ADMM Based Privacy-Preserving Decentralized Optimization

Chunlei Zhang, Muaz Ahmad, and Yongqiang Wang

***Abstract*— Privacy preservation is addressed for decentralized optimization, where N agents cooperatively minimize the sum of N convex functions private to these individual agents. In most existing decentralized optimization approaches, participating agents exchange and disclose states explicitly, which may not be desirable when the states contain sensitive information of individual agents. The problem is more acute when adversaries exist which try to steal information from other participating agents. To address this issue, we propose a privacy-preserving decentralized optimization approach based on alternating direction method of multipliers (ADMM) and partially homomorphic cryptography. To the best of our knowledge, this is the first time that cryptographic techniques are incorporated in a fully decentralized setting to enable privacy preservation in decentralized optimization in the absence of any third party or aggregator. To facilitate the incorporation of encryption in a fully decentralized manner, we introduce a new ADMM, which allows time-varying penalty matrices and rigorously prove that it has a convergence rate of $O(1/t)$. Numerical and experimental results confirm the effectiveness and low-computational complexity of the proposed approach.**

***Index Terms*— ADMM, privacy preservation, decentralized optimization.**

I. Introduction

In recent years, decentralized optimization has been playing key roles in applications as diverse as rendezvous in multi-agent systems [1], spectrum sensing in cognitive networks [2], support vector machine in machine learning [3], online learning [4], classification [5], data regression in statistics [6], source localization in sensor networks [7], and monitoring of smart grids [8]. In these applications, the problem can be formulated in the following general form, in which N agents cooperatively solve an unconstrained optimization problem:

$$\min_{\boldsymbol{y}} \quad \sum_{i=1}^{N} f_i(\boldsymbol{y}), \tag{1}$$

where variable $\boldsymbol{y} \in \mathbb{R}^D$ is common to all agents, function $f_i : \mathbb{R}^D \to \mathbb{R}$ is the local objective function of agent i.

Manuscript received January 11, 2018; revised May 25, 2018 and June 22, 2018; accepted June 28, 2018. Date of publication July 12, 2018; date of current version August 17, 2018. This work was supported by the National Science Foundation under Grant 1738902. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ivan Visconti. *(Corresponding author: Yongqiang Wang.)*

The authors are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29634 USA (e-mail: chunlez@clemson.edu; muaza@clemson.edu; yongqiw@clemson.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2018.2855169

Typical decentralized solutions to the optimization problem (1) include distributed (sub)gradient based algorithms [9], augmented Lagrangian methods (ALM) [10], and the alternating direction method of multipliers (ADMM) as well as its variants [10]–[13], etc. In (sub)gradient based solutions, (sub)gradient computations and averaging among neighbors are conducted iteratively to achieve convergence to the minimum. In augmented Lagrangian and ADMM based solutions, iterative Lagrangian minimization is employed, which, coupled with dual variable update, guarantees that all agents agree on the same minimization solution.

However, most of the aforementioned decentralized approaches require agents to exchange and disclose their states explicitly to neighboring agents in every iteration [9]–[13]. This brings about serious privacy concerns in many practical applications [14]. For example, in projection based source localization, intermediate states are positions of points lying on the circles centered at individual nodes' positions [15], and thus a node may infer the exact position of a neighboring node using three intermediate states, which is undesirable when agents want to keep their position private [16]. In the rendezvous problem where a group of individuals want to meet at an agreed time and place [1], exchanging explicit states may leak their initial locations which may need to be kept secret instead [17]. Other examples include the agreement problem [18], where a group of individuals want to reach consensus on a subject without leaking their individual opinions to others [17], and the regression problem [6], where individual agent's training data may contain sensitive information (e.g., salary, medical record) and should be kept private. In addition, exchanging explicit states without encryption is susceptible to eavesdroppers which try to intercept and steal information from exchanged messages.

To enable privacy preservation in decentralized optimization, one commonly used approach is differential privacy [19]–[21], which adds carefully-designed noise to exchanged states or objective functions to cover sensitive information. However, the added noise also unavoidably compromises the accuracy of optimization results, leading to a trade-off between privacy and accuracy [19]–[21]. In fact, as indicated in [21], even when no noise perturbation is added, differential-privacy based approaches may fail to converge to the accurate optimal solution. It is worth noting that although some differential-privacy based optimization approaches can converge to the optimal solution in the mean-square sense with the assistance of a third party such as a cloud (e.g., [22], [23]), those results are not applicable to the

completely decentralized setting discussed here where no third parties or aggregators exist. Observability-based design has been proposed for privacy preservation in linear multi-agent networks [24], [25]. By properly designing the weights for the communication graph, agents' information will not be revealed to non-neighboring agents. However, this approach cannot protect the privacy of the direct neighbors of compromised agents and it is susceptible to external eavesdroppers. Another approach to enabling privacy preservation is encryption. However, despite successful applications in cloud based control and optimization [26]–[29], conventional cryptographic techniques cannot be applied directly in a completely *decentralized* setting without the assistance of aggregators/third parties (note that traditional secure multi-party computation schemes like fully homomorphic encryption [30] and Yao's garbled circuit [31] are computationally too heavy to be practical for real-time optimization [14]). Other privacy-preserving optimization approaches include [32], [33] which protect privacy via perturbing problems or states. Recently, by using linear dynamical systems theory to facilitate cryptographic design, we proposed a privacy-preserving decentralized linear consensus approach [34]. However, to the best of our knowledge, results are still lacking on cryptography based approaches that can enable privacy for the more complicated general nonlinear optimization problem like (1) in completely *decentralized* setting without any aggregator/third party.

Given that the compromised accuracy of differential-privacy based approaches makes them inappropriate for applications where both accuracy and privacy are of primary concern (for example, when dealing with medical treatment data, even minimal noise may interfere with users' learning [35], [36]), we propose a new privacy-preserving decentralized optimization approach based on ADMM and partially homomorphic cryptography. We used ADMM because it has several advantages. First, ADMM has a fast convergence speed in both primal and dual iterations [13]. By incorporating a quadratic regularization term, ADMM has been shown to be able to obtain satisfactory convergence speed even in ill-conditioned dual functions [12]. Secondly, the convergence of ADMM has been established for non-convex and non-smooth objective function [37]. Moreover, from the implementation point of view, not only is ADMM easy to parallelize and implement, but it is also robust to noise and computation errors [38].

It is worth noting that privacy has different meanings under different settings. For example, in the distributed optimization literature, privacy has been defined as the non-disclosure of agents' states [22], objective functions or subgradients [4], [21], [39]. In this paper, we define privacy as preserving the confidentiality of agents' intermediate states, gradients of objective functions, and objective functions. We protect the privacy of objective functions through protecting intermediate states. In fact, if left unprotected, intermediate states could be used by an adversary to infer the gradients or even objective functions of other nodes through, e.g., data mining techniques. For example, in the regression problem in [6], the objective functions take the form $f_i(\boldsymbol{y}) = \frac{1}{2} \parallel \boldsymbol{s}_i - B_i \boldsymbol{y} \parallel_2^2$, in which s_i and B_i are raw data containing sensitive information such as salary and medical record. When the subgradient method in [9] is used to solve the optimization problem $\min_{\boldsymbol{y}} \sum_{i=1}^{N} f_i(\boldsymbol{y})$, agent i updates its intermediate states in the following way:

$$\boldsymbol{y}_i^{k+1} = \sum_{j=1}^{N} a_{ij} \boldsymbol{y}_j^k + \alpha_k \bigtriangledown f_i(\boldsymbol{y}_i^k)$$

where a_{ij} are weights, α_k is the stepsize, and $\bigtriangledown f_i(\boldsymbol{y}) = B_i^T B_i \boldsymbol{y} - B_i^T \boldsymbol{s}_i$ is the gradient. In this case, an adversary can infer $\bigtriangledown f_i(\boldsymbol{y}_i^k)$ based on exchanged intermediate states $\boldsymbol{y}_i$ if the weights a_{ij} and stepsize α_k are publicly known. We consider two adversaries: *Honest-but-curious adversaries* are agents who follow all protocol steps correctly but are curious and collect all intermediate and input/output data in an attempt to learn some information about other participating agents [40]. *External eavesdroppers* are adversaries who steal information through wiretapping all communication channels and intercepting exchanged messages between agents. Protecting agents' intermediate states can avoid eavesdroppers from inferring any information in optimization.

Contributions: The main contribution of this paper is a privacy-preserving decentralized optimization approach based on ADMM and partially homomorphic cryptography. To our knowledge, this is the first time that cryptographic techniques are incorporated in a fully decentralized setting to enable privacy preservation in decentralized optimization without the assistance of any third party or aggregator. To facilitate the incorporation of homomorphic encryption in ADMM in a fully decentralized manner, we also propose a new ADMM which allows time-varying penalty matrices and rigorously characterize its convergence rate of $O(1/t)$. It is worth noting that the $O(1/t)$ convergence rate requires the subproblems (primary update) to be efficiently computed. When the subproblems are difficult, some suboptimal solution (obtained by, e.g., the approach in [41]) can be used to approximately solve the subproblem. In contrast to differential-privacy based optimization approaches [19]–[22], our approach can enable privacy preservation without sacrificing accuracy. Moreover, our approach does not require strong convexity, Lipschitz or twice continuous differentiability in the objective function. (It is worth noting that [22] requires a trusted cloud and hence is not applicable to the completely decentralized setting discussed here.) Different from the privacy-preserving optimization approach in [39] which only protects the privacy of gradients, our approach preserves the privacy of both intermediate states and gradients. In addition, [39] assumes that an adversary does not have access to the adjacency matrix of the network graph while our approach does not need this assumption.

II. A New ADMM With Time-Varying Penalty Matrices

In this section, we propose a new ADMM with time-varying penalty matrices for (1), which is key for enabling the incorporation of partially homomorphic cryptography in a completely decentralized optimization problem for privacy protection.

A. Problem Formulation

We assume that each f_i in (1) is private and only known to agent i, and all N agents form a bidirectional connected network. Using the graph theory [42], we represent the communication pattern of a multi-agent network by a graph $G = \{V, E\}$, where V denotes the set of agents and E denotes the set of communication links (undirected edges) between agents. Denote the total number of communication links in E as $|E|$. If there exists a communication link between agents i and j, we say that agent j is a neighbor of i (agent i is a neighbor of j as well) and denote the communication link as $e_{i,j} \in E$ if $i < j$ is true or $e_{j,i} \in E$ if $i > j$ is true. Moreover, we denote the set of all neighboring agents of i as $\mathcal{N}_i$ (we consider agent i to be a neighbor of itself in this paper, i.e., $i \in \mathcal{N}_i$, but $e_{i,i} \notin E$).

B. Proximal Jacobian ADMM

To solve (1) in a decentralized manner, we reformulate (1) as follows (which avoids using dummy variables in conventional ADMM [43]):

$$\begin{aligned} \min_{\boldsymbol{x}_i \in \mathbb{R}^D,\, i \in \{1,2,\ldots,N\}} \quad & \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) \\ \text{subject to} \quad & \boldsymbol{x}_i = \boldsymbol{x}_j, \quad \forall e_{i,j} \in E, \end{aligned} \tag{2}$$

where $\boldsymbol{x}_i$ is considered as a copy of $\boldsymbol{x}$ and belongs to agent i. To solve (2), each agent first exchanges its current state $\boldsymbol{x}_i$ with its neighbors. Then it carries out local computations based on its private local objective function f_i and the received state information from neighbors to update its state. Iterating these computations will make every agent reach consensus on a solution that is optimal to (1) when (1) is convex. Detailed implementation of the ADMM algorithm based on Jacobian update is elaborated as follows [44]:

$$\begin{cases} \boldsymbol{x}_i^{t+1} = \underset{\boldsymbol{x}_i}{\operatorname{argmin}} \mathcal{L}(\boldsymbol{x}_1^t, \boldsymbol{x}_2^t, \ldots, \boldsymbol{x}_{i-1}^t, \boldsymbol{x}_i, \boldsymbol{x}_{i+1}^t, \ldots, \boldsymbol{x}_N^t, \boldsymbol{\lambda}^t) \\ \qquad\qquad + \dfrac{\gamma_i}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_i^t \parallel^2, \qquad (3) \\ \boldsymbol{\lambda}_{i,j}^{t+1} = \lambda_{i,j}^t + \rho(\boldsymbol{x}_i^{t+1} - \boldsymbol{x}_j^{t+1}), \quad \forall j \in \mathcal{N}_i \qquad (4) \end{cases}$$

for $i = 1, 2, \ldots, N$. Here, t is the iteration index, $\gamma_i > 0$ $(i = 1, 2, \ldots, N)$ are proximal coefficients, and $\mathcal{L}$ is the augmented Lagrangian function

$$\begin{aligned} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = & \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) \\ & + \sum_{e_{i,j} \in E} (\boldsymbol{\lambda}_{i,j}^T(\boldsymbol{x}_i - \boldsymbol{x}_j) + \frac{\rho}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_j \parallel^2). \end{aligned} \tag{5}$$

In (5), $\boldsymbol{x} = [\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, \ldots, \boldsymbol{x}_N^T]^T \in \mathbb{R}^{ND}$ is the augmented state, $\boldsymbol{\lambda}_{i,j}$ is the Lagrange multiplier corresponding to the constraint $\boldsymbol{x}_i = \boldsymbol{x}_j$, and all $\boldsymbol{\lambda}_{i,j}$ for $e_{i,j} \in E$ are stacked into $\boldsymbol{\lambda} \in \mathbb{R}^{|E|D}$. ρ is the penalty parameter, which is a positive constant scalar.

The above ADMM algorithm cannot protect the privacy of participating agents as states are exchanged and disclosed explicitly among neighboring agents. To facilitate privacy design, we propose a new ADMM with time-varying penalty matrices in the following subsection, which will enable the integration of homomorphic cryptography and decentralized optimization in Sec. III.

C. ADMM With Time-Varying Penalty Matrices

Motivated by the fact that ADMM allows time-varying penalty matrices [45], [46], we present in the following an ADMM with time-varying penalty matrices. It is worth noting that [45], [46] deal with a two-block ($N = 2$) problem. While in this paper, we consider a more general problem with $N \geq 3$ blocks, whose convergence is more difficult to analyze. The generalization from $N = 2$ to $N \geq 3$ is highly non-trivial. In fact, as indicated in [47], a direct extension from two-block to multi-block convex minimization is not necessarily convergent.

We first reformulate (1) in a more compact form:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & A\boldsymbol{x} = \boldsymbol{0}, \end{aligned} \tag{6}$$

where $\boldsymbol{x} = [\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, \ldots, \boldsymbol{x}_N^T]^T \in \mathbb{R}^{ND}$, $f(\boldsymbol{x}) = \sum_{i=1}^{N} f_i(\boldsymbol{x}_i)$, and $A = [a_{m,n}] \otimes I_D \in \mathbb{R}^{|E|D \times ND}$ is the edge-node incidence matrix of graph G as defined in [48], with its $|E|D$ rows corresponding to the $|E|$ communication links and the ND columns corresponding to the N agents. The symbol $\otimes$ denotes Kronecker product. The $a_{m,n}$ element is defined as

$$a_{m,n} = \begin{cases} 1 & \text{if the } m^{th} \text{ edge originates from agent } n, \\ -1 & \text{if the } m^{th} \text{ edge terminates at agent } n, \\ 0 & \text{otherwise.} \end{cases}$$

Here we define that each edge $e_{i,j}$ originates from agent i and terminates at agent j.

Let $\boldsymbol{\lambda}_{i,j}$ be the Lagrange multiplier corresponding to the constraint $\boldsymbol{x}_i = \boldsymbol{x}_j$, then we can form an augmented Lagrangian function of problem (6) as

$$\begin{aligned} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\rho}) = & \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) \\ & + \sum_{e_{i,j} \in E} (\boldsymbol{\lambda}_{i,j}^T(\boldsymbol{x}_i - \boldsymbol{x}_j) + \frac{\rho_{i,j}}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_j \parallel^2), \end{aligned} \tag{7}$$

or in a more compact form:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\rho}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T A\boldsymbol{x} + \frac{1}{2} \parallel A\boldsymbol{x} \parallel_{\boldsymbol{\rho}}^2, \tag{8}$$

where $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_{i,j}]_{ij, e_{i,j} \in E} \in \mathbb{R}^{|E|D}$ is the augmented Lagrange multiplier,

$$\boldsymbol{\rho} = \operatorname{diag}\{\rho_{i,j} I_D\}_{ij, e_{i,j} \in E} \in \mathbb{R}^{|E|D \times |E|D}, \quad \rho_{i,j} > 0$$

is the time-varying penalty matrix, and $\parallel A\boldsymbol{x} \parallel_{\boldsymbol{\rho}}^2 = \boldsymbol{x}^T A^T \boldsymbol{\rho} A\boldsymbol{x}$.

Note that if $\rho_{i,j} I_D$ is the mth block in $\boldsymbol{\rho}$, then $e_{i,j}$ is the mth edge in E, i.e., for the one-dimensional case, $a_{m,i} = 1$ and $a_{m,j} = -1$, and for high dimensional cases, the (m, i)th block of A is I_D and the (m, j)th block of A is $-I_D$.

Now, inspired by [46], we propose a new ADMM which allows time-varying penalty matrices based on Jacobian

update [49]:

$$\begin{cases} \boldsymbol{x}_i^{t+1} = \underset{\boldsymbol{x}_i}{\operatorname{argmin}} \mathcal{L}(\boldsymbol{x}_1^t, \boldsymbol{x}_2^t, \ldots, \boldsymbol{x}_{i-1}^t, \boldsymbol{x}_i, \boldsymbol{x}_{i+1}^t, \ldots, \boldsymbol{x}_N^t, \boldsymbol{\lambda}^t, \boldsymbol{\rho}^t) \\ \qquad\qquad + \dfrac{\gamma_i}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_i^t \parallel^2, & (9) \\ \rho_{i,j}^t \to \rho_{i,j}^{t+1}, & (10) \\ \boldsymbol{\lambda}_{i,j}^{t+1} = \lambda_{i,j}^t + \rho_{i,j}^{t+1}(\boldsymbol{x}_i^{t+1} - \boldsymbol{x}_j^{t+1}), \quad \forall j \in \mathcal{N}_i & (11) \end{cases}$$

for $i = 1, 2, \ldots, N$. It is worth noting that although the communication graph is undirected, we introduce both $\boldsymbol{\lambda}_{i,j}$ and $\boldsymbol{\lambda}_{j,i}$ for $e_{i,j} \in E$ in (3) and (9) to unify the algorithm description. More specifically, we set $\boldsymbol{\lambda}_{i,j}^0 = \rho_{i,j}^0(\boldsymbol{x}_i^0 - \boldsymbol{x}_j^0)$ at $t = 0$ such that $\boldsymbol{\lambda}_{i,j}^t = -\boldsymbol{\lambda}_{j,i}^t$ holds for all $i = 1, 2, \cdots, N$, $j \in \mathcal{N}_i$. In this way, we can unify the update rule of agent i without separating $i > j$ and $i < j$ for $j \in \mathcal{N}_i$, as shown in (12).

Remark 1: The proximal Jacobian ADMM (3)-(4) *can be considered as a special case of* (9)-(11) *by assigning the same and constant weight $\rho_{i,j} = \rho$ to different equality constraints $\boldsymbol{x}_i = \boldsymbol{x}_j$. Different from the ADMM which uses the same ρ (which might be time-varying in, e.g., the two-block optimization problem [50]) for all equality constraints, the new approach uses different and time-varying $\rho_{i,j}$ for different equality constraints $\boldsymbol{x}_i = \boldsymbol{x}_j$. As indicated later, this is key for enabling privacy preservation.*

Remark 2: We did not use Gauss-Seidel update [48], which requires a predefined global order and hence as indicated in [44], is not amenable to parallelism. Different from [44] which has a constant penalty parameter, we intentionally introduce time-varying penalty matrix to enable privacy preservation. Despite enabling new capabilities in privacy protection (with the assistance of partially homomorphic Paillier encryption), introducing time-varying penalty matrix also reduces convergence rate to $O(1/t)$, in contrast to the $o(1/t)$ rate in [44]. Besides giving new capabilities in privacy and different result in convergence rate, the novel idea of intentional time-varying penalty matrix also leads to difference in theoretical analysis in comparison with [44]. For example, different from [44] which relies on constant-penalty based monotonically non-increasing sequences to prove convergence, our introduction of time-varying penalty matrix results in non-monotonic sequences which prompted us to use a variational inequality to facilitate the analysis.

It is obvious that the new ADMM (9)–(11) can be implemented in a decentralized manner. The detailed implementation procedure is outlined in Algorithm 1.

Remark 3: A weighted ADMM which also assigns different weights to different equality constraints is proposed in [13]. However, the weights in [13] are constant while Algorithm 1 allows time-varying weights in each iteration, which, as shown later, is key to enable the integration of partially homomorphic cryptography with decentralized optimization.

D. Convergence Analysis

In this subsection, we rigorously prove the convergence of Algorithm 1 under the following standard assumptions [48]:

Assumption 1: Each private local function $f_i : \mathbb{R}^D \to \mathbb{R}$ is convex and continuously differentiable.

Algorithm 1

Initial Setup: Each agent i initializes $\boldsymbol{x}_i^0$, $\rho_{i,j}^0$.
Input: $\boldsymbol{x}_i^t$, $\boldsymbol{\lambda}_{i,j}^{t-1}$, $\rho_{i,j}^t$
Output: $\boldsymbol{x}_i^{t+1}$, $\boldsymbol{\lambda}_{i,j}^t$, $\rho_{i,j}^{t+1}$

1) Each agent i sends $\boldsymbol{x}_i^t$, $\rho_{i,j}^t$ to its neighboring agents, and then set $\rho_{i,j}^t = \min\{\rho_{i,j}^t, \rho_{j,i}^t\}$. It is clear that $\rho_{i,j}^t = \rho_{j,i}^t$ holds.
2) Each agent i updates $\boldsymbol{\lambda}_{i,j}^t$ as follows for $j \in \mathcal{N}_i$
$$\boldsymbol{\lambda}_{i,j}^t = \boldsymbol{\lambda}_{i,j}^{t-1} + \rho_{i,j}^t(\boldsymbol{x}_i^t - \boldsymbol{x}_j^t). \quad (12)$$
It is clear that $\boldsymbol{\lambda}_{i,j}^t = -\boldsymbol{\lambda}_{j,i}^t$ holds (note that when $t = 0$, we set $\boldsymbol{\lambda}_{i,j}^0 = \rho_{i,j}^0(\boldsymbol{x}_i^0 - \boldsymbol{x}_j^0)$).
3) All agents update their local vectors in parallel:
$$\boldsymbol{x}_i^{t+1} \in \operatorname{argmin}_{\boldsymbol{x}_i} f_i(\boldsymbol{x}_i) + \frac{\gamma_i}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_i^t \parallel^2 + \sum_{j \in \mathcal{N}_i} ((\boldsymbol{\lambda}_{i,j}^t)^T \boldsymbol{x}_i + \frac{\rho_{i,j}^t}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_j^t \parallel^2). \quad (13)$$
Here we added two proximal terms $\frac{\rho_{i,i}^t}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_i^t \parallel^2$ and $\frac{\gamma_i}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_i^t \parallel^2$ to accommodate the influence of $\boldsymbol{x}_i^t$. For all $\gamma_i > 0$, $\rho_{i,i}^t$ is set to
$$\rho_{i,i}^t = 1 - \sum_{j \in \mathcal{N}_i, j \neq i} \rho_{i,j}^t. \quad (14)$$
4) Each agent i updates $\rho_{i,j}^{t+1}$ for all $j \in \mathcal{N}_i$ and sets $t = t + 1$. The detailed update rule for $\rho_{i,j}$ will be elaborated later in Theorem 1.

Assumption 2: Problem (6) *has an optimal solution, i.e., the Lagrangian function*

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T A \boldsymbol{x} \quad (15)$$

has a saddle point $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$ such that

$$L(\boldsymbol{x}^*, \boldsymbol{\lambda}) \leq L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) \leq L(\boldsymbol{x}, \boldsymbol{\lambda}^*)$$

holds for all $\boldsymbol{x} \in \mathbb{R}^{ND}$ and $\boldsymbol{\lambda} \in \mathbb{R}^{|E|D}$.

Denote the iterating results in the kth step in Algorithm 1 as follows:

$$\begin{aligned} \boldsymbol{x}^k &= [\boldsymbol{x}_1^{kT}, \boldsymbol{x}_2^{kT}, \ldots, \boldsymbol{x}_N^{kT}]^T \in \mathbb{R}^{ND}, \\ \boldsymbol{\lambda}^k &= [\boldsymbol{\lambda}_{i,j}^k]_{ij, e_{i,j} \in E} \in \mathbb{R}^{|E|D}, \\ \boldsymbol{\rho}^k &= \operatorname{diag}\{\rho_{i,j}^k I_D\}_{ij, e_{i,j} \in E} \in \mathbb{R}^{|E|D \times |E|D}. \end{aligned}$$

Further augment the coefficients γ_i $(i = 1, 2, \ldots, N)$ in (13) into the matrix form

$$Q_P = \operatorname{diag}\{\gamma_1, \gamma_2, \ldots, \gamma_N\} \otimes I_D \in \mathbb{R}^{ND \times ND},$$

and augment $\rho_{i,j}^k$ into the following matrix form

$$Q_C^k = \operatorname{diag}\{\sum_{j \in \mathcal{N}_1} \rho_{1,j}^k, \sum_{j \in \mathcal{N}_2} \rho_{2,j}^k, \ldots, \sum_{j \in \mathcal{N}_N} \rho_{N,j}^k\} \otimes I_D,$$

and $Q_C^k \in \mathbb{R}^{ND \times ND}$. By plugging (14) into Q_C^k, we have $Q_C^k = I_{ND}$, i.e., Q_C^k is an identity matrix.

Now we are in position to give the main results of this subsection:

Theorem 1: Under Assumption 1 and Assumption 2, Algorithm 1 is guaranteed to converge to an optimal solution to (6) *if the following two conditions are met:*

Condition A: The sequence $\{\boldsymbol{\rho}^k\}$ *satisfies*

$$0 \prec \boldsymbol{\rho}^0 \preceq \boldsymbol{\rho}^k \preceq \boldsymbol{\rho}^{k+1} \preceq \bar{\boldsymbol{\rho}}, \quad \forall k \geq 0,$$

where $\boldsymbol{\rho}^0 \succ 0$ *means that* $\boldsymbol{\rho}^0$ *is positive definite, and similarly* $\boldsymbol{\rho}^k \preceq \boldsymbol{\rho}^{k+1}$ *means that* $\boldsymbol{\rho}^{k+1} - \boldsymbol{\rho}^k$ *is positive semi-definite.*

Condition B: $Q_P + Q_C^k \succ A^T \bar{\boldsymbol{\rho}} A$.

Proof: The proof is provided in the Appendix. ■

Theorem 2: The convergence rate of Algorithm 1 is $O(1/t)$, *where* t *is the iteration time.*

Proof: The proof is provided in the Appendix. ■

III. Privacy-Preserving Decentralized Optimization

Algorithm 1 requires agents to exchange and disclose states explicitly in each iteration among neighboring agents to reach consensus on the final optimal solution. In this section, we combine partially homomorphic cryptography with Algorithm 1 to propose a privacy-preserving approach for decentralized optimization. We first give the definition of privacy used in this paper.

Definition 1: A mechanism $\mathcal{M} : \mathcal{M}(\mathcal{X}) \to \mathcal{Y}$ *is defined to be privacy preserving if the input* $\mathcal{X}$ *cannot be uniquely derived from the output* $\mathcal{Y}$.

This definition of privacy is inspired by the privacy-preservation definitions in [4], [39], and [51]–[54] which take advantages of the fact that if a system of equations has infinite number of solutions, it is impossible to derive the exact value of the original input data from the output data. Therefore, privacy preservation is achieved (see [53, Part 4.2.2]). Next, we introduce the Paillier cryptosystem and our privacy-preserving approach.

A. Paillier Cryptosystem

Our method uses the flexibility of time-varying penalty matrices in Algorithm 1 to enable the incorporation of Paillier cryptosystem [55] in a completely decentralized setting. The Paillier cryptosystem is a public-key cryptosystem which uses a pair of keys: a public key and a private key. The public key can be disseminated publicly and used by any person to encrypt a message, but the message can only be decrypted by the private key. The Paillier cryptosystem includes three algorithms, which are detailed in the following.

A notable feature of Paillier cryptosystem is that it is additively homomorphic, i.e., the ciphertext of $m_1 + m_2$ can be obtained from the ciphertext of m_1 and m_2 directly when $0 \leq m_1 + m_2 < n$ holds:

$$\mathcal{E}(m_1, r_1) \cdot \mathcal{E}(m_2, r_2) = \mathcal{E}(m_1 + m_2, r_1 r_2), \tag{16}$$

$$\mathcal{E}(m)^k = \mathcal{E}(km), \quad k \in \mathbb{Z}^+. \tag{17}$$

Due to the existence of random r, the Paillier cryptosystem is resistant to the dictionary attack [56]. Since r_1 and r_2 play no role in the decryption process, (16) can be simplified as

$$\mathcal{E}(m_1) \cdot \mathcal{E}(m_2) = \mathcal{E}(m_1 + m_2). \tag{18}$$

Paillier Cryptosystem

Key generation:

1) Choose two large prime numbers p and q of equal bit-length and compute $n = pq$.
2) Let $g = n + 1$.
3) Let $\lambda = \phi(n) = (p-1)(q-1)$, where $\phi(\cdot)$ is the Euler's totient function.
4) Let $\mu = \phi(n)^{-1} \mod n$ which is the modular multiplicative inverse of $\phi(n)$.
5) The public key k_p for encryption is (n, g).
6) The private key k_s for decryption is (λ, μ).

Encryption $(c = \mathcal{E}(m))$**:**

Recall the definitions of $\mathbb{Z}_n = \{z | z \in \mathbb{Z}, 0 \leq z < n\}$ and $\mathbb{Z}_n^* = \{z | z \in \mathbb{Z}, 0 \leq z < n, \gcd(z, n) = 1\}$.

1) Choose a random $r \in \mathbb{Z}_n^*$.
2) The ciphertext is given by $c = g^m \cdot r^n \mod n^2$, where $m \in \mathbb{Z}_n, c \in \mathbb{Z}_{n^2}^*$.

Decryption $(m = D(c))$**:**

1) Define the integer division function $L(\mu) = \frac{\mu - 1}{n}$.
2) The plaintext is $m = L(c^\lambda \mod n^2) \cdot \mu \mod n$.

B. Privacy-Preserving Decentralized Optimization

In this subsection, we combine Paillier cryptosystem with Algorithm 1 to enable privacy preservation in the decentralized solving of optimization problem (1). First, note that solving (13) amounts to solving the following problem:

$$\nabla f_i(\boldsymbol{x}_i) + \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{i,j}^t + \rho_{i,j}^t (\boldsymbol{x}_i - \boldsymbol{x}_j^t)) + \gamma_i (\boldsymbol{x}_i - \boldsymbol{x}_i^t) = \mathbf{0}. \tag{19}$$

Let $\boldsymbol{\lambda}_i = \sum_{j \in N_i} \boldsymbol{\lambda}_{i,j}$, then (19) reduces to the following equation

$$\nabla f_i(\boldsymbol{x}_i) + (\sum_{j \in \mathcal{N}_i} \rho_{i,j}^t + \gamma_i) \boldsymbol{x}_i + \boldsymbol{\lambda}_i^t - \sum_{j \in \mathcal{N}_i} \rho_{i,j}^t \boldsymbol{x}_j^t - \gamma_i \boldsymbol{x}_i^t = \mathbf{0}. \tag{20}$$

Given that we have set $\rho_{i,i}^t = 1 - \sum_{j \in \mathcal{N}_i, j \neq i} \rho_{i,j}^t$ in (14), we can further reduce (20) to

$$\nabla f_i(\boldsymbol{x}_i) + (1 + \gamma_i) \boldsymbol{x}_i + \boldsymbol{\lambda}_i^t - \sum_{j \in \mathcal{N}_i} \rho_{i,j}^t (\boldsymbol{x}_j^t - \boldsymbol{x}_i^t) - (1 + \gamma_i) \boldsymbol{x}_i^t = \mathbf{0}. \tag{21}$$

By constructing $\rho_{i,j}^t, i \neq j$ as the product of two random positive numbers, i.e., $\rho_{i,j}^t = b_{i \to j}^t \times b_{j \to i}^t = \rho_{j,i}^t$, with $b_{i \to j}^t$ only known to agent i and $b_{j \to i}^t$ only known to agent j, we can propose the following privacy-preserving solution to (1) based on Algorithm 1:

Algorithm 2

Initial Setup: Each agent initializes $\boldsymbol{x}_i^0$.
Input: $\boldsymbol{x}_i^t$, $\boldsymbol{\lambda}_{i,j}^{t-1}$
Output: $\boldsymbol{x}_i^{t+1}$, $\boldsymbol{\lambda}_{i,j}^t$

1) Agent i encrypts $-\boldsymbol{x}_i^t$ with its public key k_{pi}:
$$\boldsymbol{x}_i^t \rightarrow \mathcal{E}_i(-\boldsymbol{x}_i^t).$$
Here the subscript i denotes encryption using the public key of agent i.
2) Agent i sends $\mathcal{E}_i(-\boldsymbol{x}_i^t)$ and its public key k_{pi} to neighboring agents.
3) Agent $j \in \mathcal{N}_i$ encrypts $\boldsymbol{x}_j^t$ with agent i's public key k_{pi}:
$$\boldsymbol{x}_j^t \rightarrow \mathcal{E}_i(\boldsymbol{x}_j^t).$$
4) Agent $j \in \mathcal{N}_i$ computes the difference directly in ciphertext:
$$\mathcal{E}_i(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t) = \mathcal{E}_i(\boldsymbol{x}_j^t) \cdot \mathcal{E}_i(-\boldsymbol{x}_i^t).$$
5) Agent $j \in \mathcal{N}_i$ computes the $b_{j\rightarrow i}^t$-weighted difference in ciphertext:
$$\mathcal{E}_i(b_{j\rightarrow i}^t(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t)) = (\mathcal{E}_i(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t))^{b_{j\rightarrow i}^t}.$$
6) Agent $j \in \mathcal{N}_i$ sends $\mathcal{E}_i(b_{j\rightarrow i}^t(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t))$ back to agent i.
7) Agent i decrypts the message received from j with its private key k_{si} and multiples the result with $b_{i\rightarrow j}^t$ to get $\rho_{i,j}^t(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t)$.
8) Computing (12), agent i obtains $\boldsymbol{\lambda}_{i,j}^t$.
9) Computing (21), agent i obtains $\boldsymbol{x}_i^{t+1}$.
10) Each agent updates $b_{i\rightarrow j}^t$ to $b_{i\rightarrow j}^{t+1}$ and sets $t = t + 1$.

Several remarks are in order:

1) The only situation that a neighbor knows the state of agent i is when $\boldsymbol{x}_i^t = \boldsymbol{x}_j^t$ is true for $j \in \mathcal{N}_i$. Otherwise, agent i's state $\boldsymbol{x}_i^t$ is encrypted and will not be revealed to its neighbors.
2) Agent i's state $\boldsymbol{x}_i^t$ and its intermediate communication data $b_{j\rightarrow i}^t(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t)$ will not be revealed to outside eavesdroppers, since they are encrypted.
3) The state of agent $j \in \mathcal{N}_i$ will not be revealed to agent i, because the decrypted message obtained by agent i is $b_{j\rightarrow i}^t(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t)$ with $b_{j\rightarrow i}^t$ only known to agent j and varying in each iteration.
4) We encrypt $\mathcal{E}_i(-\boldsymbol{x}_i^t)$ because it is much easier to compute addition in ciphertext. The issue regarding encryption of signed values using Paillier will be addressed in Sec. V.
5) Paillier encryption cannot be performed on vectors directly. For vector messages $\boldsymbol{x}_i^t \in \mathbb{R}^D$, each element of the vector (a real number) has to be encrypted separately. For notation convenience, we still denote it in the same way as scalars, e.g., $\mathcal{E}_i(-\boldsymbol{x}_i^t)$.
6) Paillier cryptosystem only works for integers, so additional steps have to be taken to convert real values in optimization to integers. This may lead to quantization errors. A common workaround is to scale the real value before quantization, as discussed in detail in Sec. V.
7) By incorporating Paillier cryptosystem, it is obvious that the computation complexity and communication load will increase. However, we argue that the privacy provided matters more than this disadvantage when privacy is of primary concern. Furthermore, our experimental results on Raspberry Pi boards confirm that the added communication and computation overhead is fully manageable on embedded microcontrollers (cf. Sec. VII).
8) Our approach is more suitable for small and medium sized optimization problems such as the source localization problem [7] and power system monitoring problem [57] addressed in our prior work.

The key to achieve privacy preservation is to construct $\rho_{i,j}^t, i \neq j$ as the product of two random positive numbers $b_{i\rightarrow j}^t$ and $b_{j\rightarrow i}^t$, with $b_{i\rightarrow j}^t$ generated by and only known to agent i and $b_{j\rightarrow i}^t$ generated by and only known to agent j. Next we show that the privacy preservation mechanism does not affect the convergence to the optimal solution.

Theorem 3: The privacy-preserving algorithm 2 will generate a solution in an ε ball around the optimum if $b_{i\rightarrow j}^t$, $b_{j\rightarrow i}^t$, and γ_i are updated in the following way (where ε depends on the quantization error):

1) *$b_{i\rightarrow j}^t$ is randomly chosen from $[b_{i\rightarrow j}^{t-1}, \bar{b}_{i\rightarrow j}]$, with $\bar{b}_{i\rightarrow j} > 0$ denoting a predetermined constant only known to agent i;*
2) *γ_i is chosen randomly in the interval $[N\bar{b}^2, \bar{\bar{b}}]$, with $\bar{b} > \max\{\bar{b}_{i\rightarrow j}\}$ denoting a predetermined positive constant known to everyone and $\bar{\bar{b}}$ a threshold chosen arbitrarily by agent i and only known to agent i.*

Proof: It can be easily obtained that if $b_{i\rightarrow j}^t$ is updated following 1), and γ_i is updated following 2), then Condition A and Condition B in Theorem 1 will be met automatically. Therefore, the states in algorithm 2 should converge to the optimal solution. However, since Paillier cryptosystem only works on unsigned integers, it requires converting real-valued states to integers using e.g., fixed-point arithmetic encoding [58] (after scaled by a large number $N_{\max}$, cf. Sec. V), which leads to quantization errors. The quantization errors lead to numerical errors on the final solution and hence the "ε-ball" statement in Theorem 3. It is worth noting that the numerical error here is no different from the conventional quantization errors met by all algorithms when implemented in practice on a computer. A quantized analysis of the ε-ball is usually notoriously involved and hence we refer interested readers to [59] which is dedicated to this problem. Furthermore, we would like to emphasize that this quantization error can be made arbitrarily small by using an arbitrarily large $N_{\max}$. In fact, our simulation results in Sec. VI B showed that under $N_{\max} = 10^6$, the final error was on the order of 10^{-14}. ■

IV. Privacy Analysis

As indicated in the introduction, our approach aims to protect the privacy of agents' intermediate states $\boldsymbol{x}_i^t$s and gradients of f_is as well as the objective functions. In this

section, we rigorously prove that these private information cannot be inferred by honest-but-curious adversaries and external eavesdroppers, which are commonly used attack models in privacy studies [40] (cf. definition in Sec. I). It is worth noting that the form of each agent's local objective function can also be totally blind to others, e.g., whether it is a quadratic, exponential, or other forms of convex functions is only known to an agent itself.

As indicated in Sec. III, our approach in Algorithm 2 guarantees that state information is not leaked to any neighbor in one iteration. However, would some information get leaked over time? More specifically, if an honest-but-curious adversary observes carefully its communications with neighbors over several steps, can it put together all the received information to infer its neighbor's state?

We can rigorously prove that an honest-but-curious adversary cannot infer the exact states of its neighbors even by collecting samples from multiple steps.

Theorem 4: Assume that all agents follow Algorithm 2. Then agent j's exact state value $\boldsymbol{x}_j^k$ cannot be inferred by an honest-but-curious agent i unless $\boldsymbol{x}_i^k = \boldsymbol{x}_j^k$ is true.

Proof: Suppose that an honest-but-curious agent i collects information from K iterations to infer the information of a neighboring agent j. From the perspective of adversary agent i, the measurements (corresponding to neighboring agent j) seen in each iteration k are $\boldsymbol{y}^k = b_{i\to j}^k b_{j\to i}^k(\boldsymbol{x}_j^k - \boldsymbol{x}_i^k)$ $(k = 0, 1, \ldots, K)$, i.e., adversary agent i can establish $(K+1)D$ equations based on received information:

$$\begin{cases} \boldsymbol{y}^0 = b_{i\to j}^0 b_{j\to i}^0(\boldsymbol{x}_j^0 - \boldsymbol{x}_i^0), \\ \boldsymbol{y}^1 = b_{i\to j}^1 b_{j\to i}^1(\boldsymbol{x}_j^1 - \boldsymbol{x}_i^1), \\ \quad\vdots \\ \boldsymbol{y}^{K-1} = b_{i\to j}^{K-1} b_{j\to i}^{K-1}(\boldsymbol{x}_j^{K-1} - \boldsymbol{x}_i^{K-1}), \\ \boldsymbol{y}^K = b_{i\to j}^K b_{j\to i}^K(\boldsymbol{x}_j^K - \boldsymbol{x}_i^K). \end{cases} \tag{22}$$

To the adversary agent i, in the system of equations (22), $\boldsymbol{y}^k, b_{i\to j}^k, \boldsymbol{x}_i^k$ $(k = 0, 1, 2, \ldots, K)$ are known, but $\boldsymbol{x}_j^k, b_{j\to i}^k$ $(k = 0, 1, 2, \ldots, K)$ are unknown. So the above system of $(K+1)D$ equations contains $(K+1)D + K + 1$ unknown variables. It is clear that adversary agent i cannot solve the system of equations (22) to infer the exact values of unknowns $\boldsymbol{x}_j^k$ and $b_{j\to i}^k$ $(k = 0, 1, 2, \ldots, K)$ of agent j. It is worth noting that if for some time index k, $\boldsymbol{x}_j^k = \boldsymbol{x}_i^k$ happens to be true, then adversary agent i will be able to know that agent j has the same state at this time index based on the fact that $\boldsymbol{y}^k$ is $\boldsymbol{0}$. ■

Using a similar way of reasoning, we can obtain that an honest-but-curious adversary agent i cannot infer the exact gradient of objective function f_j from a neighboring agent j at any point when agent j has another legitimate neighbor other than the honest-but curious neighbor i.

Theorem 5: In Algorithm 2, the exact gradient of f_j at any point cannot be inferred by an honest-but-curious agent i if agent j has another legitimate neighbor.

Proof: Suppose that an honest-but-curious adversary agent i collects information from K iterations to infer the gradient of function f_j of a neighboring agent j. The adversary agent i can establish KD equations corresponding to the gradient of f_j by making use of the fact that the update rule (21) is publicly known, i.e.,

$$\begin{cases} \nabla f_j(\boldsymbol{x}_j^1) + (1+\gamma_j)\boldsymbol{x}_j^1 + \boldsymbol{\lambda}_j^0 \\ \qquad - \sum_{m\in\mathcal{N}_j} \rho_{j,m}^0(\boldsymbol{x}_m^0 - \boldsymbol{x}_j^0) - (1+\gamma_j)\boldsymbol{x}_j^0 = \boldsymbol{0}, \\ \nabla f_j(\boldsymbol{x}_j^2) + (1+\gamma_j)\boldsymbol{x}_j^2 + \boldsymbol{\lambda}_j^1 \\ \qquad - \sum_{m\in\mathcal{N}_j} \rho_{j,m}^1(\boldsymbol{x}_m^1 - \boldsymbol{x}_j^1) - (1+\gamma_j)\boldsymbol{x}_j^1 = \boldsymbol{0}, \\ \qquad\vdots \\ \nabla f_j(\boldsymbol{x}_j^{K-1}) + (1+\gamma_j)\boldsymbol{x}_j^{K-1} + \boldsymbol{\lambda}_j^{K-2} \\ \quad - \sum_{m\in\mathcal{N}_j} \rho_{j,m}^{K-2}(\boldsymbol{x}_m^{K-2} - \boldsymbol{x}_j^{K-2}) - (1+\gamma_j)\boldsymbol{x}_j^{K-2} = \boldsymbol{0}, \\ \nabla f_j(\boldsymbol{x}_j^K) + (1+\gamma_j)\boldsymbol{x}_j^K + \boldsymbol{\lambda}_j^{K-1} \\ \quad - \sum_{m\in\mathcal{N}_j} \rho_{j,m}^{K-1}(\boldsymbol{x}_m^{K-1} - \boldsymbol{x}_j^{K-1}) - (1+\gamma_j)\boldsymbol{x}_j^{K-1} = \boldsymbol{0}. \end{cases} \tag{23}$$

In the system of KD equations (23), $\nabla f_j(\boldsymbol{x}_j^k)$ $(k = 1, 2, \ldots, K)$, γ_j, and $\boldsymbol{x}_j^k$ $(k = 0, 1, 2, \ldots, K)$ are unknown to adversary agent i. Parameters $\boldsymbol{\lambda}_j^k$ and $\sum_{m\in\mathcal{N}_j} \rho_{j,m}^k(\boldsymbol{x}_m^k - \boldsymbol{x}_j^k)$ $(k = 0, 1, 2, \ldots, K-1)$ are known to adversary agent i only when agent j has agent i as the only neighbor. Otherwise, $\boldsymbol{\lambda}_j^k$ and $\sum_{m\in\mathcal{N}_j} \rho_{j,m}^k(\boldsymbol{x}_m^k - \boldsymbol{x}_j^k)$ $(k = 0, 1, 2, \ldots, K-1)$ are unknown to adversary agent i. Noting that $\boldsymbol{\lambda}_j^{k+1} = \boldsymbol{\lambda}_j^k - \sum_{m\in\mathcal{N}_j} \rho_{j,m}^{k+1}(\boldsymbol{x}_m^{k+1} - \boldsymbol{x}_j^{k+1})$ and $\boldsymbol{\lambda}_j^0 = -\sum_{m\in\mathcal{N}_j} \rho_{j,m}^0(\boldsymbol{x}_m^0 - \boldsymbol{x}_j^0)$, we can see that the above system of KD equations contains $3KD + D + 1$ unknowns when agent j has more than one neighbor. Therefore, adversary agent i cannot infer the exact values of $\nabla f_j(\boldsymbol{x}_j^k)$ by solving (23).

It is worth noting that after the optimization converges, adversary agent i can have another piece of information according to the KKT conditions [44]:

$$\nabla f_j(\boldsymbol{x}_j^*) = -\boldsymbol{\lambda}_j^* \tag{24}$$

where $\boldsymbol{x}_j^*$ denotes the optimal solution and $\boldsymbol{\lambda}_j^*$ denotes the optimal multiplier. However, since $\boldsymbol{\lambda}_j^*$ is known to adversary agent i only when agent j has agent i as the only neighbor, we have that adversary agent i cannot infer the exact value of f_j at any point when agent j has another legitimate neighbor besides an honest-but curious neighbor i. ■

Using a similar way of reasoning, we have the following corollary corresponding to the situation where agent j has honest-but-curious agent i as the only neighbor.

Corollary 1: In Algorithm 2, the exact gradient of f_j at the optimal solution can be inferred by an honest-but-curious agent i if agent j has adversary agent i as the only neighbor. However, at any other point, the gradient of f_j is uninferrable by the adversary agent i.

Proof: Following a similar line of reasoning of Theorem 5, we can obtain the above Corollary. ■

Based on Theorem 4, Theorem 5, and Corollary 1, we can obtain that agent i cannot infer agent j's local objective function f_j.

Corollary 2: In Algorithm 2, agent j's local objective function f_j cannot be inferred by an honest-but-curious agent i.

Proof: According to Theorem 4, Theorem 5, and Corollary 1, the intermediate states and corresponding gradients of f_j cannot be inferred by adversary i. Therefore, adversary i cannot infer agent j's local objective function f_j as well. ■

Furthermore, we have that an external eavesdropper cannot infer any private information of all agents.

Corollary 3: All agents' intermediate states, gradients of objective functions, and objective functions cannot be inferred by an external eavesdropper.

Proof: Since all exchanged messages are encrypted and that cracking the encryption is practically infeasible [56], an external eavesdropper cannot learn anything by intercepting these messages. Therefore, it cannot infer any agent's intermediate states, gradients of objective functions, and objective functions. ■

From the above analysis, it is obvious that agent j's private information cannot be uniquely derived by adversaries. However, an honest-but-curious neighbor i can still get some range information about the state $\boldsymbol{x}_j^k$ and this range information will become tighter as $\boldsymbol{x}_j^k$ converges to the optimal solution as $k \to \infty$ (cf. the simulation results in Fig. 4). We argue that this is completely unavoidable for any privacy-preserving approaches because all agents have to agree on the same final state, upon which the privacy of $\boldsymbol{x}_j^k$ will disappear. In fact, this is also acknowledged in [19], which shows that the privacy of $\boldsymbol{x}_j^k$ will vanish as $k \to \infty$ and the noise variance converges to zero at the state corresponding to the optimal solution. It is worth noting that when the constraint is of a form different from consensus, it may be possible to protect the privacy of $\boldsymbol{x}_j^k$ when $k \to \infty$. However, how to incorporate the proposed privacy mechanism in decentralized optimization under non-consensus constraint is difficult and will be addressed in our future work.

Remark 4: It is worth noting that an adversary agent i can combine systems of equations (22) *and* (23) *to infer the information of a neighboring agent j. However, this will not increase the ability of adversary agent i because the combination will not change the fact that the number of unknowns is greater than the number of establishable relevant equations. In addition, if all other agents collude to infer $\boldsymbol{x}_j^k$ of agent j, these agents can be considered as one agent which amounts to having a network consisting of two agents.*

Remark 5: From Theorem 4, we can see that in decentralized optimization, an agent's information will not be disclosed to other agents no matter how many neighbors it has. This is in distinct difference from the average consensus problem in [17], [34] where privacy cannot be protected for an agent if it has an honest-but-curious adversary as the only neighbor.

V. Implementation Details

In this section, we discuss several technical issues that have to be addressed in the implementation of Algorithm 2.

1) In modern communication, a real number is represented by a floating point number, while encryption techniques only work for unsigned integers. To deal with this problem, we uniformly multiplied each element of the vector message $\boldsymbol{x}_i^t \in \mathbb{R}^D$ (in floating point representation) by a sufficiently large number $N_{\max}$ and round off the fractional part during the encryption to convert it to an integer. After decryption, the result is divided by $N_{\max}$. This process is conducted in each iteration and this quantization brings an error upper-bounded by $\frac{1}{N_{\max}}$. In implementation, $N_{\max}$ can be chosen according to the used data structure.
2) As indicated in 1), encryption techniques only work for unsigned integers. In our implementation all integer values are stored in fix-length integers (i.e., long int in C) and negative values are left in 2's complement format. Encryption and intermediate computations are carried out as if the underlying data were unsigned. When the final message is decrypted, the overflown bits (bits outside the fixed length) are discarded and the remaining binary number is treated as a signed integer which is later converted back to a real value.

VI. Numerical Experiments

In this section, we first illustrate the efficiency of the proposed approach using C/C++ implementations. Then we compare our approach with the algorithm in [19] and the algorithm in [39]. The open-source C implementation of the Paillier cryptosystem [60] is used in our simulations. We conducted numerical experiments on the following global objective function

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \frac{1}{p_i} \parallel H_i \boldsymbol{x} - \boldsymbol{\theta}_i \parallel^2, \tag{25}$$

which makes the optimization problem (1) become

$$\min_{\boldsymbol{x}} \quad \sum_{i=1}^{N} \frac{1}{p_i} \parallel H_i \boldsymbol{x} - \boldsymbol{\theta}_i \parallel^2 \tag{26}$$

with $\boldsymbol{\theta}_i \in \mathbb{R}^D$, $H_i = h_i \boldsymbol{I}_D$ ($h_i \in \mathbb{R}$), and $p_i > 0$ ($p_i \in \mathbb{R}$). Hence, each agent i deals with a private local objective function

$$f_i(\boldsymbol{x}_i) = \frac{1}{p_i} \parallel H_i \boldsymbol{x}_i - \boldsymbol{\theta}_i \parallel^2, \quad \forall i \in \{1, 2, \ldots, N\}. \tag{27}$$

We used the above function (25) because it is easy to verify whether the obtained solution is the minimal value of the original optimization problem, which should be $\frac{\sum_{i=1}^{N} \frac{2h_i}{p_i} \boldsymbol{\theta}_i}{\sum_{i=1}^{N} \frac{2h_i^2}{p_i}}$. Furthermore, (25) makes it easy to compare with [19], whose verification is also based on (25).

In the implementation, the parameters are set as follows: $N_{\max}$ was set to 10^6 to convert each element in $\boldsymbol{x}_i$ to a 64-bit integer during intermediate computations. $b_{i \to j}^t$ was also scaled up in the same way and represented by a 64-bit integer. The encryption and decryption keys were chosen as 256-bit long.

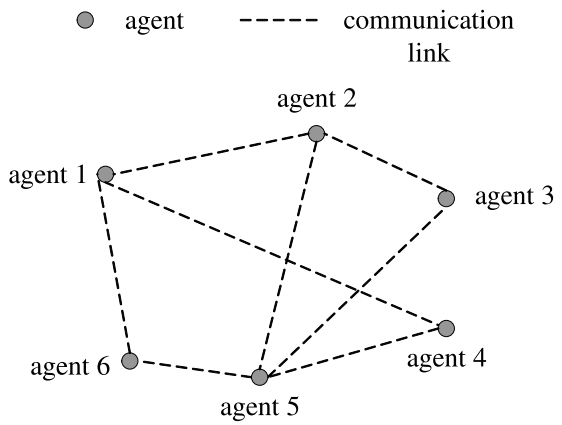

Fig. 1. A network of six agents ($N = 6$).

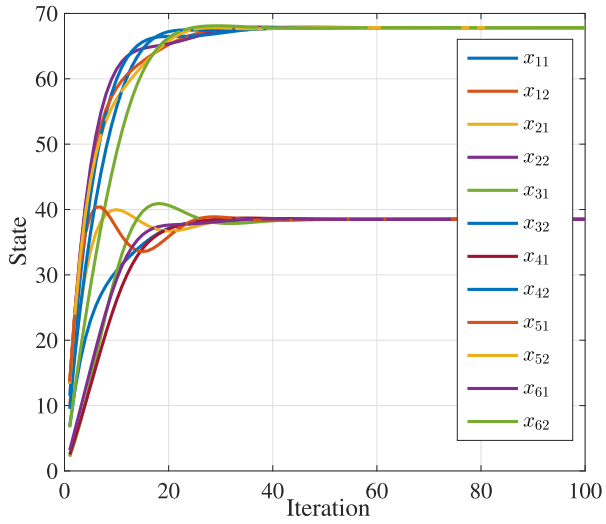

Fig. 2. The evolution of $\boldsymbol{x}_i$ $(i = 1, 2, \ldots, 6)$.

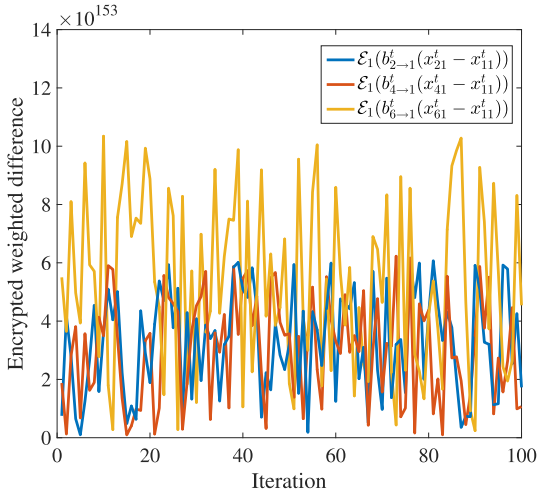

Fig. 3. The evolution of the encrypted weighted differences (in ciphertext) $\mathcal{E}_1(b^t_{2\to1}(\boldsymbol{x}^t_{21} - \boldsymbol{x}^t_{11}))$, $\mathcal{E}_1(b^t_{4\to1}(\boldsymbol{x}^t_{41} - \boldsymbol{x}^t_{11}))$, and $\mathcal{E}_1(b^t_{6\to1}(\boldsymbol{x}^t_{61} - \boldsymbol{x}^t_{11}))$.

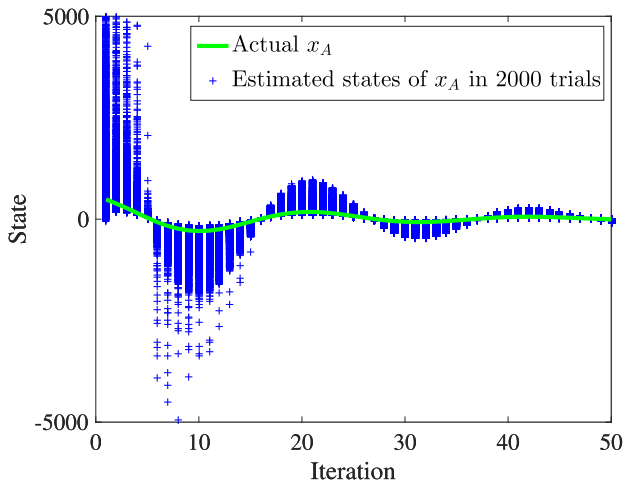

Fig. 4. Estimated states of x_A in 2,000 trials.

A. Evaluation of Our Approach

We implemented Algorithm 2 on different network topologies, all of which gave the right optimal solution. Simulation results confirmed that our approach always converged to the optimal solution of (26). Fig. 2 visualizes the evolution of $\boldsymbol{x}_i$ $(i = 1, 2, \ldots, 6)$ in one specific run where the network deployment is illustrated in Fig. 1. In Fig. 2, x_{ij} $(i = 1, 2, \ldots, 6, j = 1, 2)$ denotes the jth element of $\boldsymbol{x}_i$. All $\boldsymbol{x}_i$ $(i = 1, 2, \ldots, 6)$ converged to the optimal solution $[38.5; \frac{407}{6}]$. In this run, $\bar{b}$ was set to 0.65 and γ_is were set to 3.

Fig. 3 visualizes the encrypted weighted differences (in ciphertext) $\mathcal{E}_1(b^t_{2\to1}(\boldsymbol{x}^t_{21} - \boldsymbol{x}^t_{11}))$, $\mathcal{E}_1(b^t_{4\to1}(\boldsymbol{x}^t_{41} - \boldsymbol{x}^t_{11}))$, and $\mathcal{E}_1(b^t_{6\to1}(\boldsymbol{x}^t_{61} - \boldsymbol{x}^t_{11}))$. It is worth noting that although the states of all agents have converged after about 40 iterations, the encrypted weighted differences (in ciphertext) still appeared random to an outside eavesdropper.

We also simulated an honest-but-curious adversary who tries to estimate its neighbors' intermediate states and gradients in order to estimate the objective function. We considered the worse case of two agents (A and B) where agent B is the honest-but-curious adversary and intends to estimate the objective function f_A of agent A. The individual local objective functions are the same as (27) with $\theta_i \in \mathbb{R}$. Because agent B knows the constraints on agent A's generation of $b^t_{A\to B}$ and γ_A (cf. Theorem 3), it generates estimates of $b^t_{A\to B}$ and γ_A in the same random way. Then it obtained a series of estimated x^t_A and $\nabla f_A(x^t_A)$ according to (23). Finally, agent B used the estimated x^t_A and $\nabla f_A(x^t_A)$ to estimate f_A.

Fig. 4 and Fig. 5 show the estimated x_A and f_A in 2,000 trials when agent B used simple linear regression to estimate $\nabla f_A(x)$. Fig. 5 suggests that agent B cannot get a good estimate of f_A. Moreover, it is worth noting that all these estimated functions give the same optimal solution as f_A to the optimization problem (26).

In addition, the encryption/decryption computation took about 1ms for each agent to communicate with one neighbor at each iteration on a 3.6 GHz CPU, which is manageable in small or medium sized real-time optimization problems such as the source localization problem [7] and power system monitoring problem [57] addressed in our prior work. For large sized optimization problems like machine learning with

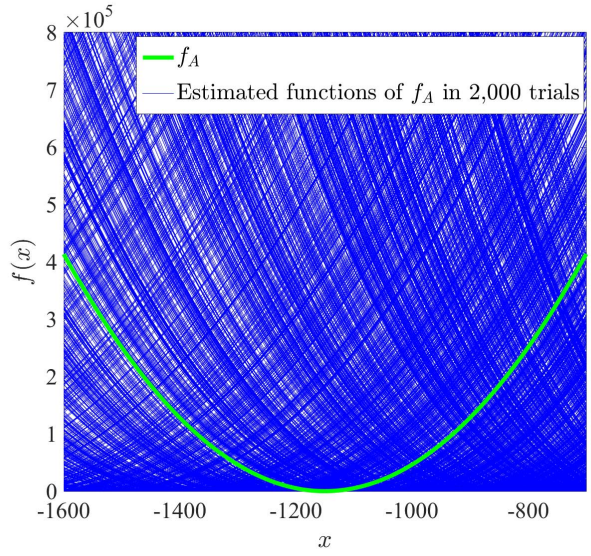

Fig. 5. Estimated functions of f_A in 2,000 trials.

extremely large dimensions, the approach may be computationally too heavy due to the underlying Paillier encryption scheme.

B. Comparison With the Algorithm in [19]

We then compared our approach with the differential-privacy based privacy-preserving optimization algorithm in [19]. Under the communication topology in Fig. 1, we simulated the algorithm in [19] under seven different privacy levels: $\epsilon = 0.2, 1, 10, 20, 30, 50, 100$. The global function we used for comparison was (25) with p_i $(i = 1, 2, .., 6)$ fixed to 2, h_i $(i = 1, 2, .., 6)$ fixed to 1, and $\boldsymbol{\theta}_i = [0.1 \times (i-1) + 0.1; 0.1 \times (i-1) + 0.2]$. The domain of optimization was set to $\mathcal{X} = \{(x, y) \in \mathbb{R}^2 | x^2 + y^2 \leq 1\}$ for the algorithm in [19]. Note that the optimal solution [0.35; 0.45] resided in $\mathcal{X}$. Parameter settings for the algorithm in [19] are detailed as follows: $n = 2$, $c = 0.5$, $q = 0.8$, $p = 0.9$, and

$$a_{ij} = \begin{cases} 0.2 & j \in \mathcal{N}_i \backslash i, \\ 0 & j \notin \mathcal{N}_i, \\ 1 - \sum_{j \in \mathcal{N}_i \backslash i} a_{ij} & i = j, \end{cases} \tag{28}$$

for $i = 1, 2, \ldots, 6$. Here $\mathcal{N}_i \backslash i$ denotes all values except i in set $\mathcal{N}_i$. Furthermore, we used the performance index d in [19] to quantify the optimization error, which was computed as the average value of squared distances with respect to the optimal solution over M runs [19], i.e.,

$$d = \frac{\sum_{i=1}^{6} \sum_{k=1}^{M} \| \boldsymbol{x}_i^k - [0.35; 0.45] \|^2}{6M}$$

with $\boldsymbol{x}_i^k$ the obtained solution of agent i in the kth run.

Simulation results from 5,000 runs showed that our approach converged to [0.35; 0.45] with an error $d = 3.14 \times 10^{-14}$, which is negligible compared with the simulation results under the algorithm in [19] (cf. Fig. 6, where each differential privacy level was implemented

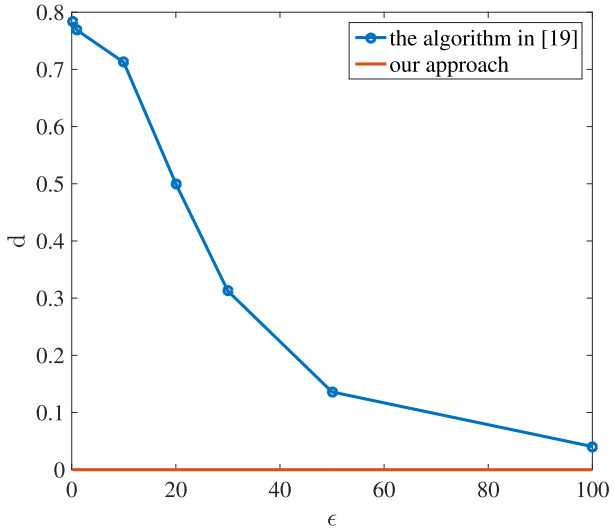

Fig. 6. The comparison of our approach with the algorithm in [19] in terms of optimization error.

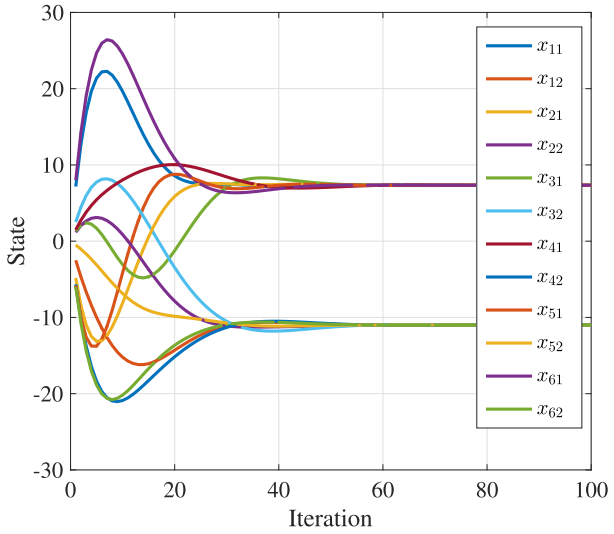

Fig. 7. The evolution of $\boldsymbol{x}_i$ in our approach.

for 5,000 times). The results confirm the trade-off between privacy and accuracy for differential-privacy based approaches and demonstrate the advantages of our approach in terms of optimization accuracy.

C. Comparison With the Algorithm in [39]

We also compared our approach with the privacy-preserving optimization algorithm in [39]. The network communication topology used for comparison is still the one in Fig. 1 and the global objective function used is (25) with p_i $(i = 1, 2, .., 6)$ fixed to 2, h_i $(i = 1, 2, .., 6)$ fixed to 1, and $\theta_i \in \mathbb{R}^2$. The adjacency matrix of network graph is defined in (28) for the algorithm in [39]. Moreover, we let every agent update at each iteration and $c_i = 1$ $(i = 1, \ldots, 6)$ for [39]. The initial states are set to the same values for both algorithms.

Fig. 7 and Fig. 8 show the evolution of $\boldsymbol{x}_i$ in our approach and the algorithm in [39] respectively. It is clear that our approach converged faster than the algorithm in [39].

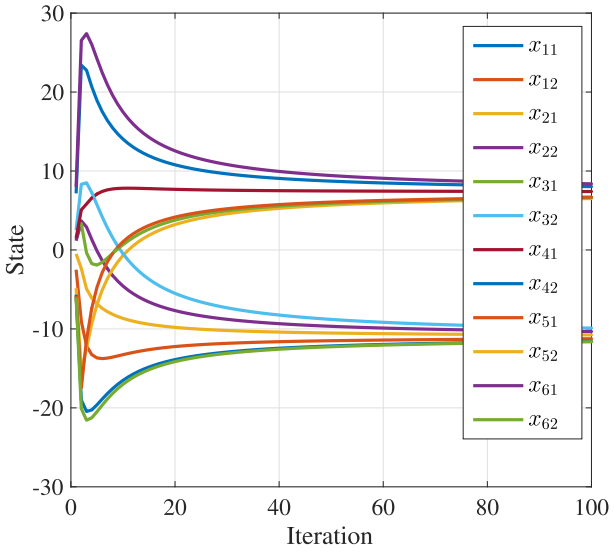

Fig. 8. The evolution of $\boldsymbol{x}_i$ in the algorithm of [39].

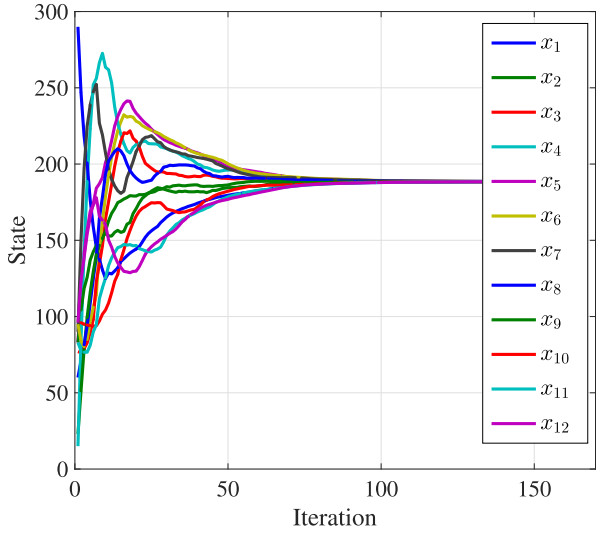

Fig. 10. The evolution of x_i in the experimental verification using Raspberry Pi boards.

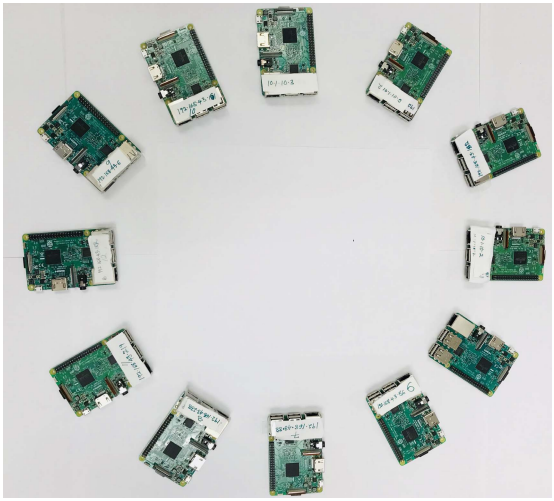

Fig. 9. The twelve Raspberry Pi boards.

VII. Implementation on Raspberry Pi Boards

We also implemented our privacy-preserving approach on twelve Raspberry Pi boards to confirm the efficiency of the approach in real-world physical systems. Each board has 64-bit ARMv8 CPU and 1 GB RAM (cf. Fig. 9). The optimization problem (26) was used in implementation with p_i $(i = 1, 2, .., 6)$ fixed to 2, h_i $(i = 1, 2, .., 6)$ fixed to 1, and $\theta_i \in \mathbb{R}$. In the implementation, "libpaillier-0.8" library [61] was used to realize the Paillier encryption and decryption process, "sys/socket.h" C library was used to conduct communication through Wi-Fi, and "pthread" C library was used to generate multiple parallel threads to realize parallelism in multi-agent networks. The encryption and decryption keys were chosen as 512-bit long.

Implementation results confirmed that our approach always converged to the optimal solution. Fig. 10 visualizes the evolution of x_i $(i = 1, 2, \ldots, 12)$ in one specific implementation where the network topology used is a cycle graph. We can see that each x_i converged to the optimal solution 188.417.

VIII. Conclusions

In this paper, we presented a privacy-preserving decentralized optimization approach by proposing a new ADMM and leveraging partially homomorphic cryptography. By incorporating Paillier cryptosystem into the newly proposed decentralized ADMM, our approach provides guarantee for privacy preservation without compromising the solution in the absence of any aggregator or third party. This is in sharp contrast to differential-privacy based approaches which protect privacy through injecting noise and are subject to a fundamental trade-off between privacy and accuracy. Theoretical analysis confirms that an honest-but-curious adversary cannot infer the information of neighboring agents even by recording and analyzing the information exchanged in multiple iterations. The new ADMM allows time-varying penalty matrices and have a theoretically guaranteed convergence rate of $O(1/t)$, which makes it of mathematical interest by itself. Numerical and experimental results are given to confirm the effectiveness and efficiency of the proposed approach.

Appendix

A. Proof of Theorem 1

The key idea to prove Theorem 1 is to show that Algorithm 1 converges to the saddle point of the Lagrangian function $L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T A\boldsymbol{x}$. To achieve this goal, we introduce a variational inequality $MVI(Q, U)$ first and prove that the solution of $MVI(Q, U)$ is also the saddle point of the Lagrangian function $L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T A\boldsymbol{x}$ (which is formulated as Lemma 1). Then we introduce a sufficient condition for solving $MVI(Q, U)$ in Lemma 2. After the two steps, what is left is to prove that the iterates of Algorithm 1 satisfy the condition in Lemma 2 when $k \to \infty$,

i.e., Algorithm 1 converges to the solution of $MVI(Q,U)$ (Theorem 6 and Theorem 7).

We form a variational inequality $MVI(Q,U)$ similar to [62, eqs. (5) and (6)] first:

$$\langle \boldsymbol{u}-\boldsymbol{u}^*,\, \boldsymbol{Q}(\boldsymbol{u}^*)\rangle \geq \mathbf{0}, \quad \forall \boldsymbol{u}, \tag{29}$$

where

$$\boldsymbol{u}^* := \begin{pmatrix} \boldsymbol{x}_1^* \\ \boldsymbol{x}_2^* \\ \vdots \\ \boldsymbol{x}_N^* \\ \boldsymbol{\lambda}^* \end{pmatrix}, \quad \boldsymbol{Q}(\boldsymbol{u}^*) := \begin{pmatrix} \xi_1^* + [A]_1^T\boldsymbol{\lambda}^* \\ \xi_2^* + [A]_2^T\boldsymbol{\lambda}^* \\ \vdots \\ \xi_N^* + [A]_N^T\boldsymbol{\lambda}^* \\ A\boldsymbol{x}^* \end{pmatrix},$$
$$\xi_i^* \in \partial f_i(\boldsymbol{x}_i^*), \quad \forall i \in \{1,2,\ldots,N\}. \tag{30}$$

In (30), $[A]_i$ denotes the columns of matrix A that are associated with agent i. By recalling the first-order necessary and sufficient condition for convex programming [62], it is easy to see that solving problem (6) amounts to solving the above $MVI(Q,U)$ [62]. Denote the solution set of $MVI(Q,U)$ as $\mathcal{U}^*$. Since f_i is convex, $\partial f_i(\boldsymbol{x}_i)$ is monotone, the $MVI(Q,U)$ is solvable and $\mathcal{U}^*$ is nonempty [62].

Next, we introduce several lemmas and theorems that contribute to the proof of Theorem 1.

Lemma 1: Each $\boldsymbol{u}^* = (\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$ in $\mathcal{U}^*$ is also the saddle point of the Lagrangian function $L(\boldsymbol{x},\boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T A\boldsymbol{x}$.

Proof: The results can be obtained from [10, Part 2.1] directly. ■

Lemma 2: If $A\boldsymbol{x}^{k+1} = \mathbf{0}$ and $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k$ hold, then $(\boldsymbol{x}_1^{k+1}, \boldsymbol{x}_2^{k+1}, \ldots, \boldsymbol{x}_N^{k+1}, \boldsymbol{\lambda}^{k+1})$ is a solution to $MVI(Q,U)$.

Proof: Using the definition of matrix A and the update rule of $\boldsymbol{\lambda}^{k+1}$ in (9), we can see that the assumption $A\boldsymbol{x}^{k+1} = \mathbf{0}$ implies $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k$ and $\boldsymbol{x}_1^{k+1} = \boldsymbol{x}_2^{k+1} = \ldots = \boldsymbol{x}_N^{k+1}$.

On the other hand, we know that $\boldsymbol{x}_i^{k+1}$ is the optimizer of (13). By using the first-order optimality condition, we get

$$(\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T(\xi_i^{k+1} + \sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_{i,j}^k + \rho_{i,j}^k(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_j^k)) + \gamma_i(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k)) \geq 0. \tag{31}$$

where $\xi_i^{k+1} \in \partial f_i(\boldsymbol{x}_i^{k+1})$. Then based on the assumption $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k$, the fact $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k$, and the definition of matrix A, we have $(\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T(\xi_i^{k+1} + [A]_i^T\boldsymbol{\lambda}^{k+1}) \geq 0$. Therefore, $(\boldsymbol{x}_1^{k+1}, \boldsymbol{x}_2^{k+1}, \ldots, \boldsymbol{x}_N^{k+1}, \boldsymbol{\lambda}^{k+1})$ is a solution to $MVI(Q,U)$. ■

Lemma 2 provides a sufficient condition for solving $MVI(Q,U)$. According to Lemma 1, we know that the solution to $MVI(Q,U)$ is also the saddle point of the Lagrangian function. Next, we prove that the iterates in Algorithm 1 satisfy $\lim_{k\to\infty} A\boldsymbol{x}^{k+1} = \mathbf{0}$ and $\lim_{k\to\infty} \boldsymbol{x}^{k+1} - \boldsymbol{x}^k = \mathbf{0}$, i.e., Algorithm 1 converges to the solution to $MVI(Q,U)$. To achieve this goal, we first establish the relationship (32) about iterates k and $k+1$ in Theorem 6, whose proof is mainly based on convex properties. Then based on the relationship, we further prove $\lim_{k\to\infty} A\boldsymbol{x}^{k+1} = \mathbf{0}$ and $\lim_{k\to\infty} \boldsymbol{x}^{k+1} - \boldsymbol{x}^k = \mathbf{0}$ in Theorem 7.

Theorem 6: Let $\boldsymbol{\rho}^k$ satisfy Condition A, $\bar{Q} \triangleq Q_P + Q_C^k$ satisfy Condition B, and $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$ be the saddle point of the Lagrangian function $L(\boldsymbol{x},\boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T A\boldsymbol{x}$, then we have

$$\begin{aligned} &\| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}} + \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \|^2_{\bar{Q}} \\ &\leq \| \boldsymbol{\lambda}^{k} - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k})^{-1}} + \| \boldsymbol{x}^{k} - \boldsymbol{x}^* \|^2_{\bar{Q}} \\ &\quad -(\| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^k} + \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \|^2_{-A^T\boldsymbol{\rho}^k A + \bar{Q}}) \\ &\quad + \| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^{k+1}} - \| A\boldsymbol{x}^{k} \|^2_{\boldsymbol{\rho}^{k}}. \end{aligned} \tag{32}$$

To prove Theorem 6, we first introduce two lemmas:

Lemma 3: Let $\boldsymbol{x}^k = [\boldsymbol{x}_1^{kT}, \boldsymbol{x}_2^{kT}, \ldots, \boldsymbol{x}_N^{kT}]^T$ and $\boldsymbol{\lambda}^k = [\boldsymbol{\lambda}_{i,j}^k]_{ij,e_{i,j}\in E}$ be the intermediate results of iteration k in Algorithm 1, then the following inequality holds for all k:

$$\begin{aligned} &f(\boldsymbol{x}) - f(\boldsymbol{x}^{k+1}) + (\boldsymbol{x} - \boldsymbol{x}^{k+1})^T A^T\boldsymbol{\lambda}^k + (\boldsymbol{x} - \boldsymbol{x}^{k+1})^T \\ &\cdot A^T\boldsymbol{\rho}^k A\boldsymbol{x}^k + (\boldsymbol{x} - \boldsymbol{x}^{k+1})^T\bar{Q}(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) \geq 0, \end{aligned} \tag{33}$$

where $\bar{Q} \triangleq Q_P + Q_C^k$.

Proof: The proof follows from [7]. For completeness, we sketch the proof here. Denote by g_i the function

$$g_i^k(\boldsymbol{x}_i) = \sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_{i,j}^{kT}\boldsymbol{x}_i + \frac{\rho_{i,j}^k}{2}\| \boldsymbol{x}_i - \boldsymbol{x}_j^k \|^2) + \frac{\gamma_i}{2}\| \boldsymbol{x}_i - \boldsymbol{x}_i^k \|^2. \tag{34}$$

Using $\xi_i^{k+1} \in \partial f_i(\boldsymbol{x}_i^{k+1})$, we can get $\xi_i^{k+1} + \triangledown g_i(\boldsymbol{x}_i^{k+1}) = \mathbf{0}$ and $(\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T[\xi_i^{k+1} + \triangledown g_i(\boldsymbol{x}_i^{k+1})] = 0$ based on the fact that $\boldsymbol{x}_i^{k+1}$ is the optimizer of $g_i^k + f_i$. On the other hand, as f_i is convex, the following relationship holds:

$$f_i(\boldsymbol{x}_i) \geq f_i(\boldsymbol{x}_i^{k+1}) + (\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T\xi_i^{k+1}.$$

Then we can get $f_i(\boldsymbol{x}_i) - f_i(\boldsymbol{x}_i^{k+1}) + (\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T\triangledown g_i(\boldsymbol{x}_i^{k+1}) \geq 0$.

Substituting $\triangledown g_i(\boldsymbol{x}_i^{k+1})$ with (34), we obtain

$$\begin{aligned} &f_i(\boldsymbol{x}_i) - f_i(\boldsymbol{x}_i^{k+1}) + (\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T \\ &\cdot(\sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_{i,j}^k + \rho_{i,j}^k(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_j^k)) + \gamma_i(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k)) \geq 0. \end{aligned}$$

Noting $\boldsymbol{\lambda}_{i,i} = \mathbf{0}$ and $\boldsymbol{\lambda}_{i,j} = -\boldsymbol{\lambda}_{j,i}$, based on the definition of matrices A and $\boldsymbol{\rho}$, we can rewrite the above inequality as

$$\begin{aligned} &f_i(\boldsymbol{x}_i) - f_i(\boldsymbol{x}_i^{k+1}) + (\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T\cdot \\ &([A]_i^T\boldsymbol{\lambda}^k + \sum_{j\in\mathcal{N}_i}\rho_{i,j}^k(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_j^k) + \gamma_i(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k)) \geq 0. \end{aligned} \tag{35}$$

Summing both sides of (35) over $i = 1, 2, \ldots, N$, and using

$$\sum_{i=1}^N(\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T[A]_i^T\boldsymbol{\lambda}^k = (\boldsymbol{x} - \boldsymbol{x}^{k+1})^T A^T\boldsymbol{\lambda}^k,$$

$$\sum_{i=1}^N(\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T\sum_{j\in\mathcal{N}_i}\rho_{i,j}^k\boldsymbol{x}_i^{k+1} = (\boldsymbol{x} - \boldsymbol{x}^{k+1})^T Q_C^k\boldsymbol{x}^{k+1},$$

$$\begin{aligned} &\sum_{i=1}^N(\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T\sum_{j\in\mathcal{N}_i}\rho_{i,j}^k\boldsymbol{x}_j^k \\ &\quad = (\boldsymbol{x} - \boldsymbol{x}^{k+1})^T(-A^T\boldsymbol{\rho}^k A + Q_C^k)\boldsymbol{x}^k, \end{aligned}$$

$$\begin{aligned} &\sum_{i=1}^N(\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T\gamma_i(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k) \\ &\quad = (\boldsymbol{x} - \boldsymbol{x}^{k+1})^T Q_P(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k), \end{aligned}$$

we can get the lemma. ■

Lemma 4: Let $\boldsymbol{x}^k = [\boldsymbol{x}_1^{kT}, \boldsymbol{x}_2^{kT}, \ldots, \boldsymbol{x}_N^{kT}]^T$ *and* $\boldsymbol{\lambda}^k = [\boldsymbol{\lambda}_{i,j}^k]_{ij,e_{i,j}\in E}$ *be the intermediate results of iteration* k *in Algorithm 1, then the following equality holds for all* k*:*

$$\begin{aligned}
&-(\boldsymbol{x}^{k+1})^T A^T(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^*)\\
&-(\boldsymbol{x}^{k+1})^T A^T \boldsymbol{\rho}^k A\boldsymbol{x}^k + (\boldsymbol{x}^* - \boldsymbol{x}^{k+1})^T \bar{Q}(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k)\\
&\quad = -\frac{1}{2}(\| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}} - \| \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}})\\
&\qquad + \frac{1}{2} \| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}} + \frac{1}{2} \| A(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) \|^2_{\boldsymbol{\rho}^k}\\
&\qquad - \frac{1}{2} \| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^k} - \frac{1}{2} \| A\boldsymbol{x}^k \|^2_{\boldsymbol{\rho}^k} - \frac{1}{2} \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \|^2_{\bar{Q}}\\
&\qquad - \frac{1}{2}(\| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \|^2_{\bar{Q}} - \| \boldsymbol{x}^k - \boldsymbol{x}^* \|^2_{\bar{Q}}). \qquad (36)
\end{aligned}$$

Proof: For a scalar a, we have $a^T = a$. Recall $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \boldsymbol{\rho}^{k+1} A\boldsymbol{x}^{k+1}$ and notice that $\boldsymbol{\rho}^{k+1}$ is a positive definite diagonal matrix, we can get

$$(\boldsymbol{x}^{k+1})^T A^T(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^*) = (\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k)^T (\boldsymbol{\rho}^{k+1})^{-1}(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^*). \quad (37)$$

On the other hand, since $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$ is the saddle point of the Lagrangian function (15), we can get $A\boldsymbol{x}^* = \mathbf{0}$ [48]. Moreover, the following equalities can be established by using algebraic manipulations:

$$\begin{aligned}
(\boldsymbol{x}^{k+1} - \boldsymbol{x}^*)^T \bar{Q}(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) &= \frac{1}{2} \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \|^2_{\bar{Q}}\\
&\quad + \frac{1}{2}(\| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \|^2_{\bar{Q}} - \| \boldsymbol{x}^k - \boldsymbol{x}^* \|^2_{\bar{Q}}), \qquad (38)\\
-\boldsymbol{x}^{(k+1)T} A^T \boldsymbol{\rho}^k A\boldsymbol{x}^k &= \frac{1}{2} \| A(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) \|^2_{\boldsymbol{\rho}^k}\\
&\quad - \frac{1}{2} \| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^k} - \frac{1}{2} \| A\boldsymbol{x}^k \|^2_{\boldsymbol{\rho}^k}, \qquad (39)\\
(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k)^T (\boldsymbol{\rho}^{k+1})^{-1}(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^*)&\\
= \frac{1}{2}(\| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}} - \| \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}}),&\\
-\frac{1}{2} \| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}}. \qquad (40)&
\end{aligned}$$

Then we can obtain (36) by plugging equalities (37)-(40) into the left hand side of (36). ■

Now we can proceed to prove Theorem 6. By setting $\boldsymbol{x} = \boldsymbol{x}^*$ in (33), we can get

$$\begin{aligned}
f(\boldsymbol{x}^*) - f(\boldsymbol{x}^{k+1}) + (\boldsymbol{x}^* - \boldsymbol{x}^{k+1})^T A^T \boldsymbol{\lambda}^k + (\boldsymbol{x}^* - \boldsymbol{x}^{k+1})^T&\\
\cdot A^T \boldsymbol{\rho}^k A\boldsymbol{x}^k + (\boldsymbol{x}^* - \boldsymbol{x}^{k+1})^T \bar{Q}(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) \geq 0.&
\end{aligned}$$

Recalling $A\boldsymbol{x}^* = \mathbf{0}$, the above inequality can be rewritten as

$$\begin{aligned}
f(\boldsymbol{x}^*) - f(\boldsymbol{x}^{k+1}) - \boldsymbol{x}^{(k+1)T} A^T \boldsymbol{\lambda}^k - \boldsymbol{x}^{(k+1)T} A^T \boldsymbol{\rho}^k A\boldsymbol{x}^k&\\
+ (\boldsymbol{x}^* - \boldsymbol{x}^{k+1})^T \bar{Q}(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) \geq 0. \qquad (41)&
\end{aligned}$$

Now adding and subtracting the term $\boldsymbol{\lambda}^{*T} A\boldsymbol{x}^{k+1}$ from the left hand side of (41) gives

$$\begin{aligned}
f(\boldsymbol{x}^*) - f(\boldsymbol{x}^{k+1}) - \boldsymbol{\lambda}^{*T} A\boldsymbol{x}^{k+1} - \boldsymbol{x}^{(k+1)T} A^T(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^*)&\\
- \boldsymbol{x}^{(k+1)T} A^T \boldsymbol{\rho}^k A\boldsymbol{x}^k + (\boldsymbol{x}^* - \boldsymbol{x}^{k+1})^T \bar{Q}(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) \geq 0.&\\
(42)&
\end{aligned}$$

Using $L(\boldsymbol{x}, \boldsymbol{\lambda}^*) - L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) \geq 0$ and $A\boldsymbol{x}^* = \mathbf{0}$, we have

$$\begin{aligned}
&-\boldsymbol{x}^{(k+1)T} A^T(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^*)\\
&\quad - \boldsymbol{x}^{(k+1)T} A^T \boldsymbol{\rho}^k A\boldsymbol{x}^k + (\boldsymbol{x}^* - \boldsymbol{x}^{k+1})^T \bar{Q}(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k)\\
&\geq f(\boldsymbol{x}^{k+1}) + \boldsymbol{\lambda}^{*T} A\boldsymbol{x}^{k+1} - f(\boldsymbol{x}^*) \geq 0.
\end{aligned}$$

Now by plugging (36) into the left hand side of the above inequality, we can obtain

$$\begin{aligned}
&-\frac{1}{2}(\| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}} - \| \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}})\\
&+\frac{1}{2} \| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}} + \frac{1}{2} \| A\boldsymbol{x}^{k+1} - A\boldsymbol{x}^k \|^2_{\boldsymbol{\rho}^k}\\
&-\frac{1}{2} \| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^k} - \frac{1}{2} \| A\boldsymbol{x}^k \|^2_{\boldsymbol{\rho}^k} - \frac{1}{2} \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \|^2_{\bar{Q}}\\
&+\frac{1}{2} \| \boldsymbol{x}^k - \boldsymbol{x}^* \|^2_{\bar{Q}} - \frac{1}{2} \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \|^2_{\bar{Q}} \geq 0.
\end{aligned}$$

Noting $\| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}} = \| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^{k+1}}$, the above inequality can be rewritten as

$$\begin{aligned}
&\| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}} + \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \|^2_{\bar{Q}}\\
&\quad \leq \| \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}} + \| \boldsymbol{x}^k - \boldsymbol{x}^* \|^2_{\bar{Q}}\\
&\qquad - (\| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^k} + \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \|^2_{-A^T \boldsymbol{\rho}^k A + \bar{Q}})\\
&\qquad + \| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^{k+1}} - \| A\boldsymbol{x}^k \|^2_{\boldsymbol{\rho}^k}. \qquad (43)
\end{aligned}$$

Recall that from Condition A, $\boldsymbol{\rho}^{k+1} \geq \boldsymbol{\rho}^k$ and $\boldsymbol{\rho}^k$ $(k = 1, 2, ...)$ are positive definite diagonal matrices. So we have $(\boldsymbol{\rho}^{k+1})^{-1} \preceq (\boldsymbol{\rho}^k)^{-1}$ [45], and consequently $\| \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}} \preceq \| \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^k)^{-1}}$, which proves Theorem 6. ■

Theorem 6 established the relationship between iterates k and $k+1$ in Algorithm 1. Based on this relationship, we can have the following theorem which shows that Algorithm 1 converges to the solution to $MVI(Q, U)$.

Theorem 7: Let $\boldsymbol{u}^k = (\boldsymbol{x}^k, \boldsymbol{\lambda}^k)$ *be the sequence generated by Algorithm 1, then we have*

$$\lim_{k\to\infty} (\| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^k} + \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \|^2_{-A^T \boldsymbol{\rho}^k A + \bar{Q}}) = 0. \quad (44)$$

Proof: Let $\alpha^k = \| \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^k)^{-1}} + \| \boldsymbol{x}^k - \boldsymbol{x}^* \|^2_{\bar{Q}}$. According to Theorem 6, we have

$$\begin{aligned}
\alpha^{k+1} &\leq \alpha^k + \| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^{k+1}} - \| A\boldsymbol{x}^k \|^2_{\boldsymbol{\rho}^k}\\
&\quad - (\| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^k} + \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \|^2_{-A^T \boldsymbol{\rho}^k A + \bar{Q}})\\
&\leq \ldots\\
&\leq \alpha^0 + \| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^{k+1}} - \| A\boldsymbol{x}^0 \|^2_{\boldsymbol{\rho}^0}\\
&\quad - \sum_{i=0}^{k} (\| A\boldsymbol{x}^{i+1} \|^2_{\boldsymbol{\rho}^i} + \| \boldsymbol{x}^{i+1} - \boldsymbol{x}^i \|^2_{-A^T \boldsymbol{\rho}^i A + \bar{Q}})\\
&\leq \alpha^0 + \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \|^2_{A^T \boldsymbol{\rho}^{k+1} A}\\
&\quad - \sum_{i=0}^{k} (\| A\boldsymbol{x}^{i+1} \|^2_{\boldsymbol{\rho}^i} + \| \boldsymbol{x}^{i+1} - \boldsymbol{x}^i \|^2_{-A^T \boldsymbol{\rho}^i A + \bar{Q}}).
\end{aligned} \quad (45)$$

The last inequality comes from the fact that $A\boldsymbol{x}^* = \mathbf{0}$ and $\| A\boldsymbol{x}^{k+1} - A\boldsymbol{x}^* \|^2_{\boldsymbol{\rho}^{k+1}}$ can be written as $\| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \|^2_{A^T \boldsymbol{\rho}^{k+1} A}$. Recall that $\boldsymbol{\rho}^0 \preceq \boldsymbol{\rho}^k \preceq \boldsymbol{\rho}^{k+1} \preceq \bar{\boldsymbol{\rho}}$ holds and $\bar{Q} - A^T \bar{\boldsymbol{\rho}} A$ is

positive definite. Moving the term $\| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \|^2_{A^T \boldsymbol{\rho}^{k+1} A}$ to the left hand side of the above inequality, we have

$$\begin{aligned}&\lim_{k\to\infty} (\alpha^{k+1} - \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \|^2_{A^T \boldsymbol{\rho}^{k+1} A}) \\ &= \lim_{k\to\infty} (\| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}} + \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \|^2_{\bar{Q} - A^T \boldsymbol{\rho}^{k+1} A}) \\ &\geq 0 \end{aligned} \tag{46}$$

Since α^0 is positive and bounded and $\| A\boldsymbol{x}^{i+1} \|^2_{\boldsymbol{\rho}^i} + \| \boldsymbol{x}^{i+1} - \boldsymbol{x}^i \|^2_{-A^T \boldsymbol{\rho}^i A + \bar{Q}}$ is nonnegative, following [46, Th. 3], we have

$$\lim_{k\to\infty} (\| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^k} + \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \|^2_{-A^T \boldsymbol{\rho}^k A + \bar{Q}}) = 0. \tag{47}$$

■

Given that $\boldsymbol{\rho}^k$ satisfies Condition A and $\bar{Q}$ satisfies Condition B, we have that both $-A^T \boldsymbol{\rho}^k A + \bar{Q}$ and $\boldsymbol{\rho}^k$ are positive symmetric definite. Then according to Theorem 7, we have $A\boldsymbol{x}^{k+1} = \mathbf{0}$ and $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k$ when $k \to \infty$.

Therefore, based on Lemma 2, we have that $(\boldsymbol{x}^{k+1}, \boldsymbol{\lambda}^{k+1})$ in Algorithm 1 converges to a solution to $MVI(Q, U)$, i.e., a saddle point of the Lagrangian function (15) according to Lemma 1. Since the objective function is convex, we can conclude Theorem 1 [48]. ■

B. Proof of Theorem 2

Now we prove that the convergence rate of Algorithm 1 is $O(1/t)$. By plugging (36) into the left hand side of (42), we can obtain

$$\begin{aligned}&f(\boldsymbol{x}^*) - f(\boldsymbol{x}^{k+1}) - \boldsymbol{\lambda}^{*T} A\boldsymbol{x}^{k+1} \\ &\quad -\frac{1}{2}(\| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}} - \| \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}}) \\ &\quad +\frac{1}{2} \| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}} + \frac{1}{2} \| A\boldsymbol{x}^{k+1} - A\boldsymbol{x}^k \|^2_{\boldsymbol{\rho}^k} \\ &\quad -\frac{1}{2} \| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^k} - \frac{1}{2} \| A\boldsymbol{x}^k \|^2_{\boldsymbol{\rho}^k} - \frac{1}{2} \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \|^2_{\bar{Q}} \\ &\quad +\frac{1}{2} \| \boldsymbol{x}^k - \boldsymbol{x}^* \|^2_{\bar{Q}} - \frac{1}{2} \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \|^2_{\bar{Q}} \geq 0. \end{aligned} \tag{48}$$

Summing both sides of the above inequality over $k = 0, 1, \ldots, t$, we have

$$\begin{aligned}&(t+1) f(\boldsymbol{x}^*) - \sum_{k=0}^{t} f(\boldsymbol{x}^{k+1}) - \boldsymbol{\lambda}^{*T} A \sum_{k=0}^{t} \boldsymbol{x}^{k+1} \\ &\quad -\frac{1}{2} \| \boldsymbol{\lambda}^{t+1} - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{t+1})^{-1}} + \frac{1}{2} \| \boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^1)^{-1}} \\ &\quad -\sum_{k=1}^{t} \frac{1}{2}(\| \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^k)^{-1}} - \| \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}}) \\ &\quad +\frac{1}{2} \| A\boldsymbol{x}^{t+1} \|^2_{\boldsymbol{\rho}^{t+1}} - \sum_{k=0}^{t} \frac{1}{2} \| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^k} - \frac{1}{2} \| A\boldsymbol{x}^0 \|^2_{\boldsymbol{\rho}^0} \\ &\quad -\frac{1}{2} \| \boldsymbol{x}^{t+1} - \boldsymbol{x}^* \|^2_{\bar{Q}} + \frac{1}{2} \| \boldsymbol{x}^0 - \boldsymbol{x}^* \|^2_{\bar{Q}} \\ &\quad -\sum_{k=0}^{t} \frac{1}{2} \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \|^2_{\bar{Q} - A^T \boldsymbol{\rho}^k A} \geq 0. \end{aligned}$$

Following the above inequality, It is easy to obtain

$$\begin{aligned}&(t+1) f(\boldsymbol{x}^*) - \sum_{k=0}^{t} f(\boldsymbol{x}^{k+1}) - \boldsymbol{\lambda}^{*T} A \sum_{k=0}^{t} \boldsymbol{x}^{k+1} \\ &\quad +\frac{1}{2} \| \boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^1)^{-1}} + \frac{1}{2} \| \boldsymbol{x}^0 - \boldsymbol{x}^* \|^2_{\bar{Q}} \\ &\quad +\frac{1}{2} \| A\boldsymbol{x}^{t+1} \|^2_{\boldsymbol{\rho}^{t+1}} - \frac{1}{2} \| A\boldsymbol{x}^{t+1} \|^2_{\boldsymbol{\rho}^t} \geq 0. \end{aligned}$$

Recall that in (47), we have proven $\lim_{k\to\infty} \| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^k} = 0$. Then the relationship $\boldsymbol{\rho}^0 \preceq \boldsymbol{\rho}^k \preceq \boldsymbol{\rho}^{k+1} \preceq \bar{\boldsymbol{\rho}}$ implies

$$\lim_{t\to\infty} (\frac{1}{2} \| A\boldsymbol{x}^{t+1} \|^2_{\boldsymbol{\rho}^{t+1}} - \frac{1}{2} \| A\boldsymbol{x}^{t+1} \|^2_{\boldsymbol{\rho}^t}) = 0.$$

Therefore, there exists some constant c such that

$$\frac{1}{2} \| A\boldsymbol{x}^{t+1} \|^2_{\boldsymbol{\rho}^{t+1}} - \frac{1}{2} \| A\boldsymbol{x}^{t+1} \|^2_{\boldsymbol{\rho}^t} \leq c.$$

On the other hand, as our function is convex, we have $\sum_{k=0}^{t} f(\boldsymbol{x}^{k+1}) \geq (t+1) f(\bar{\boldsymbol{x}}^{t+1})$ where $\bar{\boldsymbol{x}}^{t+1} = \frac{1}{t+1} \sum_{k=0}^{t} \boldsymbol{x}^{k+1}$. Therefore, we have

$$\begin{aligned}&(t+1) f(\boldsymbol{x}^*) - (t+1) f(\bar{\boldsymbol{x}}^{t+1}) - (t+1) \boldsymbol{\lambda}^{*T} A \bar{\boldsymbol{x}}^{t+1} \\ &\quad +\frac{1}{2} \| \boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^1)^{-1}} + \frac{1}{2} \| \boldsymbol{x}^0 - \boldsymbol{x}^* \|^2_{\bar{Q}} + c \geq 0. \end{aligned}$$

By dividing both sides by $-(t+1)$, we can obtain

$$\begin{aligned}&f(\bar{\boldsymbol{x}}^{t+1}) + \boldsymbol{\lambda}^{*T} A \bar{\boldsymbol{x}}^{t+1} - f(\boldsymbol{x}^*) \\ &\quad \leq \frac{1}{t+1} (\frac{1}{2} \| \boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^1)^{-1}} + \frac{1}{2} \| \boldsymbol{x}^0 - \boldsymbol{x}^* \|^2_{\bar{Q}} + c). \end{aligned}$$

Combining the above relationship with the Lagrangian function (15), we can conclude Theorem 2.

REFERENCES

[1] J. Lin, A. S. Morse, and B. D. O. Anderson, "The multi-agent rendezvous problem—The asynchronous case," in *Proc. 43rd IEEE Conf. Decis. Control*, vol. 2, Dec. 2004, pp. 1926–1931.

[2] Z. Fanzi, C. Li, and Z. Tian, "Distributed compressive spectrum sensing in cooperative multihop cognitive networks," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 1, pp. 37–48, Feb. 2011.

[3] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[4] F. Yan, S. Sundaram, S. V. N. Vishwanathan, and Y. Qi, "Distributed autonomous online learning: Regrets and intrinsic privacy-preserving properties," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 11, pp. 2483–2493, Nov. 2013.

[5] 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, Jan. 2017.

[6] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5262–5276, Oct. 2010.

[7] C. L. Zhang and Y. Q. Wang, "Distributed event localization via alternating direction method of multipliers," *IEEE Trans. Mobile Comput.*, vol. 17, no. 2, pp. 348–361, Feb. 2017.

[8] S. Nabavi, J. Zhang, and A. Chakrabortty, "Distributed optimization algorithms for wide-area oscillation monitoring in power systems using interregional PMU-PDC architectures," *IEEE Trans. Smart Grid*, vol. 6, no. 5, pp. 2529–2538, Sep. 2015.

[9] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[10] B. S. He, H. K. Xu, and X. M. Yuan, "On the proximal jacobian decomposition of ALM for multiple-block separable convex minimization problems and its relationship to ADMM," *J. Sci. Comput.*, vol. 66, no. 3, pp. 1204–1217, 2016.

[11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[12] Q. Ling and A. Ribeiro, "Decentralized dynamic optimization through the alternating direction method of multipliers," in *Proc. IEEE 14th Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jun. 2013, pp. 170–174.

[13] Q. Ling, Y. Liu, W. Shi, and Z. Tian, "Weighted ADMM for fast decentralized network optimization," *IEEE Trans. Signal Process.*, vol. 64, no. 22, pp. 5930–5942, Nov. 2016.

[14] R. L. Lagendijk, Z. Erkin, and M. Barni, "Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 82–105, Jan. 2013.

[15] Q. Shi and C. He, "Distributed source localization via projection onto the nearest local minimum," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Mar./Apr. 2008, pp. 2553–2556.

[16] A. Alanwar, Y. Shoukry, S. Chakraborty, P. Martin, P. Tabuada, and M. B. Srivastava, "PrOLoc: Resilient localization with private observers using partial homomorphic encryption," in *Proc. IPSN*, Apr. 2017, pp. 41–52.

[17] Y. Mo and R. M. Murray, "Privacy preserving average consensus," *IEEE Trans. Autom. Control*, vol. 62, no. 2, pp. 753–765, Feb. 2017.

[18] M. H. DeGroot, "Reaching a consensus," *J. Amer. Statist. Assoc.*, vol. 69, no. 345, pp. 118–121, Mar. 1974.

[19] Z. Huang, S. Mitra, and N. Vaidya, "Differentially private distributed optimization," in *Proc. Int. Conf. Distrib. Comput. Netw.*, 2015, Art. no. 4.

[20] S. Han, U. Topcu, and G. J. Pappas, "Differentially private distributed constrained optimization," *IEEE Trans. Autom. Control*, vol. 62, no. 1, pp. 50–64, Jan. 2017.

[21] E. Nozari, P. Tallapragada, and J. Cortés, "Differentially private distributed convex optimization via objective perturbation," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 1, pp. 395–408, Mar. 2018.

[22] M. T. Hale and M. Egerstedt, "Differentially private cloud-based multi-agent optimization with constraints," in *Proc. Amer. Control Conf.*, Jul. 2015, pp. 1235–1240.

[23] M. T. Hale and M. Egerstedt, "Cloud-enabled differentially private multi-agent optimization with constraints," *IEEE Trans. Control Netw. Syst.*, to be published.

[24] A. Alaeddini, K. Morgansen, and M. Mesbahi. (2017). "Adaptive communication networks with privacy guarantees." [Online]. Available: https://arxiv.org/abs/1704.01188

[25] S. Pequito, S. Kar, S. Sundaram, and A. P. Aguiar, "Design of communication networks for distributed computation with privacy guarantees," in *Proc. 53rd IEEE Conf. Decis. Control*, Dec. 2014, pp. 1370–1376.

[26] 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.

[27] N. M. Freris and P. Patrinos, "Distributed computing over encrypted data," in *Proc. IEEE 54th Annu. Allerton Conf. Commun., Control, Comput.*, Sep. 2016, pp. 1116–1122.

[28] Y. Shoukry *et al.*, "Privacy-aware quadratic optimization using partially homomorphic encryption," in *Proc. IEEE 55th Conf. Decis. Control*, Dec. 2016, pp. 5053–5058.

[29] C. Wang, K. Ren, and J. Wang, "Secure and practical outsourcing of linear programming in cloud computing," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 820–828.

[30] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. STOC*, vol. 9, 2009, pp. 169–178.

[31] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annu. Symp. Found. Comput. Sci.*, Nov. 1982, pp. 160–164.

[32] O. L. Mangasarian, "Privacy-preserving horizontally partitioned linear programs," *Optim. Lett.*, vol. 6, no. 3, pp. 431–436, 2012.

[33] S. Gade and N. H. Vaidya. (2017). "Private learning on networks: Part II." [Online]. Available: https://arxiv.org/abs/1703.09185

[34] M. H. Ruan, M. Ahmad, and Y. Q. Wang, "Secure and privacy-preserving average consensus," in *Proc. Workshop Cyber-Phys. Syst. Secur. Privacy*, 2017, pp. 123–129.

[35] I. Jo *et al.*, "Safe & efficient privacy-policy enforcement on hadoop," *Int. Inf. Inst. (Tokyo)*, vol. 15, no. 5, pp. 1973–1984, 2012.

[36] A. Machanavajjhala, A. Korolova, and A. D. Sarma, "Personalized social recommendations: Accurate or private," *Proc. VLDB Endowment*, vol. 4, no. 7, pp. 440–450, 2011.

[37] Y. Wang, W. Yin, and J. Zeng. (2015). "Global convergence of ADMM in nonconvex nonsmooth optimization." [Online]. Available: https://arxiv.org/abs/1511.06324

[38] A. Simonetto and G. Leus, "Distributed maximum likelihood sensor network localization," *IEEE Trans. Signal Process.*, vol. 62, no. 6, pp. 1424–1437, Mar. 2014.

[39] Y. Lou, L. Yu, S. Wang, and P. Yi, "Privacy preservation in distributed subgradient optimization algorithms," *IEEE Trans. Cybern.*, vol. 48, no. 7, pp. 2154–2165, Jul. 2018.

[40] F. Li, B. Luo, and P. Liu, "Secure information aggregation for smart grids using homomorphic encryption," in *Proc. 1st IEEE Int. Conf. Smart Grid Commun.*, Oct. 2010, pp. 327–332.

[41] J. Barzilai and J. M. Borwein, "Two-point step size gradient methods," *IMA J. Numer. Anal.*, vol. 8, no. 1, pp. 141–148, 1988.

[42] J. A. Bondy and U. S. R. Murty, *Graph Theory With Applications*, vol. 290. London, U.K.: Macmillan, 1976.

[43] 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 Process.*, vol. 62, no. 7, pp. 1750–1761, Apr. 2014.

[44] W. Deng, M. Lai, Z. Peng, and W. Yin, "Parallel multi-block ADMM with $o(1/k)$ convergence," *J. Sci. Comput.*, vol. 71, no. 2, pp. 712–736, 2017.

[45] S. Kontogiorgis and R. R. Meyer, "A variable-penalty alternating directions method for convex optimization," *Math. Program.*, vol. 83, nos. 1–3, pp. 29–53, 1998.

[46] B. He, L. Z. Liao, D. Han, and H. Yang, "A new inexact alternating directions method for monotone variational inequalities," *Math. Program.*, vol. 92, no. 1, pp. 103–118, 2002.

[47] C. Chen, B. He, Y. Ye, and X. Yuan, "The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent," *Math. Program.*, vol. 155, nos. 1–2, pp. 57–79, 2016.

[48] E. Wei and A. Ozdaglar, "Distributed alternating direction method of multipliers," in *Proc. 51st IEEE Conf. Decis. Control*, Dec. 2012, pp. 5445–5450.

[49] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Philadelphia, PA, USA: SIAM, 2003.

[50] B. He and H. Yang, "Some convergence properties of a method of multipliers for linearly constrained monotone variational inequalities," *Oper. Res. Lett.*, vol. 23, nos. 3–5, pp. 151–161, 1998.

[51] W. Du, Y. S. Han, and S. Chen, "Privacy-preserving multivariate statistical analysis: Linear regression and classification," in *Proc. SIAM Int. Conf. Data Mining*, 2004, pp. 222–233.

[52] K. Liu, H. Kargupta, and J. Ryan, "Random projection-based multiplicative data perturbation for privacy preserving distributed data mining," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 1, pp. 92–106, Jan. 2006.

[53] S. Han, W. K. Ng, L. Wan, and V. C. S. Lee, "Privacy-preserving gradient-descent methods," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 6, pp. 884–899, Jun. 2010.

[54] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.

[55] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 1999, pp. 223–238.

[56] O. Goldreich, *Foundations of Cryptography: Basic Applications*, vol. 2. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[57] Y. Q. Wang and J. P. Hespanha, "Distributed estimation of power system oscillation modes under attacks on GPS clocks," *IEEE Trans. Instrum. Meas.*, vol. 67, no. 7, pp. 1626–1637, Jul. 2018.

[58] *Python-Paillier Library*. Accessed: Jul. 19, 2018. [Online]. Available: http://python-paillier.readthedocs.io/en/latest/phe.html

[59] S. Zhu, M. Hong, and B. Chen, "Quantized consensus ADMM for multi-agent distributed optimization," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Mar. 2016, pp. 4134–4138.

[60] J. Bethencourt. *Advanced Crypto Software Collection*. Accessed: Jul. 19, 2018. [Online]. Available: http://acsc.cs.utexas.edu/libpaillier

[61] *Paillier Library*. Accessed: Jul. 19, 2018. [Online]. Available: http://acsc.cs.utexas.edu/libpaillier/

[62] D. Han and X. Yuan, "A note on the alternating direction method of multipliers," *J. Optim. Theory Appl.*, vol. 155, no. 1, pp. 227–238, 2012.

Chunlei Zhang received the B.S. degree in electronic and information engineering from Beihang University, Beijing, China, in 2015. She is currently pursuing the Ph.D. degree in electrical engineering with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, USA. Her research focuses on localization in mobile sensor networks and privacy preservation in decentralized optimization.

Muaz Ahmad is a currently pursuing the master's degree in computer engineering with Clemson University. He is currently an Intern test Engineer with Itron Inc. He has co-authored a couple of papers on decentralized average consensus. His research is focused on improving timing security of GPS receivers against spoofing attacks. His interests lie in software and embedded programming.

Yongqiang Wang was born in Shandong, China. He received the B.S. degree in electrical engineering and automation, the B.S. degree in computer science and technology from Xi'an Jiaotong University, Xi'an, Shaanxi, China, in 2004, and the M.Sc. and Ph.D. degrees in control science and engineering from Tsinghua University, Beijing, China, in 2009. From 2007 to 2008, he was with the University of Duisburg-Essen, Germany, as a Visiting Student. He was a Project Scientist with the University of California at Santa Barbara, Santa Barbara. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Clemson University. His research interests are cooperative and networked control, synchronization of wireless sensor networks, systems modeling and analysis of biochemical oscillator networks, and model-based fault diagnosis.

Dr. Wang was a recipient of the 2008 Young Author Prize from the IFAC Japan Foundation for a paper presented at the 17th IFAC World Congress in Seoul.