Privacy-preserving Decentralized Optimization Based on ADMM
Chunlei Zhang and Yongqiang Wang

Abstract—In this paper, we address the problem of privacy-preservation in decentralized optimization, where $N$ agents cooperatively minimize an objective function that is the sum of $N$ strongly convex functions private to these individual agents. In most existing decentralized optimization approaches, participating agents exchange and disclose estimates explicitly, which may not be desirable when the estimates 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 ADMM and partially homomorphic cryptography. In contrast to differential privacy based approaches which use noise to cover sensitive information and are subject to a trade-off between privacy and accuracy, our approach can provide privacy without compromising the optimality of the solution. 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 in the absence of any third party or aggregator. To facilitate the incorporation of encryption in a fully decentralized manner, we also introduce a new ADMM algorithm which allows time-varying penalty matrices and rigorously proves its convergence. Numerical simulations confirm the effectiveness and low computational complexity of the proposed approach.

I. INTRODUCTION

The problem of decentralized optimization has attracted remarkable attention in recent years due to its wide applications in various domains, ranging from multi-agent systems [1]–[4], machine learning [5], communication and networking [2]–[4], [6]–[10], to power and grid systems [11], [12]. In these applications, data are collected and processed in a decentralized and cooperative manner among multiple agents. This decentralized processing without using any central computing server provides advantages in terms of robustness to network traffic bottlenecks and single point failure, and greatly enhances scalability and flexibility.

In many applications of decentralized optimization, the problem can be formulated in the following form, in which $N$ agents cooperatively solve an unconstrained problem [6]–[12]:

$$\min_y \sum_{i=1}^{N} f_i(y),$$

where variable $y \in \mathbb{R}^D$ is common to all agents, function $f_i : \mathbb{R}^D \to \mathbb{R}$ is the local objective function of agent $i$. Formulation (1) has found applications in domains as diverse as spectrum sensing in cognitive networks [6], [7], path estimation in stochastic process [13], source localization in sensor networks [8]–[10], and monitoring of smart grids [11], [12].

Typical decentralized solutions to the optimization problem (1) include distributed (sub)gradient based algorithms [1]–[4], [14], [15], augmented Lagrangian methods (ALM) [16], [17], and the alternating direction method of multipliers (ADMM) as well as its variants [5], [10]–[19], 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, iteration of projection based source localization, a node can infer the exact position of neighboring nodes using three intermediate estimate values from those nodes [10], which is undesirable when agents do not want to reveal their position information [21]. Secondly, exchanging explicit estimates (states) without encryption is susceptible to eavesdroppers which try to intercept and steal information from exchanged packets. Such attackers may steal information about both exchanged estimates and the private function $f_i$.

To enable privacy preservation in decentralized optimization, one commonly used approach is differential privacy [22]–[29], which adds carefully-designed noise to exchanged estimates 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 [23], [25]. Observability-based design has been proposed for privacy preservation in linear multi-agent networks [30], [31]. 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 enable privacy is encryption. However, despite successful application in cloud based control and optimization [32], [33], conventional cryptographic techniques cannot be applied directly to a completely decentralized optimization problem without the assist of aggregators/third parties (note that traditional secure multiparty computation schemes like fully homomorphic encryption [34] and Yao’s garbled circuit [35] are too heavy to be computationally feasible for real-
time optimization \cite{20}). Recently, by using linear dynamical systems theory to facilitate cryptographic design, we proposed a privacy-preserving decentralized linear consensus approach \cite{36}. However, to the best of our knowledge, results are still lacking on cryptography based approaches that can enable privacy for a more complicated general nonlinear optimization problem like (1) in completely decentralized setting without any aggregators/third parties.

Based on our work on cryptography based privacy-preserving consensus seeking for linear systems, we address privacy-preserving decentralized optimization based on ADMM and partially homomorphic cryptography. We used ADMM because it has several unique advantages compared with other decentralized optimization approaches. First, ADMM has a fast convergence speed in both primal and dual iterations. By incorporating a quadratic regularization term, ADMM has a fast convergence speed in both primal and dual iterations. Second, the convergence of ADMM has been established \cite{5,18}. 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 error \cite{39}. Therefore, in this paper, we adopt the ADMM framework.

Contributions: The main contribution of this paper is a privacy-preserving decentralized optimization approach based on ADMM and partially homomorphic cryptography. To our knowledge, our approach is the first to enable privacy-preserving in completely decentralized optimization without compromising the optimality of the solution. To facilitate the incorporation of homomorphic encryption in ADMM in a fully decentralized manner, we also introduce a new ADMM which allows time-varying penalty matrices and rigorously prove its convergence. In contrast to differential privacy based approaches which use noise to cover sensitive information and are subject to a trade-off between privacy and accuracy, our approach can enable privacy preservation without sacrificing accuracy.

Organization: The rest of this paper is organized as follows: Section II first presents the conventional ADMM solution to (1), and then introduces a new ADMM which allows time-varying penalty matrices with guaranteed convergence. Based on the new ADMM and partially homomorphic cryptography, a completely decentralized privacy-preserving approach to solving problem (1) is proposed in Sec. III. Rigorous analysis of the guaranteed privacy of the approach are addressed in Sec. IV and its implementation details are discussed in Section V. Numerical simulation results are given in Section VI to confirm the effectiveness and computational efficiency of the proposed approach. In the end, we draw conclusions in Section VII.

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 will enable the incorporation of partially homomorphic cryptography in a completely decentralized manner to protect the privacy of all parties in decentralized optimization.

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 \cite{20}, 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 to itself in this paper, i.e., $i \in \mathcal{N}_i$, but $e_{i,i} \notin E$).

B. Conventional ADMM

To solve (1) in a decentralized manner, the conventional ADMM reformulates (1) as follows:

$$\min_{x_i \in \mathbb{R}^p, i \in \{1, 2, ..., N\}} \sum_{i=1}^{N} f_i(x_i)$$

subject to $x_i = x_j$, $\forall e_{i,j} \in E$,

where $x_i$ is considered as a copy of $x$ and belongs to agent $i$. To solve (2), each agent first exchanges its current state $x_i$ with its neighbors. Then it carries out local computations based on its private local objective function $f_i$ and the received state (estimate) 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 strongly convex. Detailed implementation of the conventional ADMM based on Jacobian update is elaborated as follows \cite{17,41,42}:

$$\begin{align}
    x_i^{t+1} &= \arg\min_{x_i} L_{\rho}(x_i^{t}, x_2^{t}, ..., x_{i-1}^{t}, x_i, x_{i+1}^{t}, ..., x_N^{t}, \lambda^{t}) \\
    &= \arg\min_{x_i} \frac{\gamma_i}{2} \| x_i - x_i^{t} \|^2 + \frac{\gamma_i}{2} \| x_i - x_i^{t} \|^2, \\
    \lambda^{t+1}_{i,j} &= \lambda^{t}_{i,j} + \rho (x_i^{t+1} - x_j^{t+1}),
\end{align}$$

(3)

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

$$L_{\rho}(x, \lambda) = \sum_{i=1}^{N} f_i(x_i) + \sum_{e_{i,j} \in E} (\lambda^{T}_{i,j} (x_i - x_j) + \frac{\rho}{2} \| x_i - x_j \|^2).$$

(5)

In (5), $x = [x_1^T, x_2^T, ..., x_N^T]^T$ is the augmented states, $\lambda_{i,j}$ is the Lagrange multiplier corresponding to the constraint $x_i = x_j$, and all $\lambda_{i,j}$ for $e_{i,j} \in E$ are stacked into $\lambda$. $\rho$ is the penalty parameter, which is a positive constant scalar.

The conventional ADMM is effective in solving (1). However, it cannot protect the privacy of participating agents
as estimates 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 alternating directions method (ADM) allows time-varying penalty matrices [43, 44], we present in the following an ADMM with time-varying penalty matrices. To this end, we first reformulate (1) in a more compact form:

\[ \min_x f(x) \quad \text{subject to} \quad Ax = 0, \]

where \( x = [x_1^T, x_2^T, \ldots, x_N^T]^T, f(x) = \sum_{i=1}^N f_i(x_i) \), and \( A \) is the edge-node incidence matrix of graph \( G \) as defined in [45]. For example, in the one-dimensional case (\( D = 1 \)), \( A \in \mathbb{R}^{[|E|] \times mD} \) is defined as

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

Here we define that each edge \( e_{i,j} \) originates from \( i \) and terminates at \( j \). For high dimensional cases (\( D \geq 2 \)), \( A \in \mathbb{R}^{[|E|] \times mD} \) can be obtained by replacing the elements 1 and -1 with \( I_D \) and \( -I_D \) (\( I_D \) denotes the \( D \) dimensional identity matrix), respectively.

Let \( \lambda_{i,j} \) be the Lagrange multiplier corresponding to the constraint \( x_i = x_j \), then we can form an augmented Lagrangian function of problem (6) as

\[
\mathcal{L}(x, \lambda, \rho) = \sum_{i=1}^N f_i(x_i) + \sum_{e_{i,j} \in E} \left( \lambda_{i,j}^T (x_i - x_j) + \frac{\rho_{i,j}}{2} \| x_i - x_j \|^2 \right),
\]

or in a more compact form:

\[
\mathcal{L}(x, \lambda, \rho) = f(x) + \lambda^T Ax + \frac{1}{2} \| Ax \|^2_{\rho},
\]

where

\[
\lambda = [\lambda_{i,j}]_{i,j \in E} \in \mathbb{R}^{[|E|] \times D},
\]

\[
\rho = \text{diag} \{ \rho_{i,j} I_D \}_{i,j \in E} \in \mathbb{R}^{[|E| \times D] \times [|E| \times D]}, \quad \rho_{i,j} > 0
\]

is the time-varying penalty matrix, and

\[
\| Ax \|^2_{\rho} = x^T A^T \rho A x.
\]

Now, inspired by [44], we propose a new ADMM which allows time-varying penalty matrices based on Jacobian update

\[
\begin{aligned}
x_i^{t+1} &= \arg\min_{x_i} \mathcal{L}(x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_N, \lambda, \rho) \\
&= x_i^t - \frac{\gamma_i}{2} A_i \left( x_i^t - x_j^t \right), \\
\rho_{i,j}^{t+1} &= \rho_{i,j}^t + \left( x_i^{t+1} - x_j^{t+1} \right)^2, \\
\lambda_{i,j}^{t+1} &= \lambda_{i,j}^t + \rho_{i,j}^{t+1} (x_i^{t+1} - x_j^{t+1}),
\end{aligned}
\]

for \( i = 1, 2, \ldots, N \).

Remark 1. The conventional ADMM 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 \( x_i = x_j \).

Remark 2. We did not use Gauss-Seidel update [48]-[47] because it requires a predefined global order and is not amenable to completely decentralized implementation, where no global coordinator exists.

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

Algorithm I

**Initial Setup:** Each agent initializes \( x_i^0, \rho_{i,j}^0 \).

**Input:** \( x_i^t, \lambda_{i,j}^{t-1}, \rho_{i,j}^t \)

**Output:** \( x_i^{t+1}, \lambda_{i,j}^t, \rho_{i,j}^{t+1} \)

1) Each agent sends \( x_i^t, \rho_{i,j}^t \) 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 updates \( \lambda_{i,j}^t \) as follows for \( j \in N_i \)

\[
\lambda_{i,j}^t = \lambda_{i,j}^{t-1} + \rho_{i,j}^t (x_i^t - x_j^t).
\]

It is clear that \( \lambda_{i,j}^t = -\lambda_{j,i}^t \) holds (note that when \( t = 0 \), we set \( \lambda_{i,j}^0 = \rho_{i,j}^0 (x_i^0 - x_j^0) \)).

3) All agents update their local vectors in parallel:

\[
x_i^{t+1} = \arg\min_{x_i} f_i(x_i) + \frac{\gamma_i}{2} \| x_i - x_i^t \|^2 + \frac{1}{2} \sum_{j \in N_i} (\lambda_{i,j}^T x_i + \rho_{i,j}^t \| x_i - x_j^t \|^2).
\]

Here we added two proximal terms \( \frac{\rho_{i,j}^t}{2} \| x_i - x_i^t \|^2 \) and \( \frac{\rho_{i,j}^t}{2} \| x_i - x_j^t \|^2 \) to accommodate the influence of \( x_i^t \). For all \( \gamma_i > 0 \), \( \rho_{i,j}^t \) is set to

\[
\rho_{i,j}^t = \frac{1}{\gamma_i} \sum_{j \in N_i, j \neq i} \rho_{j,i}^t.
\]

4) Each agent updates \( \rho_{i,j}^{t+1} \) and sets \( t = t + 1 \).

Remark 3. A weighted ADMM which also assigns different weights to different equality constraints is proposed in [19] to solve (1). However, the weights in [19] are kept constant while Algorithm I 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 I under the following standard assumptions [45]:

**Assumption 1.** Each private local function \( f_i : \mathbb{R}^D \rightarrow \mathbb{R} \) is strongly convex and continuously differentiable.

**Assumption 2.** Problem (6) has an optimal solution, i.e., the Lagrangian function
\[
L(x, \lambda) = f(x) + \lambda^T Ax
\]
has a saddle point \((x^*, \lambda^*)\) such that
\[
L(x^*, \lambda^*) \leq L(x^i, \lambda^i) \leq L(x, \lambda^*)
\]
holds for all \( x \in \mathbb{R}^{ND} \) and \( \lambda \in \mathbb{R}^{E|D|} \).

Denote the iterating results in the kth step in Algorithm I as follows:
\[
x^k = [x_1^k, x_2^k, \ldots, x_N^k]^T, \\
\lambda^k = [\lambda_{i,j}^k]_{i,j \in E}, \\
\rho^k = \text{diag}(\rho_{i,i}^k)_{i \in [1, N], j \in \mathbb{E}}.
\]
Further augment the coefficients \( \gamma_i \) \( (i = 1, 2, \ldots, N) \) in (13) into the matrix form
\[
Q_P = \text{diag}(\gamma_1 I_D, \gamma_2 I_D, \ldots, \gamma_N I_D),
\]
and augment \( \rho_{i,j}^k \) into the following matrix form
\[
Q_C = \text{diag}(\sum_{j \in N_i} \rho_{i,j}^k I_D, \sum_{j \in N_2} \rho_{2,j}^k I_D, \ldots, \sum_{j \in N_N} \rho_{N,j}^k I_D).
\]
Given that the following relationship is true for \( k \geq 0 \) in Algorithm I
\[
\rho_{i,i}^k = 1 - \sum_{j \in N_i, j \neq i} \rho_{i,j}^k,
\]
we have the following equation
\[
Q_C^k = I_{ND},
\]
i.e., \( Q_C^k \) is an identity matrix.

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

**Condition A:** The sequence \( \{\rho^k\} \) satisfies
\[
0 < \rho^0 \leq \rho^k \leq \rho^{k+1} \leq 0, \quad \forall k \geq 0,
\]
where \( \rho^0 > 0 \) means that \( \rho^0 \) is positive definite, and similarly \( \rho^k \leq \rho^{k+1} \) means that \( \rho^{k+1} - \rho^k \) is positive semi-definite.

**Condition B:** \( Q_P + Q_C^k > A^T A \).

**Proof:** The proof is provided in the Appendix.

III. PRIVACY-PRESERVING DECENTRALIZED OPTIMIZATION

Like most decentralized algorithms in the literature, e.g., [1]-[5], [14]-[19]. Algorithm I requires agents to exchange explicitly and disclose estimates (states) in each iteration among neighboring agents to reach consensus on the final optimal solution. In this section, we combine partially homomorphic cryptography with Algorithm I to propose a privacy-preserving approach for decentralized optimization.

A. Paillier Cryptosystem

Our method use the flexibility of time-varying penalty matrices in Algorithm I to enable the incorporation of Paillier cryptosystem [48] 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 publically 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 below:

**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 \( (\lambda, \mu) \).

**Encryption** \( (c = E(m)) \):
Recall the definitions of \( \mathbb{Z}_n = \{ z | z \in \mathbb{Z}, 0 \leq z < n \} \) and \( \mathbb{Z}_n^* = \{ z | z \in \mathbb{Z}, 0 \leq z < n, \text{gcd}(z, n) = 1 \} \).
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 \( m \in \mathbb{Z}_n, c \in \mathbb{Z}_n^* \).

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

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:
\[
E(m_1, r_1) \cdot E(m_2, r_2) = E(m_1 + m_2, r_1 r_2) \\
E(m)^k = E(km), \quad k \in \mathbb{Z}_n^+
\]
Due to the existence of random \( r \), the Paillier cryptosystem is resistant to the dictionary attack [49]. Since \( r_1 \) and \( r_2 \) play no role in the decryption process, (16) can be simplified as
\[
E(m_1) \cdot E(m_2) = E(m_1 + m_2)
\]
B. Privacy-Preserving Decentralized Optimization

In this subsection, we combine Paillier cryptosystem with Algorithm I to enable privacy preservation in the decentralized solving of optimization problem \((1)\). First, note that solving (15) amounts to solving the following problem:

\[
\nabla f_i(x(t)) + \sum_{j \in N_i} (\lambda^t_{i,j} + \rho^t_{i,j}(x_i(t) - x_j(t))) + \gamma_i(x_i(t) - x^t_i) = 0.
\]

Let \(\lambda_i = \sum_{j \in N_i} \lambda_{i,j}\), then (19) reduces to the following equation

\[
\nabla f_i(x_i(t)) + (\sum_{j \in N_i} \rho^t_{i,j}(x_j(t) - x_i(t)) - \sum_{j \in N_i} \rho^t_{i,j}(x_j^t - \gamma_i x_i^t)) = 0.
\]

Given that we have set \(\rho^k_{i,j} = 1 - \sum_{j \in N_i, j \neq i} \rho^k_{i,j}\) in (14), we can further reduce (20) to

\[
\nabla f_i(x_i(t)) + (1 + \gamma_i) x_i^t - \sum_{j \in N_i} \rho^t_{i,j}(x_j(t) - x_i(t)) - (1 + \gamma_i)x_i^t = 0.
\]

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

Algorithm II

**Initial Setup:** Each agent initializes \(x_i^0\).

**Input:** \(x_i^t, \lambda^t_{i,j}\), \(\lambda^t_{i,j}\)

**Output:** \(x_i^{t+1}, \lambda^t_{i,j}\)

1. Agent \(i\) encrypts \(-x_i^t\) with its public key \(k_{pi}\):
   \[x_i^t \rightarrow E_i(-x_i^t)\]
   Here subscript \(i\) denotes encryption using the public key of agent \(i\).
2. Agent \(i\) sends \(E_i(-x_i^t)\) and its public key \(k_{pi}\) to neighboring agents.
3. Agent \(j \in N_i\) decrypts \(x_j^t\) with agent \(i\)’s public key \(k_{pi}\):
   \[x_j^t \rightarrow E_i^t(x_j^t)\]
4. Agent \(j \in N_i\) computes the difference directly in ciphertext:
   \[E_i(x_j^t - x_i^t) = E_i^t(x_j^t) - E_i(-x_i^t)\]
5. Agent \(j \in N_i\) computes the \(b^{-1}_{j \rightarrow i}\)-weighted difference in ciphertext:
   \[E_i^t(b^{-1}_{j \rightarrow i}(x_j^t - x_i^t)) = (E_i^t(x_j^t - x_i^t))(b^{-1}_{j \rightarrow i})\]
6. Agent \(j \in N_i\) sends \(E_i^t(b^{-1}_{j \rightarrow i}(x_j^t - x_i^t))\) back to agent \(i\).
7. Agent \(i\) decrypts the message received from \(j\) with its private key \(k_{si}\) and multiplies the result with \(b^{-1}_{i \rightarrow j}\) to get \(\rho^t_{i,j}(x_j^t - x_i^t)\).
8. Computing (12), agent \(i\) obtains \(\lambda^t_{i,j}\).
9. Computing (21), agent \(i\) obtains \(x_i^{t+1}\).
10. Each agent updates \(b_{i \rightarrow j}^{t+1}\) 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 \(x_i^t = x_j^t\) is true for \(j \in N_i\). Otherwise, agent \(i\)’s state \(x_i^t\) is encrypted and will not be revealed to its neighbors.
2. Agent \(i\)’s state \(x_i^t\) and its intermediate communication data \(b_{j \rightarrow i}^t(x_j^t - x_i^t)\) will not be revealed to outside eavesdroppers, since they are encrypted.
3. The state of agent \(j \in N_i\) will not be revealed to agent \(i\), because the decrypted message obtained by agent \(i\) is \(b_{j \rightarrow i}^t(x_j^t - x_i^t)\) with \(b_{j \rightarrow i}^t\) only known to agent \(j\) and varying in every iteration.
4. We encrypt \(E_i(-x_i^t)\) 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 to vectors directly. For vector messages \(x_i^t \in \mathbb{R}^D\), each element of the vector (a real number) has to be encrypted separately. For notation convenience, we still denoted it in the same way as scalars, e.g., \(E_i(-x_i^t)\).
6. Paillier cryptosystem only works on 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.

The key to achieve privacy preservation is to construct \(\rho^t_{i,j}, i \neq j\) as the product of two random positive numbers \(b^t_{i \rightarrow j}, b^t_{j \rightarrow i}\), with \(b^t_{i \rightarrow j}\) generated by and only known to agent \(i\) and \(b^t_{j \rightarrow i}\) generated by and only known to agent \(j\). Next we show that the privacy preservation mechanism does not affect convergence.

**Theorem 2.** The privacy-preserving algorithm II will converge to the optimal solution if \(b^{-1}_{i \rightarrow j}, b^{-1}_{j \rightarrow i}\), and \(\gamma_i\) are updated in the following way:

1. \(b_{i \rightarrow j}^t\) is randomly chosen from \([b_{i \rightarrow j}^t, b_{i \rightarrow j}^t]\) with \(b_{i \rightarrow j}^t > 0\) denoting a predetermined constant only known to agent \(i\);
2. \(\gamma_i\) is chosen randomly in the interval \([N\frac{b^2}{2}, \frac{b}{2}])\) with \(b > \max\{b_{i \rightarrow j}\}\) denoting a predetermined positive constant known to everyone and \(b\) a threshold set 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 \(\gamma_i\) is updated following 2), then Condition A and Condition B in Theorem 1 will be met automatically. Therefore, the theorem is proven.

IV. Privacy Analysis

In this section, we rigorously prove that each agent’s private information, e.g., immediate estimate (state) \(x_i^t\) and private local objective function \(f_i\), cannot be inferred by honest-but-curious adversaries, which are commonly used in privacy study. It is worth noting that since all exchanged messages are encrypted, an outside eavesdropper cannot learn anything by intercepting these messages.
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 parties.

As indicated in Sec. III, our approach in Algorithm II guarantees that state information is not leaked to any neighbor in one iteration. However, would some information get leaked to an honest-but-curious adversary over time? More specifically, if an honest-but-curious adversary observes carefully its interactions 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 states of its neighbors even by collecting samples from multiple steps.

**Theorem 3.** Assume that all agents follow Algorithm II. Then agent $i$’s state $x_k^i$ cannot be inferred by an honest-but-curious adversary agent $i$ unless $x_k^i = x_{k+1}^i$ is true.

**Proof:** Suppose that an honest-but-curious adversary 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 $y_k^j = b_{k-j}^i b_{j-i}^i (x_k^j - x_i^k)$ ($k = 0, 1, \ldots, K$), i.e., adversary agent $i$ can establish $K + 1$ equations based on received information:

$$
\begin{align*}
\{ & y_0^j = b_{0-j}^i b_{j-i}^i (x_0^j - x_0^i), \\
& y_1^j = b_{1-j}^i b_{j-i}^i (x_1^j - x_1^i), \\
& \vdots \\
& y_{K-1}^j = b_{K-1-j}^i b_{j-i}^i (x_{K-1}^j - x_{K-1}^i), \\
& y_K^j = b_{K-j}^i b_{j-i}^i (x_K^j - x_K^i). \\
\}
\end{align*}
$$

(22)

To the adversary agent $i$, in the system of equations (22), $y_k^j, b_{k-j}^i, x_k^j$ ($k = 0, 1, 2, \ldots, K$) are known, but $x_k^i, b_{k-i}^j$ ($k = 0, 1, 2, \ldots, K$) are unknown. So the above system of $K + 1$ equations contains $2(K + 1)$ unknown variables. It is clear that adversary agent $i$ cannot solve the system of equations to infer the unknowns $x_k^i$ and $b_{k-i}^j$ ($k = 0, 1, 2, \ldots, K$) of agent $j$. It is worth noting that if for some time index $k$, $x_k^i = x_{k+1}^i$ 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 $y_k^j$ is 0.

**Corollary 1.** The privacy-preserving approach in Algorithm II is able to guarantee privacy against honest-but-curious adversaries even when $b_{k-i}^j$ is constant.

**Proof:** Following the line of reasoning in Theorem 3 we can see that when $b_{k-i}^j$ is time invariant, i.e., $b_{0-j}^i = b_{1-j}^i = \ldots = b_{K-j}^i$, inferring the information of agent $j$ by adversary agent $i$ boils down to solving a system of $K + 1$ equations with $K + 2$ unknowns. Therefore, adversary agent $i$ cannot infer the information of agent $j$ by even using the information it acquired in several steps. Of course, similar to the time-varying $b_{k-i}^j$ case, if for some time index $k$, $x_k^i = x_{k+1}^i$ happens to be true, then adversary agent $i$ will be able to know that agent $j$ has the same state at time index $k$ based on the fact that $y_k^j$ is 0.

Based on a similar way of reasoning, we can obtain that an honest-but-curious adversary agent $i$ cannot infer the private information of function $f_j$ from a neighboring agent $j$ either.

**Corollary 2.** In Algorithm II, agent $j$’s private local function $f_j$ will not be revealed to an honest-but-curious adversary agent $i$.

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

$$
\begin{align*}
\nabla f_j(x_j^1) + (1 + \gamma_j) x_j^1 + \lambda_j^0 \\
- \sum_{m \in N_j} \rho_{j,m}^0 (x_m^0 - x_j^0) - (1 + \gamma_j) x_j^0 = 0, \\
\nabla f_j(x_j^2) + (1 + \gamma_j) x_j^2 + \lambda_j^1 \\
- \sum_{m \in N_j} \rho_{j,m}^1 (x_m^1 - x_j^1) - (1 + \gamma_j) x_j^1 = 0, \\
& \vdots \\
\nabla f_j(x_j^{K-1}) + (1 + \gamma_j) x_j^{K-1} + \lambda_j^{K-2} \\
- \sum_{m \in N_j} \rho_{j,m}^{K-2} (x_m^{K-2} - x_j^{K-2}) - (1 + \gamma_j) x_j^{K-2} = 0, \\
\nabla f_j(x_j^K) + (1 + \gamma_j) x_j^K + \lambda_j^{K-1} \\
- \sum_{m \in N_j} \rho_{j,m}^{K-1} (x_m^{K-1} - x_j^{K-1}) - (1 + \gamma_j) x_j^{K-1} = 0.
\end{align*}
$$

(23)

In the system of $K$ equations (23), $\nabla f_j(x_j^k)$ ($k = 1, 2, \ldots, K$), $\gamma_j$, and $x_j^k$ ($k = 0, 1, 2, \ldots, K$) are unknown to adversary agent $i$. Parameters $\lambda_j^k$ and $\sum_{m \in N_j} \rho_{j,m}^k (x_m^k - x_j^k)$ are known to adversary agent $i$ only when agent $j$ has agent $i$ as the only neighbor. Otherwise, $\lambda_j^k$ and $\sum_{m \in N_j} \rho_{j,m}^k (x_m^k - x_j^k)$ are unknown to adversary agent $i$. So the above system of $K$ equations contains $2K + 2$ unknown variables when agent $j$ has agent $i$ as the only neighbor, and $4K + 2$ unknowns when agent $j$ has more than one neighbor. In neither case can adversary agent $i$ infer function $f_j$ by solving (23).

**Remark 4.** It is worth noting that 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 $i$ in inferring the information of agent $j$ because the combination will not change the fact that the number of unknowns is larger than the number of establishable equations.

**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 [26], [29], [30] where privacy cannot be protected for an agent if it has the honest-but-curious adversary as the
only neighbor. This shows the disparate difference between decentralized optimization and the linear consensus problem.

V. IMPLEMENTATION DETAILS

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

1) Paillier encryption cannot be used on vectors directly. For vector messages \( x_i^t \in \mathbb{R}^D \), each element of the vector (a real number) has to be encrypted separately. For notation convenience, we still denoted it in the same way as scalars, e.g., \( E_i(-x_i^t) \).

2) In modern communication, a real number is represented by a floating point number, while encryption techniques only work on unsigned integers. To deal with this problem, we uniformly multiplied each element of the vector message \( x_i^t \in \mathbb{R}^D \) (in floating point representation) by a sufficiently large number \( N_{\text{max}} \) and round off the fractional part during the encryption to convert it to an integer. After decryption, \( N_{\text{max}} \) is divided from the result. This process is conducted in each iteration and this quantization brings an error upper-bounded of \( \frac{1}{N_{\text{max}}} \).

3) As indicated in 2), encryption techniques only work on 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 two’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 illustrated the efficiency of the proposed approach using C/C++ implementations. Then we compared our approach with the algorithm in [23], which is based on differential privacy. The open-source C implementation of the Paillier cryptosystem [51] is used in our simulations. We conducted numerical experiments on the following global objective function

\[
f(x) = \sum_{i=1}^{N} \frac{1}{2} \| x - \theta_i \|^2,
\]

which makes the optimization problem \( \mathbf{1} \) become

\[
\min_x \sum_{i=1}^{N} \frac{1}{2} \| x - \theta_i \|^2
\]

with \( \theta_i \in \mathbb{R}^2 \). Hence, each agent \( i \) deals with a private local objective function

\[
f_i(x_i) = \frac{1}{2} \| x_i - \theta_i \|^2, \forall i \in \{1, 2, \ldots, N\}.
\]

We used the above function \( \mathbf{24} \) because it is easy to verify whether the obtained solution is the minimal to the original optimization problem, which should be \( \frac{\sum_{i=1}^{N} \theta_i}{N} \). Furthermore, \( \mathbf{24} \) makes it easy to compare with \( \mathbf{23} \), whose verification is also based on \( \mathbf{24} \).

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

A. Evaluation of Our Approach

We implemented Algorithm II on different network topologies, all of which gave the right optimal solution. Simulation results suggested that our approach always converged to the optimal solution (i.e., \( \sum_{i=1}^{N} \theta_i \)) of \( \mathbf{25} \). Fig. 2 visualizes the evolution of \( x_i^t \) \( (i = 1, 2, \ldots, 6) \) in one specific run where the network structure is illustrated in Fig. 1. In Fig. 2, \( x_{ij}^t \) \( (i = 1, 2, \ldots, 6, j = 1, 2) \) denotes the \( j \)th element of \( x_i^t \). Each \( x_i^t \) \( (i = 1, 2, \ldots, 6) \) converged to the optimal value \([38.5; 407/6]\).

In this run, \( b \) was set to \( 0.65 \) and \( \gamma_i \)s were set to \( 3 \).

![Fig. 1: A network of six agents (N = 6).](image)

![Fig. 2: The evolution of x_i^t (i = 1, 2, ..., 6).](image)

Fig. 3 visualizes the encrypted weighted differences (in ciphertext) \( E_i(b_{1 \rightarrow 2}^t(x_{2}^t - x_{1}^t)), E_i(b_{1 \rightarrow 3}^t(x_{3}^t - x_{1}^t)), \) and \( E_i(b_{5 \rightarrow 6}^t(x_{6}^t - x_{1}^t)) \). It is worth noting that although the estimates of all agents have converged after about 40 iterations, the encrypted weighted differences (in ciphertext) still appeared random to an outside eavesdropper.

In addition, the encryption/decryption computation took about 1ms for each iteration on a 3.6 GHz CPU, which is manageable in most applications.
B. Comparison with the algorithm in [23]

We also compared our approach with the differential privacy based privacy-preserving optimization algorithm in [23]. The network structure used for comparison is still Fig. 1. We simulated the algorithm in [23] under seven different privacy levels: \( \epsilon = 0.2, 1, 10, 20, 30, 50, 100 \). The global function we used for comparison was fixed to

\[
 f(x) = \sum_{i=1}^{6} \frac{1}{2} \| x - \theta_i \|^2
\]

where \( \theta_i = [0.1 \times (i-1) + 1, 0.1 \times (i-1) + 0.2] \). 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 [23]. Note that the optimal solution \([0.35; 0.45]\) was in \( \mathcal{X} \). Parameter settings for the algorithm in [23] are detailed as follows: \( n = 2 \), \( c = 0.5 \), \( q = 0.8 \), \( p = 0.9 \), and

\[
 a_{ij} = \begin{cases} 
 0.2 & j \in \mathcal{N}_i \setminus i, \\
 0 & j \notin \mathcal{N}_i, \\
 1 - \sum_{j \in \mathcal{N}_i \setminus i} a_{ij} & i = j,
\end{cases}
\]

for \( i = 1, 2, ..., 6 \). Here \( \mathcal{N}_i \setminus i \) denotes all values except \( i \) in set \( \mathcal{N}_i \). Furthermore, we used the performance index \( d \) in [23] to quantify the optimization error, which was computed by the average value of squared distances over \( M \) runs [23], i.e.,

\[
 d = \frac{1}{M} \sum_{i=1}^{M} \| x_i - [0.35; 0.45] \|^2
\]

with \( x_i \) the obtained solution in the \( i \)th run.

We ran our algorithm for 5000 times. Simulation results showed that our approach converged to \([0.35; 0.45]\) with error \( d = 3.14 \times 10^{-11} \), which is negligible compared with the simulation results for the algorithm in [23], as shown in Fig. 4 (each differential privacy level was implemented for 5000 times). The results confirm the trade-off between privacy and accuracy for differential privacy based approaches and demonstrate the clear advantages of our approach in terms of optimization accuracy.

VII. CONCLUSIONS

In this paper, we proposed a privacy-preserving decentralized optimization approach based on ADMM and partially homomorphic cryptography. By incorporating Paillier cryptosystem into a newly proposed decentralized ADMM, our approach provides guarantee for privacy preservation without compromising the optimality of optimization. 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 estimates of neighboring nodes even by recording and analyzing the information exchanged in multiple iterations. The new ADMM allows time-varying penalty matrices and have theoretically guaranteed convergence, which makes it of mathematical interest by itself.

APPENDIX

To prove Theorem 1, we form a variational inequality \( MVI(Q, U) \) similar to (5)-(6) in [47] first:

\[
 \langle u - u^*, Q(u^*) \rangle \geq 0, \quad \forall u, \quad (26)
\]

where

\[
 u^* := \begin{pmatrix}
 x_1^* \\
 x_2^* \\
 \vdots \\
 x_N^*
\end{pmatrix},
 Q(u^*) := \begin{pmatrix}
 \xi_1^* + [A]^T_{11} \lambda^* \\
 \xi_2^* + [A]^T_{22} \lambda^* \\
 \vdots \\
 \xi_N^* + [A]^T_{NN} \lambda^*
\end{pmatrix}, \quad (27)
\]

\( \lambda^* \in \partial f_i(x_i^*), \forall i \in \{1, 2, ..., N\} \).

In (27), \( [A] \) denotes the columns of matrix \( A \) that are associated with agent \( i \). By recalling the first-order necessary and sufficient condition for convex programming [47], it is easy to see that solving problem (3) amounts to solving the above \( MVI(Q, U) \) [47]. Denote the solution set of \( MVI(Q, U) \) as \( \mathcal{U}^* \). Since \( f_i \) is strongly convex, \( \partial f_i(x_i) \) is strongly monotone, the \( MVI(Q, U) \) is solvable and \( \mathcal{U}^* \) is nonempty (In fact, using the strongly convexity of \( f_i \), we can get that \( x_i^* \) is unique) [47].

Next, we introduce several lemmas and theorems that contribute to the proof of Theorem 1.
Lemma 1. Each \( u^* = (x^*, \lambda^*) \in U^* \) is also the saddle point of the Lagrangian function \( L(x, \lambda) = f(x) + \lambda^T A x \).

Proof: The results can be obtained from Part 2.1 in [17] directly.

Lemma 2. If \( A x^{k+1} = 0 \) and \( x^{k+1} = x^k \) hold, then \((x_{k+1}^1, x_{k+1}^2, ..., x_{k+1}^N)^T, \lambda^{k+1}\) is a solution to \( MV1(Q, U) \).

Proof: Using the definition of matrix \( A \) and the update rule of \( \lambda^{k+1} \) in (11), we can easily see that the assumption \( A x^{k+1} = 0 \) implies \( \lambda^{k+1} = \lambda^k \) and

\[
x_{1}^{k+1} = x_{2}^{k+1} = ... = x_{N}^{k+1}.
\]

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

\[
(x_i - x_{i}^{k+1})^T (\xi_{k+1}^i + \sum_{j \in N_i} (\lambda_{i,j}^k + \rho_{i,j}^k (x_{i,j}^{k+1} - x_{j}^{k+1})) + \rho_{i,j}^k (x_{i,j}^{k+1} - x_{j}^{k+1})) \geq 0.
\]

where \( \xi_{k+1}^i \in \partial f_i(x_{i}^{k+1}) \). Then based on the assumption \( x^{k+1} = x^k \), the fact \( \lambda^{k+1} = \lambda^k \), and the definition of matrix \( A \), we have

\[
(x_i - x_{i}^{k+1})^T (\xi_{k+1}^i + [A]_i^T \lambda^{k+1}) \geq 0.
\]

Therefore, \((x_{1}^{k+1}, x_{2}^{k+1}, ..., x_{N}^{k+1}, \lambda^{k+1})\) is a solution to \( MV1(Q, U) \).

Theorem 4. Let \( \rho^k \) satisfy Condition A, \( \bar{Q} \triangleq Q_P + Q_C^{K} \) satisfy Condition B, and \((x^*, \lambda^*)\) be the saddle point of the Lagrangian function \( L(x, \lambda) = f(x) + \lambda^T A x \), then we have

\[
\begin{align*}
&\|\lambda^{k+1} - \lambda^*\|_{\rho^k+1}^2 + \|x^{k+1} - x^*\|_Q^2 \\
&\leq \|\lambda^k - \lambda^*\|_{\rho^k+1}^2 + \|x^k - x^*\|_Q^2 \\
&- (\|Ax^{k+1}\|_{\rho^k}^2 + \|x^{k+1} - x^k\|_A^2 + \rho_{k+1}^a + A^T \rho_{k+1}^a + Q) \\
&+ \|Ax^{k+1}\|_{\rho_{k+1}^a}^2 - \|Ax^k\|_{\rho_{k+1}^a}^2 + \rho_{k+1}^a.
\end{align*}
\]

To prove Theorem 4, we first introduce two lemmas:

Lemma 3. Let \( x^k = [x_1^k, x_2^k, ..., x_N^k]^T \) and \( \lambda^k = [\lambda_{i,j}^k]_{i,j \in E} \) be the intermediate results of iteration \( k \) in Algorithm I, then the following inequality holds for all \( k \):

\[
\begin{align*}
&f(x) - f(x^{k+1}) + (x - x^{k+1})^T A \lambda^k + (x - x^{k+1})^T A \rho^k A x^k + (x - x^{k+1})^T Q(x^{k+1} - x^k) \\
&\geq 0,
\end{align*}
\]

where \( \bar{Q} \triangleq Q_P + Q_C^{K} \).

Proof: The proof follows from [9]. For the purpose of completeness, we sketch the proof here. Denote by \( g_i \) the function

\[
g_i^k(x_i) = \sum_{j \in N_i} (\lambda_{i,j}^{kT} x_j + \frac{k_{ij}^2}{2} \|x_i - x_j^k\|^2 + \frac{\gamma_{ij}}{2} \|x_i - x_j^k\|^2).
\]

Using \( \xi_{k+1}^i \in \partial f_i(x_{i}^{k+1}) \), we get

\[
\xi_{k+1}^i + \nabla g_i(x_{i}^{k+1}) = 0
\]

and

\[
(x_i - x_{i}^{k+1})^T [\xi_{k+1}^i + \nabla g_i(x_{i}^{k+1})] = 0
\]

based on the fact that \( x_{i}^{k+1} \) is the optimizer of \( g_i^k \). On the other hand, as \( f_i \) is convex, the following relationship holds:

\[
f_i(x_i) \geq f_i(x_{i}^{k+1}) + (x_i - x_{i}^{k+1})^T (\xi_{k+1}^i + \nabla g_i(x_{i}^{k+1})).
\]

Then we can get

\[
f_i(x_i) - f_i(x_{i}^{k+1}) + (x_i - x_{i}^{k+1})^T \nabla g_i(x_{i}^{k+1}) \geq 0.
\]

Substituting \( \nabla g_i(x_{i}^{k+1}) \) with (21), we obtain

\[
\begin{align*}
&f_i(x_i) - f_i(x_{i}^{k+1}) + (x_i - x_{i}^{k+1})^T \sum_{j \in N_i} (\lambda_{i,j}^k + \rho_{i,j}^k (x_{i,j}^{k+1} - x_{j}^{k+1})) + \gamma_i (x_{i}^{k+1} - x_i^k) \\
&\geq 0.
\end{align*}
\]

Noting \( \lambda_{i,i} = 0 \) and \( \lambda_{i,j} = -\lambda_{j,i} \), based on the definition of matrices \( A \) and \( \rho \), we can rewrite the above inequality as

\[
f_i(x_i) - f_i(x_{i}^{k+1}) + (x_i - x_{i}^{k+1})^T \sum_{j \in N_i} \rho_{i,j}^k (x_{i,j}^{k+1} - x_{j}^{k+1}) + \gamma_i (x_{i}^{k+1} - x_i^k) \geq 0,
\]

where \( [A]_i \) denotes the columns of matrix \( A \) corresponding to agent \( i \).

Summing both sides of (32) over \( i = 1, 2, ..., N \), and using

\[
\begin{align*}
&\sum_{i=1}^{N} (x_i - x_{i}^{k+1})^T [A]_i^T \lambda^k = (x - x^{k+1})^T A^T \lambda^k, \\
&\sum_{i=1}^{N} (x_i - x_{i}^{k+1})^T \sum_{j \in N_i} \rho_{i,j}^k x_{j}^{k+1} = (x - x^{k+1})^T Q_P^x x^{k+1}, \\
&\sum_{i=1}^{N} (x_i - x_{i}^{k+1})^T \sum_{j \in N_i} \rho_{i,j}^k x_{j}^{k} = (x - x^{k+1})^T (-A^T \rho^k A + Q_C^k) x^{k}, \\
&\sum_{i=1}^{N} (x_i - x_{i}^{k+1})^T \gamma_i (x_{i}^{k+1} - x_i^k) = (x - x^{k+1})^T Q_P^x (x^{k+1} - x^k),
\end{align*}
\]

we can get the lemma.

Lemma 4. Let \( x^k = [x_1^k, x_2^k, ..., x_N^k]^T \) and \( \lambda^k = [\lambda_{i,j}^k]_{i,j \in E} \) be the intermediate results of iteration \( k \) in Algorithm I, then the following equality holds for all \( k \):

\[
\begin{align*}
&- (x^{k+1})^T A \lambda^k + A \lambda^k + (x - x^{k+1})^T Q(x^{k+1} - x^k) \\
&= -\frac{1}{2} \left( \| \lambda^{k+1} - \lambda^* \|_{\rho_{k+1}}^2 + \| \lambda^k - \lambda^* \|_{\rho_k}^2 \right) \\
&+ \frac{1}{2} \left( \| A x^{k+1} \|_{\rho_{k+1}}^2 + \| A x^k \|_{\rho_k}^2 \right) \\
&+ \frac{1}{2} \left( \| x^{k+1} - x^k \|_Q^2 - \| x^* - x^k \|_Q^2 \right) \\
&+ \frac{1}{2} \left( \| x^{k+1} - x^* \|_Q^2 - \| x^* - x^k \|_Q^2 \right).
\end{align*}
\]

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

\[
(x^{k+1})^T A \lambda^k + (x - x^{k+1})^T Q(x^{k+1} - x^k) = (x^{k+1} - \lambda^k)^T (\rho^{k+1} - 1) (\lambda^k - \lambda^*).
\]
Moreover, the following equalities can be established by using the Lagrangian function (15), we can get $A\mathbf{x}^* = 0$ [45].

Moreover, the following equalities can be established by using algebraic manipulations:

\[
(x^{k+1} - x^*)^T \bar{Q}(x^{k+1} - x^k) = \frac{1}{2} \| x^{k+1} - x^k \|^2_{\bar{Q}} + \frac{1}{2} \| x^{k+1} - x^* \|^2_{\bar{Q}} - \| x^k - x^* \|^2_{\bar{Q}},
\]

\[
(x^{k+1})^T A^T \rho^k A \mathbf{x} = \frac{1}{2} \| A(x^{k+1} - x^k) \|^2_{\rho^k} - \frac{1}{2} \| A \mathbf{x}^{k+1} \|^2_{\rho^k} - \frac{1}{2} \| A \mathbf{x}^k \|^2_{\rho^k}, \tag{35}
\]

\[
(\lambda^{k+1} - \lambda^k)^T (\rho^{k+1})^{-1} (\lambda^{k+1} - \lambda^k) = \frac{1}{2} \| \lambda^{k+1} - \lambda^* \|^2_{(\rho^{k+1})^{-1}} - \| \lambda^k - \lambda^* \|^2_{(\rho^{k+1})^{-1}},
\]

\[
-\frac{1}{2} \| \lambda^{k+1} - \lambda^k \|^2_{(\rho^{k+1})^{-1}}. \tag{36}
\]

Then we can obtain [33] by plugging equalities [34]-[37] into the left hand side of [33].

Now we can proceed to prove Theorem 4. By setting $x = x^*$ in [30], we can get

\[
f(x^*) - f(x^{k+1}) + (x^* - x^{k+1})^T A^T \lambda^k + (x^* - x^{k+1})^T \cdot A^T \rho^k A \mathbf{x} + (x^* - x^{k+1})^T \bar{Q}(x^{k+1} - x^k) \geq 0.
\]

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

\[
f(x^*) - f(x^{k+1}) - (x^{k+1})^T A^T \lambda^k - (x^* - x^{k+1})^T \bar{Q}(x^{k+1} - x^k) \geq 0. \tag{38}
\]

Now adding and subtracting the term $\lambda^T A \mathbf{x}^{k+1}$ from the left hand side of [33] gives

\[
f(x^*) - f(x^{k+1}) - \lambda^T A \mathbf{x}^{k+1} - (x^{k+1})^T A^T (\lambda^k - \lambda^*) - (x^* - x^{k+1})^T \bar{Q}(x^{k+1} - x^k) \geq 0.
\]

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

\[
-\frac{1}{2} (\| \lambda^{k+1} - \lambda^* \|^2_{(\rho^{k+1})^{-1}} - \| \lambda^k - \lambda^* \|^2_{(\rho^{k+1})^{-1}}) + \frac{1}{2} \| \lambda^{k+1} - \lambda^k \|^2_{(\rho^{k+1})^{-1}} + \frac{1}{2} \| A \mathbf{x}^{k+1} - A \mathbf{x}^k \|^2_{\rho^k} - \frac{1}{2} \| A \mathbf{x}^{k+1} \|^2_{\rho^k} - \frac{1}{2} \| A \mathbf{x}^k \|^2_{\rho^k} - \frac{1}{2} \| \lambda^{k+1} - \lambda^* \|^2_{(\rho^{k+1})^{-1}} - \| \lambda^k - \lambda^* \|^2_{(\rho^{k+1})^{-1}} - \| x^{k+1} - x^* \|^2_{\bar{Q}} - \frac{1}{2} \| x^{k+1} - x^k \|^2_{\bar{Q}} \geq 0.
\]

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

\[
\| \lambda^{k+1} - \lambda^k \|^2_{(\rho^{k+1})^{-1}} + \| x^{k+1} - x^* \|^2_{\bar{Q}} - \| x^k - x^* \|^2_{\bar{Q}} \leq \| \lambda^{k+1} - \lambda^* \|^2_{(\rho^{k+1})^{-1}} + \| x^{k+1} - x^* \|^2_{\bar{Q}} - \frac{1}{2} \| A \mathbf{x}^{k+1} \|^2_{\rho^k} - \| A \mathbf{x}^k \|^2_{\rho^k} - \frac{1}{2} \| A \mathbf{x}^{k+1} - A \mathbf{x}^k \|^2_{\rho^k}.
\]

Recall that from Condition A, $\rho^{k+1} \geq \rho^k$ and $\rho^k$ ($k = 1, 2, \ldots$) are positive definite diagonal matrices. So we have [43]

\[
(\rho^{k+1})^{-1} \leq (\rho^k)^{-1}
\]

and consequently

\[
\| \lambda^{k+1} - \lambda^* \|^2_{(\rho^{k+1})^{-1}} \leq \| \lambda^k - \lambda^* \|^2_{(\rho^k)^{-1}},
\]

which proves Theorem 4.

Theorem 5. Let $u^k = (x^k, \lambda^k)$ be the sequence generated by Algorithm I, then we have

\[
\lim_{k \to \infty} \| A \mathbf{x}^{k+1} \|^2_{\rho^{k+1}} + \| x^{k+1} - x^k \|^2_{A^T \rho^k A + \bar{Q}} = 0. \tag{40}
\]

Proof: Let

\[
\alpha^k = \| \lambda^k - \lambda^* \|^2_{(\rho^k)^{-1}} + \| x^k - x^* \|^2_{\bar{Q}}.
\]

According to Theorem 4 we have

\[
\alpha^k \leq \alpha^k + \| A \mathbf{x}^{k+1} \|^2_{\rho^{k+1}} - \| A \mathbf{x}^k \|^2_{\rho^k} - (\| A \mathbf{x}^{k+1} \|^2_{\rho^k} + \| x^{k+1} - x^k \|^2_{A^T \rho^k A + \bar{Q}}) \leq \cdots \leq \alpha^k + \| A \mathbf{x}^{k+1} \|^2_{\rho^{k+1}} - \| A \mathbf{x}^k \|^2_{\rho^k} - \sum_{i=0}^{k-1} \| A \mathbf{x}^{i+1} \|^2_{\rho^{i+1}} \leq \alpha^k + \| A \mathbf{x}^{k+1} \|^2_{\rho^{k+1}} - \| A \mathbf{x}^k \|^2_{\rho^k} - \sum_{i=0}^{k-1} \| A \mathbf{x}^{i+1} \|^2_{\rho^{i+1}} + \| x^{i+1} - x^i \|^2_{A^T \rho^i A + \bar{Q}}.
\]

Recall $\rho^0 \leq \rho^k \leq \rho^{k+1} \leq \bar{\rho}$. We have $\rho^k = \rho^{k+1}$ when $k \to \infty$ and consequently

\[
\lim_{k \to \infty} \| A \mathbf{x}^{k+1} \|^2_{\rho^{k+1}} = \| A \mathbf{x}^{k+1} \|^2_{\rho^k}.
\]

Then following Theorem 3 in [43], we have

\[
\lim_{k \to \infty} \| A \mathbf{x}^{k+1} \|^2_{\rho^k} + \| x^{k+1} - x^k \|^2_{A^T \rho^k A + \bar{Q}} = 0.
\]

Given that $\rho^k$ satisfies Condition A and $\bar{Q}$ satisfies Condition B, we have that both $-A^T \rho^k A + \bar{Q}$ and $\rho^k$ are positive symmetric definite. Then according to Theorem 5 we have $A \mathbf{x}^k = 0$ and $x^{k+1} = x^k$ when $k \to \infty$.

Therefore, based on Lemma 2 we have that $(x^{k+1}, \lambda^{k+1})$ in Algorithm I converges to a solution to $MV^I(Q, U)$, i.e., a saddle point of the Lagrangian function (13) according to Lemma 1. Since the objective function is strongly convex, we can conclude Theorem 1 [45].
[51] J. Bethencourt. Advanced crypto software collection. *URL: http://acsc.cs.utexas.edu/libpallier.*