Privacy-Preserving Distributed Optimization via Subspace Perturbation

A General Framework

Li, Qiongxiu; Heusdens, Richard; Christensen, Mads Græsbøll

Published in:
I E E E Transactions on Signal Processing

DOI (link to publication from Publisher):
10.1109/TSP.2020.3029887

Publication date:
2020

Document Version
Accepted author manuscript, peer reviewed version

Privacy-Preserving Distributed Optimization via Subspace Perturbation: A General Framework

Qiongxiu Li, *Student Member, IEEE*, Richard Heusdens and Mads Græsbøll Christensen, *Senior Member, IEEE*

***Abstract*—As the modern world becomes increasingly digitized and interconnected, distributed signal processing has proven to be effective in processing its large volume of data. However, a main challenge limiting the broad use of distributed signal processing techniques is the issue of privacy in handling sensitive data. To address this privacy issue, we propose a novel yet general subspace perturbation method for privacy-preserving distributed optimization, which allows each node to obtain the desired solution while protecting its private data. In particular, we show that the dual variable introduced in each distributed optimizer will not converge in a certain subspace determined by the graph topology. Additionally, the optimization variable is ensured to converge to the desired solution, because it is orthogonal to this non-convergent subspace. We therefore propose to insert noise in the non-convergent subspace through the dual variable such that the private data are protected, and the accuracy of the desired solution is completely unaffected. Moreover, the proposed method is shown to be secure under two widely-used adversary models: passive and eavesdropping. Furthermore, we consider several distributed optimizers such as ADMM and PDMM to demonstrate the general applicability of the proposed method. Finally, we test the performance through a set of applications. Numerical tests indicate that the proposed method is superior to existing methods in terms of several parameters like estimated accuracy, privacy level, communication cost and convergence rate.**

***Index Terms*—Distributed optimization, privacy, subspace, noise insertion, consensus, least squares, LASSO.**

I. Introduction

In a world of interconnected and digitized systems, new and innovative signal processing tools are needed to take advantage of the sheer scale of information/data. Such systems are often characterized by "big data". Another central aspect of such systems is their distributed nature, in which the data are usually located in different computational units that form a network. In contrast to the traditional centralized systems, in which all the data must be firstly collected from different units and then processed at a central server, distributed signal processing circumvents this limitation by utilizing the network nature. That is, instead of relying on a single centralized coordination, each node/unit is able to collect information from its neighbors and also to conduct computation on a subset of the overall networked data. This distributed processing has many advantages, such as allowing for flexible scalability of the number of nodes and robustness to dynamical changes in the graph topology. Currently, the computational unit/node in distributed systems is usually limited in resources, as tablets and phones become the primary computing devices used by many people [1], [2]. These devices often contain sensors that can use wireless communication to form so-called ad-hoc networks. Therefore, these devices can collaborate in solving problems by sharing computational resources and sensor data. However, the information collected from sensors such as GPS, cameras and microphones often includes personal data, thus posing a major concern, because such data are private in nature.

There has been a considerable growth of optimization techniques in the field of distributed signal processing, as many traditional signal processing problems in distributed systems can be equivalently formed as convex optimization problems. Owing to the general applicability and flexibility of distributed optimization, optimization has emerged in a wide range of applications such as acoustic signal processing [3], [4], control theory [5] and image processing [6]. Typically, the paradigm of distributed optimization is to separate the global objective function over the network into several local objective functions, which can be solved for each node through exchanging data only with its neighbors. This data exchange is a major concern regarding privacy, because the exchanged data usually contain sensitive information, and traditional distributed optimization schemes do not address this privacy issue. Therefore, how to design a distributed optimizer for processing sensitive data, is a challenge to be overcome in the field.

A. Related works

To address the privacy issues in distributed optimization, the literature has mainly used techniques from differential privacy [7], [8] and secure multiparty computation (SMPC) [9]. Differential privacy is one of the most commonly used non-cryptographic techniques for privacy preservation, because it is computationally lightweight, and it also uses a strict privacy metric to quantify that the posterior guess of the private data is only slightly better than the prior (quantified by a small positive number ϵ). This method of protecting private data has been applied in [10]–[16] through carefully adding noise to the exchanged states or objective functions. However, this noise insertion mechanism involves an inherent trade-off between the privacy and the accuracy of the optimization outputs. Additionally, some approaches [17]–[19] have applied differential privacy with the help of a trusted third party (TTP) like a server/cloud. However, requiring a TTP for

Q. Li and M. G. Christensen are with the Audio Analysis Lab, CREATE, Aalborg University, 9000 Aalborg, Denmark (emails: {qili,mgc}@create.aau.dk).

R. Heusdens is with the Netherlands Defence Academy (NLDA), Het Nieuwe Diep 8, 1781 AC Den Helder, The Netherlands, and with the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands (email: r.heusdens@{mindef.nl,tudelft.nl}).

coordination makes the protocol not completely decentralized (i.e., peer-to-peer setting). Consequently, it thus hinders use in many applications such as wireless sensor networks in which centralized coordinations are unavailable.

SMPC, in contrast, has been widely used in distributed processing, because it provides cryptographic techniques to ensure privacy in a distributed network. More specifically, it aims to compute the result of a function of a number of parties' private data while protecting each party's private data from being revealed to others. Examples of how to preserve privacy by using SMPC have been applied in [20]–[24], in which partially homomorphic encryption (PHE) [25] was used to conduct computations in the encrypted domain. However, PHE requires the assistance of a TTP and thus cannot be directly applied in a fully decentralized setting. Additionally, although PHE is more computationally lightweight than other encryption techniques, such as fully homomorphic encryption [26] and Yao's garbled circuit [27], [28], it is more computationally demanding than the noise insertion techniques, such as differential privacy, because it relies on the computational hardness assumption. To alleviate the bottleneck of computational complexity, another technique in SMPC, called secret sharing [29], has become a popular alternative for distributed processing, because its computational cost is comparable to that of differential privacy. It has been applied in [30] to preserve privacy by splitting sensitive data into pieces and sending them to the so-called computing parties afterward. However, secret sharing generally is expensive in terms of communication cost, because it requires multiple communication rounds for each splitting process.

B. Paper contributions

The main contribution of this paper is that we propose a novel subspace perturbation method, which circumvents the limitations of both the differential privacy and SMPC approaches for distributed signal processing. We propose to insert noise in the subspace such that not only the private data is protected from being revealed to others but also the accuracy of results is not affected. The proposed subspace perturbation method has several attractive properties:

- Compared to differential privacy based approaches, the proposed approach is ensured to converge to the optimum results without compromising privacy. Additionally, it is defined in a completely decentralized setting, because no central aggregator is required.
- In contrast to SMPC based approaches, the proposed approach is efficient in both computation and communication. Because it does not require complex encryption functions (such as those involved in PHE), and it does not have high communication costs (such as those required in the secret sharing approaches).
- The proposed subspace perturbation method is generally applicable to many distributed optimization algorithms like ADMM, PDMM or the dual ascent method.
- The convergence rate of the proposed method is invariant with respect to the amount of inserted noise and thus to the privacy level.

We published preliminary results in [31], [32] where the main concept of subspace perturbation was introduced using PDMM based on a specific application. Here we give a more complete analysis of the proposed subspace perturbation in a broader context, i.e., for all convex problems, and further generalize it into other optimizers such as ADMM and dual ascent.

C. Outline and notation

The remainder of this paper is organized as follows: Section II reviews distributed convex optimization and some important concepts for privacy preservation. Section III defines the problem to be solved and provides qualitative metrics to evaluate the performance. Section IV introduces the primal-dual method of multipliers (PDMM), explaining its key properties used in the proposed approach. Section V introduces the proposed subspace perturbation method based on the PDMM. Section VI shows the general applicability of the proposed method by considering other types of distributed optimizers, such as ADMM and the dual ascent method. In Section VII the proposed approach is applied to a wide range of applications including distributed average consensus, distributed least squares and distributed LASSO. Section VIII demonstrates the numerical results for each application and compares the proposed method with existing approaches. Finally, Section IX concludes the paper.

The following notations are used in this paper. Lowercase letters (x), lowercase boldface letters $(\boldsymbol{x})$, uppercase boldface letters $(\boldsymbol{X})$, overlined uppercase letters $(\bar{X})$ and calligraphic letters $(\mathcal{X})$ denote scalars, vectors, matrices, subspaces and sets, respectively. An uppercase letter (X) denotes the random variable of its lowercase argument, which means that the lowercase letter x is assumed to be a realization of random variable X. $\mathrm{null}\{\cdot\}$ and $\mathrm{span}\{\cdot\}$ denote the nullspace and span of their argument, respectively. $(\boldsymbol{X})^\dagger$ and $(\boldsymbol{X})^\top$ denote the Moore-Penrose pseudo inverse and transpose of $\boldsymbol{X}$, respectively. $\boldsymbol{x}_i$ denotes the i-th entry of the vector $\boldsymbol{x}$, and $\boldsymbol{X}_{ij}$ denotes the (i,j)-th entry of the matrix $\boldsymbol{X}$. $\mathbf{0}$, $\mathbf{1}$ and $\boldsymbol{I}$ denote the vectors with all zeros and all ones, and the identity matrix of appropriate size, respectively.

II. Fundamentals

In this section, we review the fundamentals and some important concepts related to privacy preservation. We first review the distributed convex optimization and highlight its privacy concerns. Then we describe the adversary models that will be addressed later in this paper.

A. Distributed convex optimization

A distributed network is usually modeled as a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{1, 2, ..., n\}$ is the set of nodes, and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ is the set of edges. Let $n = |\mathcal{N}|$ and $m = |\mathcal{E}|$ denote the numbers of nodes and edges, respectively. $\mathcal{N}_i = \{j \,|\, (i,j) \in \mathcal{E}\}$ denotes the neighborhood of node i, and $d_i = |\mathcal{N}_i|$ denotes the degree of node i.

Let $\boldsymbol{x}_i \in \mathbb{R}^{u_i}$ denote the local optimization variable at node i. A standard constrained convex optimization problem over the network can then be expressed as

$$\begin{aligned} \min_{\boldsymbol{x}_i} \quad & \sum_{i\in\mathcal{N}} f_i(\boldsymbol{x}_i) \\ \text{s.t.} \quad & \boldsymbol{B}_{i|j}\boldsymbol{x}_i + \boldsymbol{B}_{j|i}\boldsymbol{x}_j = \boldsymbol{b}_{i,j} \ \forall (i,j) \in \mathcal{E} \end{aligned} \tag{1}$$

where $f_i : \mathbb{R}^{u_i} \mapsto \mathbb{R}\cup\{\infty\}$ denote the local objective function at node i, which we assume to be convex for all nodes $i \in \mathcal{N}$. Additionally, let $v_{i,j}$ denote the dimension of constraints at each edge $(i,j) \in \mathcal{E}$, $\boldsymbol{B}_{i|j} \in \mathbb{R}^{v_{i,j}\times u_i}$, $\boldsymbol{b}_{i,j} \in \mathbb{R}^{v_{i,j}}$ are defined for the constraints. Note that we distinct the subscripts $i|j$ and i,j, where the former is a directed identifier that denotes the directed edge from node i to j and the later i,j is an undirected identifier. Stacking all the local information and let $N_n = \sum_{i\in\mathcal{N}} u_i$ and $M_m = \sum_{(i,j)\in\mathcal{E}} v_{i,j}$, we can compactly express (1) as

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{B}\boldsymbol{x} = \boldsymbol{b} \end{aligned}, \tag{2}$$

where $f : \mathbb{R}^{N_n} \mapsto \mathbb{R}\cup\{\infty\}$, $\boldsymbol{x} \in \mathbb{R}^{N_n}$, $\boldsymbol{B} \in \mathbb{R}^{M_m\times N_n}$, $\boldsymbol{b} \in \mathbb{R}^{M_m}$. For simplicity, we assume the dimension of $\boldsymbol{x}_i$ of all nodes are the same and set it as u, i.e., $u = u_i, \forall i \in \mathcal{N}$, all $\boldsymbol{B}_{i|j}$ be square matrices, i.e., $v_{i,j} = u_i = u, \forall (i,j) \in \mathcal{E}$, and the constraints be zeros, i.e., $\boldsymbol{b} = \boldsymbol{0}$. We thus have $M_m = m\times u$ and $N_n = n \times u$. We further define matrix $\boldsymbol{B}$ based on the incidence matrix of the graph: $\boldsymbol{B}_{i|j} = \boldsymbol{I}$, $\boldsymbol{B}_{j|i} = -\boldsymbol{I}$ if and only if $(i,j) \in \mathcal{E}$ and $i < j$ and $\boldsymbol{B}_{i|j} = -\boldsymbol{I}$, $\boldsymbol{B}_{j|i} = \boldsymbol{I}$ if and only if $(i,j) \in \mathcal{E}$ and $i > j$. Note that $\boldsymbol{B}$ reduces to the graph incidence matrix if $u = 1$.

To solve the above problem without any centralized coordination, several distributed optimizers have been proposed, such as ADMM [33] and PDMM [34], [35], to iteratively solve the problem by communicating only with the local neighborhood. That is, at each iteration (denoted by index k), each node i updates its optimization variable $\boldsymbol{x}_i^{(k)}$ by exchanging data only with its neighbors. The goal of distributed optimizers is to design certain updating functions to ensure that $\boldsymbol{x}_i^{(k)} \to \boldsymbol{x}_i^*$, where $\boldsymbol{x}_i^*$ denotes the optimum solution for node i.

B. Privacy concerns

As mentioned in the introduction, the sensor data captured by an individual's device are usually private in nature. For example, health conditions like Parkinson's disease can be detected by voice signals [36], [37], and activities of householders can be revealed by power consumption data [38]. In the context of distributed optimization, such private information regarding each node i is contained in its local objective function $f_i(\boldsymbol{x}_i)$ [12]. Recall that after each iteration, node i will send the updated optimization variable $\boldsymbol{x}_i^{(k+1)}$ to all of its neighbors. Since this variable is related to $f_i(\boldsymbol{x}_i)$, the revealed $\boldsymbol{x}_i^{(k+1)}$ leaks information about $f_i(\boldsymbol{x}_i)$, e.g., its subgradient $\partial f_i(\boldsymbol{x}_i)$, thereby violating privacy. This privacy concern, however, has not been addressed in existing distributed optimizers. Therefore, in this paper, we attempt to investigate this privacy issue and propose a general solution to achieve privacy-preserving distributed optimization.

C. Adversary model

When designing a privacy-preserving algorithm, it is important to determine the adversary model that qualifies its robustness under various types of security attack. By colluding with a number of nodes, the adversary aims to conduct certain malicious behaviors, such as learning private data or manipulating the function result to be incorrect. These colluding and non-colluding nodes are referred to as corrupted nodes and honest nodes, respectively. Most of the literature has considered only a passive (also called honest-but-curious or semi-honest) adversary, where the corrupted nodes are assumed to follow the instructions of the designed protocol, but are curious about the honest nodes' private data. Another common adversary is the external eavesdropping adversary, which is assumed to infer the private data of the honest nodes by eavesdropping all the communication channels in the network. The eavesdropping adversary in the context of privacy-preserving distributed optimization is relatively unexplored. In fact, many SMPC based approaches, such as secret sharing [30], [39], [40], assume that all messages are transmitted through securely encrypted channels [41], such that the communication channels cannot be eavesdropped upon. However, channel encryption is computationally demanding and is therefore very expensive for iterative algorithms, such as those used here, because they require use of communication channels between nodes many times. In this paper, we design the privacy-preserving distributed optimizers in an efficient way, such that the channel encryption needs to be used only once.

III. Problem Definition

Given the above-mentioned fundamentals, we thus conclude that the goal of privacy-preserving distributed convex optimization is to jointly optimize the constrained convex problem while protecting the private data of each node from being revealed under defined adversary models. More specifically, there are two requirements that should be satisfied simultaneously:

1) Output correctness: at the end of the algorithm, each node i obtains its optimum solution $\boldsymbol{x}_i^*$ and its correct function result $f_i(\boldsymbol{x}_i^*)$, which implies that the global function result $f(\boldsymbol{x}^*)$ has been also achieved.
2) Individual privacy: throughout the execution of the algorithm, the private data, i.e., the information regarding $f_i(\boldsymbol{x}_i)$, held by each honest node should be protected against both passive and eavesdropping adversaries; except for the information that can be directly inferred from the knowledge of the function output and the private data of the corrupted nodes (in Section VII we will explain this in detail).

To quantify the above requirements, two metrics must be defined.

A. Output correctness metric

For each node i, achieving the optimum solution $\boldsymbol{x}_i^*$ implies obtaining the correct function output $f_i(\boldsymbol{x}_i^*)$ as well. To

measure the output correctness for the whole network in terms of the amount of communication, we thus use the mean squared error $\|\boldsymbol{x}^{(k)}-\boldsymbol{x}^*\|_2^2$ over all the nodes as a function of number of transmission: one transmission denotes that one message package is transmitted from one node to another.

B. Individual privacy metric

In the literature, information-theoretic measures like mutual information and ϵ-differential privacy are often adopted as the privacy metric (see [42] for details). In this paper, we deploy mutual information as the metric for quantifying the individual privacy. The reason of choosing mutual information over ϵ-differential privacy is because ϵ-differential privacy corresponds to a worst-case metric and the worst-case privacy leakage can in practice be quite far from the typical leakage of the average user [43]. Notably, mutual information and ϵ-differential privacy are closely related with each other and it is shown in [44] that Bayesian mutual information is a relaxation of ϵ-differential privacy.

Given a continuous random variable X with a probability density function f_X, the differential entropy of X is defined as $h(X)=-\int f(x)\log f(x)dx$. Let Y be another random variable, the conditional entropy $h(X|Y)$ quantifies the uncertainty of X after knowing Y. The mutual information $I(X;Y)$ [45] measures the amount of information learned about X by observing Y, or vice versa, which is given by[1]

$$I(X;Y)=h(X)-h(X|Y). \tag{3}$$

C. Lower bound of information leakage

When defining the individual privacy, we explicitly exclude the information that can be deduced from the function output and the private data of the corrupted nodes, because each node will eventually obtain its function output from the algorithm, and in some cases, this output may contain certain information regarding the private data held by the individual honest node. To explain this scenario more explicitly, take the distributed average consensus as an example. Let $\mathcal{N}_c$ and $\mathcal{N}_h$ denote the set of corrupted and honest nodes, respectively. A group of n people would like to compute their average salary, denoted by s_{ave}, while keeping each person's salary $\boldsymbol{s}_i$ unknown to the others. If the average result is accurate, the salary sum of the honest people can always be computed by $\sum_{i\in\mathcal{N}_h}\boldsymbol{s}_i=n\times s_{\text{ave}}-\sum_{i\in\mathcal{N}_c}\boldsymbol{s}_i$ assuming the adversary knows n, regardless of the underlying algorithms. With the mutual information metric, the salary sum will leak $I(S_i;\sum_{j\in\mathcal{N}_h}S_j)$ amount of information about the salary of the honest node i. Provided that this information leakage is unavoidable, we therefore refer to it as the lower bound of information leakage. We now give a definition of perfect security in the context of distributed processing.

Definition 1. *(Perfect privacy-preserving algorithms.) Given a specific application, let $\delta\geq 0$ denote its lower bound of information leakage. A privacy-preserving algorithm is considered perfect (or achieves perfect security) as long as it reveals no more information than this lower bound δ.*

[1]In the case of discrete random variables, the condition is expressed in terms of the Shannon entropy $H(\cdot)$

IV. Primal-dual method of multipliers

To introduce the main idea of subspace perturbation, we first use PDMM as an example and then generalize it to other distributed optimizers in Section VI, like ADMM or dual ascent. The main reasons for choosing PDMM over other optimizers are its general applicability and its broadcasting property (see Section IV-B) which allows for simplification of the individual privacy analysis. In this section, we first provide a review of the fundamentals of the PDMM and introduce its main properties, which will be used later in the proposed approach.

A. Fundamentals

PDMM is an instance of Peaceman-Rachford splitting of the extended dual problem (refer to [35] for details). It is an alternative distributed optimization tool to ADMM for solving constrained convex optimization problems and is often characterized by a faster convergence rate [34]. For the distributed optimization problem stated in (1), the extended augmented Lagrangian of PDMM is given by

$$L(\boldsymbol{x},\boldsymbol{\lambda})=f(\boldsymbol{x})+(\boldsymbol{P}\boldsymbol{\lambda}^{(k)})^\top\boldsymbol{C}\boldsymbol{x}+\frac{c}{2}\|\boldsymbol{C}\boldsymbol{x}+\boldsymbol{P}\boldsymbol{C}\boldsymbol{x}^{(k)}\|_2^2, \tag{4}$$

and the updating equations of PDMM are given by

$$\boldsymbol{x}^{(k+1)}=\arg\min_{\boldsymbol{x}} L\left(\boldsymbol{x},\boldsymbol{x}^{(k)},\boldsymbol{\lambda}^{(k)}\right), \tag{5}$$

$$\boldsymbol{\lambda}^{(k+1)}=\boldsymbol{P}\boldsymbol{\lambda}^{(k)}+c(\boldsymbol{C}\boldsymbol{x}^{(k+1)}+\boldsymbol{P}\boldsymbol{C}\boldsymbol{x}^{(k)}), \tag{6}$$

where $\boldsymbol{P}\in\mathbb{R}^{2M_m\times 2M_m}$ is a symmetric permutation matrix exchanging the first M_m with the last M_m rows of the matrix it applies, $c>0$ is a constant controlling the convergence rate. $\boldsymbol{\lambda}^{(k)}\in\mathbb{R}^{2M_m}$ denotes the dual variable at iteration k, introduced for controlling the constraints. Each edge $(i,j)\in\mathcal{E}$ is related to two dual variables $\boldsymbol{\lambda}_{i|j},\boldsymbol{\lambda}_{j|i}\in\mathbb{R}^u$, controlled by node i and j, respectively. Additionally, $\boldsymbol{C}\in\mathbb{R}^{2M_m\times N_n}$ is a matrix related to $\boldsymbol{B}$: $\boldsymbol{C}=[\boldsymbol{B}_+^\top\ \boldsymbol{B}_-^\top]^\top$, where $\boldsymbol{B}_+$ and $\boldsymbol{B}_-$ are the matrices containing only the positive and negative entries of $\boldsymbol{B}$, respectively. Of note, $\boldsymbol{C}+\boldsymbol{P}\boldsymbol{C}=[\boldsymbol{B}^\top\ \boldsymbol{B}^\top]^\top$ and $\forall(i,j)\in\mathcal{E}:\boldsymbol{\lambda}_{j|i}=(\boldsymbol{P}\boldsymbol{\lambda})_{i|j}$.

B. Broadcast PDMM

On the basis of (5), the local updating function at each node i is given by

$$\boldsymbol{x}_i^{(k+1)}=\arg\min_{\boldsymbol{x}_i}\left(f_i(\boldsymbol{x}_i)+\sum_{j\in\mathcal{N}_i}\boldsymbol{\lambda}_{j|i}^{(k)\top}\boldsymbol{B}_{i|j}\boldsymbol{x}_i+\frac{c}{2}\sum_{j\in\mathcal{N}_i}\|\boldsymbol{B}_{i|j}\boldsymbol{x}_i+\boldsymbol{B}_{j|i}\boldsymbol{x}_j^{(k)}\|_2^2\right) \tag{7}$$

$$\forall j\in\mathcal{N}_i:\boldsymbol{\lambda}_{i|j}^{(k+1)}=\boldsymbol{\lambda}_{j|i}^{(k)}+c\big(\boldsymbol{B}_{i|j}\boldsymbol{x}_i^{(k+1)}+\boldsymbol{B}_{j|i}\boldsymbol{x}_j^{(k)}\big). \tag{8}$$

We can see that updating $\boldsymbol{\lambda}_{i|j}^{(k+1)}$ requires only $\boldsymbol{\lambda}_{j|i}^{(k)},\boldsymbol{x}_j^{(k)}$ and $\boldsymbol{x}_i^{(k+1)}$, of which $\boldsymbol{\lambda}_{j|i}^{(k)}$ and $\boldsymbol{x}_j^{(k)}$ are already available at node j. Thus, node i needs to broadcast only $\boldsymbol{x}_i^{(k+1)}$ after which the neighboring nodes can update $\boldsymbol{\lambda}_{i|j}^{(k+1)}$ themselves. As a

consequence, the dual variables do not need to be transmitted at all, except for the initialization step.

More specifically, each node i, regarding each iteration k, has the following knowledge:

$$\{\boldsymbol{x}_i^{(k)}, \boldsymbol{\lambda}_{i|j}^{(k)}\}_{j\in\mathcal{N}_i} \cup \{\boldsymbol{x}_j^{(k)}, \boldsymbol{\lambda}_{j|i}^{(k)}\}_{j\in\mathcal{N}_i}, \tag{9}$$

where the first term represents the local variables of node i, and the latter is the variables of its neighbors that are related to node i. Note that the optimization variables $\{\boldsymbol{x}_j^{(k)}\}_{j\in\mathcal{N}_i}$ are sent by the neighbouring nodes during the (iterative) optimization process whereas the dual variables $\{\boldsymbol{\lambda}_{j|i}^{(k)}\}_{j\in\mathcal{N}_i}$ are computed and kept locally at node i, except for the initialization step ($k = 0$) where these variables need to be exchanged through securely encrypted channels.

C. Convergence of dual variables

Consider two successive $\boldsymbol{\lambda}$-updates in (6), in which we have

$$\boldsymbol{\lambda}^{(k+2)} = \boldsymbol{\lambda}^{(k)} + c(\boldsymbol{C}\boldsymbol{x}^{(k+2)} + 2\boldsymbol{P}\boldsymbol{C}\boldsymbol{x}^{(k+1)} + \boldsymbol{C}\boldsymbol{x}^{(k)}), \tag{10}$$

as $\boldsymbol{P}^2 = \boldsymbol{I}$. Let $\bar{H}_p = \text{span}(\boldsymbol{C}) + \text{span}(\boldsymbol{P}\boldsymbol{C})$ and $\bar{H}_p^{\perp} = \text{null}(\boldsymbol{C}^\top) \cap \text{null}((\boldsymbol{P}\boldsymbol{C})^\top)$. Denote $\Pi_{\bar{H}_p}$ as the orthogonal projection onto $\bar{H}_p$. From (10), we conclude that every two $\boldsymbol{\lambda}$-updates affect only $\Pi_{\bar{H}_p}\boldsymbol{\lambda} \in \bar{H}_p$, and $(\boldsymbol{I} - \Pi_{\bar{H}_p})\boldsymbol{\lambda} \in \bar{H}_p^{\perp}$ remains the same. Moreover, as shown in [35], $\left(\boldsymbol{I} - \Pi_{\bar{H}_p}\right)\boldsymbol{\lambda}$ will only be permuted in each iteration and $\Pi_{\bar{H}_p}\boldsymbol{\lambda}$ will eventually converge to $\boldsymbol{\lambda}^*$ given by

$$\boldsymbol{\lambda}^* = -\begin{pmatrix} \boldsymbol{C}^\top \\ (\boldsymbol{P}\boldsymbol{C})^\top \end{pmatrix}^{\dagger} \begin{pmatrix} \partial f(\boldsymbol{x}^*) + c\boldsymbol{C}^\top\boldsymbol{C}\boldsymbol{x}^* \\ \partial f(\boldsymbol{x}^*) + c\boldsymbol{C}^\top\boldsymbol{P}\boldsymbol{C}\boldsymbol{x}^* \end{pmatrix} + c\boldsymbol{C}\boldsymbol{x}^*. \tag{11}$$

We thus can separate the dual variable into two parts:

$$\begin{aligned} \boldsymbol{\lambda}^{(k)} &= \Pi_{\bar{H}_p}\boldsymbol{\lambda}^{(k)} + (\boldsymbol{I} - \Pi_{\bar{H}_p})\boldsymbol{\lambda}^{(k)}, \\ &\to \boldsymbol{\lambda}^* + \boldsymbol{P}^k\left(\boldsymbol{I} - \Pi_{\bar{H}_p}\right)\boldsymbol{\lambda}^{(0)} \end{aligned} \tag{12}$$

since $\boldsymbol{P}^2 = \boldsymbol{I}$. Below, $\bar{H}_p$ and $\bar{H}_p^{\perp}$ are respectively referred to as the convergent subspace and non-convergent subspace associated with PDMM, and similarly $\Pi_{\bar{H}_p}\boldsymbol{\lambda}$ and $\left(\boldsymbol{I} - \Pi_{\bar{H}_p}\right)\boldsymbol{\lambda}$ are called the convergent and non-convergent component of the dual variable, respectively.

V. Proposed approach using PDMM

Having introduced the PDMM algorithm, we now introduce the proposed approach. To achieve a computationally and communicationally efficient solution for privacy preservation, one of the most used techniques is obfuscation by inserting noise, such as those used in differential privacy based approaches. However, inserting noise usually compromises the function accuracy, because the updates are disturbed by noise. To alleviate this trade-off, we propose to insert noise in the non-convergent subspace only so that the accuracy of the optimization solution is not affected (see also Remark 4), thus achieving both privacy and accuracy at the same time. The proposed noise insertion method is referred to as subspace perturbation. Below, we first present some information-theoretic results regarding privacy, after which we explain the proposed subspace perturbation in detail and prove that it satisfies both the output correctness and individual privacy requirements stated in Section III.

A. Privacy preservation using noise insertion

We first present the following information theoretic results regarding privacy, which serve as fundamentals for the proposed approach.

Proposition 1. *Let $\{X_i\}_{i=1,\dots,n}$ denote a set of continuous random variables with mean and variance μ_{X_i} and $\sigma_{X_i}^2$, respectively, assuming they exist. Let $\{Y_i\}_{i=1,\dots,n}$ be a set of mutually independent random variables, i.e., $I(Y_i, Y_j) = 0, i \neq j$, which is independent of $\{X_i\}_{i=1,\dots,n}$, i.e., $I(X_i, Y_j) = 0$ for all $i, j \in \mathcal{N}$. Let $Z_i = X_i + Y_i$ and let $Z_i' = Z_i/\sigma_{Z_i}$ be the normalized random variable with unit variance. We have*

$$\lim_{\sigma_{Y_i}^2\to\infty} I(X_1, \dots, X_n; Z_1, \dots, Z_n) = 0, \tag{13}$$

Proof. See Appendix A. □

Proposition 1 states that, if the lower bound of information leakage $\delta = 0$, we have perfect security if the variance of the inserted noise goes to infinity. That is, the system is asymptotically optimal. In the context of distributed signal processing, however, δ is usually positive as the optimum solution is often an aggregation of the local information of all nodes. In these cases, perfect security can be achieved through finite noise insertion. The following result gives a lower bound on the noise variance for guaranteeing perfect security.

Proposition 2. *Consider two independent random variables X and Y with variance σ_X^2 and σ_Y^2, where X denotes the private data and Y denotes the inserted noise for protecting X. Let $Z = X + Y$. Given $\delta > 0$, if we choose to insert Gaussian noise, we can obtain*

$$I(X; Z) \leq \delta$$

as long as

$$\sigma_Y^2 \geq \frac{\sigma_X^2}{2^{2\delta} - 1}, \tag{14}$$

Proof. See Appendix B. □

Proposition 2 provides a simple way to set the noise variance for achieving perfect security. As an example, if $\delta = 7 \times 10^{-2}$ bits, then perfect security can be guaranteed by setting the variance of the inserted Gaussian noise to be 10 times that of the variance of the private data.

B. Subspace perturbation

We first give the following assumption.

Assumption 1. *The communication channels in the network are securely encrypted when transmitting the initialized dual variable $\boldsymbol{\lambda}^{(0)}$.*

Because of the broadcasting property of the PDMM, after transmission of the initialized dual variables, the updated optimization variable $\boldsymbol{x}_i^{(k+1)}$ is the only information transmitted in the network at each iteration. Based on (7), $\boldsymbol{x}_i^{(k+1)}$ is computed by [2]

$$\boldsymbol{0} \in \partial f_i(\boldsymbol{x}_i^{(k+1)}) + \sum_{j\in\mathcal{N}_i} \boldsymbol{B}_{i|j}\boldsymbol{\lambda}_{j|i}^{(k)} + c\sum_{j\in\mathcal{N}_i}(\boldsymbol{x}_i^{(k+1)} - \boldsymbol{x}_j^{(k)}). \tag{15}$$

[2]Note that $\boldsymbol{B}_{i|j}^\top = \boldsymbol{B}_{i|j}$, $\boldsymbol{B}_{i|j}\boldsymbol{B}_{j|i} = -\boldsymbol{I}$, and $\boldsymbol{B}_{i|j}\boldsymbol{B}_{i|j} = \boldsymbol{I}$.

We can see that the information about the local objective function $f_i(\boldsymbol{x}_i^{(k+1)})$ is contained in the subgradient $\partial f_i(\boldsymbol{x}_i^{(k+1)})$. The main goal of privacy preservation thus becomes minimizing the information loss about $\partial f_i(\boldsymbol{x}_i^{(k+1)})$ given the information known by the adversary. To do so, we first need to analyze how much knowledge the adversary knows regarding (15). Based on the defined adversary models, there are two ways for the adversary to collect information. The first way is to eavesdrop the communication channels. By doing so, the adversary is able to collect, at every iteration $k \geq 0$, all optimization variables $\{\boldsymbol{x}_i^{(k)}\}_{i\in\mathcal{N}}$ in the network. The dual variables $\{\boldsymbol{\lambda}^{(k)}\}$, on the other hand, cannot be eavesdropped because they are only transmitted during the initialization phase ($k = 0$) through securely encrypted channels (see Assumption 1). The second way is to collect the information of all corrupted nodes (a passive adversary model), which is given by (9) for every $i \in \mathcal{N}_c$. Combining the above information together, we conclude that the adversary has the following knowledge regarding $\boldsymbol{x}$ and $\boldsymbol{\lambda}$:

$$\{\boldsymbol{x}_i^{(k)}\}_{i\in\mathcal{N}} \cup \{\boldsymbol{\lambda}_{i|j}^{(k)}\}_{(i,j)\in\mathcal{E}_c}, \tag{16}$$

where $\mathcal{E}_c = \{(i,j) : (i,j) \in \mathcal{E}, (i,j) \notin \mathcal{N}_h \times \mathcal{N}_h\}$ denotes the corrupted edge set. Let $\mathcal{N}_{i,c} = \mathcal{N}_i \cap \mathcal{N}_c$ and $\mathcal{N}_{i,h} = \mathcal{N}_i \cap \mathcal{N}_h$ denote the corrupted and honest neighborhood of node i, respectively. By inspecting (15), we can see that with the knowledge (16), the adversary is able to compute both $c\sum_{j\in\mathcal{N}_i}(\boldsymbol{x}_i^{(k+1)} - \boldsymbol{x}_j^{(k)})$ and the partial sum contributed by the corrupted neighborhood of node i, i.e., $\sum_{j\in\mathcal{N}_{i,c}} \boldsymbol{B}_{i|j}\boldsymbol{\lambda}_{j|i}^{(k)}$. Therefore, after deducing the known terms from (15), what the adversary observes is

$$\partial f_i(\boldsymbol{x}_i^{(k+1)}) + \sum_{j\in\mathcal{N}_{i,h}} \boldsymbol{B}_{i|j}\boldsymbol{\lambda}_{j|i}^{(k)}. \tag{17}$$

In order to achieve perfect security (see Definition 1), the information loss at every iteration should not exceed δ, i.e.,

$$I\left(\partial f_i(X_i^{(k+1)}) ; \partial f_i(X_i^{(k+1)}) + \sum_{j\in\mathcal{N}_{i,h}} \boldsymbol{B}_{i|j}\Lambda_{j|i}^{(k)}\right) \leq \delta. \tag{18}$$

Note that the lower bound δ is given by the application. The only freedom we have, in order to obtain perfect security, is to control the variance of $\sum_{j\in\mathcal{N}_{i,h}} \boldsymbol{B}_{i|j}\Lambda_{j|i}^{(k)}$.

Using (12), we can express (17) as

$$\begin{aligned}\partial f_i(\boldsymbol{x}_i^{(k+1)}) &+ \sum_{j\in\mathcal{N}_{i,h}} \boldsymbol{B}_{i|j}\left(\Pi_{\bar{H}_p}\boldsymbol{\lambda}^{(k)}\right)_{j|i} \\ &+ \sum_{j\in\mathcal{N}_{i,h}} \boldsymbol{B}_{i|j}\left(\boldsymbol{P}^k\left(\boldsymbol{I} - \Pi_{\bar{H}_p}\right)\boldsymbol{\lambda}^{(0)}\right)_{j|i},\end{aligned} \tag{19}$$

from which we conclude that the variance of the convergent term $\sum_{j\in\mathcal{N}_{i,h}} \boldsymbol{B}_{i|j}\left(\Pi_{\bar{H}_p}\boldsymbol{\lambda}^{(k)}\right)_{j|i}$ can not be manipulated to be large as it will always converge since $\Pi_{\bar{H}_p}\boldsymbol{\lambda}^{(k)} \to \boldsymbol{\lambda}^*$. On the contrary, the variance of the non-convergent term $\sum_{j\in\mathcal{N}_{i,h}} \boldsymbol{B}_{i|j}\left(\boldsymbol{P}^k\left(\boldsymbol{I} - \Pi_{\bar{H}_p}\right)\boldsymbol{\lambda}^{(0)}\right)_{j|i}$ can be made arbitrarily large as it only depends on the initialization of the dual variable. As a consequence, we propose to exploit this non-convergent term to guarantee perfect security. That is, given an arbitrary $\delta > 0$, we can adjust the variance of $\sum_{j\in\mathcal{N}_{i,h}} \boldsymbol{B}_{i|j}\left(\boldsymbol{P}^k\left(\boldsymbol{I} - \Pi_{\bar{H}_p}\right)\boldsymbol{\lambda}^{(0)}\right)_{j|i}$ such that (18) is satisfied. In particular, by applying Proposition 2 to the problem at hand, we conclude that a sufficient condition to guarantee perfect security (18) is to initialize the dual variable with Gaussian distributed noise with

$$\exists j \in \mathcal{N}_{i,h} : \mathrm{var}\left(((\boldsymbol{I} - \Pi_{\bar{H}_p})\Lambda^{(0)})_{j|i}\right) \geq \frac{\mathrm{var}(\partial f_i(X_i^{(k+1)}))}{2^{2\delta} - 1}. \tag{20}$$

By inspecting the above condition, we have the following remarks.

Remark 1. *($\left(\boldsymbol{I} - \Pi_{\bar{H}_p}\right)\boldsymbol{\lambda}^{(0)} \neq \boldsymbol{0}$ can be realized by randomly initializing $\boldsymbol{\lambda}^{(0)}$). Of note, $[\boldsymbol{C}\ \boldsymbol{PC}] \in \mathbb{R}^{2M_m\times 2N_n}$ can be viewed as a new graph incidence matrix with $2N_n$ nodes and $2M_m$ edges (see (c) in Fig. 1 for an example); we thus have $\dim(\bar{H}_p) \leq 2N_n - 1$, and $\bar{H}_p^{\perp}$ is non-empty. For a connected graph with the number of edges not less than the number of nodes (i.e., $M_m \geq N_n$), we conclude that a randomly initialized $\boldsymbol{\lambda}^{(0)} \in \mathbb{R}^{2M_m}$ will achieve $\left(\boldsymbol{I} - \Pi_{\bar{H}_p}\right)\boldsymbol{\lambda}^{(0)} \neq \boldsymbol{0}$ with probability 1.*

Remark 2. *(The privacy is still guaranteed if the adversary has full knowledge of the subspace $\bar{H}_p$). If the network topology, i.e., $\boldsymbol{B}$, is known to the adversary, it is able to construct the subspace $\bar{H}_p$ by using $\boldsymbol{C}$ and $\boldsymbol{PC}$. However, knowing $\bar{H}_p$ will not compromise the privacy because the term $\sum_{j\in\mathcal{N}_{i,h}} \boldsymbol{B}_{i|j}\boldsymbol{\lambda}_{j|i}^{(k)}$ in (17) can not be determined by the adversary. The reason is that as long as $\boldsymbol{\lambda}^{(0)} \notin \bar{H}_p$, the adversary is not able to reconstruct the dual variables transmitted between the honest nodes.*

Remark 3. *(The privacy will not be compromised even though the adversary collects information over iterations). Since the updates are only conducted in the convergent subspace, even if the adversary collects information over iterations, it can only know the difference $\boldsymbol{\lambda}^{(k+2)} - \boldsymbol{\lambda}^{(k)} \in \bar{H}_p$. With the knowledge of the dual variables associated with the corrupted nodes only, again the adversary is not able to determine the dual variables related to the honest nodes. Hence, $\sum_{j\in\mathcal{N}_{i,h}} \boldsymbol{B}_{i|j}\boldsymbol{\lambda}_{j|i}^{(k)}$ in (17) can not be determined and the privacy is thus guaranteed.*

Remark 4. *(No trade-off between privacy and accuracy). No matter how much noise is inserted in the non-convergent subspace, the convergence of the optimization variable $\boldsymbol{x} \to \boldsymbol{x}^*$ is guaranteed. By inspecting (4), we can see that the $\boldsymbol{x}$-update is independent of $(\boldsymbol{I} - \Pi_{\bar{H}_p})\boldsymbol{\lambda}$ as $\boldsymbol{\lambda}^{\top}\left(\boldsymbol{I} - \Pi_{\bar{H}_p}\right)\boldsymbol{PC} = 0$.*

Details of the proposed approach using PDMM are shown in algorithm 1. And the analysis of both output correctness and individual privacy is summarized below.

1) Output correctness: As proven in [35], with strictly convex $f(\boldsymbol{x}_i)$, the optimization variable $\boldsymbol{x}_i^{(k)}$ of each node i of the PDMM is guaranteed to converge geometrically (linearly on a logarithmic scale) to the optimum solution $\boldsymbol{x}_i^*$, regardless of the initialization of both $\boldsymbol{x}^{(0)}$ and $\boldsymbol{\lambda}^{(0)}$, thereby ensuring the correctness. Moreover, for convex functions that are not strictly convex, a slightly modified version called averaged PDMM (see Section VII-C for an example) can be used to guarantee the convergence.

Algorithm 1 Privacy-preserving distributed optimization via subspace perturbation using PDMM

1: Each node $i \in \mathcal{N}$ initializes its optimization variable $\boldsymbol{x}_i^{(0)}$ arbitrarily, and its dual variables $\boldsymbol{\lambda}_{i|j}^{(0)}, j \in \mathcal{N}_i$ are randomly initialized with a certain distribution with large variance (specified by the required privacy level).
2: Node i broadcasts $\boldsymbol{x}_i^{(0)}$ and sends the initialized $\{\boldsymbol{\lambda}_{i|j}^{(0)}\}$ to its neighbor j through secure channels [41].
3: **while** $\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|_2 <$ threshold **do**
4: Activated a node uniformly at random, e.g., node i, updates $\boldsymbol{x}_i^{(k+1)}$ using (5).
5: Node i broadcasts $\boldsymbol{x}_i^{(k+1)}$ to its neighbors $j \in \mathcal{N}_i$ (through non-secure channels).
6: After receiving $\boldsymbol{x}_i^{(k+1)}$, each neighbor $j \in \mathcal{N}_i$ updates $\boldsymbol{\lambda}_{i|j}^{(k+1)}$ using (6).
7: **end while**

2) Individual privacy: From (20), we conclude that the proposed algorithm is able to achieve perfect security, against both passive and eavesdropping adversaries as long as the honest node has at least one honest neighbor, i.e., $\mathcal{N}_{i,h} \neq \emptyset$ and Assumption 1 holds.

Overall, the proposed subspace perturbation approach is able to achieve perfect security without compromising the accuracy.

C. Discussions

In Remark 4 we mentioned that the proposed approach has no trade-off between privacy and accuracy. When considering practical signal processing tasks that require quantization, the above claim still holds if the quantizer has a fixed resolution (cell width), but there will be a trade-off between privacy and bit rate as an increase in variance will require a higher bit rate. On the other hand, there will be a trade-off between privacy and accuracy if the quantizer has a fixed bit rate, since increasing the noise will end up with a lower resolution. These quantization related trade-offs, however, exist in all approaches using noise insertion for example the differential privacy approaches. One way to circumvent these trade-offs is to adopt a fixed-rate quantizer that change the cell width adaptively during the iterations [46], [47].

In connection to the above quantization effects, we note that the lower bound of information leakage is very important in reducing these trade-offs, i.e., minimizing the communication bandwidth (bit rate) or the error in the algorithm output, because it helps to specify the minimum noise variance that needs to be added for perfect security; it is not necessary to set the noise variance to be large if the lower bound is not low enough.

VI. Proposed approach using other optimizers

In this section, we demonstrate the general applicability of the proposed subspace perturbation method. In fact, the proposed method can be generally applied to any distributed optimizer if the introduced dual variables converge only in a subspace (i.e., there is a non-empty nullspace), which is indeed usually true, because these optimizers often work in a subspace determined by the incidence matrix of a graph. To substantiate this claim, we will show that the subspace perturbation also applies to ADMM and the dual ascent method. We then illustrate their differences by linking the convergent subspaces to their graph topologies.

A. ADMM

Given a standard distributed optimization problem stated in (1), the augmented Lagrangian function of ADMM [33] is given by

$$L(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z}) = f(\boldsymbol{x}) + \boldsymbol{v}^\top(\boldsymbol{M}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{z}) + \frac{c}{2}\|\boldsymbol{M}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{z}\|^2, \tag{21}$$

where $\boldsymbol{M} \in \mathbb{R}^{2M_m \times N_n}$, like PDMM, is a matrix related to the graph incidence matrix, and $\boldsymbol{M} = \left[\boldsymbol{B}_+^\top \ -\boldsymbol{B}_-^\top\right]^\top$, $\boldsymbol{W} = \left[-\boldsymbol{I}^\top \ -\boldsymbol{I}^\top\right]^\top \in \mathbb{R}^{2M_m \times M_m}$. $\boldsymbol{v} \in \mathbb{R}^{2M_m}$ and $\boldsymbol{z} \in \mathbb{R}^{2M_m}$ are the introduced dual variables and auxiliary variable for constraints, respectively. The updating functions of ADMM are given by

$$\boldsymbol{x}^{(k+1)} = \arg\min_{\boldsymbol{x}} L\left(\boldsymbol{x}, \boldsymbol{z}^{(k)}, \boldsymbol{v}^{(k)}\right) \tag{22}$$

$$\boldsymbol{z}^{(k+1)} = \arg\min_{\boldsymbol{z}} L\left(\boldsymbol{x}^{(k+1)}, \boldsymbol{z}, \boldsymbol{v}^{(k)}\right) \tag{23}$$

$$\boldsymbol{v}^{(k+1)} = \boldsymbol{v}^{(k)} + c\left(\boldsymbol{M}\boldsymbol{x}^{(k+1)} + \boldsymbol{W}\boldsymbol{z}^{(k+1)}\right). \tag{24}$$

By inspecting the $\boldsymbol{v}$-update in (24), we can see that it has a similar structure to that of the $\boldsymbol{\lambda}$-update in (10) in PDMM. Let $\bar{H}_a = \mathrm{span}(\boldsymbol{M}) + \mathrm{span}(\boldsymbol{W})$ and the matrix $[\boldsymbol{M} \ \boldsymbol{W}]$ can also be seen as an incidence matrix of an extended bipartite graph (see (b) in Fig. 1 for an example). Therefore, we have $\dim(\bar{H}_a) \leq M_m + N_n - 1$ and every $\boldsymbol{v}$-update only effects $\left(\Pi_{\bar{H}_a}\right)\boldsymbol{v} \in \bar{H}_a$ and leaves $(\boldsymbol{I} - \Pi_{\bar{H}_a})\boldsymbol{v} \in \bar{H}_a^\perp, \bar{H}_a^\perp = \mathrm{null}(\boldsymbol{M}^\top) \cap \mathrm{null}(\boldsymbol{W}^\top)$, unchanged. In addition to this, similar as (15) in PDMM, the local optimization variable $\boldsymbol{x}_i^{(k+1)}$ of node i is computed by

$$\boldsymbol{0} \in \partial f_i(\boldsymbol{x}_i^{(k+1)}) + \sum_{j \in \mathcal{N}_i} \boldsymbol{v}_{i|j}^{(k)} + c\sum_{j \in \mathcal{N}_i}(\boldsymbol{x}_i^{(k+1)} - \boldsymbol{z}_{i,j}^{(k)}). \tag{25}$$

Note that ADMM is not a broadcasting protocol, i.e., it requires pairwise communications for the auxiliary variable. The individual privacy is thus dependent on both $\boldsymbol{x}$ and $\boldsymbol{z}$. To simplify the analysis, we remark that revealing the auxiliary variable $\boldsymbol{z}$ will not disclose more information than revealing $\boldsymbol{x}$ by the data processing inequality [45]. As a consequence, it is sufficient to analyze the individual privacy through (25). As for the knowledge of the adversary, we note that, in addition to the optimization variable and the dual variable, all the auxiliary variables $\{\boldsymbol{z}_{i,j}^{(k)}\}_{(i,j)\in\mathcal{E}}$ are assumed to be known by the adversary as they can be eavesdropped. We then conclude that after deducing all the known terms, i.e., $\sum_{j \in \mathcal{N}_{i,c}} \boldsymbol{v}_{i|j}^{(k)}$ and $c\sum_{j \in \mathcal{N}_i}(\boldsymbol{x}_i^{(k+1)} - \boldsymbol{z}_{i,j}^{(k)})$, from (25), what the adversary observes is

$$\partial f_i(\boldsymbol{x}_i^{(k+1)}) + \sum_{j \in \mathcal{N}_{i,h}} \boldsymbol{v}_{i|j}^{(k)} = \partial f_i(\boldsymbol{x}_i^{(k+1)}) + \sum_{j \in \mathcal{N}_{i,h}} \left(\Pi_{\bar{H}_a}\boldsymbol{v}^{(k)}\right)_{i|j}$$

$$+ \sum_{j\in\mathcal{N}_{i,h}} \left((\boldsymbol{I}-\Pi_{\bar{H}_a})\boldsymbol{v}^{(0)}\right)_{i|j}. \tag{26}$$

The proof for both the output correctness and individual privacy using ADMM follows a similar structure as that of PDMM. The sufficient condition for perfect security using ADMM becomes

$$\exists j \in \mathcal{N}_{i,h} : \text{var}\left(((\boldsymbol{I}-\Pi_{\bar{H}_a})\boldsymbol{v}^{(0)})_{i|j}\right) \geq \frac{\text{var}(\partial f_i(X_i^{(k+1)}))}{2^{2\delta}-1}. \tag{27}$$

We thus conclude that perfect security can be achieved using ADMM via subspace perturbation.

Of note, there are variations in ADMM. For example, [48] showed that the auxiliary variable can be eliminated and the dual variable update can be simplified by proper initialization; it requires that the dual variable should be initialized properly such that it is in the column space of $[(\boldsymbol{B})^\top \ (-\boldsymbol{B})^\top]^\top$. However, such initialization ensures $(\boldsymbol{I}-\Pi_{\bar{H}_a})\boldsymbol{v}^{(0)} = \boldsymbol{0}$ and thus there is no subspace noise for protecting the private data. Instead, we need to randomly initialize the dual variable $\boldsymbol{v}^{(0)}$ such that (27) can be satisfied.

B. Dual ascent method

The Lagrangian of the dual ascent method for solving (1) is given by

$$L(\boldsymbol{x},\boldsymbol{u}) = f(\boldsymbol{x}) + \boldsymbol{u}^\top \boldsymbol{B}\boldsymbol{x}, \tag{28}$$

where $\boldsymbol{u}\in\mathbb{R}^{M_m}$ is the introduced dual variable. The updating function is given by

$$\boldsymbol{x}^{(k+1)} = \arg\min_{\boldsymbol{x}} L\left(\boldsymbol{x},\boldsymbol{u}^{(k)}\right) \tag{29}$$

$$\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + t^{(k)}\boldsymbol{B}\boldsymbol{x}^{(k+1)}, \tag{30}$$

where $t^{(k)}$ denotes the step size at iteration k. Likewise, the $\boldsymbol{u}$-update in (30) has a similar structure as the $\boldsymbol{\lambda}$-update of PDMM and the $\boldsymbol{v}$-update of ADMM. Here the convergent subspace is $\bar{H}_d = \text{span}(\boldsymbol{B})$ and $\boldsymbol{B}$ is also rank deficient as it is related to the graph incidence matrix. Hence, we conclude that the proposed subspace perturbation method also works for the dual ascent method.

C. Linking graph topologies and subspaces

Thus far, we have shown that the dual variable updates of PDMM, ADMM and the dual ascent method are dependent only on their corresponding subspaces: $\bar{H}_p = \text{span}(\boldsymbol{C}) + \text{span}(\boldsymbol{PC})$, $\bar{H}_a = \text{span}(\boldsymbol{M})+\text{span}(\boldsymbol{W})$ and $\bar{H}_d = \text{span}(\boldsymbol{B})$. As mentioned before, each of the matrices $[\boldsymbol{C}\ \boldsymbol{PC}]$, $[\boldsymbol{M}\ \boldsymbol{W}]$ and $\boldsymbol{B}$ can be seen as an incidence matrix of a graph, therefore they all have a non-empty left nullspace for subspace perturbation as long as $m \geq n$ (Remark 1). To examine the appearance of these constructed graphs, in Fig. 1 we give an example of these graphs and provide illustrative insights into the differences among these optimizers.

VII. Applications

To demonstrate the potential of the proposed approach to be used in a wide range of applications, we now present the application of the proposed subspace perturbation to three fundamental problems: distributed average consensus, distributed least squares and distributed LASSO, because they serve as building blocks for many other signal processing tasks, such as denoising [49], interpolation [50], machine learning [51], [52] and compressed sensing [53], [54]. We first introduce the application and then perform the individual privacy analysis. We will continue using PDMM to introduce the details, but the numerical results of using all the discussed optimizers will be presented in the next section.

A. Privacy-preserving distributed average consensus

The optimization problem setup (1) for distributed average consensus becomes

$$\begin{aligned} \min_{\boldsymbol{x}_i} \quad & \sum_{i\in\mathcal{N}} \frac{1}{2}\|\boldsymbol{x}_i - \boldsymbol{s}_i\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{x}_i = \boldsymbol{x}_j, \forall (i,j)\in\mathcal{E}, \end{aligned} \tag{31}$$

where $\boldsymbol{s}_i$ denotes the initial state value held by node i. The optimum solution for each optimization variable is $\boldsymbol{x}_i^* = n^{-1}\sum_{i\in\mathcal{N}}\boldsymbol{s}_i$ and $\boldsymbol{x}^* = (\boldsymbol{x}_1^*,\ldots,\boldsymbol{x}_n^*)^\top$. Privacy-preserving distributed average consensus is widely investigated in the literature [55]–[63] and it aims estimate the average of all the nodes' initial state values over a network and keep each node's initial state value unknown to others. Such privacy-preserving solutions are highly desired in practice. For example, in face recognition applications, computing mean faces is usually required, thus prompting privacy concerns. Here, a group of people may cooperate to compute the mean face, but none of them would like to reveal their own facial images during the computation.

1) Individual privacy: Note that in distributed average consensus, the requirement of protecting the initial state value $\boldsymbol{s}_i$ is equivalent to protecting $\partial f_i(\boldsymbol{x}_i^{(k+1)}) = \boldsymbol{x}_i^{(k+1)} - \boldsymbol{s}_i$, since the optimization variable $\boldsymbol{x}_i^{(k+1)}$ is known to the adversary and can thus be seen as a constant. To comply with existing approaches, in what follows we will analyze the privacy in terms of the initial state value $\boldsymbol{s}_i$. Substitute $\partial f_i(\boldsymbol{x}_i^{(k+1)}) = \boldsymbol{x}_i^{(k+1)} - \boldsymbol{s}_i$ in (17) and remove the known term $\boldsymbol{x}_i^{(k+1)}$, we then have

$$-\boldsymbol{s}_i + \sum_{j\in\mathcal{N}_{i,h}} \boldsymbol{B}_{i|j}\boldsymbol{\lambda}_{j|i}^{(k)}. \tag{32}$$

As shown in [31], [55], the only revealed information here would be the partial sums of all the honest components (connected subgraphs consist of honest nodes only) after removal of all the corrupted nodes. Let $\mathcal{H}$ denote the node set of the component that the honest node i belongs to, the lower bound of information leakage for node i is thus given by $I(S_i;\sum_{j\in\mathcal{H}}S_j) = \delta$. We conclude that, given δ, the proposed approach is able to achieve perfect security, i.e.,

$$I\left(S_i\,;\, S_i + \sum_{j\in\mathcal{N}_{i,h}} \boldsymbol{B}_{i|j}\Lambda_{j|i}^{(k)}\right) \leq \delta, \tag{33}$$

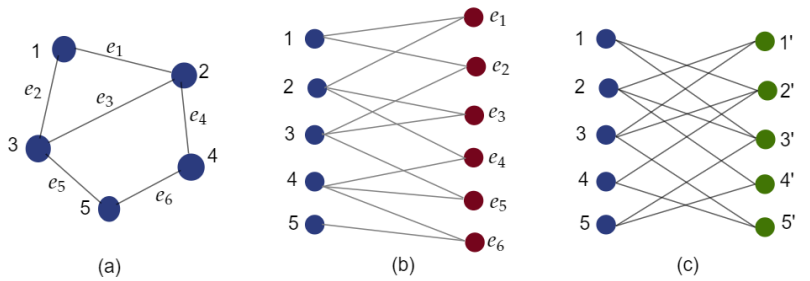


Fig. 1: An example of graph topologies associated with dual ascent, ADMM and PDMM with $u = 1$: (a) A graph with $n = 5$ nodes and $m = 6$ edges. (b) The bipartite graph constructed by ADMM with $n + m$ nodes and $2m$ edges. (c) The bipartite graph constructed by PDMM with $2n$ nodes and $2m$ edges.

by satisfying (20) in which $\mathrm{var}(\partial f_i(X_i^{(k+1)}))$ is replaced by $\mathrm{var}(S_i)$.

B. Privacy-preserving distributed least squares

Privacy-preserving distributed least squares aims to find a solution for a linear system (here we consider an overdetermined system in which there are more equations than unknowns) over a network in which each node has only partial knowledge of the system and is only able to communicate with its neighbors, and in the meantime the local information held by each node should not be revealed to others. More specifically, here the local information of node i means both the input observations, denoted by $\boldsymbol{Q}_i \in \mathbb{R}^{p_i \times u}$, $p_i > u$ and decision vector, denoted by $\boldsymbol{y}_i \in \mathbb{R}^{p_i}$. That is, each node i has p_i observations, and each contains an u-dimensional feature vector. Collecting all the local information, we thus have $\boldsymbol{Q} = [\boldsymbol{Q}_1^\top, \ldots, \boldsymbol{Q}_n^\top]^\top \in \mathbb{R}^{P_n \times u}$ and $\boldsymbol{y} = [\boldsymbol{y}_1^\top, \ldots, \boldsymbol{y}_n^\top]^\top \in \mathbb{R}^{P_n}$, where $P_n = \sum_{i\in\mathcal{N}} p_i$.

The least-squares problem in a distributed network can be formulated as a distributed linearly constrained convex optimization problem, and the problem setup in (1) becomes

$$\begin{array}{ll} \min\limits_{\boldsymbol{x}_i} & \sum_{i\in\mathcal{N}} \frac{1}{2}\|\boldsymbol{y}_i - \boldsymbol{Q}_i\boldsymbol{x}_i\|_2^2 \\ \text{s.t.} & \boldsymbol{x}_i = \boldsymbol{x}_j, \forall (i,j) \in \mathcal{E}, \end{array} \tag{34}$$

where the optimum solution is given by $\boldsymbol{x}_i^* = (\boldsymbol{Q}^\top\boldsymbol{Q})^{-1}\boldsymbol{Q}^\top\boldsymbol{y} \in \mathbb{R}^u, \forall i \in \mathcal{N}$.

1) Individual privacy: Note that the local information $\boldsymbol{y}_i$ and $\boldsymbol{Q}_i$ usually contain users' sensitive information [24]. Take the distributed linear regression as an example, which is widely used in the field of machine learning, and consider the case that several hospitals want to collaboratively learn a predictive model by exploring all the data in their datasets. However, such collaborations are limited because they must comply with policies such as the general data protection regulation (GDPR) and because individual patients/customers may feel uncomfortable with revealing their private information to others, such as insurance data and health condition. In this context, since $\partial f_i(\boldsymbol{x}_i^{(k+1)}) = \boldsymbol{Q}_i^\top(\boldsymbol{Q}_i\boldsymbol{x}_i^{(k+1)} - \boldsymbol{y}_i)$ contains sensitive information regarding the local information $\boldsymbol{Q}_i$ and $\boldsymbol{y}_i$ of node i, it is thus important to protect it from being revealed. We can see that at the end each node obtains the optimum solution $\forall i \in \mathcal{N} : \boldsymbol{x}_i^* = (\boldsymbol{Q}^\top\boldsymbol{Q})^{-1}\boldsymbol{Q}^\top\boldsymbol{y}$. The lower bound δ is thus the amount of information learned about $\partial f_i(\boldsymbol{x}_i)$ of honest node $i \in \mathcal{N}_h$ by knowing $\boldsymbol{x}_i^*$ given the knowledge of the corrupted nodes, i.e., $\{f_i(\boldsymbol{x}_i)\}_{i\in\mathcal{N}_c}$. Hence, the propose approach is able to achieve privacy-preserving distributed least squares, i.e., perfect security (18) is guaranteed as long as (20) is satisfied.

C. Privacy-preserving distributed LASSO

Privacy-preserving distributed LASSO aims to securely find a sparse solution when solving an underdetermined system (in which the number of equations is less than number of unknowns). We thus have a network similar to the previous least squares section but with the dimension $P_n < u$ to ensure an underdetermined system. The distributed LASSO is formulated as a ℓ_1-regularized distributed least squares problem given by

$$\begin{aligned} &\min_{\boldsymbol{x}_i} \sum_{i\in\mathcal{N}} \left(\frac{1}{2}\|\boldsymbol{y}_i - \boldsymbol{Q}_i\boldsymbol{x}_i\|_2^2 + \alpha|\boldsymbol{x}_i|\right) \\ &\text{s.t. } \boldsymbol{x}_i = \boldsymbol{x}_j, \forall (i,j) \in \mathcal{E} \end{aligned} \tag{35}$$

where α the constant controlling the sparsity of the solution. Because the objective function is convex but not strictly convex, we use averaged PDMM to ensure convergence, the $\boldsymbol{x}$-updating function remains the same, and the $\boldsymbol{\lambda}$-updating function in (6) is replaced with a weighted average by

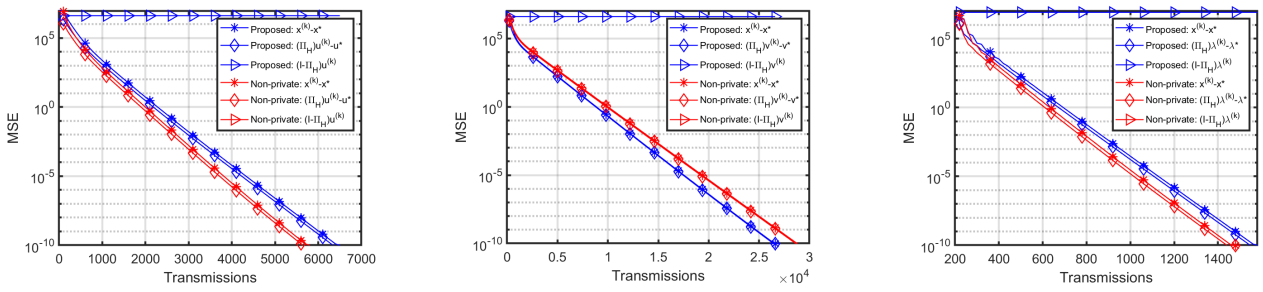
$$\begin{aligned} \boldsymbol{\lambda}^{(k+1)} &= \theta(\boldsymbol{\lambda}^{(k)} + c\boldsymbol{C}(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)})) \\ &+ (1-\theta)\left(\boldsymbol{P}\boldsymbol{\lambda}^{(k)} + c(\boldsymbol{C}\boldsymbol{x}^{(k+1)} + \boldsymbol{P}\boldsymbol{C}\boldsymbol{x}^{(k)})\right), \end{aligned} \tag{36}$$

where $0 < \theta < 1$ is the constant controlling the average weight. The output correctness is ensured by simply replacing the equation (6) in step 6 of algorithm 1 with the above equation. The rest analysis follows similarly as the example for distributed least squares described above. Hence, with (20), we are able to achieve perfect security in distributed LASSO.

VIII. Numerical results

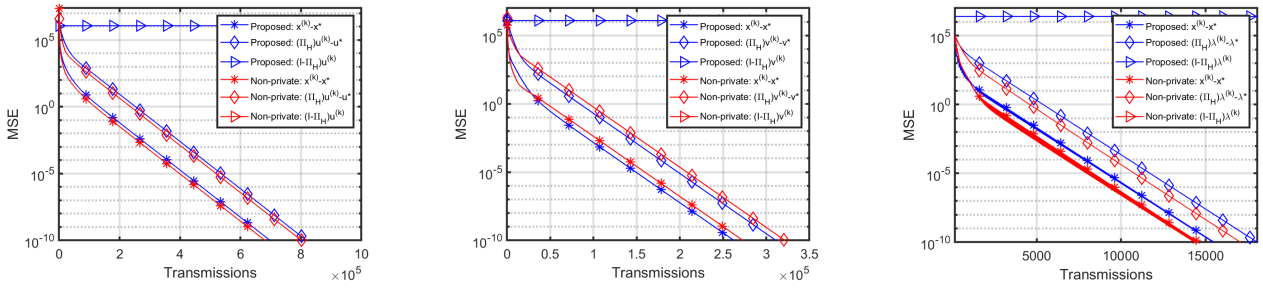
In this section, several numerical tests[3] are conducted to demonstrate both the generally applicability and the benefits of the proposed subspace perturbation in terms of several important parameters including accuracy, convergence rate, communication cost and privacy level.

[3]The code for reproducing these results is available at https://github.com/qiongxiu/PrivacyOptimizationSubspace



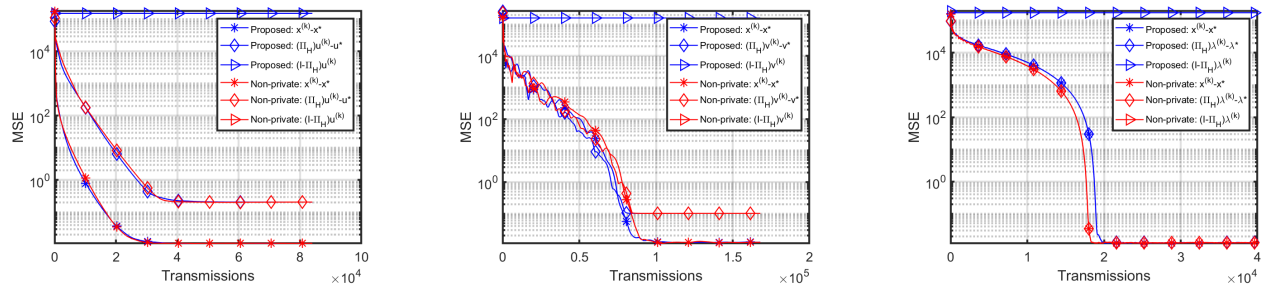
(a) (b) (c)

Fig. 2: Distributed average consensus with two different initializations of the dual variable with a variance of 10^6: convergence of the optimization variable, the convergent and non-convergent component of the dual variable, using (a) dual ascent, (b) ADMM and (c) PDMM.



(a) (b) (c)

Fig. 3: Distributed least squares with two different initializations of the dual variable with a variance of 10^6: convergence of the optimization variable, the convergent and non-convergent component of the dual variable of (a) dual ascent, (b) ADMM and (c) PDMM.



(a) (b) (c)

Fig. 4: Distributed LASSO with two different initializations of the dual variable with a variance of 10^6: convergence of the optimization variable, the convergent and non-convergent component of the dual variable of (a) dual ascent, (b) ADMM and (c) PDMM.

We simulated a distributed network by generating a random graph with $n = 20$ nodes and the communication radius r was set at $r^2 = 2\frac{\log n}{n}$ so ensure that the graph is connected with high probability [64]. For simplicity, all the local data regarding $f(\boldsymbol{x}_i)$ in each application, i.e., $\boldsymbol{s}_i$ in distributed average consensus, $\boldsymbol{Q}_i$ and $\boldsymbol{y}_i$ in distributed least squares and LASSO, are randomly generated from a Gaussian distribution with unit variance, and the optimization variables are initialized with zeros. Additionally, we initialize all the dual variables with a Gaussian distribution with different variances.

A. General applicability

In Fig. 2, 3 and 4, we compare the convergence behavior of the proposed subspace perturbation methods (blue lines) with traditional non-private approaches (red lines) by using three distributed optimizers in three applications: distributed average consensus, least squares and LASSO. More specifically, the blue lines indicate that the dual variables are randomly initialized with a variance of 10^6, such that the non-convergent component (blue line with triangle markers) can protect the private data, whereas the red lines mean that the dual variables are initialized within the convergent subspace, and the private

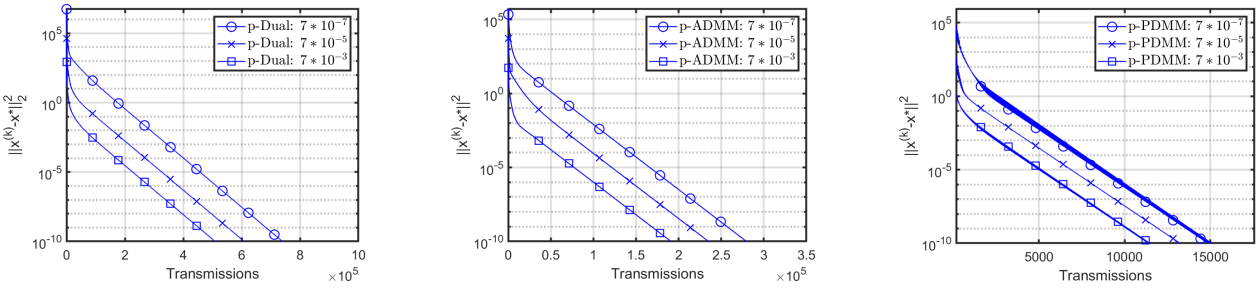


Fig. 5: Convergence of the optimization variable in terms of three different privacy levels, i.e., approximately 7×10^{-3}, 7×10^{-5} and 7×10^{-7} bits, for (a) proposed dual ascent (p-Dual), (b) proposed ADMM (p-ADMM) and (c) proposed PDMM (p-PDMM) in a distributed least squares application.

data are therefore not protected, because the non-convergent component is zero (the red lines with triangle markers are not shown in the plots). We can see that the proposed approach has no effect on the accuracy, because all the optimization variables converge to the same optimum solution as the non-private counterparts. Overall, we can conclude that

1) the proposed approach is able to achieve both accuracy and privacy simultaneously;
2) it is able to solve a variety of convex problems;
3) it is generally applicable to a broad range of distributed convex optimization methods.

B. Privacy level-invariant convergence rate

Another important aspect to quantify the performance is the influence on the convergence rate when considering privacy. Because the convergence rates of the discussed distributed optimizers depend only on the underlying graph topologies rather than the initializations, initializing the dual variables with greater variance will therefore not change the convergence rate; and it will only result in only a larger offset in the initial error. To validate these results, in Fig. 5 we show the convergence behavior of the proposed approaches in the distributed least squares problem under three different privacy levels: $\delta = 7\times10^{-3}$, 7×10^{-5}, and 7×10^{-7} bits. In order to achieve perfect security, based on Proposition 2, the variance of each dual variable is set as $10^2, 10^4$ and 10^6, respectively. As expected, in all optimizers, the convergence rate remains identical regardless of the privacy level.

C. Comparison with differential privacy

To demonstrate the benefits of the proposed method, in Fig. 6, we compare the proposed approaches (both p-PDMM and p-ADMM) with a differential privacy approach [60] by using the distributed average consensus application. We consider three cases in which the variance of each dual variable is set as $0, 10^2, 10^4$. We can see that the accuracy of the differential privacy approach decreases with increasing privacy level. Hence, there is a trade-off between privacy and accuracy. Additionally, the convergence rates of differential privacy approaches will also be affected when increasing the level of privacy, because noise is inserted at every iteration, and a higher privacy level will also result in a slower convergence rate.

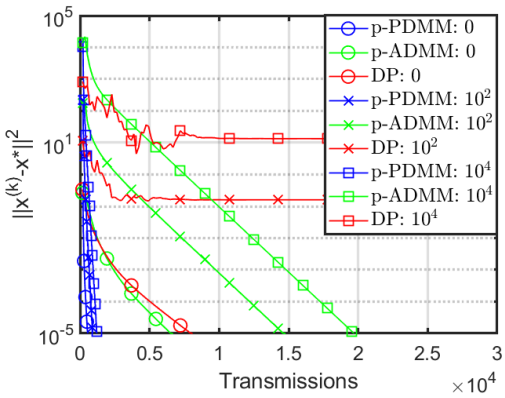


Fig. 6: Performance comparison: convergence behavior under three different noise variances using the proposed p-PDMM, p-ADMM and an existing differential privacy (DP) approach.

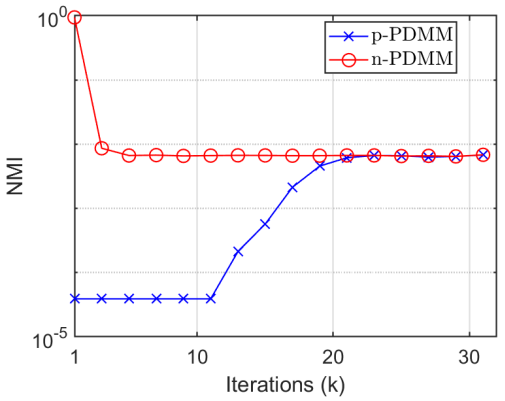


Fig. 7: Normalized mutual information of an arbitrary node i (i.e., $\frac{I(S_i;X_i^{(k)})}{I(S_i;S_i)}$) using the proposed p-PDMM and non-private PDMM (n-PDMM) for each iteration.

D. Information loss over the iterative process

To visualize how the information loss behaves during the iterative process, we perform 10^5 Monte Carlo simulations and estimate the normalized mutual information (NMI) $\frac{I(S_i;X_i^{(k)})}{I(S_i;S_i)}$ of distributed average consensus using the non-parametric entropy estimation toolbox (npeet) [65]. In Fig. 7, we show the estimated normalized mutual information of the proposed p-PDMM with a noise variance of 10^6 and traditional non-private PDMM, in which the dual variables are initialized with zeros. Since both approaches converge to the same average result $\boldsymbol{x}_i^* = n^{-1}\sum_{i\in\mathcal{N}}\boldsymbol{s}_i$, they ultimately have the same information loss $I(S_i;X_i^*)$, which corresponds to the lower bound of information leakage under the condition that there is no passively corrupted nodes. As expected, the information loss of the proposed p-PDMM never exceeds the lower bound; hence, the proposed approach achieves perfect security. However, the n-PDMM reveals all the information about s_i at the first iteration as no noise is inserted; thus the privacy is not protected at all.

IX. Conclusions

In this paper, a novel and general subspace perturbation method was proposed for privacy-preserving distributed optimization. As a noise insertion approach, this method is more practical than SMPC based approaches in terms of both computation and communication costs. Additionally, by inserting noise in subspace, it circumvents the trade-off between privacy and accuracy in traditional noise insertion approaches such as differential privacy. Moreover, the proposed method guarantees perfect security and is generally applicable to various optimizers and all convex problems. Furthermore, we consider both passive and eavesdropping adversary models; in which the private data of each honest node are protected as long as the node has one honest neighbor, and only secure channel encryption in the initialization is required.

Appendix A
Proof of Proposition 1

Proof.

$$\begin{aligned}
&I(X_1,\dots,X_n;Z_1,\dots,Z_n)\\
&\quad= h(Z_1,\dots,Z_n)-h(Z_1,\dots,Z_n|X_1,\dots,X_n)\\
&\quad\overset{\text{(a)}}{=} h(Z_1,\dots,Z_n)-h(Y_1,\dots,Y_n)\\
&\quad\overset{\text{(b)}}{=} \sum_{i=1}^{n} h\big(Z_i|Z_1,\dots,Z_{i-1}\big)-\sum_{i=1}^{n} h\big(Y_i\big)\\
&\quad\overset{\text{(c)}}{\leq} \sum_{i=1}^{n} h\big(Z_i\big)-\sum_{i=1}^{n} h(Y_i)\\
&\quad\overset{\text{(d)}}{=} \sum_{i=1}^{n} I(X_i;Z_i)\\
&\quad\overset{\text{(e)}}{=} \sum_{i=1}^{n} I(X_i/\sigma_{Z_i};Z_i').
\end{aligned}$$

Step (a) holds, as $h(Z_i|X_i)=h(Y_i)$, (b) holds from the chain rule of differential entropy and the condition that the Y_i's are independent random variables, (c) holds, as conditioning decreases entropy, (d) holds, as $h\big(Z_i\big)-h(Y_i)=h(Z_i)-h(Z_i|X_i)=I(X_i;Z_i)$, and (e) holds from the fact that mutual information is scaling invariant. As a consequence,

$$\begin{aligned}
\lim_{\sigma_{Y_i}^2\to\infty}\sum_{i=1}^{n} I(X_i;Z_i) &= \lim_{\sigma_{Z_i}\to\infty}\sum_{i=1}^{n} I(X_i/\sigma_{Z_i};Z_i')\\
&= \sum_{i=1}^{n} I(0;Z_i')=0,
\end{aligned}$$

thereby proving our claims. □

Appendix B
Proof of Proposition 2

Proof. As X and Y are independent, we have $\sigma_Z^2=\sigma_X^2+\sigma_Y^2$. For a Gaussian random variable with variance σ^2, the differential entropy is given by $\frac{1}{2}\log(2\pi e\sigma^2)$, so that

$$\begin{aligned}
\delta = I(X;Z) &= h(Z)-h(Z|X)\\
&= h(Z)-h(Y)\\
&\overset{\text{(a)}}{\leq} \frac{1}{2}\log(2\pi e\sigma_Z^2)-\frac{1}{2}\log(2\pi e\sigma_Y^2)\\
&= \frac{1}{2}\log(1+\sigma_X^2/\sigma_Y^2), \qquad (37)
\end{aligned}$$

where (a) holds, because the maximum entropy of a random variable with fixed variance is achieved by a Gaussian distribution; and equality holds if X is also Gaussian distributed. □

Acknowledgement

We thank the anonymous reviewers for their helpful comments to improve this manuscript. In particular the reviewer who suggested to directly define the subgradient as the private data helped to simplify and clarify this manuscript.

References

[1] M. Anderson, *Technology device ownership, 2015*, Pew Research Center, 2015.
[2] J. Poushter and others, "Smartphone ownership and internet usage continues to climb in emerging economies," *Pew Research Center*, vol. 22, pp. 1–44, 2016.
[3] T. Sherson, W. B. Kleijn, R. Heusdens, "A distributed algorithm for robust lcmv beamforming," in *ICASSP, pp. 101-105*, 2016.
[4] A. I. Koutrouvelis, T. W. Sherson, R. Heusdens, and R. C. Hendriks, "A low-cost robust distributed linearly constrained beamformer for wireless acoustic sensor networks with arbitrary topology," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 26, no. 8, pp. 1434–1448, 2018.
[5] M. Zhu, S. Martínez, "On distributed convex optimization under inequality and equality constraints," *IEEE Trans. Autom. Control*, vol. 57, no. 1, pp. 151–164, 2011.
[6] M. Zibulevsky, M. Elad, "L1-L2 optimization in signal and image processing," *IEEE Signal Process. Mag.*, vol. 27, no. 3, pp. 76–88, 2010.
[7] C. Dwork, "Differential privacy," in *ICALP, pp. 1–12*, 2006.
[8] C. Dwork, F. McSherry, K. Nissim, A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory of Cryptography Conf. , pp. 265-284*, 2006.
[9] R. Cramer, I. B. Damgård, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*, Cambridge University Press, 2015.
[10] Z. Huang, S. Mitra, and N. Vaidya, "Differentially private distributed optimization," in *Proc. Int. Conf. Distrib. Comput. Netw*, 2015, pp. 1–10.
[11] S. Han, U. Topcu, and G. J. Pappas, "Differentially private distributed constrained optimization," *IEEE Trans. Autom. Control*, vol. 62, no. 1, pp. 50–64, 2016.

[12] E. Nozari, P. Tallapragada, and J. Cortés, "Differentially private distributed convex optimization via functional perturbation," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 1, pp. 395–408, 2018.
[13] T. Zhang and Q. Zhu, "Dynamic differential privacy for ADMM-based distributed classification learning," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 1, pp. 172–187, 2016.
[14] X. Zhang, M. M. Khalili, and M. Liu, "Recycled ADMM: Improve privacy and accuracy with less computation in distributed algorithms," in *in Proc. 56th Annu. Allerton Conf. Commun., Control, Comput. pp.959–965*, 2018.
[15] X. Zhang, M. M. Khalili, and M. Liu, "Improving the privacy and accuracy of ADMM-based distributed algorithms," *arXiv:1806.02246*, 2018.
[16] Y. Xiong, J. Xu, K. You, J. Liu and L. Wu, "Privacy preserving distributed online optimization over unbalanced digraphs via subgradient rescaling," *IEEE Trans. Control Netw. Syst.*, 2020.
[17] Z. Huang, R. Hu, Y. Gong, and E. Chan-Tin, "DP-ADMM: ADMM-based distributed learning with differential privacy," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1002–1012, 2019.
[18] M. T. Hale, M. Egerstedt, "Differentially private cloud-based multi-agent optimization with constraints," in *Proc. American Control Conf., pp. 1235-1240*, 2015.
[19] M. T. Hale, M. Egerstedt, "Cloud-enabled differentially private multi-agent optimization with constraints," *IEEE Trans. Control Netw. Syst., vol. 5, no. 4, pp. 1693–1706*, 2018.
[20] Z. Xu and Q. Zhu, "Secure and resilient control design for cloud enabled networked control systems," in *Proc. 1st ACM Workshop Cyber-Phys. Syst.-Secur. Privacy*, 2015, pp. 31–42.
[21] N. M. Freris and P. Patrinos, "Distributed computing over encrypted data," in *Proc. IEEE 54th Annu. Allerton Conf. Commun., Control, Comput.*, 2016, pp. 1116–1122.
[22] Y. Shoukry et al., "Privacy-aware quadratic optimization using partially homomorphic encryption," in *Proc. IEEE 55th Conf. Decis. Control*, 2016, pp. 5053–5058.
[23] C. Wang, K. Ren, and J. Wang, "Secure and practical outsourcing of linear programming in cloud computing," in *INFOCOM*, 2011, pp. 820–828.
[24] C. Zhang, M. Ahmad, and Y. Wang, "ADMM based privacy-preserving decentralized optimization," *IEEE Trans. Inf. Forensics Security, vol. 14, no. 3, pp. 565–580*, 2019.
[25] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT, pp. 223–238*, 1999.
[26] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *STOC*, 2009, pp. 169–178.
[27] A. C. Yao, "Protocols for secure computations," in *FOCS, pp. 160–164*, 1982.
[28] A. C. Yao, "How to generate and exchange secrets," in *FOCS, pp. 162–167*, 1986.
[29] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Advances in Cryptology–CRYPTO, pp. 643–662*. Springer, 2012.
[30] K. Tjell and R. Wisniewski, "Privacy preservation in distributed optimization via dual decomposition and ADMM," in *CDC*, 2020.
[31] Q. Li, R. Heusdens and M. G. Christensen, "Convex optimisation-based privacy-preserving distributed average consensus in wireless sensor networks," in *ICASSP, pp. 5895-5899*, 2020.
[32] Q. Li, R. Heusdens and M. G. Christensen, "Convex optimization-based privacy-preserving distributed least squares via subspace perturbation," in *EUSIPCO, to appear*, 2020.
[33] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al., "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine learning, vol. 3, no. 1, pp. 1–122*, 2011.
[34] G. Zhang and R. Heusdens, "Distributed optimization using the primal-dual method of multipliers," *IEEE Trans. Signal Process., vol. 4, no. 1, pp. 173–187*, 2018.
[35] T. Sherson, R. Heusdens, W. B. Kleijn, "Derivation and analysis of the primal-dual method of multipliers based on monotone operator theory," *IEEE Trans. Signal Inf. Process. Netw., vol. 5, no. 2, pp 334-347*, 2018.
[36] A. H. Poorjam, Y. P. Raykov, R. Badawy, J. R. Jensen, M. G. Christensen and M.A. Little, "Quality control of voice recordings in remote parkinson's disease monitoring using the infinite hidden markov model," in *ICASSP*, 2019.
[37] A. H. Poorjam, M. S. Kavalekalam, L. Shi, Y. P. Raykov, J. R. Jensen, M. A. Little and M. G. Christensen, "Automatic quality control and enhancement for voice-based remote parkinson's disease detection," *arXiv preprint arXiv:1905.11785*, 2019.
[38] G. Giaconi, D. Gündüz, H. V. Poor, "Privacy-aware smart metering: Progress and challenges," *IEEE Signal Process. Mag., vol. 35, no. 6, pp. 59-78*, 2018.
[39] Q. Li and M. G. Christensen, "A privacy-preserving asynchronous averaging algorithm based on shamir's secret sharing," in *EUSIPCO, pp. 1-5*, 2019.
[40] K. Tjell, I. Cascudo and R. Wisniewski, "Privacy preserving recursive least squares solutions," in *ECC, pp.3490–3495*, 2019.
[41] D. Dolev, C. Dwork, O. Waarts, M. Yung, "Perfectly secure message transmission," *J. Assoc. Comput. Mach., vol. 40, no. 1, pp. 17-47,*, 1993.
[42] I. Wagner and D. Eckhoff, "Technical privacy metrics: a systematic survey," *ACM Comput. Surv., vol. 51, no. 3, pp. 1–38*, 2018.
[43] M. Lopuhaä-Zwakenberg, B. Škorić and N. Li, "Information-theoretic metrics for local differential privacy protocols," *arXiv preprint arXiv:1910.07826*, 2019.
[44] P. Cuff and L. Yu, "Differential privacy as a mutual information constraint," in *Proc. 23rd ACM SIGSAC Conf. Comput. Commun. Secur., pp 43–54*, 2016.
[45] T. M. Cover and J. A. Tomas, *Elements of information theory*, John Wiley & Sons, 2012.
[46] D. H. M. Schellekens, T. Sherson, and R. Heusdens, "Quantisation effects in PDMM: A first study for synchronous distributed averaging," in *ICASSP, pp. 4237–4241*, 2017.
[47] D. H. M. Schellekens, T. Sherson, and R. Heusdens, "Quantisation effects in distributed optimisation," in *ICASSP, pp. 3649–3653*, 2018.
[48] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Trans. Signal Processing, vol. 62, no. 7, pp. 1750–1761*, 2014.
[49] J. Pang, G. Cheung, A. Ortega, O. C. Au, "Optimal graph Laplacian regularization for natural image denoising," in *ICASSP, pp 2294-2298*, 2015.
[50] SK Narang, A Gadde, A Ortega, "Signal processing techniques for interpolation in graph structured data," in *ICASSP, pp 5445-5449*, 2013.
[51] Tibshirani, Robert, "Regression shrinkage and selection via the lasso," *J. Royal Statistical Soc. B*, vol. 58, no. 1, pp. 267–288, 1996.
[52] J. Wright, A. Yang, A. Ganesh, S. Sastry, Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, 2008.
[53] E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?," *IEEE Trans. Inf. Theory.*, vol. 52, no. 12, pp. 5406–5425, 2006.
[54] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory.*, vol. 52, no. 4, pp. 1289–1306, 2006.
[55] Q. Li, I. Cascudo, and M. G. Christensen, "Privacy-preserving distributed average consensus based on additive secret sharing," in *EUSIPCO, pp. 1-5*, 2019.
[56] N. Gupta, J. Katz, N. Chopra, "Privacy in distributed average consensus," *IFAC-PapersOnLine, vol. 50, no. 1, pp. 9515-9520*, 2017.
[57] N. Gupta, J. Kat and N. Chopra, "Statistical privacy in distributed average consensus on bounded real inputs," in *ACC, pp 1836-1841*, 2019.
[58] M. Kefayati, M. S. Talebi, B. H. Khalajand H. R. Rabiee , "Secure consensus averaging in sensor networks using random offsets," in *Proc. of the IEEE Int. Conf. on Telec., and Malaysia Int. Conf. on Commun., pp. 556–560*, 2007.
[59] Z. Huang, S. Mitra, and G. Dullerud, "Differentially private iterative synchronous consensus," in *ACM workshop Privacy electron. Soc., pp. 81–90*, 2012.
[60] E. Nozari, P. Tallapragada, and J. Cortés, "Differentially private average consensus: Obstructions, trade-offs, and optimal algorithm design," *Automatica, vol. 81, pp. 221–231*, 2017.
[61] N. E. Manitara and C. N. Hadjicostis, "Privacy-preserving asymptotic average consensus," in *ECC, pp. 760–765*, 2013.
[62] Y. Mo and R. M. Murray, "Privacy preserving average consensus," *IEEE Trans. Automat Contr., vol. 62, no. 2, pp. 753–765*, 2017.
[63] J. He, L. Cai, C. Zhao, P. Cheng, X. Guan, "Privacy-preserving average consensus: privacy analysis and algorithm design," *IEEE Trans. Signal Inf. Process. Netw., vol. 5, no. 1, pp. 127–138*, 2019.
[64] J. Dall and M. Christensen, "Random geometric graphs," *Physical review E, vol. 66, no. 1, pp. 016121*, 2002.
[65] G. Ver Steeg, "Non-parametric entropy estimation toolbox (npeet)," 2000.