Privacy Preservation in Distributed Optimization via Dual Decomposition and ADMM

Tjell, Katrine ; Wisniewski, Rafal

Published in:
2019 IEEE 58th Conference on Decision and Control (CDC)

DOI (link to publication from Publisher):
10.1109/CDC40024.2019.9028969

Publication date:
2020

Document Version
Accepted author manuscript, peer reviewed version

Privacy Preservation in Distributed Optimization via Dual Decomposition and ADMM

Katrine Tjell* and Rafael Wisniewski*

***Abstract*— In this work, we explore distributed optimization problems, as they are often stated in energy and resource optimization. More precisely, we consider systems consisting of a number of subsystems that are solely connected through linear constraints on the optimized solutions. The focus is put on two approaches; namely dual decomposition and *alternating direction method of multipliers (ADMM)*, and we are interested in the case where it is desired to keep information about subsystems secret. To this end, we propose a privacy preserving algorithm based on secure multiparty computation (SMPC) and secret sharing that ensures privacy of the subsystems while converging to the optimal solution. To gain efficiency in our method, we modify the traditional ADMM algorithm.**

I. INTRODUCTION

Developments in technology have enabled efficient collection of data and ensured powerful signal transmissions. This is beneficial in many cases since collected data has the potential of aiding accurate estimations and predictions as well as the calculation of solutions customized to individuals. Take for instance electricity production and pressure control in water networks as examples of control systems with a potential of reducing resource losses by accurately predicting the consumption through fine grained measurements. Another example could be the future smart house that will adapt more and more to the occupants in pace with the more data it collects. To this end, collected data is valuable, however, more often than not it contains privacy sensitive information. For instance, the activities of occupants in a household can be inferred from electricity consumption measurements as showed in [1]. Most people are cautious to share this kind of information; hence, they will also be reluctant to share their consumption data. The idea, thus, is to develop methods that can exploit the benefits of the data without risking the privacy of individuals.

The focus in this paper is on distributed optimization where several agents each hold a part of the optimization problem. This kind of problem appears in many applications, take for instance resource allocation between subsystems, formation control of autonomous vehicles and distributed resource generation.

The problem in focus can be described as a system containing N agents that each have a private objective. Moreover, the overall system describes a set of linear constraints that couples the N individual optimization problems. To exemplify, the agents could be power production units and a constraint could be that the sum of produced power meet a certain requirement or that the sum of resources used may not exceed a given limit. Each agent is unwilling to share information concerning their system, so the individual objectives as well as optimized solutions must be kept private. Furthermore, the constraints must also remain secret as they may reveal information about the individual systems. Specifically, we assume that the constraints are designed by a *superintendent* and will only be used in encrypted form.

We study distributed optimization via dual decomposition and Alternating Direction Method of Multipliers (ADMM) and propose privacy preserving solutions. In particular, we put forward a method based on Secure Multiparty Computation (SMPC) and cloud computing, that minimizes the objective while leaking no information about the system.

Related work. The number of papers focusing on privacy preserving control and optimization is increasing. Zhang et al.'s work [2] is closely related to ours as they also consider an ADMM based privacy preserving distributed optimization solution. However, they allow every agent to communicate with their neighbour using homomorphic encryption, which is in contrast to our solution, where the agents communicate only with a number of so-called computing parties using secure multiparty computation. We believe that this solution scales better with the number of agents and is computationally less demanding. Many recent papers within this subject are based on homomorphic encryption, for instance [3] and [4]. There is also a substantial amount of work, basing the privacy preservation on differential privacy, for instance [5], [6], and [7]. This approach requires at least some trust in the system, which our solution does not exhibit.

Also of interest to our work is [8] and [9] that investigates how to preserve privacy and integrity in cloud computations.

Structure. In Section II, the specific problem studied throughout the paper is presented. Subsequently, Section III gives an introduction to the two building blocks of the paper, namely SMPC and secret sharing. In the Sections IV and V, we introduce the main contribution, which is privacy preserving distributed optimization. In Section IV, our ideas are presented using a dual decomposition approach. The main purpose of this section is to provide intuition about the methods, since the ADMM based approach presented in Section V is more likely to have a practical relevance. Finally, a discussion is provided in Section VI.

II. PROBLEM FORMULATION

In the paper, we consider the setup consisting of N agents, $A_1, \ldots, A_N$, each having a local objective function

*Faculty of IT and Design, Department of Electronic Systems, Automation and Control, Aalborg University, Denmark,
{kst,raf}@es.aau.dk

$f_1(\boldsymbol{x}_1), \ldots, f_N(\boldsymbol{x}_N)$ and a *superintendent* having requirements for the agents. Remark, we assume that $f_i(\boldsymbol{x}_i)$ is known only to agent i and that the constraints are known only to the superintendent. The agents want to minimize their individual objective function, while the superintendent wants the result achieved by the agents to fulfill certain requirements. These requirements are formulated as M linear constraints on the form

$$\sum_{i=1}^{N} \boldsymbol{B}_i \boldsymbol{x}_i = \boldsymbol{c}, \tag{1}$$

where $\boldsymbol{x}_i \in \mathbb{R}^q$, $\boldsymbol{B}_i \in \mathbb{R}^{M \times q}$, and $\boldsymbol{c} \in \mathbb{R}^{M \times 1}$. The overall problem can thus be stated as

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) \\ \text{subject to} \quad & \sum_{i=1}^{N} \boldsymbol{B}_i \boldsymbol{x}_i - \boldsymbol{c} = \boldsymbol{0}, \end{aligned} \tag{2}$$

where $\boldsymbol{x} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$.

Concerning the approach to solving (2), we are interested in the case where each agent are not willing to share information about their system with the other agents or the public in general. A reason for this could for instance be that the system information is considered corporate secrets. At the same time, we assume that the agents are honest in the sense that they will follow computational instructions and not tamper with any intermediate results. Nonetheless, they may attempt to disclose the secret information of other participants and our methods must prevent this.

The goal in the paper is to solve (2), without leaking any private information. We base the methods on SMPC and secret sharing. However, one could exchange SMPC with homomorphic encryption in our protocols. This would entail less communication but more computation, thus the choice is application dependent. A short introduction to SMPC and secret sharing is provided in the following section, and the reader is referred to [10] for a more elaborate explanation.

III. SECURE MULTIPARTY COMPUTATION BASED ON SECRET SHARING

Consider the evaluation of a function $f(x_1, \ldots, x_n)$, where each of the inputs are provided by a distinct party, say $P_1, \ldots, P_n$. The parties want to learn the output of the function, nonetheless, they are not willing to reveal their individual inputs.

This scenario is the central problem in SMPC, that has the aim of developing protocols that allow the parties to evaluate the function without disclosing their private information. This is achieved by applying encryption techniques to hide the secret data. In particular, secret sharing methods are popular to use as the encryption technique in SMPC, since this usually entails protocols with a low computational complexity.

Secret sharing aims to avoid the situation where one entity holds a secret, since in case of an attack, the adversary then has to attack multiple entities to learn the secret. The secret is "chunked" into pieces that can be distributed among several parties. One piece of information is referred to as a *share* and by itself it reveals nothing about the secret. Specifically, it takes some or all shares to recreate the secret. In Definition 1, we give a definition of a secret sharing scheme.

Definition 1 (Secret Sharing Scheme): A secret sharing scheme consists of two algorithms namely Share and Reconstruct. The first one takes a secret s and creates the *shares*, which are values $s_1, \ldots, s_n$, where n is the number of parties. The latter one outputs s upon given any set of at least t shares, where t is the threshold for the scheme. Additionally, it holds that no information about the secret can be gained from a set of fewer than t shares.

Since a set of fewer than t shares reveals nothing about the secret, the threshold can be used to adopt the protocol to the expectation of so-called *corrupted* parties. If a set of parties collude in inferring information about private values of other parties they are referred to as corrupted parties. Moreover, we distinguish between corrupted parties that follow the protocol and ones that do not. The former is referred to passive corruption, while the latter is referred to as active.

We use $\{\cdot\}$ to denote the secret sharing scheme, i.e., $\{s\}$ is the shares $(s_1, \ldots, s_n)$ of the secret s.

The particular choice of the algorithm Share in the SMPC protocol, determines the computations that can be done on the secrets. In this paper, one can use any linear secret sharing scheme that takes an integer secret modulo some prime p and outputs integer shares also modulo p. Moreover, the scheme should satisfy that a shared version of the sum $\{x + y\}$ and product $\{xy\}$ modulo p, can be computed given $\{x\}$ and $\{y\}$. One can for instance use *Shamir's secret sharing scheme*, for which the above mentioned properties hold when the threshold $t < \frac{n}{2}$.

To give an overview, the privacy preserving protocols proposed in this paper works by hiding secret values using the Share algorithm. A caveat is that, as mentioned, secret values are integers modulo a prime p. We can choose p large enough, so that we do not have wrap-arounds, but eventually the values $\boldsymbol{x}, \boldsymbol{B}$ and $\boldsymbol{c}$ needs to be truncated before Share can be used to create shares of them. Scaling can be used to minimize the truncation error, however, it is unavoidable not to induce quatization errors. We will not touch upon this issue any further.

The desired output is computed as each party alternately distributes the shares among the parties and performs local operations on the shares. At the end, all parties can use the Reconstruct algorithm to learn the output.

In our case, the aforementioned parties can be the agents themselves or to reduce communication, independent computing parties (for instance cloud servers), can be applied to perform the calculations. For the latter choice, efficient frameworks such as ABY3 and SecureML introduced by Mohassel and Rindal in [11] and Mohassel and Zhang in [12] respectively, can be employed. The number of computing parties are three in the case of ABY3 and two in the case of SecureML, and security is achieved against one actively

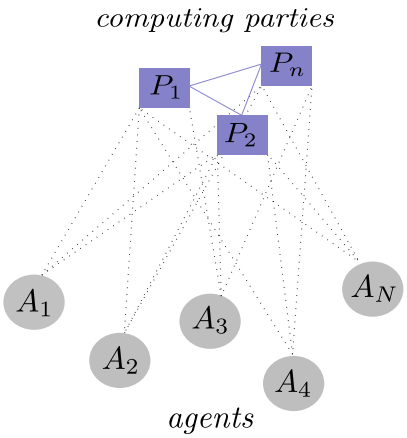

Fig. 1. The N agents connected to the n computing parties.

corrupted server.

For this work, we consider the case where n computing parties perform the SMPC, but one can adapt the protocols to the other case as well. That is, only the computing parties will do computations on shares, and the agents and superintendent will only provide inputs to the protocol. Fig. 1 gives an illustration of how the agents are connected to the computing parties while the computing parties are all connected to each other.

A. Preliminaries

As already mentioned, we will use $\{x\}$ to denote the shares of the value x. When we say that a number of parties *hold* $\{x\}$, we mean that each party has one share of x. In continuation, when a number of parties hold $\{\boldsymbol{x}\}$ and $\{\boldsymbol{X}\}$, it means that each party has a share of each of the entries in the vector $\boldsymbol{x}$, respectively the matrix $\boldsymbol{X}$. Furthermore, the following is a recap of the assumptions we make.

- The agents are honest but curious. That is, they will follow instructions but may attempt to disclose information. However, we assume the agents do not collude.
- A majority of the computing parties are honest and not colluding. If active attacks on the computing parties are expected, one can for instance use the techniques presented in [13] to prevent this, otherwise one can use for instance Shamir's Secret sharing scheme to ensure privacy against a passive adversary.
- The superintendent is honest and not colluding with any other entity.
- Only A_i knows f_i.
- Only the superintendent knows the constraints.

IV. DUAL DECOMPOSITION BASED PRIVACY PRESERVING OPTIMIZATION

In this section, we use the idea in dual decomposition to solve (2). In the following, we merely present the standard dual decomposition algorithm, thus the interested reader is referred to [14] for a more thorough introduction to the subject.

For (2), the Lagrangian is

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = \sum_{i=1}^{N} (f_i(\boldsymbol{x}_i)) + \boldsymbol{\lambda}^\top (\boldsymbol{B}\boldsymbol{x} - \boldsymbol{c}), \tag{3}$$

where $\boldsymbol{B} = [\boldsymbol{B}_1, \ldots, \boldsymbol{B}_N]$. However, it can equivalently be written as

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = \sum_{i=1}^{N} \left(f_i(\boldsymbol{x}_i) + \boldsymbol{\lambda}^\top \boldsymbol{B}_i \boldsymbol{x}_i \right) - \boldsymbol{\lambda}\boldsymbol{c}. \tag{4}$$

The idea in dual decomposition is then to minimize the Lagrangian by solving N sub problems, one for each $\boldsymbol{x}_i$. This is done in iterations, where subsequently the dual variables are updated. The algorithm can be described by the following to steps, where the first is performed in parallel by each agent i:

$$\begin{aligned} \boldsymbol{x}_i^{k+1} &= \min_{\boldsymbol{x}_i} f_i(\boldsymbol{x}_i) + \boldsymbol{\lambda}^{k\top} \boldsymbol{B}_i \boldsymbol{x}_i \\ \boldsymbol{\lambda}^{k+1} &= \boldsymbol{\lambda}^k + \alpha(\boldsymbol{B}\boldsymbol{x}^{k+1} - \boldsymbol{c}), \end{aligned} \tag{5}$$

where $\alpha > 0$ is a step size and $\boldsymbol{x}^k = [\boldsymbol{x}_1^{k\top}, \ldots, \boldsymbol{x}_N^{k\top}]^\top$. In the following section, SMPC and cloud computing are employed to introduce a privacy preserving version of dual decomposition, where $\boldsymbol{x}^k$, $\boldsymbol{B}$ and $\boldsymbol{c}$ remain secret during protocol execution.

A. Privacy Preserving Optimization using Dual Decomposition

We start by make a couple of remarks. The first one is that by sampling a uniform random matrix $\boldsymbol{T}$ and applying it on the constraints, i.e.,

$$\boldsymbol{T}\boldsymbol{B}\boldsymbol{x} - \boldsymbol{T}\boldsymbol{c} = \boldsymbol{0},$$

yields the same constraints, but they are now masked by a random matrix. This will only work if $\boldsymbol{T}$ is non-singular. A singular $\boldsymbol{T}$ can almost always be avoided by constructing it in a special way. This result is stated in Lemma 1, which follows from application of Theorem 2 in [15].

Lemma 1: Let K be a finite field with cardinality $|K|$. Suppose $\boldsymbol{T} = \boldsymbol{U}\boldsymbol{L}$, where

$$\boldsymbol{U} = \begin{bmatrix} 1 & u_2 & u_3 & \ldots & u_n \\ 0 & u_1 & u_2 & \ldots & u_{n-1} \\ \ldots & & & & \\ 0 & 0 & 0 & \ldots & u_2 \\ 0 & 0 & 0 & \ldots & 1 \end{bmatrix},$$

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ v_2 & 1 & 0 & \ldots & 0 \\ v_3 & v_2 & 1 & \ldots & 0 \\ \ldots & & & & \\ v_n & v_{n-1} & v_{n-2} & \ldots & 1 \end{bmatrix}$$

are $K^{n\times n}$ Toeplitz matrices with independent uniformly distributed random entries. Then

$$\mathbb{P}[\det(\boldsymbol{T}) \neq 0] \geq 1 - \frac{n(n+1)}{|K|}.$$

We propose to create $\boldsymbol{T}$ using Lemma 1 and a preprocessing phase that the computing parties will run before protocol

execution, i.e., each party will obtain a share of each entry in $\boldsymbol{T}$ and no party will know the actual $\boldsymbol{T}$. For details about how $\boldsymbol{T}$ is created in a preprocessing phase, we refer to [10]. After $\boldsymbol{T}$ is created, the computing parties must securely check the determinant of $\boldsymbol{T}$, which can be done directly on the shares as shown in [16]. If the determinant is zero they will start over, thus ensuring a non-singular $\boldsymbol{T}$ at the beginning of protocol execution.

The second remark is that the minimization of agent i can be written as

$$\boldsymbol{x}_i^{k+1} = \min_{\boldsymbol{x}_i} f_i(\boldsymbol{x}_i) + \boldsymbol{v}_i^{k\top}\boldsymbol{x}_i, \tag{6}$$

where

$$\boldsymbol{v}_i^{k\top} = \boldsymbol{\lambda}^{k\top}\boldsymbol{T}\boldsymbol{B}_i. \tag{7}$$

The privacy preserving algorithm will then alternate between the agents solving their individual minimization problem in parallel and the computing parties doing SMPC. When each agent has solved its problem, it will create shares of the result and send one share to each of the computing parties. Each computing party will also receive a share of $\boldsymbol{B}$ and $\boldsymbol{c}$ from the superintendent, meaning that the parties can compute $\boldsymbol{v}_i^k$, by performing computations directly on shares. In order to avoid having the computing parties working with α, which is not an integer, we introduce an integer $d > 1$ and let $\alpha = \frac{1}{d}$. The computing parties will then compute $\tilde{\boldsymbol{v}}_i^k = d\boldsymbol{v}_i^k$.

Each agent i will receive $\tilde{\boldsymbol{v}}_i^k$ from the computing parties where after it can compute $\boldsymbol{v}_i^k = \frac{1}{d}\tilde{\boldsymbol{v}}_i^k$. It is of course vital that $\boldsymbol{v}_i^k$ does not leak information allowing agent i to learn either $\boldsymbol{B}$, $\boldsymbol{c}$, or $\boldsymbol{x}_j$ for $j \neq i$. We provide a proof that $\boldsymbol{v}_i^k$ does not leak information in the following.

Lemma 2: Disclosing $\boldsymbol{v}_i^k$ to agent i at time k does not leak information.

Proof: Because $\boldsymbol{T}$ is applied on $\boldsymbol{B}$, each entry in $\boldsymbol{v}_i^k$ is a sum of uniformly distributed random variables. Thus, agent i does not learn more from $\boldsymbol{v}_i^k$ than he could from drawing numbers from a uniform distribution. ∎

The protocol is written formally in Protocol 1.

We now expand the solution presented in this section, to a solution based on the ADMM algorithm.

V. ALTERNATING DIRECTION METHOD OF MULTIPLIERS

The ADMM method is based on the *augmented* Lagrangian, which entails that the ADMM has convergence advantages over dual decomposition. We refer the reader to [14] for an introduction to ADMM; here, we merely present the algorithm. In regards to this, we note that the standard ADMM problem is on the form

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f(\boldsymbol{x}) + g(\boldsymbol{z}) \\ \text{subject to} \quad & \boldsymbol{A}_1\boldsymbol{x} + \boldsymbol{A}_2\boldsymbol{z} = \boldsymbol{b}, \end{aligned} \tag{11}$$

and the augmented Lagrangian for this problem is

$$\begin{aligned} L_\rho(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) =& f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{\lambda}^\top(\boldsymbol{A}_1\boldsymbol{x} + \boldsymbol{A}_2\boldsymbol{z} - \boldsymbol{b}) \\ &+ \frac{1}{2}\rho||\boldsymbol{A}_1\boldsymbol{x} + \boldsymbol{A}_2\boldsymbol{z} - \boldsymbol{b}||_2^2. \end{aligned} \tag{12}$$

Protocol 1 Privacy Preserving Dual Decomposition Method

1: The computing parties hold $\{\boldsymbol{T}\}$, where $\{\boldsymbol{T}\}$ is a uniformly random $M \times M$ matrix obtained in a preprocessing phase.
2: The computing parties hold $\{\boldsymbol{B}\}$ and $\{\boldsymbol{c}\}$, received from the super intended.
3: $d > 1$ is an integer.
4: $\alpha = \frac{1}{d}$, $\boldsymbol{v}^0 = \boldsymbol{0}$, $\boldsymbol{\lambda}^0 = \boldsymbol{0}$.
5: **for all** $k = 0, \ldots$ **do**
6: Each agent i computes

$$\boldsymbol{v}_i^k = \frac{1}{d}\tilde{\boldsymbol{v}}_i^k. \tag{8}$$

after receiving all shares of $\tilde{\boldsymbol{v}}_i^k$ from the computing parties.
7: Each agent i computes $\boldsymbol{x}_i^{k+1}$ by solving (6).
8: Each agent i creates shares of $\boldsymbol{x}_i^{k+1}$ and send one share to each of the computing parties.
9: The computing parties compute jointly:

$$\{\boldsymbol{\lambda}^{k+1}\} = \{\boldsymbol{\lambda}^k\} + (\{\boldsymbol{T}\}\{\boldsymbol{B}\} - \{\boldsymbol{T}\}\{\boldsymbol{c}\})\{\boldsymbol{x}^{k+1}\}, \tag{9}$$

and

$$\{\tilde{\boldsymbol{v}}_i^{k+1^\top}\} = \{\boldsymbol{\lambda}^{k+1^\top}\}\{\boldsymbol{T}\}\{\boldsymbol{B}_i\}. \tag{10}$$

for $i = 1, \ldots, N$.
10: The computing parties send all shares of $\{\tilde{\boldsymbol{v}}_i\}$ to agent i for $i = 1, \ldots, N$.

Consequently, the ADMM algorithm consists of the following steps

$$\begin{aligned} \boldsymbol{x}^{k+1} &= \min_x L_\rho(\boldsymbol{x}, \boldsymbol{z}^k, \boldsymbol{\lambda}^k) \\ \boldsymbol{z}^{k+1} &= \min_z L_\rho(\boldsymbol{x}^{k+1}, \boldsymbol{z}, \boldsymbol{\lambda}^k) \\ \boldsymbol{\lambda}^{k+1} &= \boldsymbol{\lambda}^k + \rho(\boldsymbol{A}_1\boldsymbol{x}^{k+1} + \boldsymbol{A}_2\boldsymbol{z}^{k+1} - \boldsymbol{b}), \end{aligned} \tag{13}$$

where $\rho > 0$.

The problem in (2) is almost on the standard ADMM form, the main difference is that instead of two blocks in (11), there are N blocks in (2). This modification is important to avoid communication among agents. The augmented Lagrangian for (2) is

$$L_\rho(\boldsymbol{x}, \boldsymbol{\lambda}) = \sum_{i=1}^N (f_i(\boldsymbol{x}_i)) + \boldsymbol{\lambda}^\top(\boldsymbol{B}\boldsymbol{x} - \boldsymbol{c}) + \frac{1}{2}\rho||\boldsymbol{B}\boldsymbol{x} - \boldsymbol{c}||_2^2. \tag{14}$$

Depending on the size of N, performing the sub optimizations sequentially may take a long time. Therefore, we propose that each agent solve their individual minimization in parallel, i.e., we propose the modified ADMM algorithm

$$\begin{aligned} \boldsymbol{x}_i^{k+1} &= \min_{\boldsymbol{x}_i} L_\rho(\boldsymbol{x}_1^k, \ldots, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_N^k, \boldsymbol{\lambda}^k) \quad \text{for } i = 1, \ldots, N \\ \boldsymbol{\lambda}^{k+1} &= \boldsymbol{\lambda}^k + \rho\left(\sum_{i=1}^N \boldsymbol{B}_i\boldsymbol{x}_i^{k+1} - \boldsymbol{c}\right). \end{aligned} \tag{15}$$

Unfortunately, this approximation of the ADMM algorithm is likely to diverge unless precautions are taken. One approach is proposed (and proved) by [17] and involves adding an underrelaxation step, such that the algorithm becomes the following;

$$\begin{aligned}
\tilde{\boldsymbol{x}}_i^{k+1} &= \min_{\boldsymbol{x}_i} L_\rho(\boldsymbol{x}_1^k, \ldots, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_N^k, \boldsymbol{\lambda}^k) \quad \text{for } i = 1, \ldots, N \\
\tilde{\boldsymbol{\lambda}}^{k+1} &= \boldsymbol{\lambda}^k + \rho \left(\sum_{i=1}^N \boldsymbol{B}_i \tilde{\boldsymbol{x}}_i^{k+1} - \boldsymbol{c} \right). \\
\boldsymbol{x}_i^{k+1} &= \boldsymbol{x}_i^k - \frac{1}{N+1}(\boldsymbol{x}^k - \tilde{\boldsymbol{x}}_i^{k+1}) \\
\boldsymbol{\lambda}^{k+1} &= \boldsymbol{\lambda}^k - \frac{1}{N+1}(\boldsymbol{\lambda}^k - \tilde{\boldsymbol{\lambda}}^{k+1}).
\end{aligned} \tag{16}$$

A. Privacy Preserving ADMM

For obtaining a privacy preserving ADMM protocol, we use the same principles as we did in the privacy preserving dual decomposition method. Namely, we mask $\boldsymbol{B}$ and $\boldsymbol{c}$ with a uniformly random matrix $\boldsymbol{T}$, such that the masked constraints yields

$$\boldsymbol{TBx} = \boldsymbol{Tc}. \tag{17}$$

Next, we rewrite the minimization problem of each agent. To do this, consider the minimization of (14), where we focus on $\boldsymbol{\lambda}^\top(\boldsymbol{Bx}-\boldsymbol{c}) + \frac{\rho}{2}||\boldsymbol{Bx}-\boldsymbol{c}||_2^2$ since these terms cannot be computed locally. We consider the minimization with respect to only one element, x_e, in the vector $\boldsymbol{x}$.

$$\begin{aligned}
&\min_{x_e} \boldsymbol{\lambda Bx} + \frac{\rho}{2}\left(\sum_{j=1}^M (\boldsymbol{b_j x} - c_j)^2 \right) \\
=&\min_{x_e} \sum_{j=1}^M \lambda_j b_{j,e} x_e + \frac{\rho}{2}\left(\sum_{j=1}^M \boldsymbol{b_j x x^\top b_j^\top} - 2\boldsymbol{b_j x} c_j \right) \\
=&\min_{x_e} \sum_{j=1}^M \lambda_j b_{j,e} x_e + \frac{\rho}{2}\sum_{j=1}^M b_{j,e}^2 {x_e}^2 \\
&+ \frac{\rho}{2} x_e \sum_{j=1}^M \sum_{k=1,k\neq e}^N 2 b_{j,e} b_{j,k} x_k - \frac{\rho}{2} x_e \sum_{j=1}^M 2 b_{j,e} c_j \\
=&\min_{x_e} {x_e}^2 \frac{\rho}{2} \sum_{j=1}^M b_{j,e}^2 + \\
&x_e \sum_{j=1}^M b_{j,e} \left(\lambda_j - \rho c_j + \rho \sum_{k=1,k\neq e}^N b_{j,k} x_k \right) \\
=&\min_{x_e} \beta_{1,e} {x_e}^2 + \beta_{2,e} x_e,
\end{aligned} \tag{18}$$

where $\boldsymbol{b_j}$ is the j'th row of $\boldsymbol{B}$, $b_{j,e}$ is the j,e'th element of $\boldsymbol{B}$ and

$$\begin{aligned}
\beta_{1,e} &= \frac{\rho}{2}\sum_{j=1}^M b_{j,e}^2 \\
\beta_{2,e} &= \sum_{j=1}^M b_{j,e} \left(\lambda_j - \rho c_j + \rho \sum_{k=1,k\neq e}^N b_{j,k} x_k \right).
\end{aligned} \tag{19}$$

Most of the reductions in (18) are because terms that are constant with respect to x_e does not affect the minimization and can be disregarded.

Define $\boldsymbol{s}_{t,j} = [\beta_{t,(j-1)q+1}, \beta_{t,(j-1)q+2}, \ldots, \beta_{t,(j-1)q+q}]$ for $j = 1, \ldots, N$ and $t = 1, 2$. To improve readability, $s_{1,j}$ and $s_{2,j}$ are illustrated in the following.

$$\begin{array}{cc}
\left.\begin{matrix} \beta_{1,1} \\ \vdots \\ \beta_{1,q} \end{matrix}\right\} \boldsymbol{s}_{1,1} & \left.\begin{matrix} \beta_{2,1} \\ \vdots \\ \beta_{2,q} \end{matrix}\right\} \boldsymbol{s}_{2,1} \\
\left.\begin{matrix} \beta_{1_{q+1}} \\ \vdots \\ \beta_{1_{2q}} \end{matrix}\right\} \boldsymbol{s}_{1_2} & \left.\begin{matrix} \beta_{2_{q+1}} \\ \vdots \\ \beta_{2_{2q}} \end{matrix}\right\} \boldsymbol{s}_{2_2} \\
\vdots & \vdots \\
\left.\begin{matrix} \beta_{1,(N-1)q+1} \\ \vdots \\ \beta_{1,Nq} \end{matrix}\right\} \boldsymbol{s}_{1,N} & \left.\begin{matrix} \beta_{2,(N-1)q+1} \\ \vdots \\ \beta_{2,Nq} \end{matrix}\right\} \boldsymbol{s}_{2,N}
\end{array} \tag{20}$$

The optimization problem of node i is now formulated as;

$$\tilde{\boldsymbol{x}}_i^k = \min_{\boldsymbol{x}_i} f_i(\boldsymbol{x}_i) + \text{diag}(\{\boldsymbol{s}_{1,i}\})\boldsymbol{x}_i^2 + \text{diag}(\{\boldsymbol{s}_{2,i}\})\boldsymbol{x}_i, \tag{21}$$

where $\text{diag}(\boldsymbol{x})$ is a matrix with $\boldsymbol{x}$ on the diagonal and 0 everywhere else.

The privacy preserving algorithm will then operate by at each time k, each agent i will calculate $\tilde{\boldsymbol{x}}_i^k$ by solving (21) upon receiving $\boldsymbol{s}_{1,i}^k$ and $\boldsymbol{s}_{2,i}^k$ from the computing parties. The agents then create shares of $\tilde{\boldsymbol{x}}_i^k$ and sends one share to each computing party. The computing parties will calculate $\boldsymbol{s}_{1_i}^{k+1}$ and $\boldsymbol{s}_{2,i}^{k+1}$ for $i = 1, \ldots, n$ by operating directly on the shares. The algorithm will obviously only be privacy preserving if $\boldsymbol{s}_{1,i}^{k+1}$ and $\boldsymbol{s}_{2,i}^{k+1}$ does not leak information. We provide a proof of this immediately.

Lemma 3: Disclosing $\boldsymbol{s}_{1,i}^k$ and $\boldsymbol{s}_{2,i}^k$ to agent i does not disclose information.

Proof: Since $\boldsymbol{T}$ is a uniformly random matrix, and the sum and product of two uniform random variables are uniformly distributed, [10] page 254, it can be verified that all terms in $\boldsymbol{s}_{1,i}^k$ and $\boldsymbol{s}_{2,i}^k$ are uniformly distributed random variables. Thus, that agent i learns these two values does not reveal information. ■

In Protocol 2, the privacy preserving protocol for the modified ADMM algorithm is stated formally.

Protocol 2 Privacy Preserving ADMM

1: The computing parties hold $\{\boldsymbol{T}\}$, where $\{\boldsymbol{T}\}$ is a uniformly random $M \times M$ matrix obtained in a preprocessing phase.
2: The computing parties hold $\{\boldsymbol{B}\}$ and $\{\boldsymbol{c}\}$, received from the super intended.
3: $\rho > 1$ is an integer.
4: $\hat{\boldsymbol{B}} = \boldsymbol{TB}$, $\hat{\boldsymbol{c}} = \boldsymbol{Tc}$, $\boldsymbol{\lambda}^0 = \tilde{\boldsymbol{\lambda}}^0 = \boldsymbol{0}$, $\boldsymbol{x} = \tilde{\boldsymbol{x}} = \boldsymbol{0}$.
5: $\boldsymbol{s}_{1,i}^0 = \boldsymbol{0}$ and $\boldsymbol{s}_{2,i}^0 = \boldsymbol{0}$ for $i = 1, \ldots, N$.
6: **for all** $k = 0, \ldots$ **do**
7: Each agent i receives $\boldsymbol{s}_{1_i}^k$ and $\tilde{\boldsymbol{s}}_{2,i}^k$.
8: Each agent i computes $\boldsymbol{s}_{2,i}^k = \frac{1}{d}\tilde{\boldsymbol{s}}_{2,i}^k$ and $\tilde{\boldsymbol{x}}_i^{k+1}$ by solving (21).
9: Each agent i creates shares of $\tilde{\boldsymbol{x}}_i^{k+1}$ and send one share to each computing party.
10: The computing parties compute jointly:

$$\{\tilde{\boldsymbol{\lambda}}^{k+1}\} = \{\boldsymbol{\lambda}^k\} + \rho(\{\hat{\boldsymbol{B}}\}\{\tilde{\boldsymbol{x}}^{k+1}\} - \{\hat{\boldsymbol{c}}\}). \quad (22)$$

11: The computing parties compute

$$\begin{aligned} \{\boldsymbol{x}_i^{k+1}\} &= \{\boldsymbol{x}_i^k\} - \frac{(\{\boldsymbol{x}^k\} - \{\tilde{\boldsymbol{x}}_i^{k+1}\})}{N+1} \\ \{\boldsymbol{\lambda}^{k+1}\} &= \{\boldsymbol{\lambda}^k\} - \frac{(\{\boldsymbol{\lambda}^k\} - \{\tilde{\boldsymbol{\lambda}}^{k+1}\})}{N+1}. \end{aligned} \quad (23)$$

and $\{\boldsymbol{s}_{1,i}^{k+1}\}$ and $\{\boldsymbol{s}_{2,i}^{k+1}\}$ by using (20) and

$$\begin{aligned} \{\beta_{1,e}\} &= \left\lfloor \frac{\rho}{2} \right\rfloor \sum_{j=1}^{M} \{\hat{b}_{j,e}\}\{\hat{b}_{j,e}\} \\ \{\beta_{2,e}\} &= \sum_{j=1}^{M} \{\hat{b}_{j,e}\} \\ &\quad \left(\{\lambda_j^{k+1}\} - \rho\{\hat{c}_j\} + \rho \sum_{t=1, t\neq e}^{N} \{\hat{b}_{j,t}\}\{x_t^{k+1}\} \right), \end{aligned} \quad (24)$$

for $e = 1, \ldots, Nq$.
12: Each of the computing parties sends their share of $\{\boldsymbol{s}_{1_i}^{k+1}\}$ and $\{\boldsymbol{s}_{2_i}^{k+1}\}$ to agent i for $i = 1, \ldots, N$.

Remark, in (23) we have assumed that there exist SMCP protocols for dividing a secret with a public integer, see [18] for more on this topic.

VI. DISCUSSION

The paper presents a privacy preserving dual decomposition protocol as well as a privacy preserving ADMM protocol. The former serves mostly as a more gentle introduction to the ideas in the paper, while the main contribution is the latter protocol.

Our proposed methods enjoy a decentralized setting, all though a number of so-called *computing parties* are employed to avoid communication between sub-systems. The computing parties are feed exclusively with encrypted values and only by attacking all parties (which we assume is infeasible) can information be learned.

VII. ACKNOWLEDGMENTS

This work is supported by SECURE project at Aalborg University.

REFERENCES

[1] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin, "Private memoirs of a smart meter," in *Proceedings of the 2Nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, (New York, NY, USA), pp. 61–66, ACM, 2010.

[2] C. Zhang, M. Ahmad, and Y. Wang, "Admm based privacy-preserving decentralized optimization," *IEEE Transactions on Information Forensics and Security*, vol. 14, pp. 565–580, March 2019.

[3] Y. Lu and M. Zhu, "Privacy preserving distributed optimization using homomorphic encryption," *Automatica*, vol. 96, pp. 314 – 325, 2018.

[4] Y. Shoukry, K. Gatsis, A. Alanwar, G. J. Pappas, S. A. Seshia, M. Srivastava, and P. Tabuada, "Privacy-aware quadratic optimization using partially homomorphic encryption," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 5053–5058, Dec 2016.

[5] Z. Huang, S. Mitra, and N. Vaidya, "Differentially private distributed optimization," in *Proceedings of the 2015 International Conference on Distributed Computing and Networking*, ICDCN '15, (New York, NY, USA), pp. 4:1–4:10, ACM, 2015.

[6] V. Rostampour, R. Ferrari, A. M. Teixeira, and T. Keviczky, "Differentially-private distributed fault diagnosis for large-scale nonlinear uncertain systems," *IFAC-PapersOnLine*, vol. 51, no. 24, pp. 975 – 982, 2018. 10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 2018.

[7] E. Nozari, P. Tallapragada, and J. Corts, "Differentially private distributed convex optimization via objective perturbation," in *2016 American Control Conference (ACC)*, pp. 2061–2066, July 2016.

[8] Z. Xu and Q. Zhu, "Secure and resilient control design for cloud enabled networked control systems," in *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*, CPS-SPC '15, (New York, NY, USA), pp. 31–42, ACM, 2015.

[9] N. Drucker, S. Gueron, and B. Pinkas, "Faster secure cloud computations with a trusted proxy," *IEEE Security Privacy*, vol. 15, pp. 61–67, November 2017.

[10] R. Cramer, I. B. Damgaard, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 1 ed., 2015.

[11] P. Mohassel and P. Rindal, "Aby3: A mixed protocol framework for machine learning," *IACR Cryptology ePrint Archive*, vol. 2018, p. 403, 2018.

[12] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 19–38, May 2017.

[13] M. Pettai and P. Laud, "Automatic proofs of privacy of secure multi-party computation protocols against active adversaries," in *2015 IEEE 28th Computer Security Foundations Symposium*, pp. 75–89, July 2015.

[14] 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, pp. 1–122, Jan. 2011.

[15] E. Kaltofen and B. D. Saunders, "On wiedemann's method of solving sparse linear systems," in *Proceedings of the 9th International Symposium, on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, AAECC-9, (London, UK, UK), pp. 29–38, Springer-Verlag, 1991.

[16] R. Cramer and I. Damgård, "Secure distributed linear algebra in a constant number of rounds," in *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '01, (London, UK, UK), pp. 119–136, Springer-Verlag, 2001.

[17] B. He, L. Hou, and X. Yuan, "On full jacobian decomposition of the augmented lagrangian method for separable convex programming," *SIAM Journal on Optimization*, vol. 25, no. 4, pp. 2274–2312, 2015.

[18] O. Catrina and S. de Hoogh, "Improved primitives for secure multiparty integer computation," in *Security and Cryptography for Networks* (J. A. Garay and R. De Prisco, eds.), (Berlin, Heidelberg), pp. 182–199, Springer Berlin Heidelberg, 2010.