{V}})$. The $\hat{\mathbf{V}}$, which maximizes $q(\hat{\mathbf{V}})$ is the subspace $\mathbf{V}_K(\mathbf{A})$. We are interested in approximating $\mathbf{V}_K(\mathbf{A})$ in a distributed setting while guaranteeing differential privacy.

Next, we consider the problem of OTD. We refer the reader to the survey by Kolda and Bader [4] for related basic definitions. As mentioned before, we consider the decomposition of symmetric tensors that appear in several latent variable models. Such tensors can be orthogonally decomposed efficiently. Two examples of OTD from Anandkumar et al. [3], namely the single topic model (STM) and the mixture of Gaussian (MOG), are presented in Appendix B.

Let $\mathcal{X}$ be an M-way D dimensional symmetric tensor. Given real valued vectors $\mathbf{v}_k \in \mathbb{R}^D$, Comon et al. [28] showed that there exists a decomposition of the form

$$\mathcal{X} = \sum_{k=1}^{K} \lambda_k \mathbf{v}_k \otimes \mathbf{v}_k \otimes \cdots \otimes \mathbf{v}_k,$$

where $\otimes$ denotes the outer product. Without loss of generality, we can assume that $\|\mathbf{v}_k\|_2 = 1 \ \forall k$. If we can find a matrix $\mathbf{V} = [\mathbf{v}_1 \ldots \mathbf{v}_K] \in \mathbb{R}^{D \times K}$ with orthogonal columns, then we say that $\mathcal{X}$ has an orthogonal symmetric tensor decomposition [11]. Such tensors are generated in several applications involving latent variable models. Recall that if $\mathbf{M} \in \mathbb{R}^{D \times D}$ is a symmetric rank-K matrix then we know that the SVD of $\mathbf{M}$ is given by

$$\mathbf{M} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top = \sum_{k=1}^{K} \lambda_k \mathbf{v}_k \mathbf{v}_k^\top = \sum_{k=1}^{K} \lambda_k \mathbf{v}_k \otimes \mathbf{v}_k,$$

where $\mathbf{\Lambda} = \mathrm{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_K\}$ and $\mathbf{v}_k$ is the k-th column of the orthogonal matrix $\mathbf{V}$. As mentioned before, the orthogonal decomposition of a 3-rd order symmetric tensor $\mathcal{X} \in \mathbb{R}^{D\times D\times D}$ is a collection of orthonormal vectors $\{\mathbf{v}_k\}$ together with corresponding positive scalars $\{\lambda_k\}$ such that $\mathcal{X} = \sum_{k=1}^{K} \lambda_k \mathbf{v}_k \otimes \mathbf{v}_k \otimes \mathbf{v}_k$. Now, in a setting where the data samples are distributed over different sites, we may have local approximates $\mathcal{X}_s$. We intend to use these local approximates from all sites to find better and more accurate estimates of the $\{\mathbf{v}_k\}$, while preserving privacy.

Differential Privacy. An algorithm $\mathcal{A}(\mathbb{D})$ taking values in a set $\mathbb{T}$ provides (ϵ, δ)-differential privacy if

$$\Pr[\mathcal{A}(\mathbb{D}) \in \mathbb{S}] \leq \exp(\epsilon) \Pr[\mathcal{A}(\mathbb{D}') \in \mathbb{S}] + \delta, \qquad (1)$$

for all measurable $\mathbb{S} \subseteq \mathbb{T}$ and all data sets $\mathbb{D}$ and $\mathbb{D}'$ differing in a single entry (neighboring datasets). This definition essentially states that the probability of the output of an algorithm is not changed significantly if the corresponding database input is changed by just one entry. Here, ϵ and δ are privacy parameters, where lower ϵ and δ ensure more privacy. Note that the parameter δ can be interpreted as the probability that the algorithm fails. For more details, see recent surveys [29] or the monograph of Dwork and Roth [30].

To illustrate, consider estimating the mean $f(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N} x_n$ of N scalars $\mathbf{x} = [x_1, \ldots, x_{N-1},\ x_N]^\top$ with each $x_i \in [0, 1]$. A neighboring data vector $\mathbf{x}' = [x_1, \ldots, x_{N-1},\ x'_N]^\top$ differs in a single element. The sensitivity [1] $\max_{\mathbf{x}} |f(\mathbf{x}) - f(\mathbf{x}')|$ of the function $f(\mathbf{x})$ is $\frac{1}{N}$. Therefore, for (ϵ, δ) differentially-private estimate of the average $a = f(\mathbf{x})$, we can follow the Gaussian mechanism [1] to release $\hat{a} = a + e$, where $e \sim \mathbb{N}\left(0, \tau^2\right)$ and $\tau = \frac{1}{N\epsilon}\sqrt{2\log\frac{1.25}{\delta}}$.

Distributed Privacy-preserving Computation. In our distributed setting, we assume that the sites are "honest but curious." That is, the aggregator is not trusted and the sites can collude to get a hold of some site's data/function output. Existing approaches to distributed differentially private algorithms can introduce a significant amount of noise to guarantee privacy. Returning to the example of mean estimation, suppose now there are S sites and each site s holds a disjoint dataset $\mathbf{x}_s$ of N_s samples for $s \in [S]$. A central aggregator wishes to estimate and publish the mean of all the samples. The sites can send estimates to the aggregator but may collude to learn the data of other sites based on the aggregator output. Without privacy, the sites can send $a_s = f(\mathbf{x}_s)$ to the aggregator and the average computed by aggregator ($a_{\text{ag}} = \frac{1}{S}\sum_{s=1}^{S} a_s$) is exactly equal to the average we would get if all the data samples were available in the aggregator node. For preserving privacy, a standard differentially private approach is for each site to send $\hat{a}_s = f(\mathbf{x}_s) + e_s$, where $e_s \sim \mathbb{N}\left(0, \tau_s^2\right)$ and $\tau_s = \frac{1}{N_s\epsilon}\sqrt{2\log\frac{1.25}{\delta}}$. The aggregator computes $a_{\text{ag}} = \frac{1}{S}\sum_{s=1}^{S}\hat{a}_s$. We observe

$$a_{\text{ag}} = \frac{1}{S}\sum_{s=1}^{S}\hat{a}_s = \frac{1}{S}\sum_{s=1}^{S} a_s + \frac{1}{S}\sum_{s=1}^{S} e_s.$$

Note that this estimate is still noisy due to the privacy constraint. The variance of the estimator a_{ag} is $S \cdot \dfrac{\tau_s^2}{S^2} = \dfrac{\tau_s^2}{S} \triangleq \tau_{\text{ag}}^2$. However, if we had all the data samples in the central aggregator, then we could compute the differentially-private average as $a_c = \frac{1}{N}\sum_{n=1}^{N} x_n + e_c$, where $e_c \sim \mathbb{N}\left(0, \tau_c^2\right)$ and $\tau_c = \frac{1}{N\epsilon}\sqrt{2\log\frac{1.25}{\delta}}$. If we assume that each site has equal number of samples then $N = SN_s$ and we have $\tau_c = \frac{1}{SN_s\epsilon}\sqrt{2\log\frac{1.25}{\delta}} = \dfrac{\tau_s}{S}$. We observe the ratio

$$\frac{\tau_c^2}{\tau_{\text{ag}}^2} = \frac{\tau_s^2\ /\ S^2}{\tau_s^2\ /\ S} = \frac{1}{S},$$

showing that the conventional differentially-private distributed averaging scheme is always worse than the differentially-private pooled data case.

III. Correlated Noise Scheme

Algorithm 1 Correlation Assisted Private Estimation (CAPE)

```
Require: Data samples $\{\mathbf{x}_s\}$; privacy parameters $\epsilon$, $\delta$.
 1: Compute $\tau_s \leftarrow \frac{1}{N_s\epsilon}\sqrt{2\log\frac{1.25}{\delta}}$
 2: At the random noise generator, generate $e_s \sim \mathcal{N}(0, \tau_e^2)$, where $\tau_e^2 = (1 - \frac{1}{S})\tau_s^2$ and $\sum_{s=1}^{S} e_s = 0$
 3: At the central aggregator, generate $f_s \sim \mathcal{N}(0, \tau_f^2)$, where $\tau_f^2 = (1 - \frac{1}{S})\tau_s^2$
 4: for $s = 1,\ \ldots,\ S$ do
 5:     Get $e_s$ from the random noise generator
 6:     Get $f_s$ from the central aggregator
 7:     Generate $g_s \sim \mathcal{N}(0, \tau_g^2)$, where $\tau_g^2 = \frac{\tau_s^2}{S}$
 8:     Compute and send $\hat{a}_s \leftarrow f(\mathbf{x}_s) + e_s + f_s + g_s$
 9: end for
10: At the central aggregator, compute $a_{\text{ag}}^{\text{imp}} \leftarrow \frac{1}{S}\sum_{s=1}^{S}\hat{a}_s - \frac{1}{S}\sum_{s=1}^{S} f_s$
11: return $a_{\text{ag}}^{\text{imp}}$
```

The recently proposed Correlation Assisted Private Estimation (CAPE) [24] scheme exploits the network structure and uses a correlated noise design to achieve the same performance of the pooled data case (i.e., $\tau_{\text{ag}} = \tau_c$) in the decentralized setting. We assume there is a trusted noise generator in addition to the central aggregator (see Figure 1). The local sites and the central aggregator can also generate noise. The noise generator and the aggregator can send noise to the sites through secure (encrypted) channels. The noise addition procedure is carefully designed to ensure the privacy of the algorithm output from each site and to achieve the noise level of the pooled data scenario in the final output from the central aggregator. Considering the same distributed averaging problem as in Section II, the noise generator and central aggregator send e_s and f_s, respectively, to each site s. Site s generates noise g_s and releases/sends $\hat{a}_s = f(\mathbf{x}_s) + e_s + f_s + g_s$. The noise generator generates e_s such that $\sum_{s=1}^{S} e_s = 0$. The term e_s is needed to protect $f(\mathbf{x}_s)$ from the aggregator, since the aggregator knows f_s and g_s is not large enough to protect $f(\mathbf{x}_s)$. The noise generator need not necessarily be a separate entity and can be considered as a common randomness, or a shared coin only possessed by the sites [24]. For example, each site could generate $\hat{e}_s$ and, perhaps using standard secure multiparty computation protocol [31], [32], compute $\sum_{s=1}^{S} \hat{e}_s$ collaboratively. Each site could then use $e_s \leftarrow \hat{e}_s - \frac{1}{S}\sum_{s=1}^{S} \hat{e}_s$ to achieve $\sum_{s=1}^{S} e_s = 0$. As shown in [24], the noise terms $\{e_s, g_s, f_s\}$ are distributed according to $e_s \sim \mathcal{N}(0, \tau_e^2)$, $f_s \sim \mathcal{N}(0, \tau_f^2)$, and $g_s \sim \mathcal{N}(0, \tau_g^2)$, where

$$\tau_e^2 = \tau_f^2 = \left(1 - \frac{1}{S}\right)\tau_s^2, \text{ and } \tau_g^2 = \frac{\tau_s^2}{S}. \qquad (2)$$

For a given pair of privacy parameters (ϵ, δ), we can calculate a noise variance τ_s^2 such that adding Gaussian noise of variance τ_s^2 will guarantee (ϵ, δ)-differential privacy. Since there are many (ϵ, δ) pairs that yield the same τ_s^2, it is convenient to parameterize our method using τ_s^2.

The noise variances of $\{e_s, g_s, f_s\}$ were derived to ensure that the variance of the noise $f_s + g_s$ guarantees (ϵ, δ)-differential privacy to the output from site s, since the noise terms e_s are correlated. Additionally, the noise variances ensure that the variance of $e_s + g_s$ is sufficient to provide (ϵ, δ)-differential privacy to the output from site s – as a safeguard against the central aggregator, which knows f_s. The aggregator computes

$$a_{\text{ag}}^{\text{imp}} = \frac{1}{S}\sum_{s=1}^{S}(\hat{a}_s - f_s) = \frac{1}{N}\sum_{n=1}^{N} x_n + \frac{1}{S}\sum_{s=1}^{S} g_s,$$

where we used $\sum_s e_s = 0$ and the fact that the aggregator knows the f_s, so it can subtract all of those from $\hat{a}_s$. The variance of the estimator $a_{\text{ag}}^{\text{imp}}$ is $S \cdot \frac{\tau_g^2}{S^2} = \frac{\tau_s^2}{S^2} = \tau_c^2$, which is the same as if all the data were present at the aggregator. This claim is formalized in Lemma 1.

Lemma 1. *Let the variances of the noise terms e_s, f_s and g_s (Step 8 of Algorithm 1) be given by* (2). *If we denote the variance of the additive noise (for preserving privacy) in the pooled data scenario by τ_c^2 and the variance of the estimator $a_{\text{ag}}^{\text{imp}}$ (Step 10 of Algorithm 1) by ${\tau_{\text{ag}}^{\text{imp}}}^2$ then Algorithm 1 achieves $\tau_c^2 = {\tau_{\text{ag}}^{\text{imp}}}^2$.*

Proof. We recall that in the pooled data scenario, the sensitivity of the function $f(\mathbf{x})$ is $\frac{1}{N}$, where $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_S]$. Therefore, to approximate $f(\mathbf{x})$ satisfying (ϵ, δ) differential privacy, we need to have additive Gaussian noise standard deviation at least $\tau_c = \frac{1}{N\epsilon}\sqrt{2\log\frac{1.25}{\delta}}$. Next, consider the (ϵ, δ) differentially-private release of the function $f(\mathbf{x}_s)$. The sensitivity of this function is $\frac{1}{N_s}$. Therefore, the (ϵ, δ) differentially-private approximate of the function $f(\mathbf{x}_s)$ requires a standard deviation at least $\tau_s = \frac{1}{N_s\epsilon}\sqrt{2\log\frac{1.25}{\delta}}$. Note that, if we assume equal number of samples in each site, then we have

$$\tau_c = \frac{\tau_s}{S} \implies \tau_c^2 = \frac{\tau_s^2}{S^2}.$$

We will now show that the CAPE algorithm will yield the same noise variance of the estimator at the aggregator. Recall that at the aggregator we compute $a_{\text{ag}}^{\text{imp}} = \frac{1}{S}\sum_{s=1}^{S}(\hat{a}_s - f_s) = \frac{1}{N}\sum_{n=1}^{N} x_n + \frac{1}{S}\sum_{s=1}^{S} g_s$. The variance of the estimator ${\tau_{\text{ag}}^{\text{imp}}}^2 \triangleq S \cdot \frac{\tau_g^2}{S^2} = \frac{\tau_g^2}{S} = \frac{\tau_s^2}{S^2}$, which is exactly the same as the pooled data scenario. Therefore, the CAPE algorithm allows us to achieve the same additive noise variance as the pooled data scenario, while satisfying at least (ϵ, δ) differential privacy at the sites and (ϵ, δ) differential privacy for the final output from the aggregator. □

We show the complete algorithm in Algorithm 1. Privacy follows from previous work [24], and if $S > 2$ and number of trusted sites (the sites that would not collude with any adversary) $S_{\text{tr}} \geq 2$, the aggregator does not need to generate f_s. Note that the CAPE protocol exploits the Gaussianity of the noise terms. Therefore, other mechanisms, e.g. the staircase mechanism (SM) [33], cannot be used in the current framework to take advantage of the correlated noise scheme. We investigated the performance of SM empirically in the conventional distributed setting and observed that the CAPE protocol always outperforms SM. We believe a very interesting future work would be to incorporate SM in the correlated noise scheme.

Proposition 1. *(Performance gain [24]) Consider the gain function $G(\mathbf{n}) = \frac{\tau_{\text{ag}}^2}{{\tau_{\text{ag}}^{\text{imp}}}^2} = \frac{N^2}{S^2}\sum_{s=1}^{S}\frac{1}{N_s^2}$ with $\mathbf{n} = [N_1, \ldots, N_S]$. Then:*

- *the minimum $G(\mathbf{n})$ is S and is achieved when $\mathbf{n} = \left[\frac{N}{S}, \ldots, \frac{N}{S}\right]$*
- *the maximum $G(\mathbf{n})$ is $\frac{N^2}{S^2}\left(\frac{1}{(N-S+1)^2} + S - 1\right)$, which occurs when $\mathbf{n} = [1, \ldots, 1, N - S + 1]$*

Proof. The proof is a consequence of Schur convexity and is given in [24]. □

The proposition suggests that the CAPE achieves a gain of at least S even when we do not know N_s of different sites. Moreover, in case of site drop-out, the performance of capePCA (Algorithm 2) and capeAGN (Algorithm 3) would fall back to that of the conventional scheme [24]. That is, the output from each site remains (ϵ, δ) differentially private irrespective of the number of dropped-out sites.

A. Extension of CAPE *to Unequal Privacy Requirements*

We now propose a generalization of the CAPE scheme, which applies to scenarios where different sites have different privacy requirements and/or sample sizes. Additionally, sites may have different "quality notions", i.e., while combining the site outputs at the aggregator, the aggregator can decide to use different weights to different sites (possibly according to the quality of the output from a site). Let us assume that site s requires (ϵ_s, δ_s)-differential privacy for its output. According to the Gaussian mechanism [1], the noise to be added to the (non-private) output of site s should have standard deviation given by $\tau_s = \frac{1}{N_s\epsilon_s}\sqrt{2\log\frac{1.25}{\delta_s}}$. Site s outputs $\hat{a}_s = f(\mathbf{x}_s) + e_s + f_s + g_s$. Here, $g_s \sim \mathcal{N}(0, \tau_{gs}^2)$ is generated locally; $e_s \sim \mathcal{N}(0, \tau_{es}^2)$ and $f_s \sim \mathcal{N}(0, \tau_{fs}^2)$ are generated from the noise generator and the aggregator, respectively. As explained before, we need to satisfy

$$\tau_{fs+gs}^2 = \tau_{fs}^2 + \tau_{gs}^2 \geq \tau_s^2, \text{ and } \tau_{es+gs}^2 = \tau_{es}^2 + \tau_{gs}^2 \geq \tau_s^2.$$

The aggregator computes a weighted average with weights selected according to the quality measure of the site's data/output (e.g., if the aggregator knows that a particular site is suffering from more noisy observations than other sites, it can choose to give the output from that site less weight while combining the site results). Let us denote the weights by $\{\mu_s\}$ such that $\sum_{s=1}^{S}\mu_s = 1$ and $\mu_s \geq 0$. Note that, our proposed generalized CAPE reduces to the existing [24] CAPE for $\mu_s = \frac{N_s}{N}$. The aggregator computes

$$a_{\text{ag}}^{\text{imp}} = \sum_{s=1}^{S}\mu_s\left(\hat{a}_s - f_s\right) = \sum_{s=1}^{S}\mu_s a_s + \sum_{s=1}^{S}\mu_s e_s + \sum_{s=1}^{S}\mu_s g_s.$$

As our goal is to achieve the same level of noise as the pooled data scenario, we need

$$\text{var}\left[\sum_{s=1}^{S}\mu_s g_s\right] = \tau_c^2 \implies \sum_{s=1}^{S}\mu_s^2\tau_{gs}^2 = \tau_c^2.$$

Additionally, we need $\sum_{s=1}^{S}\mu_s e_s = 0$. With these constraints, we can formulate a feasibility problem to solve for the unknown noise variances $\{\tau_{es}^2, \tau_{gs}^2, \tau_{fs}^2\}$ as

$$\begin{aligned}
\text{minimize} \quad & 0 \\
\text{subject to} \quad & \tau_{fs}^2 + \tau_{gs}^2 \geq \tau_s^2,\ \tau_{es}^2 + \tau_{gs}^2 \geq \tau_s^2, \\
& \sum_{s=1}^{S}\mu_s^2\tau_{gs}^2 = \tau_c^2, \sum_{s=1}^{S}\mu_s e_s = 0,
\end{aligned}$$

for all $s \in [S]$, where $\{\mu_s\}$, τ_c and $\{\tau_s\}$ are known to the aggregator. For this problem, multiple solutions are possible. We present one solution here that solves the problem with equality.

Solution. We start with $\sum_{s=1}^{S}\mu_s e_s = 0$. The noise generator generates e_s for $s \in [S-1]$ independently with $e_s \sim \mathcal{N}(0, \tau_{es}^2)$ and then sets $e_S = -\frac{1}{\mu_S}\sum_{s=1}^{S-1}\mu_s e_s$. As e_s for $s \in [S-1]$ are independent Gaussians, they are also uncorrelated. Therefore, we have

$$\tau_{eS}^2 = \frac{1}{\mu_S^2}\sum_{s=1}^{S-1}\mu_s^2\tau_{es}^2 \implies \sum_{s=1}^{S-1}\mu_s^2\tau_{es}^2 - \mu_S^2\tau_{eS}^2 = 0.$$

Additionally, we have $\sum_{s=1}^{S-1}\mu_s^2\tau_{gs}^2 + \mu_S^2\tau_{gS}^2 = \tau_c^2$. Combining these, we observe $\tau_{gS}^2 - \tau_{eS}^2 = \frac{1}{\mu_S^2}\left(\tau_c^2 - \sum_{s=1}^{S-1}\mu_s^2\tau_s^2\right)$. Moreover, for the S-th site, $\tau_{gS}^2 + \tau_{eS}^2 = \tau_S^2$. Therefore, we can solve for τ_{gS}^2 and τ_{eS}^2 as

$$\begin{aligned}
\tau_{gS}^2 &= \frac{\tau_S^2}{2} + \frac{1}{2\mu_S^2}\left(\tau_c^2 - \sum_{s=1}^{S-1}\mu_s^2\tau_s^2\right) \\
\tau_{eS}^2 &= \frac{\tau_S^2}{2} - \frac{1}{2\mu_S^2}\left(\tau_c^2 - \sum_{s=1}^{S-1}\mu_s^2\tau_s^2\right).
\end{aligned}$$

Additionally, we set τ_{fS}^2 from $\tau_{gS}^2 + \tau_{fS}^2 = \tau_S^2$ as

$$\tau_{fS}^2 = \frac{\tau_S^2}{2} - \frac{1}{2\mu_S^2}\left(\tau_c^2 - \sum_{s=1}^{S-1}\mu_s^2\tau_s^2\right).$$

Now, we focus on setting the noise variances for $s \in [S-1]$. From the relation $\sum_{s=1}^{S-1}\mu_s^2\tau_{es}^2 = \mu_S^2\tau_{eS}^2$, one solution is to set

$$\tau_{es}^2 = \frac{1}{\mu_s^2(S-1)}\left[\frac{\mu_S^2}{2}\tau_S^2 - \frac{1}{2}\left(\tau_c^2 - \sum_{s=1}^{S-1}\mu_s^2\tau_s^2\right)\right].$$

Using this and $\tau_{gs}^2 = \tau_s^2 - \tau_{es}^2$, we have

$$\tau_{gs}^2 = \tau_s^2 - \frac{1}{\mu_s^2(S-1)}\left[\frac{\mu_S^2}{2}\tau_S^2 - \frac{1}{2}\left(\tau_c^2 - \sum_{s=1}^{S-1}\mu_s^2\tau_s^2\right)\right].$$

Finally, we solve for $\tau_{fs}^2 = \tau_s^2 - \tau_{gs}^2$ as

$$\tau_{fs}^2 = \frac{1}{\mu_s^2(S-1)}\left[\frac{\mu_S^2}{2}\tau_S^2 - \frac{1}{2}\left(\tau_c^2 - \sum_{s=1}^{S-1}\mu_s^2\tau_s^2\right)\right].$$

Therefore, we can solve the feasibility problem with equality using the following noise variance expressions. For the S-th site:

$$\begin{aligned}
\tau_{eS}^2 = \tau_{fS}^2 &= \frac{\tau_S^2}{2} - \frac{1}{2\mu_S^2}\left(\tau_c^2 - \sum_{s=1}^{S-1}\mu_s^2\tau_s^2\right) \\
\tau_{gS}^2 &= \frac{\tau_S^2}{2} + \frac{1}{2\mu_S^2}\left(\tau_c^2 - \sum_{s=1}^{S-1}\mu_s^2\tau_s^2\right).
\end{aligned}$$

For other sites $s \in [S-1]$:

$$\begin{aligned}
\tau_{es}^2 = \tau_{fs}^2 &= \frac{1}{\mu_s^2(S-1)}\left[\frac{\mu_S^2}{2}\tau_S^2 - \frac{1}{2}\left(\tau_c^2 - \sum_{s=1}^{S-1}\mu_s^2\tau_s^2\right)\right] \\
\tau_{gs}^2 &= \tau_s^2 - \frac{1}{\mu_s^2(S-1)}\left[\frac{\mu_S^2}{2}\tau_S^2 - \frac{1}{2}\left(\tau_c^2 - \sum_{s=1}^{S-1}\mu_s^2\tau_s^2\right)\right].
\end{aligned}$$

It is evident from the expressions that the noise variances for each site depend on the target noise variance (τ_c^2) as well as the local noise variances $\{\tau_s^2\}$ and the quality metric $\{\mu_s\}$ for all sites. This is expected as we are trying to achieve the target noise variance collectively while satisfying a different privacy requirement at each site. If all the sites had the same quality metric and the same privacy requirement, the noise variances would be uncoupled, as depicted in (2).

IV. Improved Distributed Differentially-private Principal Component Analysis

In this section, we propose an improved distributed differentially-private PCA algorithm that takes advantage of the CAPE protocol. Recall that in our distributed PCA problem, we are interested in approximating $\mathbf{V}_K(\mathbf{A})$ in a distributed setting while guaranteeing differential privacy. One naïve approach (non-private) would be to send the data matrices from the sites to the aggregator. When D and/or N_s are large, this entails a huge communication overhead. In many scenarios the local data are also private or sensitive. As the aggregator is not trusted, sending the data to the aggregator can result in a significant privacy violation. Our goals are therefore to reduce the communication cost, ensure differential privacy, and provide a close approximation to the true PCA subspace $\mathbf{V}_K(\mathbf{A})$. We previously proposed a differentially-private distributed PCA scheme [17], but the performance of the scheme is limited by the larger variance of the additive noise at the local sites due to the smaller sample sizes. We intend to alleviate this problem using the correlated noise scheme [24]. The improved distributed differentially-private PCA algorithm (capePCA) we propose here achieves the same utility as the pooled data scenario.

Algorithm 2 Improved Distributed Differentially-private PCA (capePCA)

```
Require: Data matrix $\mathbf{X}_s \in \mathbb{R}^{D \times N_s}$ for $s \in [S]$; privacy parameters $\epsilon$, $\delta$; reduced dimension $K$
1: At random noise generator: generate $\mathbf{E}_s \in \mathbb{R}^{D \times D}$, as described in the text; send to sites
2: At aggregator: generate $\mathbf{F}_s \in \mathbb{R}^{D \times D}$, as described in the text; send to sites
3: for $s = 1, 2, \ldots, S$ do                    ▷ at the local sites
4:     Compute $\mathbf{A}_s \leftarrow \frac{1}{N_s}\mathbf{X}_s\mathbf{X}_s^\top$
5:     Generate $D \times D$ symmetric matrix $\mathbf{G}_s$, as described in the text
6:     Compute $\hat{\mathbf{A}}_s \leftarrow \mathbf{A}_s + \mathbf{E}_s + \mathbf{F}_s + \mathbf{G}_s$; send $\hat{\mathbf{A}}_s$ to aggregator
7: end for
8: Compute $\hat{\mathbf{A}} \leftarrow \frac{1}{S}\sum_{s=1}^{S}\left(\hat{\mathbf{A}}_s - \mathbf{F}_s\right)$    ▷ at the aggregator
9: Perform SVD: $\hat{\mathbf{A}} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$
10: Release / send to sites: $\mathbf{V}_K$
11: return $\mathbf{V}_K$
```

We consider the same network structure as in Section III: there is a random noise generator that can generate and send noise to the sites through encrypted/secure channels. The aggregator can also generate noise and send those to the sites over encrypted/secure channels. Recall that in the pooled data scenario, we have the data matrix $\mathbf{X}$ and the sample second-moment matrix $\mathbf{A} = \frac{1}{N}\mathbf{X}\mathbf{X}^\top$. We refer to the top-$K$ PCA subspace of this sample second-moment matrix as the true (or optimal) subspace $\mathbf{V}_K(\mathbf{A})$. At each site, we compute the sample second-moment matrix as $\mathbf{A}_s = \frac{1}{N_s}\mathbf{X}_s\mathbf{X}_s^\top$. The $\mathcal{L}_2$ sensitivity [1] of the function $f(\mathbf{X}_s) = \mathbf{A}_s$ is $\Delta_2^s = \frac{1}{N_s}$ [26]. In order to approximate $\mathbf{A}_s$ satisfying (ϵ, δ) differential privacy, we can employ the AG algorithm [26] to compute $\hat{\mathbf{A}}_s = \mathbf{A}_s + \mathbf{G}_s$, where the symmetric matrix $\mathbf{G}_s$ is generated with entries i.i.d. $\sim \mathcal{N}(0, \tau_s^2)$ and $\tau_s = \frac{\Delta_2^s}{\epsilon}\sqrt{2\log\frac{1.25}{\delta}}$. Note that, in the pooled data scenario, the $\mathcal{L}_2$ sensitivity of the function $f(\mathbf{X}) = \mathbf{A}$ is $\Delta_2^{\text{pool}} = \frac{1}{N}$. Therefore, the required additive noise standard deviation should satisfy $\tau_{\text{pool}} = \frac{\Delta_2^{\text{pool}}}{\epsilon}\sqrt{2\log\frac{1.25}{\delta}} = \frac{\tau_s}{S}$, assuming equal number of samples in the sites. As we want the same utility as the pooled data scenario, we compute the following at each site s:

$$\hat{\mathbf{A}}_s = \mathbf{A}_s + \mathbf{E}_s + \mathbf{F}_s + \mathbf{G}_s.$$

Here, the noise generator generates the $D \times D$ matrix $\mathbf{E}_s$ with $[\mathbf{E}_s]_{ij}$ drawn i.i.d. $\sim \mathcal{N}(0, \tau_e^2)$ and $\sum_{s=1}^{S}\mathbf{E}_s = 0$. We set the variance τ_e^2 according to (2) as $\tau_e^2 = \left(1 - \frac{1}{S}\right)\tau_s^2$. Additionally, the aggregator generates the $D \times D$ matrix $\mathbf{F}_s$ with $[\mathbf{F}_s]_{ij}$ drawn i.i.d. $\sim \mathcal{N}(0, \tau_f^2)$. The variance τ_f^2 is set according to (2) as $\tau_f^2 = \left(1 - \frac{1}{S}\right)\tau_s^2$. Finally, the sites generate their own symmetric $D \times D$ matrix $\mathbf{G}_s$, where $[\mathbf{G}_s]_{ij}$ are drawn i.i.d. $\sim \mathcal{N}(0, \tau_g^2)$ and $\tau_g^2 = \frac{1}{S}\tau_s^2$ according to (2). Note that, these variance assignments can be readily modified to fit the unequal privacy/sample size scenario (Section III-A). However, for simplicity, we are considering the equal sample size scenario. Now, the sites send their $\hat{\mathbf{A}}_s$ to the aggregator and the aggregator computes

$$\hat{\mathbf{A}} = \frac{1}{S}\sum_{s=1}^{S}\left(\hat{\mathbf{A}}_s - \mathbf{F}_s\right) = \frac{1}{S}\sum_{s=1}^{S}\left(\mathbf{A}_s + \mathbf{G}_s\right),$$

where we used the relation $\sum_{s=1}^{S}\mathbf{E}_s = 0$. The detailed calculation is shown in Appendix A-A. Note that at the aggregator, we achieve an estimator with noise variance exactly the same as that of the pooled data scenario (by Lemma 1). Next, we perform SVD on $\hat{\mathbf{A}}$ and release the top-K eigenvector matrix $\mathbf{V}_K$, which is the (ϵ, δ) differentially private approximate to the true subspace $\mathbf{V}_K(\mathbf{A})$. To achieve the same utility level as the pooled data case, we send the full matrix $\hat{\mathbf{A}}_s$ from the sites to the aggregator instead of the partial square root of it [17]. This increases the communication cost by $SD(D-R)$, where R is the intermediate dimension of the partial square root. We consider this as the cost of performance gain.

Theorem 1 (Privacy of capePCA Algorithm)**.** *Algorithm 2 computes an (ϵ, δ) differentially private approximation to the optimal subspace $\mathbf{V}_K(\mathbf{A})$.*

Proof. The proof of Theorem 1 follows from using the Gaussian mechanism [1], the AG algorithm [26], the bound on $\|\mathbf{A}_s - \mathbf{A}'_s\|_2$ and recalling that the data samples in each site are disjoint. We start by showing that

$$\tau_e^2 + \tau_g^2 = \tau_g^2 + \tau_f^2 = \tau_s^2 = \left(\frac{1}{N_s\epsilon}\sqrt{2\log\frac{1.25}{\delta}}\right)^2.$$

Therefore, the computation of $\hat{\mathbf{A}}_s$ at each site is at least (ϵ, δ) differentially-private. As differential privacy is post-processing invariant, we can combine the noisy second-moment matrices $\hat{\mathbf{A}}_s$ at the aggregator while subtracting $\mathbf{F}_s$ for each site. By the correlated noise generation at the random noise generator, the noise $\mathbf{E}_s$ cancels out. We perform the SVD on $\hat{\mathbf{A}}$ and release $\mathbf{V}_K$. The released subspace $\mathbf{V}_K$ is thus the (ϵ, δ) differentially private approximate to the true subspace $\mathbf{V}_K(\mathbf{A})$. □

Performance Gain with Correlated Noise. The distributed differentially-private PCA algorithm of [17] essentially employs the conventional averaging (when each site sends the full $\hat{\mathbf{A}}_s$ to the aggregator). Therefore, the gain in performance of the proposed capePCA algorithm over the one in [17] is the same as shown in Proposition 1.

Theoretical Performance Guarantee. Due to the application of the correlated noise protocol, we achieve the same level of noise at the aggregator in the distributed setting as we would have in the pooled data scenario. In essence, the proposed capePCA algorithm can achieve the same performance as the AG algorithm [26] modified to account for all the samples across all the sites. Here, we present three guarantees for the captured energy, closeness to the true subspace and low-rank approximation. The guarantees are adopted from Dwork et al. [26] and modified to fit our setup and notation. Let us assume that the (ϵ, δ) differentially-private subspace output from Algorithm 2 and the true subspace are denoted by $\hat{\mathbf{V}}_K$ and $\mathbf{V}_K$, respectively. We denote the singular values of $\mathbf{X}$ with $\sigma_1 \geq \ldots \geq \sigma_D$ and the un-normalized second-moment matrix with $\mathbf{A} = \mathbf{X}\mathbf{X}^\top$. Let $\mathbf{A}_K$ and $\hat{\mathbf{A}}_K$ be the true and the (ϵ, δ)

Algorithm 3 Distributed Differentially-private OTD (capeAGN)

```
Require: Sample second-order moment matrices $\mathbf{M}_2^s \in \mathbb{R}^{D\times D}$ and third-order moment tensors $\mathcal{M}_3^s \in \mathbb{R}^{D\times D\times D}$ $\forall s \in [S]$, privacy parameters $\epsilon_1$, $\epsilon_2$, $\delta_1$, $\delta_2$, reduced dimension $K$
 1: At random noise generator: generate $\mathbf{E}_2^s \in \mathbb{R}^{D\times D}$ and $\mathcal{E}_3^s \in \mathbb{R}^{D\times D\times D}$, as described in the text; send to sites
 2: At aggregator: generate $\mathbf{F}_2^s \in \mathbb{R}^{D\times D}$ and $\mathcal{F}_3^s \in \mathbb{R}^{D\times D\times D}$, as described in the text; send to sites
 3: for $s = 1, \ldots, S$ do                          ▷ at the local sites
 4:     Generate $\mathbf{G}_2^s \in \mathbb{R}^{D\times D}$, as described in the text
 5:     Compute $\hat{\mathbf{M}}_2^s \leftarrow \mathbf{M}_2^s + \mathbf{E}_2^s + \mathbf{F}_2^s + \mathbf{G}_2^s$; send $\hat{\mathbf{M}}_2^s$ to aggregator
 6: end for
 7: Compute $\hat{\mathbf{M}}_2 \leftarrow \frac{1}{S}\sum_{s=1}^{S}\left(\hat{\mathbf{M}}_2^s - \mathbf{F}_2^s\right)$ and then SVD($K$) of $\hat{\mathbf{M}}_2$ as $\hat{\mathbf{M}}_2 = \mathbf{U}\mathbf{D}\mathbf{U}^\top$    ▷ at the aggregator
 8: Compute and send to sites: $\mathbf{W} \leftarrow \mathbf{U}\mathbf{D}^{-\frac{1}{2}}$
 9: for $s = 1, \ldots, S$ do                          ▷ at the local sites
10:     Generate symmetric $\mathcal{G}_3^s \in \mathbb{R}^{D\times D\times D}$ from the entries of $\mathbf{b} \in \mathbb{R}^{D_{\text{sym}}}$, where $[\mathbf{b}]_d \sim \mathcal{N}(0, \tau_{3g}^2)$ and $\tau_{3g}^2 = \frac{1}{S}\tau_3^{s2}$
11:     Compute $\hat{\mathcal{M}}_3^s \leftarrow \mathcal{M}_3^s + \mathcal{E}_3^s + \mathcal{F}_3^s + \mathcal{G}_3^s$ and $\tilde{\mathcal{M}}_3^s \leftarrow \hat{\mathcal{M}}_3^s(\mathbf{W},\mathbf{W},\mathbf{W})$; send $\tilde{\mathcal{M}}_3^s$ to aggregator
12: end for
13: Compute $\tilde{\mathcal{M}}_3 \leftarrow \frac{1}{S}\sum_{s=1}^{S}\left(\tilde{\mathcal{M}}_3^s - \mathcal{F}_3^s(\mathbf{W},\mathbf{W},\mathbf{W})\right)$ ▷ at the aggregator
14: return The differentially private orthogonally decomposable tensor $\tilde{\mathcal{M}}_3$, projection subspace $\mathbf{W}$
```

differentially-private rank-K approximates to $\mathbf{A}$, respectively. If we assume that the gap $\sigma_K^2 - \sigma_{K+1}^2 = \omega(\tau_{\text{pool}}\sqrt{D})$, then the following holds

- $\text{tr}\left(\hat{\mathbf{V}}_K^\top \mathbf{A}\hat{\mathbf{V}}_K\right) \geq \text{tr}\left(\mathbf{V}_K^\top \mathbf{A}\mathbf{V}_K\right) - O(\tau_{\text{pool}}K\sqrt{D})$
- $\left\|\mathbf{V}_K\mathbf{V}_K^\top - \hat{\mathbf{V}}_K\hat{\mathbf{V}}_K^\top\right\|_2 = O\left(\frac{\tau_{\text{pool}}\sqrt{D}}{\sigma_K^2-\sigma_{K+1}^2}\right)$
- $\|\mathbf{A} - \hat{\mathbf{A}}_K\|_2 \leq \|\mathbf{A} - \mathbf{A}_K\|_2 + O(\tau_{\text{pool}}\sqrt{D})$.

The detailed proofs can be found in Dwork et al. [26].

Communication Cost. We quantify the total communication cost associated with the proposed capePCA algorithm. Recall that capePCA is an one-shot algorithm. Each of the random noise generator and the aggregator send one $D \times D$ matrix to the sites. Each site uses these to compute the noisy estimate of the local second-moment matrix ($D \times D$) and sends that back to the aggregator. Therefore, the total communication cost is proportional to $3SD^2$ or $O(D^2)$. This is expected as we are computing the global $D \times D$ second-moment matrix in a distributed setting before computing the PCA subspace.

V. Distributed Differentially-private Orthogonal Tensor Decomposition

In this section, we propose an algorithm (capeAGN) for distributed differentially-private OTD. The proposed algorithm takes advantage of the correlated noise design scheme (Algorithm 1) [24]. To our knowledge, this is the first work on distributed differentially-private OTD. The definition of the differentially-private OTD is presented in Appendix C. We refer the reader to our previous work [21] for two centralized differentially-private OTD algorithms: AGN and AVN.

We start with recalling that the orthogonal decomposition of a 3-rd order symmetric tensor $\mathcal{X} \in \mathbb{R}^{D\times D\times D}$ is a collection of orthonormal vectors $\{\mathbf{v}_k\}$ together with corresponding positive scalars $\{\lambda_k\}$ such that $\mathcal{X} = \sum_{k=1}^{K}\lambda_k\mathbf{v}_k \otimes \mathbf{v}_k \otimes \mathbf{v}_k$. A unit vector $\mathbf{u} \in \mathbb{R}^D$ is an **eigenvector** of $\mathcal{X}$ with corresponding **eigenvalue** λ if $\mathcal{X}(\mathbf{I},\mathbf{u},\mathbf{u}) = \lambda\mathbf{u}$, where $\mathbf{I}$ is the $D\times D$ identity matrix [3]. To see this, one can observe

$$\begin{aligned}\mathcal{X}(\mathbf{I},\mathbf{u},\mathbf{u}) &= \sum_{k=1}^{K}\lambda_k\left(\mathbf{I}^\top\mathbf{v}_k\right)\otimes\left(\mathbf{u}^\top\mathbf{v}_k\right)\otimes\left(\mathbf{u}^\top\mathbf{v}_k\right)\\ &= \sum_{k=1}^{K}\lambda_k\left(\mathbf{u}^\top\mathbf{v}_k\right)^2\mathbf{v}_k.\end{aligned}$$

By the orthogonality of the $\mathbf{v}_k$, it is clear that $\mathcal{X}(\mathbf{I},\mathbf{v}_k,\mathbf{v}_k) = \lambda_k\mathbf{v}_k\ \forall k$. Now, the orthogonal tensor decomposition proposed in [3] is based on the mapping

$$\mathbf{u} \mapsto \frac{\mathcal{X}(\mathbf{I},\mathbf{u},\mathbf{u})}{\|\mathcal{X}(\mathbf{I},\mathbf{u},\mathbf{u})\|_2}, \tag{3}$$

which can be considered as the tensor equivalent of the well-known matrix power method. Obviously, all tensors are not orthogonally decomposable. As the tensor power method requires the eigenvectors $\{\mathbf{v}_k\}$ to be orthonormal, we need to perform *whitening* - that is, projecting the tensor on a subspace such that the eigenvectors become orthogonal to each other.

We note that the proposed algorithm applies to both of the STM and MOG problems. However, as the correlated noise scheme only works with Gaussian noise, the proposed capeAGN employs the AGN algorithm [21] at its core. Inline with our setup in Section III, we assume that there is a random noise generator that can generate and send noise to the sites through an encrypted/secure channel. The un-trusted aggregator can also generate noise and send those to the sites over encrypted/secure channels. At site s, the sample second-order moment matrix and the third-order moment tensor are denoted as $\mathbf{M}_2^s \in \mathbb{R}^{D\times D}$ and $\mathcal{M}_3^s \in \mathbb{R}^{D\times D\times D}$, respectively. The noise standard deviation required for computing the (ϵ_1, δ_1) differentially-private approximate to $\mathbf{M}_2^s$ is given by

$$\tau_2^s = \frac{\Delta_2^s}{\epsilon_1}\sqrt{2\log\left(\frac{1.25}{\delta_1}\right)}, \tag{4}$$

where the sensitivity Δ_2^s is inversely proportional to the sample size N_s. To be more specific, we can write $\Delta_{2,S}^s = \frac{\sqrt{2}}{N_s}$ for STM and $\Delta_{2,M}^s = \frac{1}{N_s}$ for MOG. The detailed derivation of the sensitivity of $\mathbf{M}_2^s$ for both STM and MOG are shown in Appendix C. Additionally, at site s, the noise standard deviation required for computing the (ϵ_2, δ_2) differentially-private approximate to $\mathcal{M}_3^s$ is given by

$$\tau_3^s = \frac{\Delta_3^s}{\epsilon_2}\sqrt{2\log\left(\frac{1.25}{\delta_2}\right)}. \tag{5}$$

Again, we can write $\Delta_{3,S}^s = \frac{\sqrt{2}}{N_s}$ for STM and $\Delta_{3,M}^s = \frac{2}{N_s} + \frac{6D\sigma^2}{N_s}$ for MOG. Appendix C contains the detailed algebra for calculating the sensitivity of $\mathcal{M}_3^s$ for both STM and MOG. We note that, as in the case of $\mathbf{M}_2^s$, the sensitivity depends only on the sample size N_s. Now, in the pooled-data scenario, the noise standard deviations would be given by:

$$\tau_2^{\text{pool}} = \frac{\Delta_2^{\text{pool}}}{\epsilon_1}\sqrt{2\log\left(\frac{1.25}{\delta_1}\right)}$$
$$\tau_3^{\text{pool}} = \frac{\Delta_3^{\text{pool}}}{\epsilon_2}\sqrt{2\log\left(\frac{1.25}{\delta_2}\right)},$$

where $\Delta_2^{\text{pool}} = \frac{\Delta_2^s}{S}$ and $\Delta_3^{\text{pool}} = \frac{\Delta_3^s}{S}$, assuming equal number of samples in the sites. Now, we need to compute the $D \times K$ whitening matrix $\mathbf{W}$ and the $D \times D \times D$ tensor $\hat{\mathcal{M}}_3$ in a distributed way while satisfying differential privacy. Although we could employ our previous differentially-private distributed PCA algorithm [17] to compute $\mathbf{W}$, to achieve the same level of accuracy as the pooled data scenario we compute the following matrix at site s:

$$\hat{\mathbf{M}}_2^s = \mathbf{M}_2^s + \mathbf{E}_2^s + \mathbf{F}_2^s + \mathbf{G}_2^s,$$

where $\mathbf{E}_2^s \in \mathbb{R}^{D\times D}$ is generated at the noise generator satisfying $\sum_{s=1}^S \mathbf{E}_2^s = 0$ and the entries $[\mathbf{E}_2^s]_{ij}$ drawn i.i.d. $\sim \mathcal{N}(0, \tau_{2e}^2)$. Here, we set the noise variance according to (2): $\tau_{2e}^2 = \left(1 - \frac{1}{S}\right)\tau_2^{s2}$. Additionally, $\mathbf{F}_2^s \in \mathbb{R}^{D\times D}$ is generated at the aggregator with the entries $[\mathbf{F}_2^s]_{ij}$ drawn i.i.d. $\sim \mathcal{N}(0, \tau_{2f}^2)$. We set the noise variance according to (2): $\tau_{2f}^2 = \left(1 - \frac{1}{S}\right)\tau_2^{s2}$. Finally, $\mathbf{G}_2^s \in \mathbb{R}^{D\times D}$ is a symmetric matrix generated at site s where $\{[\mathbf{G}_2^s]_{ij} : i \in [D], j \le i\}$ are drawn i.i.d. $\sim \mathcal{N}(0, \tau_{2g}^2)$, $[\mathbf{G}_2^s]_{ij} = [\mathbf{G}_2^s]_{ji}$ and $\tau_{2g}^2 = \frac{1}{S}\tau_2^{s2}$. The aggregator computes

$$\hat{\mathbf{M}}_2 = \frac{1}{S}\sum_{s=1}^S \left(\hat{\mathbf{M}}_2^s - \mathbf{F}_2^s\right) = \frac{1}{S}\sum_{s=1}^S \left(\mathbf{M}_2^s + \mathbf{G}_2^s\right),$$

where we used the relation $\sum_{s=1}^S \mathbf{E}_2^s = 0$. Note that the variance of the additive noise in $\hat{\mathbf{M}}_2$ is exactly the same as the pooled data scenario, as described in Lemma 1. At the aggregator, we can then compute the SVD(K) of $\hat{\mathbf{M}}_2$ as $\hat{\mathbf{M}}_2 = \mathbf{U}\mathbf{D}\mathbf{U}^\top$. We compute the matrix $\mathbf{W} = \mathbf{U}\mathbf{D}^{-\frac{1}{2}}$ and send it to the sites.

Next, we focus on computing $\hat{\mathcal{M}}_3$ in the distributed setting. For this purpose, we can follow the same steps as computing $\hat{\mathbf{M}}_2$. However, $\hat{\mathcal{M}}_3$ is a $D \times D \times D$ tensor, and for large enough D, this will entail a very large communication overhead. We alleviate this in the following way: each site receives $\mathcal{F}_3^s \in \mathbb{R}^{D\times D\times D}$ and $\mathbf{W}$ from the aggregator and $\mathcal{E}_3^s \in \mathbb{R}^{D\times D\times D}$ from the noise generator. Here, $[\mathcal{F}_3^s]_{ijk}$ are drawn i.i.d. $\sim \mathcal{N}(0, \tau_{3f}^2)$. Additionally, $[\mathcal{E}_3^s]_{ijk}$ are drawn i.i.d. $\sim \mathcal{N}(0, \tau_{3e}^2)$ and $\sum_{s=1}^S \mathcal{E}_3^s = 0$ is satisfied. We set the two variance terms according to (2): $\tau_{3f}^2 = \tau_{3e}^s = \left(1 - \frac{1}{S}\right)\tau_3^{s2}$. Finally, each site generates their own $\mathcal{G}_3^s \in \mathbb{R}^{D\times D\times D}$ in the following way: site s draws a vector $\mathbf{b} \in \mathbb{R}^{D_{\text{sym}}}$ with $D_{\text{sym}} = \binom{D+2}{3}$ and entries i.i.d. $\sim \mathcal{N}(0, \tau_{3g}^2)$, where $\tau_{3g}^2 = \frac{1}{S}\tau_3^{s2}$. The tensor $\mathcal{G}_3^s$ is generated with the entries from $\mathbf{b}$ such that $\mathcal{G}_3^s$ is symmetric. Again, for both $\hat{\mathbf{M}}_2^s$ and $\hat{\mathcal{M}}_3^s$, we are considering the equal sample size scenario for simplicity. Our framework requires only a small modification to incorporate the unequal privacy/sample size (Section III-A). Now, site s computes

$$\hat{\mathcal{M}}_3^s = \mathcal{M}_3^s + \mathcal{E}_3^s + \mathcal{F}_3^s + \mathcal{G}_3^s \text{ and } \tilde{\mathcal{M}}_3^s = \hat{\mathcal{M}}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W}).$$

We note that $\tilde{\mathcal{M}}_3^s$ is a $K \times K \times K$ dimensional tensor. Each site sends this to the aggregator. This saves a lot of communication overhead as typically $K \ll D$. To see how this results in the same estimate of $\tilde{\mathcal{M}}_3$ as the pooled data scenario, we observe

$$\tilde{\mathcal{M}}_3^s = \hat{\mathcal{M}}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W}) = \mathcal{M}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W}) + \mathcal{E}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W}) + \mathcal{F}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W}) + \mathcal{G}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W}).$$

Additionally, at the aggregator, we compute

$$\tilde{\mathcal{M}}_3 = \frac{1}{S}\sum_{s=1}^S \left(\tilde{\mathcal{M}}_3^s - \tilde{\mathcal{F}}_3^s\right) = \left(\frac{1}{S}\sum_{s=1}^S \mathcal{M}_3^s + \mathcal{G}_3^s\right)(\mathbf{W}, \mathbf{W}, \mathbf{W}),$$

where $\tilde{\mathcal{F}}_3^s = \mathcal{F}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W})$. We used the associativity of the multi-linear operation [3] and the relation $\sum_{s=1}^S \mathcal{E}_3^s = 0$. The detailed calculation is shown in Appendix A-B. Note that the $\tilde{\mathcal{M}}_3$ we achieve in this scheme is exactly the same $\tilde{\mathcal{M}}_3$ we would have achieved if all the data samples were present in the aggregator. Moreover, this is also the quantity that the aggregator would get if the sites send the full $\hat{\mathcal{M}}_3^s$ to the aggregator instead of $\tilde{\mathcal{M}}_3^s$. The complete capeAGN algorithm is shown in Algorithm 3.

Theorem 2 (Privacy of capeAGN Algorithm)**.** *Algorithm 3 computes an $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$ differentially private orthogonally decomposable tensor $\tilde{\mathcal{M}}_3$. Additionally, the computation of the projection subspace $\mathbf{W}$ is (ϵ_1, δ_1) differentially private.*

Proof. The proof of Theorem 2 follows from using the Gaussian mechanism [1], the sensitivities of $\mathbf{M}_2^s$ and $\mathcal{M}_3^s$ and recalling that the data samples in each site are disjoint. First, we show that the computation of $\mathbf{W}$ satisfies (ϵ_1, δ_1) differential privacy. Due to the correlated noise, we have

$$\tau_{2e}^2 + \tau_{2g}^2 = \tau_{2g}^2 + \tau_{2f}^2 = \tau_2^{s2} = \left(\frac{\Delta_2^s}{\epsilon_1}\sqrt{2\log\left(\frac{1.25}{\delta_1}\right)}\right)^2,$$

where Δ_2^s is the sensitivity of $\mathbf{M}_2^s$. Therefore, the release of $\hat{\mathbf{M}}_2^s$ from each site s is at least (ϵ_1, δ_1) differentially-private. As differential privacy is closed under post-processing and the samples in each site are disjoint, the computation of $\mathbf{W}$ at the aggregator also satisfies (ϵ_1, δ_1) differential privacy. Next, we show that the computation of $\tilde{\mathcal{M}}_3$ satisfies $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$ differential privacy. We recall that

$$\tau_{3e}^2 + \tau_{3g}^2 = \tau_{3g}^2 + \tau_{3f}^2 = \tau_3^{s2} = \left(\frac{\Delta_3^s}{\epsilon_2}\sqrt{2\log\left(\frac{1.25}{\delta_2}\right)}\right)^2,$$

where Δ_3^s is the sensitivity of $\mathcal{M}_3^s$. The computation of $\hat{\mathcal{M}}_3^s$ at each site is at least (ϵ_2, δ_2) differentially-private. Further, by the composition theorem [1], the computation $\tilde{\mathcal{M}}_3^s = \hat{\mathcal{M}}_3^s(\mathbf{W}, \mathbf{W}, \mathbf{W})$ at each site is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$ differentially-private. By the post-processing invariability, the computation of $\tilde{\mathcal{M}}_3$ at the aggregator is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$ differentially-private. ☐

Performance Gain with Correlated Noise. As we mentioned before, this is the first work that proposes an algorithm for distributed differentially-private OTD. As we employ the CAPE scheme for our computations, the gain in the performance over a conventional distributed differentially-private OTD is therefore the same as in the case of distributed differentially-private averaging, as described in Proposition 1.

Theoretical Performance Guarantee. Although our proposed capeAGN algorithm can reach the performance of the pooled data scenario (that is, the AGN algorithm [21] with all data samples from all sites stored in the aggregator), it is hard to characterize how the estimated $\{\hat{\mathbf{a}}_k\}$ and $\{\hat{w}_k\}$ would deviate from the true $\{\mathbf{a}_k\}$ and $\{w_k\}$, respectively. We note that although we are adding symmetric noise to the third-order moment tensor, an orthogonal decomposition need not exist for the perturbed tensor, even though the perturbed tensor is symmetric [3], [11]. Anandkumar et al. [3] provided a bound on the error of the recovered decomposition in terms of the operator norm of the tensor perturbation. For our proposed algorithm, the perturbation includes the effect of estimating the third-order moment tensor from the samples as well as the noise added for differential-privacy. Even without accounting for the error in estimating the moments from observable samples, the operator norm of the effective noise at the aggregator: $\|\mathcal{G}\|_{\text{op}} = \frac{1}{S}\left\|\sum_{s=1}^{S}\mathcal{G}_3^s\right\|_{\text{op}}$, is a random quantity, and requires new measure concentration results to analyze. Relating these bounds to the error in estimating recovering the $\{\mathbf{a}_k\}$ and $\{w_k\}$ is nontrivial. However, very recently Esmaeili and Huang [34] proposed differentially private OTD-based Latent Dirichlet Allocation (LDA) for topic modeling. The authors consider the sensitivities of different functions at different points in the flow of the LDA algorithm and propose to employ Gaussian mechanism [1] to the point with the smallest sensitivity, conditioned on some constraints. This enables the DP-LDA algorithm to achieve better utility bounds. The extension of the techniques introduced in [34] to STM and MOG is nontrivial and we defer that for future work.

Communication Cost. We note that capeAGN is a two-step algorithm: it computes the projection subspace $\mathbf{W} \in \mathbb{R}^{D\times K}$ and then orthogonally decomposable tensor $\tilde{\mathcal{M}}_3$. The random noise generator sends $\mathbf{E}_2^s \in \mathbb{R}^{D\times D}$ and $\mathcal{E}_3^s \in \mathbb{R}^{D\times D\times D}$ to each site s. Each site s sends $\hat{\mathbf{M}}_2^s \in \mathbb{R}^{D\times D}$ and $\tilde{\mathcal{M}}_3^s \in \mathbb{R}^{K\times K\times K}$ and to the aggregator. The aggregator sends $\mathbf{F}_2^s \in \mathbb{R}^{D\times D}$, $\mathbf{W} \in \mathbb{R}^{D\times K}$, and $\mathcal{F}_3^s \in \mathbb{R}^{D\times D\times D}$ to each site s. Therefore, the total communication cost is proportional to $3SD^2 + 2SD^3 + SDK + SK^3$ or $O(D^3)$. This is expected as we are computing the global $D \times D \times D$ third-order moment tensor in a distributed setting.

VI. Experimental Results

In this section, we empirically show the effectiveness of the proposed distributed differentially-private matrix and tensor factorization algorithms. We focus on investigating the privacy-utility trade-off: how the performance varies as a function of the privacy parameters and the number of samples. We start with the proposed capePCA algorithm followed by the capeAGN algorithm. In each case, we compare the proposed algorithms with existing (if any) and non-private algorithms and a conventional approach (no correlated noise).

A. Improved Distributed Differentially-private PCA

We empirically compared the proposed capePCA, the existing DPdisPCA [17] and non-private PCA on pooled data (non − dp pool). We also included the performance of differentially private PCA [26] on local data (local) (i.e. data of a single site) and the conventional approach (conv) (i.e. without correlated noise). We designed the experiments according to Imtiaz and Sarwate [17] using three datasets: a *synthetic* dataset ($D = 200$, $K = 50$) generated with zero mean and a pre-determined covariance matrix, the *MNIST* dataset ($D = 784$, $K = 50$) [35] (MNIST) and the *Covertype* dataset ($D = 54$, $K = 10$) [36] (COVTYPE). The MNIST consists of handwritten digits and has a training set of 60000 samples, each of size 28×28 pixels. The COVTYPE contains the forest cover types for 30×30 m^2 cells obtained from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. We collected the dataset from the UC Irvine KDD archive [36]. For our experiments, we randomly selected 60000 samples from the COVTYPE. We preprocessed the data by subtracting the mean (centering) and normalizing the samples with the maximum $\mathcal{L}_2$ norm in each dataset to enforce the condition $\|\mathbf{x}_n\|_2 \leq 1\ \forall n$. We note that this preprocessing step is not differentially private, although it can be modified to satisfy differential-privacy at the cost of some utility. In all cases we show the average performance over 10 runs of the algorithms. As a performance measure of the produced subspace from the algorithm, we choose the captured energy: $q^{\text{CE}} = \text{tr}(\hat{\mathbf{V}}^\top \mathbf{A}\hat{\mathbf{V}})$, where $\hat{\mathbf{V}}$ is the subspace estimated by an algorithm and $\mathbf{A}$ is the true second-moment matrix of the data. Note that, we can approximate the the captured energy in the true subspace as $\text{tr}(\mathbf{V}_K(\mathbf{A})^\top \mathbf{A}\mathbf{V}_K(\mathbf{A}))$, where $\mathbf{A}$ is achieved from the pooled-data sample second-moment matrix and $\mathbf{V}_K(\mathbf{A})$ is achieved from the non-private PCA.

Dependence on privacy parameter ϵ. First, we explore the trade-off between privacy and utility; i.e., between ϵ and q^{CE}. We note that the standard deviation of the added noise is inversely proportional to ϵ – bigger ϵ means higher privacy risk but less noise and thus, better utility. In Figure 2(a)-(c), we show the variation of q^{CE} of different algorithms for different values of ϵ. For this experiment, we kept the parameters δ, N_s and S fixed. For all the datasets, we observe that as ϵ increases (higher privacy risk), the captured energy increases. The proposed capePCA reaches the optimal utility (non − dp pool) for some parameter choices and clearly outperforms the existing DPdisPCA, the conv, and the local algorithms. One of the reasons that capePCA outperforms conv is the smaller noise variance at the aggregator, as described before. Moreover, capePCA outperforms DPdisPCA because DPdisPCA suffers from a larger variance at the aggregator due to computation of the partial square root of $\hat{\mathbf{A}}_s$ [17]. However, DPdisPCA offers a much smaller communication overhead than capePCA. Achieving better performance than local is intuitive because including the information from multiple sites should always result in better estimates of population parameters than using the data from a single site only. An interesting observation is that for datasets with lower dimensional samples, we can use smaller ϵ (i.e., lower privacy risk) for the same utility.

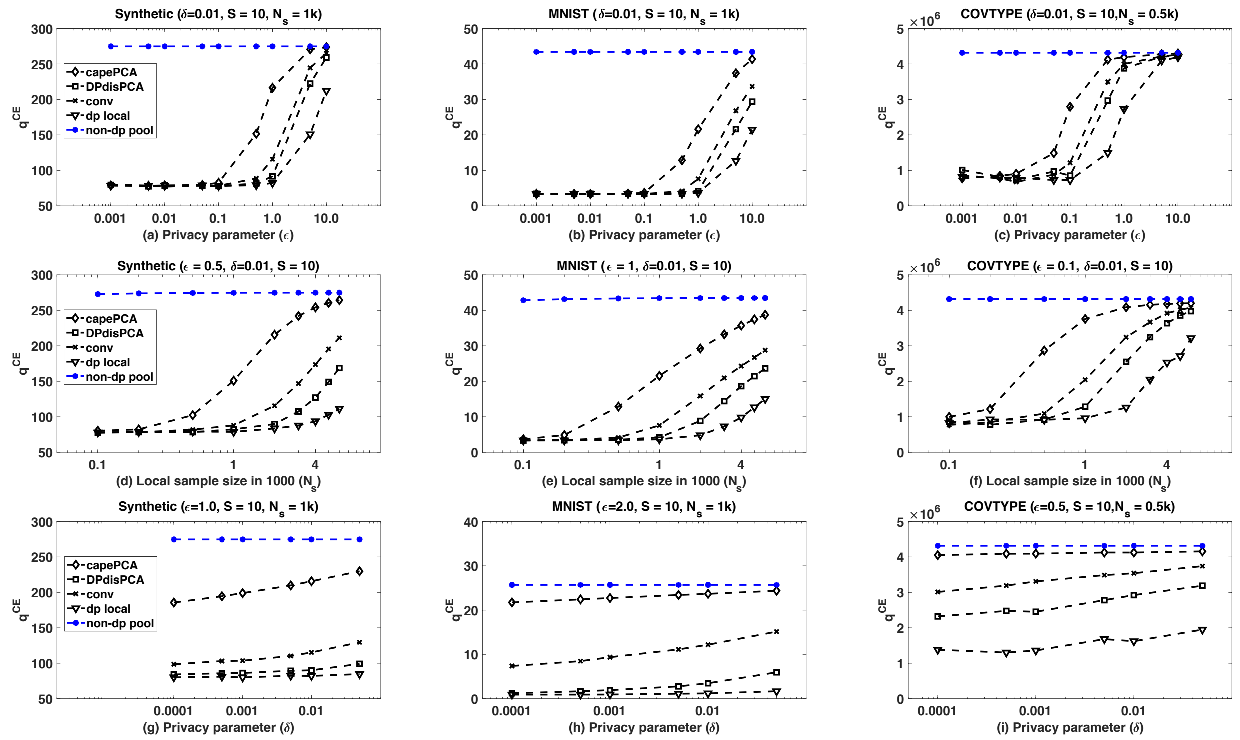

Fig. 2. Variation of performance in distributed PCA for synthetic and real data: (a) - (c) with privacy parameter ϵ; (d) - (f) with sample size N_s and (g) - (i) with privacy parameter δ

Dependence on number of samples N_s. Next, we investigate the variation in performance with sample size N_s. Intuitively, it should be easier to guarantee smaller privacy risk ϵ and higher utility q^{CE}, when the number of samples is large. Figures 2(d)-(f) show how q^{CE} increases as a function of N_s. The variation with N_s reinforces the results seen earlier with variation of ϵ. For a fixed ϵ and δ, the utility increases as we increase N_s. For sufficiently large N_s, the captured energy will reach that of non − dp pool. Again, we observe a sharper increase in utility for lower-dimensional dataset.

Dependence on privacy parameter δ. Finally, we explore the variation of performance with the other privacy parameter δ. Recall that δ can be considered as the probability that the algorithm releases the private information without guaranteeing privacy. We, therefore, want this to be as small as possible. However, lower δ results in larger noise variance. In Figure 2(g)-(i), we show how q^{CE} vary with varying δ. We observe that if δ is not too small, the proposed algorithm can achieve very good utility, easily outperforming the other algorithms.

B. *Distributed Differentially-private OTD*

For the proposed capeAGN algorithm, we focus on measuring how well the output of the proposed algorithm approximate the true components $\{\mathbf{a}_k\}$ and $\{w_k\}$. Let the recovered component vectors be $\{\hat{\mathbf{a}}_k\}$. We use the same error metric as our previous work [21], q^{comp}, to capture the disparity between $\{\mathbf{a}_k\}$ and $\{\hat{\mathbf{a}}_k\}$:

$$q^{\mathrm{comp}} = \frac{1}{K}\sum_{k=1}^{K} ED^k_{\min}, \text{ and } ED^k_{\min} = \min_{k'\in[K]} \|\hat{\mathbf{a}}_k - \mathbf{a}_{k'}\|_2.$$

For comparison, we show the error resulting from the $\hat{\mathbf{a}}_k$'s achieved from the proposed capeAGN algorithm, a conventional (but never proposed anywhere to the best of our knowledge) distributed differentially-private OTD algorithm that does not employ correlated noise (conv), a differentially-private OTD [21] on local data (local) and the non-private tensor power method [3] on pooled data (Non − priv.). We also show the error considering random vectors as $\hat{\mathbf{a}}_k$'s (Rand. vect.). The reason [21] to show (Rand. vect.) is the following: this error corresponds to the worst possible results, as we are not taking any information from data into account to estimate $\hat{\mathbf{a}}_k$'s. As recovering the component vectors is closely related with recovering the selection probabilities $\{w_k\}$, we only show the error of recovering the component vectors. We studied the dependence of q^{comp} on the privacy parameters ϵ, δ and the sample size N_s. In all cases we show the average performance over 10 runs of each algorithm. We note that the capeAGN algorithm adds noise in two stages for ensuring differential-privacy: one for estimating $\mathbf{W}$ and another for estimating $\mathcal{M}_3$. We equally divided ϵ and δ to set ϵ_1, ϵ_2 and δ_1, δ_2 for the two stages. Optimal allocation of ϵ and δ in multi-stage differentially-private algorithms is still an open question.

Performance variation in the MOG setup. First, we present the performance of the aforementioned algorithms in the setting of the mixture of Gaussians. We use two *synthetic data* sets of different feature dimensions ($D = 10$, $K = 5$ and $D = 50$, $K = 10$), where the common covariance is $\sigma^2 = 0.05$ and the components $\{\mathbf{a}_k\}$ satisfy $\|\mathbf{a}_k\|_2 \leq 1$.

We first explore the *privacy-utility tradeoff* between ϵ and q^{comp}. For the capeAGN algorithm, the variance of the noise

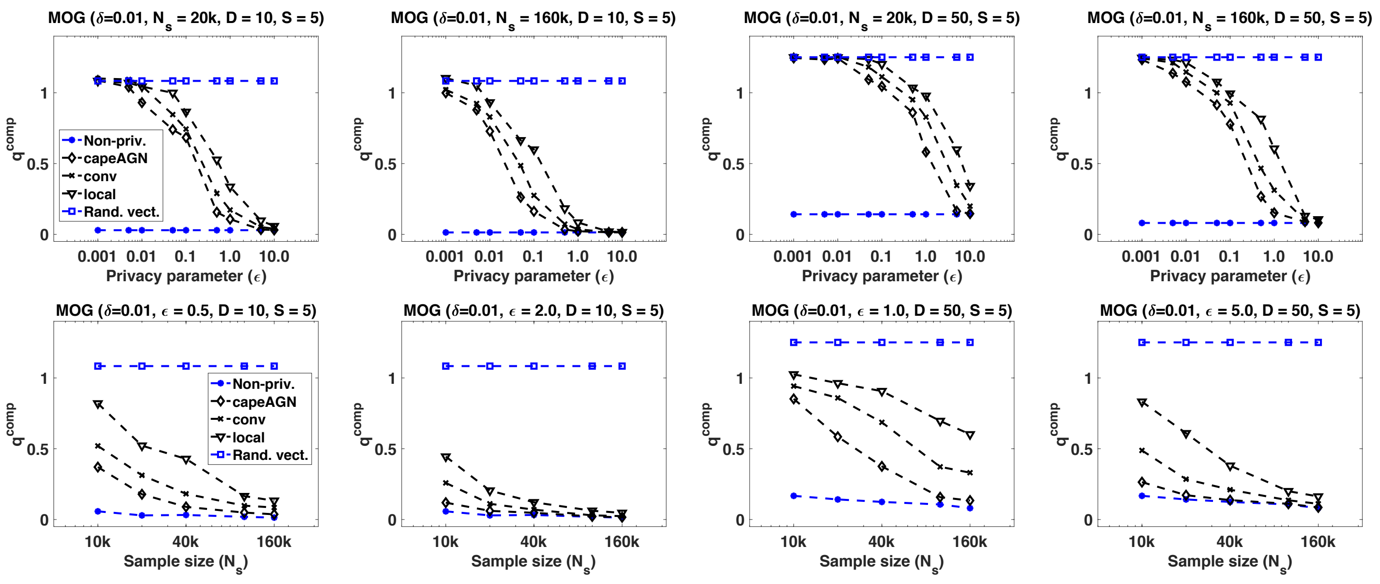

Fig. 3. Variation of performance in the MOG setup: top-row – with privacy parameter ϵ; bottom-row – with sample size N_s

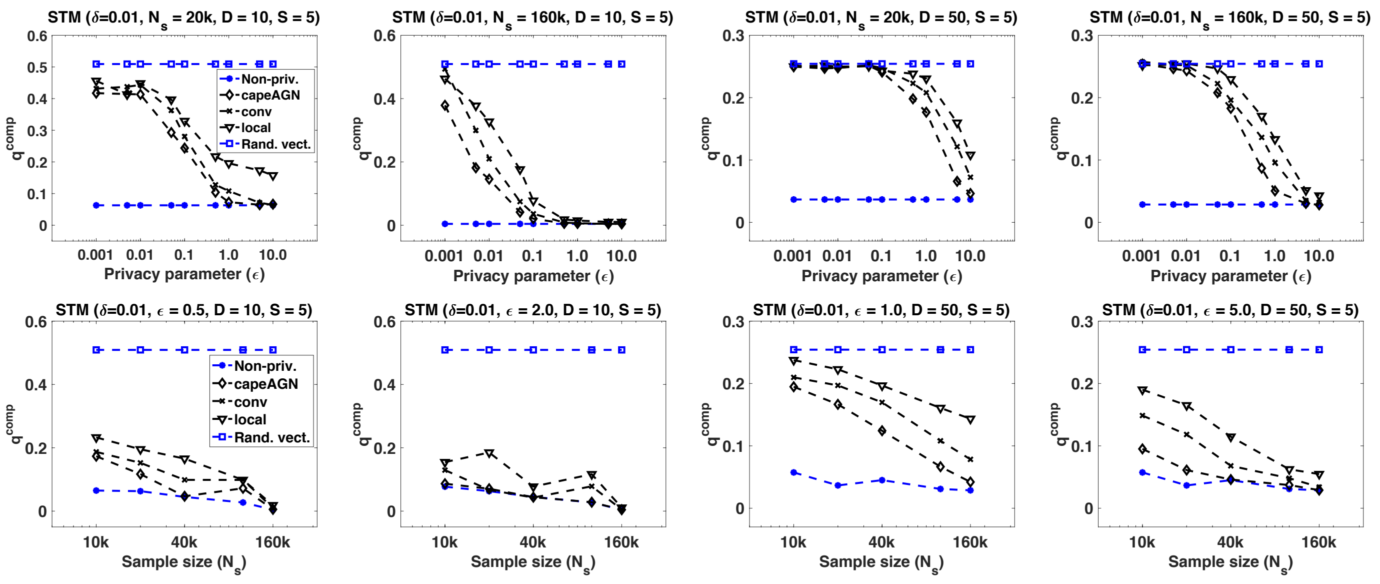

Fig. 4. Variation of performance in the STM setup: top-row – with privacy parameter ϵ; bottom-row – with sample size N_s

is inversely proportional to ϵ^2 – smaller ϵ means more noise (lower utility) and lower privacy risk. In the top-row of Figure 3, we show the variation of q^{comp} with ϵ for a fixed $\delta = 0.01$ and $S = 5$ for two different feature dimensions, each with two different samples sizes. For both of the feature dimensions, we observe that as ϵ increases (higher privacy risk), the errors decrease and the proposed capeAGN algorithm outperforms the conv and local methods. capeAGN matches the performance of Non $-$ priv. method for larger ϵ values. For a particular feature dimension, we notice that if we increase N_s, the performance of capeAGN gets even better. This is expected as the variance of the noise for capeAGN is inversely proportional to square of the sample size.

Next, we consider the performance variation with N_s. Intuitively, it should be easier to guarantee a smaller privacy risk for the same ϵ and a higher utility (lower error) when the number of samples is large. In the bottom row of Figure 3, we show how the errors vary as a function of N_s for the MOG model for two different feature dimensions, while keeping $\delta = 0.01$ and $S = 5$ fixed. The variation with the sample size reinforces the results seen earlier with variation in ϵ: the proposed capeAGN outperforms the other algorithms under investigation for both $D = 10$ and $D = 50$. In general, capeAGN approaches the performance of Non $-$ priv. as the sample size increases. When ϵ is large enough, the capeAGN algorithm achieves as much utility as Non $-$ priv. method.

Finally, we show the variation of performance with the other privacy parameter δ. Recall that δ can be interpreted as the probability that the privacy-preserving algorithm releases the private information "out in the wild" without any additive noise. Therefore, we want to ensure that δ is small. However, the smaller the δ is the larger the noise variance becomes.

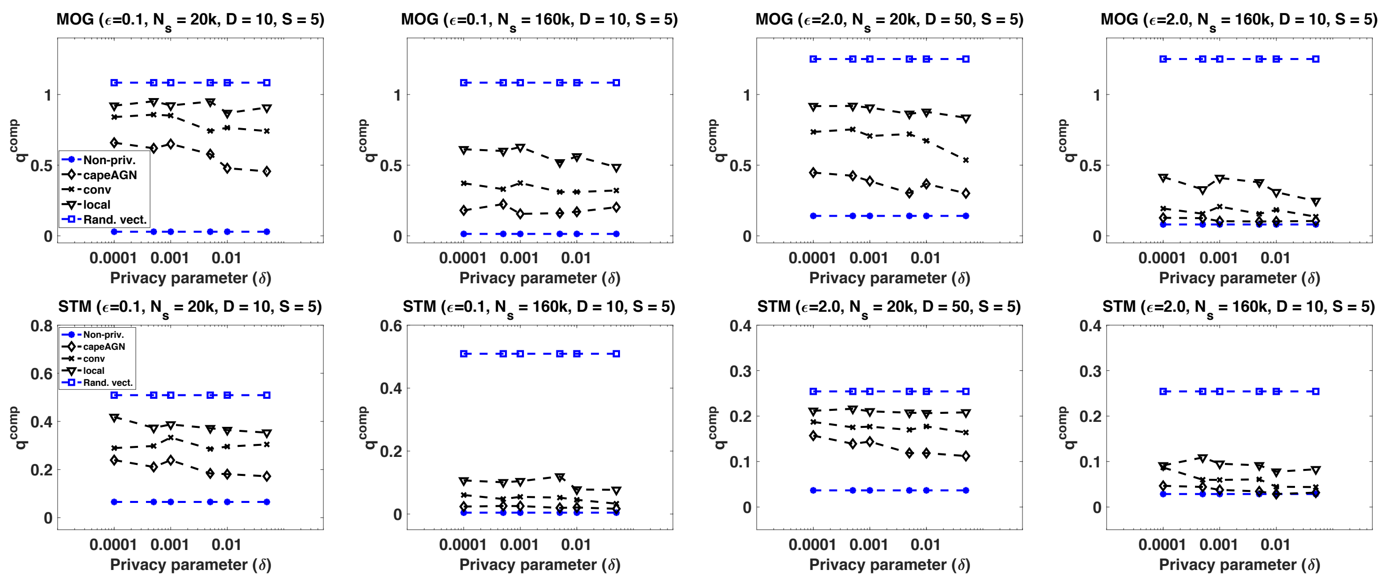

Fig. 5. Variation of performance with privacy parameter δ: top-row – in MOG setup; bottom-row – in STM setup

Thus smaller δ dictates loss in utility. We observe this in our experiments as well. In the top-row of Figure 5, we show the variation of q^{comp} with δ for two different feature dimensions and two different sample sizes. We kept $S = 5$ fixed. We observe that when ϵ is small, we need larger δ to achieve meaningful performance. This can be explained in the following way: for Gaussian mechanism, we need to ensure that the standard deviation of the noise σ satisfies $\sigma \geq \frac{\Delta}{\epsilon}\sqrt{2\log\frac{1.25}{\delta}}$, where Δ is the $\mathcal{L}_2$ sensitivity of the function under consideration. This inequality can be satisfied with infinitely many (ϵ, δ) pairs and one can keep the noise level the same for a smaller ϵ with a larger δ. We observe from the figure that when both ϵ and N_s are larger, the proposed capeAGN can achieve very close performance to the non-private one even for very small δ values.

Performance variation in the STM setup. We performed experiments on two *synthetic* datasets of different feature dimensions ($D = 10$, $K = 5$ and $D = 50$, $K = 10$) generated with pre-determined $\mathbf{w}$ and $\{\mathbf{a}_k\}$. It should be noted here that the recovery of $\{\mathbf{a}_k\}$ is difficult [21], because the recovered word probabilities from the tensor decomposition, whether private or non-private, may not always be valid probability vectors (i.e., no negative entries and sum to 1). Therefore, prior to computing the q^{comp}, we ran a post-processing step (0-out negative entries and then normalize by summation) to ensure that the recovered vectors are valid probability vectors. This process is non-linear and potentially makes the recovery error worse. However, for practical STM, D is not likely to be 10 or 50, rather it may be of the order of thousands, simulating which is a huge computational burden. In general, if we want the same privacy level for higher dimensional data, we need to increase the sample size. We refer the reader to some efficient (but non-privacy-preserving) implementations [37] for tensor factorization.

As in the case of the MOG model, we first explore the *privacy-utility tradeoff* between ϵ and q^{comp}. In the top-row of Figure 4, we show the variation of q^{comp} with ϵ for a fixed $\delta = 0.01$ and $S = 5$ for two different feature dimensions. For both of the feature dimensions, we observe that as ϵ increases (higher privacy risk), the errors decrease. The proposed capeAGN outperforms conv and local methods in all settings; and match the performance of $\mathsf{Non-priv.}$ for large enough ϵ. Increasing N_s makes the proposed algorithm perform even better.

Next, in the bottom-row of Figure 4, we show how the errors vary as a function of N_s for two different feature dimensions, while keeping $\delta = 0.01$ and $S = 5$ fixed. The variation with N_s reiterates the results seen earlier. The proposed capeAGN outperforms all other algorithms (except the $\mathsf{Non-priv.}$) for both $D = 10$ and $D = 50$. For larger N_s, it achieves almost the same utility as the $\mathsf{Non-priv.}$ algorithm. Even for smaller ϵ with a proper sample size, the error is very low. For the $D = 10$ case, the capeAGN always performs very closely with the $\mathsf{Non-priv.}$ algorithm.

Lastly, we show the variation of q^{comp} with δ in the bottom-row of Figure 5. We observe similar performance trend as in the MOG setting. For smaller ϵ and sample size, we need to compensate with larger δ to achieve a performance closer to $\mathsf{Non-priv.}$ one. However, when sample size is larger, we can get away with a smaller ϵ and δ. This is intuitive as hiding one individual among a large group is easier – the additive noise variance need not be very large and hence the performance does not take a large hit.

VII. Conclusion

In this paper, we proposed new algorithms for distributed differentially-private principal component analysis and orthogonal tensor decomposition. Our proposed algorithms achieve the same level of additive noise variance as the pooled data scenario for ensuring differential-privacy. Therefore, we attain the same utility as the differentially-private pooled data scenario in a distributed setting. This is achieved through the employment of the correlated noise design protocol, under the assumption of availability of some reasonable resources. We empirically compared the performance of the proposed algorithms with those of existing (if any) and conventional distributed algorithms on synthetic and real data sets. We varied privacy parameters and relevant dataset parameters. The proposed algorithms outperformed the existing and conventional algorithms comfortably and matched the performance of corresponding non-private algorithms for proper parameter choices. In general, the proposed algorithms offered very good utility even for strong privacy guarantees, which indicates that meaningful privacy can be attained even without loosing much utility.

Acknowledgements

The authors would like to acknowledge helpful discussions with Dr. Jafar Mohammadi, particularly regarding the design of CAPE.

Appendix A
Algebra for Various Calculations

A. Calculation of $\hat{\mathbf{A}}$ in Section IV

We show the calculation of $\hat{\mathbf{A}}$ here in detail. Recall that the sites send their $\hat{\mathbf{A}}_s$ to the aggregator and the aggregator computes

$$\begin{aligned}\hat{\mathbf{A}} &= \frac{1}{S}\sum_{s=1}^{S}\left(\hat{\mathbf{A}}_s - \mathbf{F}_s\right)\\ &= \frac{1}{S}\sum_{s=1}^{S}\left(\mathbf{A}_s + \mathbf{E}_s + \mathbf{F}_s + \mathbf{G}_s - \mathbf{F}_s\right)\\ &= \frac{1}{S}\sum_{s=1}^{S}\left(\mathbf{A}_s + \mathbf{G}_s\right) + \frac{1}{S}\sum_{s=1}^{S}\mathbf{E}_s\\ &= \frac{1}{S}\sum_{s=1}^{S}\left(\mathbf{A}_s + \mathbf{G}_s\right),\end{aligned}$$

where we used the relation $\sum_{s=1}^{S}\mathbf{E}_s = 0$.

B. Calculation of $\tilde{\mathcal{M}}_3$ in Section V

We show the calculation of $\tilde{\mathcal{M}}_3$ here in detail. We recall that $\tilde{\mathcal{M}}_3^s = \hat{\mathcal{M}}_3^s\left(\mathbf{W},\mathbf{W},\mathbf{W}\right)$. At the aggregator, we compute

$$\tilde{\mathcal{M}}_3 = \frac{1}{S}\sum_{s=1}^{S}\left(\tilde{\mathcal{M}}_3^s - \tilde{\mathcal{F}}_3^s\right),$$

where $\tilde{\mathcal{F}}_3^s = \mathcal{F}_3^s\left(\mathbf{W},\mathbf{W},\mathbf{W}\right)$.

Then we have

$$\begin{aligned}\tilde{\mathcal{M}}_3 &= \frac{1}{S}\sum_{s=1}^{S}\Big(\mathcal{M}_3^s\left(\mathbf{W},\mathbf{W},\mathbf{W}\right) + \mathcal{E}_3^s\left(\mathbf{W},\mathbf{W},\mathbf{W}\right) +\\ &\quad \tilde{\mathcal{F}}_3^s + \mathcal{G}_3^s\left(\mathbf{W},\mathbf{W},\mathbf{W}\right) - \tilde{\mathcal{F}}_3^s\Big)\\ &= \frac{1}{S}\sum_{s=1}^{S}\left(\mathcal{M}_3^s\left(\mathbf{W},\mathbf{W},\mathbf{W}\right) + \mathcal{G}_3^s\left(\mathbf{W},\mathbf{W},\mathbf{W}\right)\right) +\\ &\quad \left(\frac{1}{S}\sum_{s=1}^{S}\mathcal{E}_3^s\right)\left(\mathbf{W},\mathbf{W},\mathbf{W}\right)\\ &= \frac{1}{S}\sum_{s=1}^{S}\left(\mathcal{M}_3^s\left(\mathbf{W},\mathbf{W},\mathbf{W}\right) + \mathcal{G}_3^s\left(\mathbf{W},\mathbf{W},\mathbf{W}\right)\right)\\ &= \left(\frac{1}{S}\sum_{s=1}^{S}\mathcal{M}_3^s + \mathcal{G}_3^s\right)\left(\mathbf{W},\mathbf{W},\mathbf{W}\right),\end{aligned}$$

where we used the associativity of the multi-linear operation [3] and the relation $\sum_{s=1}^{S}\mathcal{E}_3^s = 0$.

Appendix B
Applications of Orthogonal Tensor Decomposition

We review two examples from Anandkumar et al. [3], which involve estimation of latent variables from observed samples. The lower-order moments obtained from the samples can be written as low-rank symmetric tensors.

A. Single Topic Model (STM)

Let us consider an exchangeable bag-of-words model [3] for documents. Such exchangeable models can be viewed as mixture models in which there is a latent variable h such that the L words in the document $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_L$ are conditionally i.i.d. given h. Additionally, the conditional distributions are identical at all the nodes [3]. Let us assume that h is the only topic of a given document, and it can take only K distinct values. Let D be the number of distinct words in the vocabulary, and $L \geq 3$ be the number of words in each document. The generative process for a document is as follows: the document's topic is drawn according to the discrete distribution specified by the probability vector $\mathbf{w} = \left[w_1, w_2, \ldots, w_K\right]^\top$. This is modeled as a discrete random variable h such that $\Pr\left[h = k\right] = w_k$, for $k \in [K]$. Given the topic h, the document's L words are drawn independently according to the discrete distribution specified by the probability vector $\mathbf{a}_h \in \mathbb{R}^D$. We represent the L words in the document by D-dimensional random vectors $\mathbf{t}_l \in \mathbb{R}^D$. Specifically, if the l-th word is d, we set $\mathbf{t}_l = \mathbf{e}_d$ for $l \in [L]$, where $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_D$ are the standard coordinate basis vectors for $\mathbb{R}^D$. We observe that for any topic k

$$\begin{aligned}\mathbb{E}\left[\mathbf{t}_1 \otimes \mathbf{t}_2 | h = k\right] &= \sum_{i,j}\Pr\left[\mathbf{t}_1 = i, \mathbf{t}_2 = j | h = k\right]\mathbf{e}_i \otimes \mathbf{e}_j\\ &= \mathbb{E}\left[\mathbf{t}_1 | h = k\right] \otimes \mathbb{E}\left[\mathbf{t}_2 | h = k\right]\\ &= \mathbf{a}_k \otimes \mathbf{a}_k.\end{aligned}$$

Now, we can define two moments in terms of the outer products of the probability vectors $\mathbf{a}_k$ and the distribution of the topics h as

$$\mathbf{M}_2 = \sum_{k=1}^{K} w_k \mathbf{a}_k \otimes \mathbf{a}_k,\ \mathcal{M}_3 = \sum_{k=1}^{K} w_k \mathbf{a}_k \otimes \mathbf{a}_k \otimes \mathbf{a}_k. \quad (6)$$

The method proposed in [3] to recover $\mathbf{w}$ and $\{\mathbf{a}_k\}$ proceeds as follows: we observe N documents. Each of the documents has number of words $L \geq 3$. The way we record what we observe is: we form an $D \times D \times D$ tensor whose (d_1, d_2, d_3)-th entry is the proportion of times we see a document with first word d_1, second word d_2 and third word d_3. In this setting, we can estimate the moments $\mathbf{M}_2$ and $\mathcal{M}_3$, defined in (6), from the observed data as: $\mathbf{M}_2 = \mathbb{E}[\mathbf{t}_1 \otimes \mathbf{t}_2]$ and $\mathcal{M}_3 = \mathbb{E}[\mathbf{t}_1 \otimes \mathbf{t}_2 \otimes \mathbf{t}_3]$. We then need to perform orthogonal tensor decomposition on $\mathcal{M}_3$ to recover $\mathbf{w}$ and $\{\mathbf{a}_k\}$.

B. Mixture of Gaussians (MOG)

A similar example as the single topic model is the spherical mixture of Gaussians [3]. Let us assume that there are K components and the component mean vectors are given by the set $\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_K\} \subset \mathbb{R}^D$. The probability of choosing component k is w_k. We assume that the common covariance matrix is $\sigma^2 \mathbf{I}_D$. However, the model can be extended to incorporate different covariance matrices for different component as well [3], [9]. The n-th observation of the model can be written as $\mathbf{t}_n = \mathbf{a}_h + \mathbf{z}$, where h is a discrete random variable with $\Pr[h = k] = w_k$ and $\mathbf{z}$ is an D-dimensional random vector, independent from h, drawn according to $\mathcal{N}(0, \sigma^2 \mathbf{I}_D)$. Let us denote the total number of observations by N. Without loss of generality, we assume that $\|\mathbf{a}_k\|_2 \leq 1$. Now, for $D \geq K$, it has been shown [9] that if we have estimates of the second and third order moments from the observations $\mathbf{t}_n$ as $\mathbf{M}_2 = \mathbb{E}[\mathbf{t} \otimes \mathbf{t}] - \sigma^2 \mathbf{I}_D$ and

$$\mathcal{M}_3 = \mathbb{E}[\mathbf{t} \otimes \mathbf{t} \otimes \mathbf{t}] - \sigma^2 \sum_{d=1}^{D} \left(\mathbb{E}[\mathbf{t}] \otimes \mathbf{e}_d \otimes \mathbf{e}_d + \mathbf{e}_d \otimes \mathbb{E}[\mathbf{t}] \otimes \mathbf{e}_d + \mathbf{e}_d \otimes \mathbf{e}_d \otimes \mathbb{E}[\mathbf{t}]\right),$$

then these moments are decomposable as: $\mathbf{M}_2 = \sum_{k=1}^{K} w_k \mathbf{a}_k \otimes \mathbf{a}_k$ and $\mathcal{M}_3 = \sum_{k=1}^{K} w_k \mathbf{a}_k \otimes \mathbf{a}_k \otimes \mathbf{a}_k$.

C. Orthogonal Decomposition of $\mathcal{M}_3$

For both the STM and the MOG models, in order to decompose $\mathcal{M}_3$ using the tensor power method (3), we need the $\mathbf{a}_k$'s to be orthogonal to each other. But, in general, they are not. To employ the orthogonal tensor decomposition, we can project the tensor onto some subspace $\mathbf{W} \in \mathbb{R}^{D \times K}$ to ensure $\mathbf{W}^\top \mathbf{a}_k$'s are orthogonal to each other. We note that, according to the multi-linear notation, we have

$$\mathcal{M}_3(\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3) = \sum_{k=1}^{K} w_k \left(\mathbf{V}_1^\top \mathbf{a}_k\right) \otimes \left(\mathbf{V}_2^\top \mathbf{a}_k\right) \otimes \left(\mathbf{V}_3^\top \mathbf{a}_k\right).$$

In order to find $\mathbf{W}$, we can compute the SVD(K) on the second-order moment $\mathbf{M}_2 \in \mathbb{R}^{D \times D}$ as $\mathbf{M}_2 = \mathbf{U}\mathbf{D}\mathbf{U}^\top$, where $\mathbf{U} \in \mathbb{R}^{D \times K}$ and $\mathbf{D} \in \mathbb{R}^{K \times K}$. We define $\mathbf{W} = \mathbf{U}\mathbf{D}^{-\frac{1}{2}} \in \mathbb{R}^{D \times K}$ and then compute the projection $\tilde{\mathcal{M}}_3 = \mathcal{M}_3(\mathbf{W}, \mathbf{W}, \mathbf{W})$. We note that $\tilde{\mathcal{M}}_3 \in \mathbb{R}^{K \times K \times K}$ is now orthogonally decomposable. We use the tensor power iteration (3) on $\tilde{\mathcal{M}}_3$ to recover the weights $\{w_k\}$ and the component vectors $\{\mathbf{a}_k\}$. The detail of the tensor power method can be found in Anandkumar et al. [3].

Appendix C
Differentially-private OTD

We note that the key step in the orthogonal tensor decomposition algorithm is the mapping given by (3). In order to ensure differential privacy for the orthogonal decomposition, we may either add noise at each iteration step scaled to the $\mathcal{L}_2$ sensitivity [26] of the operation given by (3) or we can add noise to the tensor $\mathcal{X}$ itself just once. Adding noise in each iteration step might result in a poor utility/accuracy of the recovered eigenvectors and eigenvalues. We intend to add noise to the tensor itself prior to employing the tensor power method. In the following, we are showing the sensitivity calculations for the pooled data scenario. Extension to the distributed case is straightforward (replacing N with N_s).

First, we focus on the exchangeable single topic model setup that we described in Appendix B-A. We observe and record N documents. Let us consider two sets of documents, which differ in only one sample (e.g., the last one). Let the empirical second-order moment matrices be $\mathbf{M}_2$ and $\mathbf{M}_2'$ and the third-order moment tensors be $\mathcal{M}_3$ and $\mathcal{M}_3'$, respectively, for these two sets. We consider the two tensors, $\mathcal{M}_3$ and $\mathcal{M}_3'$, as *neighboring*. We observe that

$$\begin{aligned}\mathbf{M}_2 &= \frac{1}{N}\sum_{n=1}^{N} \mathbf{t}_{1,n}\mathbf{t}_{2,n}^\top \\ &= \frac{1}{N}\sum_{n=1}^{N-1} \mathbf{t}_{1,n}\mathbf{t}_{2,n}^\top + \frac{1}{N}\mathbf{t}_{1,N}\mathbf{t}_{2,N}^\top, \\ \mathbf{M}'_2 &= \frac{1}{N}\sum_{n=1}^{N-1} \mathbf{t}_{1,n}\mathbf{t}_{2,n}^\top + \frac{1}{N}\mathbf{t}'_{1,N}\mathbf{t}'^\top_{2,N},\end{aligned}$$

where $\mathbf{t}_{l,n}$ denotes the l-th word of the n-th document. Similarly, we observe

$$\begin{aligned}\mathcal{M}_3 &= \frac{1}{N}\sum_{d=1}^{D} \mathbf{t}_{1,n} \otimes \mathbf{t}_{2,n} \otimes \mathbf{t}_{3,n} \\ &= \frac{1}{N}\sum_{n=1}^{N-1} \mathbf{t}_{1,n} \otimes \mathbf{t}_{2,n} \otimes \mathbf{t}_{3,n} + \frac{1}{N}\mathbf{t}_{1,N} \otimes \mathbf{t}_{2,N} \otimes \mathbf{t}_{3,N}, \\ \mathcal{M}'_3 &= \frac{1}{N}\sum_{n=1}^{N-1} \mathbf{t}_{1,n} \otimes \mathbf{t}_{2,n} \otimes \mathbf{t}_{3,n} + \frac{1}{N}\mathbf{t}'_{1,N} \otimes \mathbf{t}'_{2,N} \otimes \mathbf{t}'_{3,N}.\end{aligned}$$

As mentioned before, we perform the SVD on $\mathbf{M}_2$ first to compute $\mathbf{W}$. We intend to use the AG algorithm [26] to make this operation differentially private. We look at the quantity:

$$\|\mathbf{M}_2 - \mathbf{M}'_2\|_2 = \frac{1}{N}\|\mathbf{t}_{1,N}\mathbf{t}_{2,N}^\top - \mathbf{t}'_{1,N}\mathbf{t}'^\top_{2,N}\|_2$$
$$= \frac{1}{N}\sup_{\|\mathbf{u}\|_2,\|\mathbf{v}\|_2=1}\left\{\mathbf{u}^\top\left(\mathbf{t}_{1,N}\mathbf{t}_{2,N}^\top - \mathbf{t}'_{1,N}\mathbf{t}'^\top_{2,N}\right)\mathbf{v}\right\}$$
$$\leq \frac{\sqrt{2}}{N} = \Delta_{2,S},$$

because of the encoding $\mathbf{t}_{l,n} = \mathbf{e}_d$. For the mixture of Gaussians model, we note that we assumed $\|\mathbf{a}_k\|_2 \leq 1$ for all $k \in \{1, 2, \ldots, K\}$. To find a bound on $\|\mathbf{M}_2 - \mathbf{M}'_2\|_2$, we consider the following: for identifiability of the $\{\mathbf{a}_k\}$, we have to assume that the $\mathbf{a}_k$'s are linearly independent. In other words, we are interested in finding the directions of the components specified by $\{\mathbf{a}_k\}$. In that sense, while obtaining the samples, we can divide the samples by a constant ζ such that $\|\mathbf{t}_n\|_2 \leq 1$ is satisfied. From the resulting second- and third-order moments, we will be able to recover $\{\mathbf{a}_k\}$ up to a scale factor. It is easy to show using the definition of largest eigenvalue of a symmetric matrix [38] that

$$\|\mathbf{M}_2 - \mathbf{M}'_2\|_2 = \frac{1}{N}\sup_{\|\mathbf{u}\|_2=1}\left\{\mathbf{u}^\top\left(\mathbf{t}_N\mathbf{t}_N^\top - \mathbf{t}'_N\mathbf{t}'^\top_N\right)\mathbf{u}\right\}$$
$$= \frac{1}{N}\sup_{\|\mathbf{u}\|_2=1}\left\{\left|\mathbf{u}^\top\mathbf{t}_N\right|^2 - \left|\mathbf{u}^\top\mathbf{t}'_N\right|^2\right\}$$
$$\leq \frac{1}{N} = \Delta_{2,M},$$

where the inequality follows from the relation $\|\mathbf{t}_n\|_2 \leq 1$. We note that the largest singular value of a matrix is the square root of the largest eigenvalue of that matrix. For the distributed case, as mentioned before, the sensitivity of $\mathbf{M}_2^s$ depends only on the local sample size. We can therefore use the AG algorithm [26] (i.e., adding Gaussian noise with variance scaled to $\Delta_{2,S}$ or $\Delta_{2,M}$ to $\mathbf{M}_2$) to make the computation of $\mathbf{W}$ (ϵ_1, δ_1)-differentially private. Next, we focus on the tensor $\mathcal{M}_3$. We need to project $\mathcal{M}_3$ on $\mathbf{W}$ before using the tensor power method. We can choose between making the projection operation differentially private, or we can make the $\mathcal{M}_3$ itself differentially private before projection. We found that making the projection operation differentially private involves addition of a large amount of noise and more importantly, the variance of the noise to be added depends on the alphabet size (or feature dimension) D and the singular values of $\mathbf{M}_2$. Therefore, we choose to make the tensor itself differentially private. We are interested to find the sensitivity of the tensor valued function $f(\mathcal{M}_3) = \mathcal{M}_3$, which is simply the identity map. That is, we need to find the maximum quantity that this function can change if we replace the argument $\mathcal{M}_3$ with a neighboring $\mathcal{M}'_3$. For our exchangeable single topic model setup, we have

$$\|\mathcal{M}_3 - \mathcal{M}'_3\| = \left\|\frac{1}{N}\mathbf{t}_{1,N}\otimes\mathbf{t}_{2,N}\otimes\mathbf{t}_{3,N} - \frac{1}{N}\mathbf{t}'_{1,N}\otimes\mathbf{t}'_{2,N}\otimes\mathbf{t}'_{3,N}\right\| \leq \frac{\sqrt{2}}{N} = \Delta_{3,S},$$

because only one entry in the $D \times D \times D$ tensor $\mathbf{t}_{1,N}\otimes\mathbf{t}_{2,N}\otimes\mathbf{t}_{3,N}$ is non-zero (in fact, the only non-zero entry is 1). Now, for the mixture of Gaussians model, we define

$$\mathcal{T} = \sigma^2\sum_{d=1}^{D}\left(\mathbb{E}[\mathbf{t}]\otimes\mathbf{e}_d\otimes\mathbf{e}_d + \mathbf{e}_d\otimes\mathbb{E}[\mathbf{t}]\otimes\mathbf{e}_d + \mathbf{e}_d\otimes\mathbf{e}_d\otimes\mathbb{E}[\mathbf{t}]\right)$$

Therefore, we have

$$\mathcal{T} - \mathcal{T}' = \frac{\sigma^2}{N}\sum_{d=1}^{D}\Big((\mathbf{t}_N - \mathbf{t}'_N)\otimes\mathbf{e}_d\otimes\mathbf{e}_d + \mathbf{e}_d\otimes(\mathbf{t}_N - \mathbf{t}'_N)\otimes\mathbf{e}_d + \mathbf{e}_d\otimes\mathbf{e}_d\otimes(\mathbf{t}_N - \mathbf{t}'_N)\Big)$$
$$\implies \|\mathcal{T} - \mathcal{T}'\| \leq \frac{3D\sigma^2}{N}\|\mathbf{t}_N - \mathbf{t}'_N\|_2 \leq \frac{6D\sigma^2}{N},$$

where the last inequality follows from $\|\mathbf{t}_n\|_2 \leq 1$. We have

$$\|\mathcal{M}_3 - \mathcal{M}'_3\| = \left\|\frac{1}{N}\mathbf{t}_N\otimes\mathbf{t}_N\otimes\mathbf{t}_N - \frac{1}{N}\mathbf{t}'_N\otimes\mathbf{t}'_N\otimes\mathbf{t}'_N + \mathcal{T} - \mathcal{T}'\right\|$$
$$\leq \frac{2}{N} + \frac{6D\sigma^2}{N} = \Delta_{3,M},$$

because $\|\mathbf{t}_N\otimes\mathbf{t}_N\otimes\mathbf{t}_N\| = 1$ in our setup. Again, we note that in the distributed setting, the sensitivity of the local $\mathcal{M}_3^s$ depends only on the local sample size. We refer the reader to our previous work [21] where we proposed two algorithms for centralized differentially private OTD. The first one uses a symmetric tensor made with i.i.d. entries from a Gaussian distribution, while the second proposed method uses a symmetric tensor made with entries taken from a sample vector drawn from an appropriate distribution. Both of the algorithms guarantee (ϵ, δ)-differential privacy.

References

[1] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis," in *Proceedings of the Third Conference on Theory of Cryptography*, 2006, pp. 265–284.

[2] A. D. Sarwate, S. M. Plis, J. A. Turner, M. R. Arbabshirani, and V. D. Calhoun, "Sharing Privacy-sensitive Access to Neuroimaging and Genetics Data: a Review and Preliminary Validation," *Frontiers in Neuroinformatics*, vol. 8, no. 35, 2014.

[3] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, "Tensor Decompositions for Learning Latent Variable Models," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 2773–2832, Jan. 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2627435.2697055

[4] T. G. Kolda and B. W. Bader, "Tensor Decompositions and Applications," *SIAM REVIEW*, vol. 51, no. 3, pp. 455–500, 2009.

[5] J. D. Carroll and J.-J. Chang, "Analysis of Individual Differences in Multidimensional Scaling via an n-way Generalization of "Eckart-Young" Decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970. [Online]. Available: http://dx.doi.org/10.1007/BF02310791

[6] R. A. Harshman, "Foundations of the PARAFAC Procedure: Models and Conditions for an 'Explanatory' Multi-modal Factor Analysis," *UCLA Working Papers in Phonetics*, vol. 16, no. 1, 1970.

[7] L. R. Tucker, "Some Mathematical Notes on Three-mode Factor Analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966. [Online]. Available: http://dx.doi.org/10.1007/BF02289464

[8] L. Lathauwer, B. D. Moor, and J. Vandewalle, "On the Best Rank-1 and Rank-(R1 ,R2 ,. . .,RN) Approximation of Higher-Order Tensors," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1324–1342, Mar. 2000. [Online]. Available: http://dx.doi.org/10.1137/S0895479898346995

[9] D. J. Hsu and S. M. Kakade, "Learning Mixtures of Spherical Gaussians: Moment Methods and Spectral Decompositions," *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, pp. 11–20, 2013. [Online]. Available: http://doi.acm.org/10.1145/2422436.2422439

[10] D. Hsu, S. M. Kakade, and T. Zhang, "A Spectral Algorithm for Learning Hidden Markov Models," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1460 – 1480, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022000012000244

[11] T. G. Kolda, "Symmetric Orthogonal Tensor Decomposition is Trivial," in *eprint arXiv:1503.01375*, 2015. [Online]. Available: https://arxiv.org/abs/1503.01375

[12] Y. Liang, M.-F. Balcan, and V. Kanchanapally, "Distributed PCA and k-Means Clustering," *Online, pages.cs.wisc.edu/~yliang/distributedPCAandCoreset.pdf.*

[13] M.-F. Balcan, V. Kanchanapally, Y. Liang, and D. Woodruff, "Improved Distributed Principal Component Analysis," in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, ser. NIPS'14, 2014, pp. 3113–3121. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969033.2969174

[14] Y. L. Borgne, S. Raybaud, and G. Bontempi, "Distributed Principal Component Analysis for Wireless Sensor Networks," *Sensors*, vol. 8, no. 8, pp. 4821–4850, 2008. [Online]. Available: http://dx.doi.org/10.3390/s8084821

[15] Z.-J. Bai, R. H. Chan, and F. T. Luk, *Principal Component Analysis for Distributed Data Sets with Updating*, Berlin, Heidelberg, 2005, pp. 471–483. [Online]. Available: https://doi.org/10.1007/11573937_51

[16] S. V. Macua, P. Belanovic, and S. Zazo, "Consensus-based Distributed Principal Component Analysis in Wireless Sensor Networks," in *2010 IEEE 11th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, June 2010, pp. 1–5.

[17] H. Imtiaz and A. D. Sarwate, "Differentially Private Distributed Principal Component Analysis," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 2206–2210.

[18] D. Feldman, M. Schmidt, and C. Sohler, "Turning Big Data into Tiny Data: Constant-size Coresets for K-means, PCA and Projective Clustering," in *Proceedings of the Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '13, 2013, pp. 1434–1453. [Online]. Available: http://dl.acm.org/citation.cfm?id=2627817.2627920

[19] K. L. Clarkson and D. P. Woodruff, "Low-Rank Approximation and Regression in Input Sparsity Time," *J. ACM*, vol. 63, no. 6, pp. 54:1–54:45, Jan. 2017. [Online]. Available: http://doi.acm.org/10.1145/3019134

[20] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions," *SIAM Rev.*, vol. 53, no. 2, pp. 217–288, May 2011. [Online]. Available: http://dx.doi.org/10.1137/090771806

[21] H. Imtiaz and A. D. Sarwate, "Improved Algorithms for Differentially Private Orthogonal Tensor Decomposition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 2201–2205.

[22] H. Imtiaz, R. Silva, B. Baker, S. M. Plis, A. D. Sarwate, and V. Calhoun, "Privacy-preserving Source Separation for Distributed Data using Independent Component Analysis," in *2016 Annual Conference on Information Science and Systems (CISS)*, March 2016, pp. 123–127.

[23] Y. Wang and A. Anandkumar, "Online and Differentially-private Tensor Decomposition," in *Proc. of the 30th NIPS*, 2016, pp. 3539–3547. [Online]. Available: http://dl.acm.org/citation.cfm?id=3157382.3157493

[24] J. Mohammadi, H. Imtiaz, and A. D. Sarwate, "Assisting Differentially Private Function Computation Using Correlated Noise," under review. [Online]. Available: http://eceweb1.rutgers.edu/~hi53/DDP_ver6_hi.pdf

[25] A. Blum, C. Dwork, F. McSherry, and K. Nissim, "Practical Privacy: The SuLQ Framework," in *Proceedings of the Twenty-fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2005, pp. 128–138. [Online]. Available: http://doi.acm.org/10.1145/1065167.1065184

[26] C. Dwork, K. Talwar, A. Thakurta, and L. Zhang, "Analyze Gauss: Optimal Bounds for Privacy-preserving Principal Component Analysis," in *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, 2014, pp. 11–20.

[27] G. W. Stewart, "On the Early History of the Singular Value Decomposition," *SIAM Rev.*, vol. 35, no. 4, pp. 551–566, Dec. 1993.

[28] P. Comon, G. Golub, L.-H. Lim, and B. Mourrain, "Symmetric Tensors and Symmetric Tensor Rank," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 1254–1279, 2008. [Online]. Available: http://dx.doi.org/10.1137/060661569

[29] A. D. Sarwate and K. Chaudhuri, "Signal Processing and Machine Learning with Differential Privacy: Theory, Algorithms, and Challenges," *IEEE Signal Processing Magazine*, vol. 30, no. 5, pp. 86–94, September 2013. [Online]. Available: http://dx.doi.org/10.1109/MSP.2013.2259911

[30] C. Dwork and A. Roth, "The Algorithmic Foundations of Differential Privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2013. [Online]. Available: http://dx.doi.org/10.1561/0400000042

[31] R. Sheikh, B. Kumar, and D. K. Mishra, "A Distributed k-secure Sum Protocol for Secure Multi-party Computations," *arXiv preprint arXiv:1003.4071*, 2010.

[32] Y. Benkaouz and M. Erradi, "A Distributed Protocol for Privacy Preserving Aggregation with Non-permanent Participants," *Computing*, vol. 97, no. 9, pp. 893–912, Sep. 2015.

[33] Q. Geng, P. Kairouz, S. Oh, and P. Viswanath, "The Staircase Mechanism in Differential Privacy," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 7, pp. 1176–1184, 2015.

[34] S. A. Esmaeili and F. Huang, "An end-to-end Differentially Private Latent Dirichlet Allocation Using a Spectral Algorithm," *ArXiv e-prints*, May 2018.

[35] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[36] M. Lichman, "UCI Machine Learning Repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[37] F. Huang, S. Matusevych, A. Anandkumar, N. Karampatziakis, and P. Mineiro, "Distributed Latent Dirichlet Allocation via Tensor Factorization," in *NIPS Optimization Workshop*, 2014.

[38] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

Hafiz Imtiaz (S16) is pursuing his Ph.D. degree at Rutgers University, NJ, USA. He received his M.Sc. degree in Electrical and Computer Engineering from Rutgers University in 2017 and his M.Sc. and B.Sc. degrees in Electrical and Electronic Engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2011 and 2009, respectively. He won the ECE PhD Research Excellence Award in Spring 2016 at Rutgers. He was awarded the University Scholarship and Dean's List Scholarship in all academic terms during his time in Bangladesh University of Engineering and Technology. His research interests are in the areas of machine learning, signal/image processing, and distributed differentially private matrix/tensor factorization algorithms.

Anand D. Sarwate (S99–M09–SM14) received the B.S. degrees in electrical engineering and computer science and mathematics from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2002, and the M.S. and Ph.D. degrees in electrical engineering from the Department of Electrical Engineering and Computer Sciences (EECS), University of California, Berkeley (U.C. Berkeley), Berkeley, CA, USA. He is a currently an Assistant Professor with the Department of Electrical and Computer Engineering, Rutgers, The State University of New Jersey, New Brunswick, NJ, USA, since January 2014. He was previously a Research Assistant Professor from 2011 to 2013 with the Toyota Technological Institute at Chicago; prior to this, he was a Postdoctoral Researcher from 2008 to 2011 with the University of California, San Diego, CA. His research interests include information theory, machine learning, signal processing, optimization, and privacy and security.

Dr. Sarwate received the A. Walter Tyson Assistant Professor Award from the Rutgers School of Engineering, the NSF CAREER award in 2015, and the Samuel Silver Memorial Scholarship Award and the Demetri Angelakos Memorial Award from the EECS Department at U.C. Berkeley. He was awarded the National Defense Science and Engineering Graduate Fellowship from 2002 to 2005. He is a member of Phi Beta Kappa and Eta Kappa Nu.