Private Aggregation with Application to Distributed Optimization

Tjell, Katrine ; Wisniewski, Rafal

Published in:
IEEE Control Systems Letters

DOI (link to publication from Publisher):
10.1109/LCSYS.2020.3041611

Publication date:
2021

Document Version
Accepted author manuscript, peer reviewed version

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2020.3041611, IEEE Control Systems Letters

Private Aggregation with Application to Distributed Optimization*

Katrine Tjell[1], *Student Member, IEEE*, Rafael Wisniewski[1], *Member, IEEE*

***Abstract*— The paper presents a fully distributed private aggregation protocol that can be employed in dynamical networks where communication is only assumed on a neighbor-to-neighbor basis. The novelty of the scheme is its low overhead in communication and computation due to a pre-processing phase that can be executed even before the participants know their input to aggregation. Moreover, the scheme is resilient to node drop-outs, and it is defined without introducing any trusted or untrusted third parties. We prove the privacy of the scheme itself and subsequently, we discuss the privacy leakage caused by the output of the scheme. Finally, we discuss implementation of the proposed protocol to solve distributed optimization problems using two versions of the alternating direction method of multipliers (ADMM).**

***Index Terms*— Distributed control, Information theory and control, Optimization**

I. Introduction

DISTRIBUTED computing is emerging everywhere in fields such as signal processing, control, and machine learning. Concerns about privacy in such distributed systems are arising since typically lots of data are collected and sensitive information is held at the network's nodes. Several works have shown how collected data can be used to identify individuals, and how private information can be inferred from it. For instance, [1] discusses how private information can be derived even though data is randomized.

In this work, we seek to circumvent these privacy issues by proposing a computation framework where data is only used indirectly and will not be exposed. Typically, the price to pay for privacy is an increase in computational complexity and a communication overhead. However, we introduce a pre-processing phase that only involves none-private data and can be executed before or in-between the actual processing phases, resulting in only a minimal increase in computations and no communication overhead.

Considering already existing distributed algorithms, commonly, the nodes exchange certain values and in preceding computations, the sum of these values is used. That is, the algorithms rely on the sum of communicated values and not the individual values, see for instance [2]–[4].

Our proposed method solves the problem of privately computing the sum of values belonging to individual users while not revealing the values. We prove that the protocol itself does not leak information and moreover; we study the information leakage caused by the output of the protocol (the sum). We note that differential privacy techniques could be added on top of our method, to avoid the leakage caused by the output, at the cost of loosing precision of the solution.

We explore the use of our proposed method to achieve privacy in distributed optimization. That is, for nodes $i \in \mathcal{N} = \{0, 1, \dots, N\}$ we assume that node i has a private convex cost function $f_i(\boldsymbol{x}_i)$, and we consider the following minimization problem:

$$\begin{aligned} \underset{\boldsymbol{x}_1,\dots,\boldsymbol{x}_N}{\text{minimize}} \quad & \sum_{i\in\mathcal{N}} f_i(\boldsymbol{x}_i) \\ \text{subject to} \quad & \sum_{i\in\mathcal{N}} \boldsymbol{B}_i\boldsymbol{x}_i - \boldsymbol{c}_i = \boldsymbol{0}, \\ & \boldsymbol{x}_i \in X_i, \quad i \in \mathcal{N} \end{aligned} \tag{1}$$

where $\boldsymbol{x}_i \in \mathbb{R}^q$, $\boldsymbol{B}_i \in \mathbb{R}^{M\times q}$, and $\boldsymbol{c}_i \in \mathbb{R}^{M\times 1}$ are assumed to be known to node i, and $X_i \subset \mathbb{R}^q$ is a convex and compact set. This problem is often seen in resource allocation or load balancing problems. In section VI, we use our proposed protocol to achieve privacy in two already existing ADMM-like algorithms which solve (1) in two different scenarios: 1) each node i can communicate only with its neighboring nodes, and 2) each node i can communicate with a (untrusted) central unit.

A. Related Work

In the literature, there are three main approaches for privacy preservation in distributed computation tasks; secret sharing based secure multiparty computation (SMPC) [5], homomorphic encryption techniques [6], and differential privacy [7]. For the problem in this paper, namely secure aggregation of private data, both SMPC and homomorphic encryption based approaches are evident methods as both can compute a cipher text version of a sum based on cipher text versions of the terms in the sum, [8]–[11]. The drawback in SMPC is that the schemes typically require all participating parties to be connected by private channels. Regarding homomorphic encryption, the disadvantage is that usually a trusted third party needs to generate and distribute encryption keys. [12] is closely related to our work as they also consider private aggregation in a peer-to-peer network. In their solution, each node needs to communicate with the neighbors of its neighbors, which is in contrast to the assumption in this paper where

*This work is supported by SECURE project funded by Aalborg University

[1]K. Tjell and R. Wisniewski with the Department of Electronic Systems, Aalborg University, Denmark. e-mail: {kst},{raf}@es.aau.dk

each node can communicate with its immediate neighbors only. Also, the solution in [12] requires the presence of a trusted third party, which our solution does not.

B. Contribution

The paper puts forth a novel private aggregation scheme which bypasses the strict communication requirements in SMPC and the engagement of a trusted third party in homomorphic encryption approaches. The main contribution of the paper can be summarized as:

- The proposed scheme comes with an efficient preprocessing phase requiring only 2 communication rounds. This phase can be executed prior to the actual computations, even before the nodes have access to their individual inputs. This partitioning makes the scheme extremely light weight at computation time, compared with state-of-the-art SMPC methods.
- In contrast to SMPC and homomorphic encryption based approaches the proposed scheme can be employed in distributed settings where a fully connected communication network cannot be assumed and where an (un)trusted third party does not exist.
- Unlike most of the existing private aggregation schemes, the proposed method in this paper is resilient to node-dropouts.

C. Outline

In section II, we formally state the problem of the paper. A few cryptographic primitives are presented in section III, while the main contribution is in section IV. Section V gives the privacy analysis of the proposed method and section VI applies the method to distributed optimization.

D. Notation

We model the network of nodes as an undirected graph with the nodes $\mathcal{N} = \{1, \dots, N\}$ and the set of edges $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$ where $(i, j) \in \mathcal{E}$ iff node i can communicate with node j. The notation $\mathcal{N}_i$ is used to denote the neighborhood of i, that is $j \in \mathcal{N}_i$ iff $(i, j) \in \mathcal{E}$. Note that we do not consider node i to be a neighbor to itself, i.e. $i \notin \mathcal{N}_i$. Moreover, we use $\mathbb{F}_p$ to denote the finite field of p elements, where p is a prime.

II. Problem Statement

At the outset, we state the problem formally.

Problem 1: Let $i \in \mathcal{N}$ denote the index of nodes in a network and assume that each node has a private value $s_i \in \mathbb{F}_p$ which it would like to keep secret. Moreover, each node has a set of neighbors, $\mathcal{N}_i$, and each node are interested in learning the sum of its neighbors secret value:

$$y_i = \sum_{j \in \mathcal{N}_i} s_j. \tag{2}$$

The problem is to compute (2) *without exposing any secret value to any node in the network with the assumption that node i can only communicate with its neighbors $\mathcal{N}_i$.*

Defining the attacker model, we consider the case where up to $t - 1 < n$ nodes may collude in attempting to learn private information of the remaining nodes. However, we assume that all nodes follow the protocol, which is often referred to as the honest-but-curious adversary [1].

III. Cryptographic Tools

To put forth a privacy preserving solution to Problem 1, we use a few cryptographic primitives which are introduced in this section.

A. Secret Sharing Scheme

In this section, we give a very brief introduction to secret sharing and we refer to [5] for more elaborate explanation. Suppose a node i has a secret s_i which it would like to share with n other nodes (hereafter called participants) such that at least $t \leq n$ of the nodes need to cooperate in order to learn the value of s_i. The scheme is defined over a finite field $\mathbb{F}_p$, where p is a large prime and it uses a set $\mathcal{P}$ of distinct elements in $\mathbb{F}_p$ to identify the participants, e.g. $\mathcal{P} = \{1, 2, \dots, n\}$ and $|\mathcal{P}| = n$. The scheme is comprised of two algorithms, and the first is $\texttt{share}(s_i, t, \mathcal{P}) = \{s_i(j)\}_{j \in \mathcal{P}}$ which outputs a share $s_i(j)$ for each participant $j \in \mathcal{P}$ upon receiving a secret s_i, the threshold $t \leq |\mathcal{P}|$, and $\mathcal{P}$. Note that we use $s_i(j)$ to denote the j'th share of the secret s_i. The second algorithm is denoted by $\texttt{reconstruct}(\{s_i(j)\}_{j \in \mathcal{P}'}, t) = s_i$, and produces the secret s_i based on the threshold t and the shares from a set $\mathcal{P}' \subseteq \mathcal{P}$ of participants, where $|\mathcal{P}'| \geq t$. Note that $\mathcal{P}'$ can be any combination of at least t elements from $\mathcal{P}$.

We have the following requirements for the secret sharing scheme. Let $\{s_i(j)\}_{j \in \mathcal{P}} = \texttt{share}(s_i, t, \mathcal{P})$, $\{\bar{s}_i(j)\}_{j \in \mathcal{P}} = \texttt{share}(\bar{s}_i, t, \mathcal{P})$ for arbitrary $s_i, \bar{s}_i \in \mathbb{F}_p$, $t \leq |\mathcal{P}|$, and $\mathcal{P} \subseteq \mathbb{F}_p$.

1) For all $\mathcal{P}' \subseteq \mathcal{P}$, with $|\mathcal{P}'| \geq t$,
$$\texttt{reconstruct}(\{s_i(j)\}_{j \in \mathcal{P}'}, t) = s_i.$$
2) For all $\mathcal{P}' \subseteq \mathcal{P}$ with $|\mathcal{P}'| < t$:
$$\{s_i(j)\}_{j \in \mathcal{P}'} \sim \{\bar{s}_i(j)\}_{j \in \mathcal{P}'}, \tag{3}$$
where $\sim$ means identically distributed.
3) Finally, we require that
$$\texttt{reconstruct}(\{s_i(j)\}_{j \in \mathcal{P}'} + \{\bar{s}_i(j)\}_{j \in \mathcal{P}'}, t) = s_i + \bar{s}_i, \tag{4}$$
for all $\mathcal{P}' \subseteq \mathcal{P}$ with $|\mathcal{P}'| \geq t$.

B. Encryption

For the privacy preserving computations, it will be necessary to encrypt certain values (messages), such that these values cannot be learned by unauthorized nodes. The encryption scheme encompasses three algorithms, where the first is denoted $\texttt{keys} = (sk_i, pk_i)$. It generates a secret- and public key pair, (sk_i, pk_i) for a participant $i \in \mathcal{P}$. The third algorithm is

[1] An honest-but-curious adversary is also sometimes referred to as *a passive adversary*, see [5]

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2020.3041611, IEEE Control Systems Letters

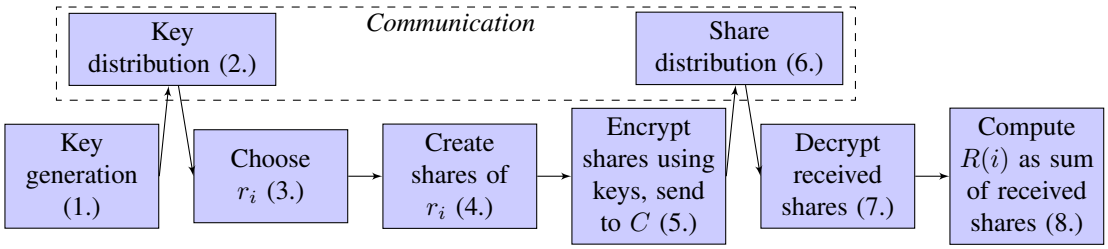

Fig. 1. Block diagram illustrating the pre-processing phase of the proposed method from the view of node $i \in \mathcal{N}_C$. The numbers in parenthesis refer to the corresponding step in Protocol 1.

the encryption algorithm $\texttt{enc}(x, pk_i) = [x]$. It takes a message x and a public key and outputs a cipher-text version $[x]$ of the message. The decryption algorithm takes a cipher-text message $[x]$ and a secret key and outputs the plain text version of the message. This algorithm is denoted by $\texttt{dec}([x], sk_i) = x$. For all $x, \bar{x} \in \mathbb{F}_p$, we have the following requirements for the scheme, where (sk_i, pk_i) are any secret- and public key pair:

1)
$$\texttt{dec}(\texttt{enc}(x, pk_i), sk_i) = x \tag{5}$$

2)
$$\texttt{enc}(x, pk_i) \sim \texttt{enc}(\bar{x}, pk_i), \tag{6}$$

see details in [10].

IV. Proposed Method

Note that both the secret sharing scheme and the encryption scheme is defined over a finite field $\mathbb{F}_p$. Consequently, modular arithmetic is used in the proposed method. Choosing $p > y_i$ and assuming that each secret s_i is an element of $\mathbb{F}_p$, the modular arithmetic will not affect the precision. In case $s_i \in \mathbb{R}$, s_i can be scaled before rounding to the nearest element in $\mathbb{F}_p$, under which circumstance the precision of the method will depend on the scaling factor.

To introduce the proposed method, consider a central node C and k nodes $i \in \mathcal{N}_C$ that are all neighbors to C but not necessarily neighbors to each other.

We present our method by focusing only on node C and the nodes $i \in \mathcal{N}_C$. Solving Problem 1 can be achieved by executing the method in parallel for all nodes $j \in \mathcal{N}$.

The idea is to solve (2) in Problem 1 by computing

$$\left(\sum_{i\in\mathcal{N}_C} s_i + r_i\right) \mod p = \left(\sum_{i\in\mathcal{N}_C} s_i + R\right) \mod p, \tag{7}$$

where $R = \left(\sum_{i\in\mathcal{N}_C} r_i\right) \mod p$ and $\{r_i \in \mathbb{F}_p\}_{i\in\mathcal{N}_C}$ are uniformly chosen by node i. In this way, s_i is masked by the random r_i. We propose a pre-processing phase where each node $i \in \mathcal{N}_C$ computes a share, $R(i)$, of R. At the execution time, node i sends $(s_i + r_i) \mod p$ and $R(i)$ to node C, that can use $R = \texttt{reconstruct}(\{R(i)\}_{i\in\mathcal{N}_C,t})$ to compute R and (7) to compute $y_C = \sum_{i\in\mathcal{N}_C} s_i$.

In Fig. 1, we give an overview of the pre-processing phase, where each communication block covers the steps where each node $i \in \mathcal{N}_C$ sends a vector of values to C who forwards the vectors to all nodes $j \in \mathcal{N}_C$. As seen, each node $i \in \mathcal{N}_C$ starts by generating the keys $(sk_i, pk_i) = \texttt{keys}$, and distributes the public key pk_i. Then each node $i \in \mathcal{N}_C$ chooses r_i uniformly from $\mathbb{F}_p$ and creates shares, $\{r_i(j)\}_{j\in\mathcal{N}_C}$, of r_i. The shares are encrypted using the public key of the corresponding node $j \in \mathcal{N}_C$ and the encrypted shares are distributed. Upon receiving encrypted shares from the other nodes, each node $i \in \mathcal{N}_C$ decrypts the shares using its own secret key, sk_i. Node i then computes its share of R by $R(i) = \sum_{j\in\mathcal{N}_C} r_j(i)$, which holds under the third requirement for the secret sharing scheme. The outcome of the pre-processing phase is that node $i \in \mathcal{N}_C$ learns $R(i)$.

We state the proposed method formally in Protocol 1 from the view of node $i \in \mathcal{N}_C$.

Protocol 1 Private Sum in Graphs

Input: $\mathbb{F}_p$ with $p > \sum_{j\in\mathcal{N}_C} s_j$, and the threshold $t < |\mathcal{N}_C|$ are publicly available.
Output: node C learns $y_C = \sum_{j\in\mathcal{N}_C} s_j$, where s_j is the secret known only to node j.

Pre-processing:

1: $(sk_i, pk_i) = \texttt{keys}$
2: Send public key pk_i to C who forwards to $j \in \mathcal{N}_C \backslash \{i\}$.
3: Draw $r_i \in \mathbb{F}_p$ uniformly.
4: $\{r_i(j)\}_{j\in\mathcal{N}_C} = \texttt{share}(r_i, t, \mathcal{N}_c)$.
5: $[r_i(j)] = \texttt{enc}(r_i(j), pk_j)$ for $j \in \mathcal{N}_C \backslash \{i\}$.
6: Send $[r_i(j)]$ to node C who forwards to node $j \in \mathcal{N}_C \backslash \{i\}$.
7: $\{r_j(i)\}_{j\in\mathcal{N}_C\backslash\{i\}} = \{\texttt{dec}([r_j(i)], sk_i)\}_{j\in\mathcal{N}_C\backslash\{i\}}$.
8: $R(i) = \sum_{j\in\mathcal{N}_C} r_j(i)$.

Execution:

9: $m_i = (s_i + r_i) \mod p.$ (8)
10: Send $\{m_i, R(i)\}$ to node C.

Node C does:

1) $R = \texttt{reconstruct}(\{R(i)\}_{i\in\mathcal{N}_C}, t)$.
2) $y_C = \left(\left(\sum_{i\in\mathcal{P}} m_i\right) - R\right) \mod p.$ (9)

A. Handling dropped nodes

The proposed protocol is inherently able to handle nodes dropping out as long as there is at least t remaining nodes. To elaborate, we use $\mathcal{P}$ to denote the nodes participating from the beginning and $\mathcal{P}'$ to denote the nodes remaining after some nodes have dropped out. Specifically, $\mathcal{P}' \subset \mathcal{P}$ and $|\mathcal{P}'| \geq t$. If nodes drop out in the pre-processing phase, the remaining nodes can carry on without any modification. If nodes drop out in the execution phase and fail to perform step 10, each node

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2020.3041611, IEEE Control Systems Letters

$i \in \mathcal{P}'$ computes $R_{new}(i) = \sum_{j\in\mathcal{P}'} r_j(i)$ and sends R_{new} to node C. Node C can then compute $\sum_{i\in\mathcal{P}'} s_i$. The advantage is that the pre-processing does not have to be run again.

V. Privacy Analysis

We use the standard simulation-based proof to prove that in executing Protocol 1, any set of fewer than t colluding nodes will not be able to infer the private values of the honest nodes. To this end, we introduce the term *view* of a node, which is the information known to it during protocol execution.

Definition 1 (View): The view of a node $i \in \mathcal{N}_C$ is a vector, $\texttt{VIEW}_i$, consisting of the values i knows and receives. For a subset $\mathcal{A} \subset \mathcal{N}_C$ of nodes, $\texttt{VIEW}_\mathcal{A}$ denotes the vector containing the view of each node $i \in \mathcal{A}$.

Note that, $\texttt{VIEW}_i$ is a random variable since it is based on random choices made by the nodes. What will be shown, is the existence of a simulator which essentially is an algorithm with the ability to simulate a view that is indistinguishable from the view of a set of nodes.

We use $\mathcal{S}_\mathcal{A} = \{s_i\}_{i\in\mathcal{A}}$ to denote the set of private values of the nodes $i \in \mathcal{A}$.

Theorem 1 (Honest-but-curious privacy): Consider a set $\mathcal{N}_C \subseteq \mathbb{F}_p$. For all integers $t \leq |\mathcal{N}_C|$ and any set $\mathcal{A} \subset \mathcal{N}_C$ with $|\mathcal{A}| < t$ and the central node $C \in \mathcal{A}$, there exist a probabilistic polynomial-time (PPT) simulator $\texttt{SIM}$ which upon the inputs; $\mathcal{S}_\mathcal{A}$, $\mathbb{F}_p$, the threshold t, and the output of Protocol 1, y_C, outputs a vector perfectly indistinguishable from $\texttt{VIEW}_\mathcal{A}$, namely

$$\texttt{VIEW}_\mathcal{A} \sim \texttt{SIM}_\mathcal{A}(\mathcal{S}_\mathcal{A}, \mathbb{F}_p, t, y_C). \tag{10}$$

Proof: The simulator must simulate each element in the view. We list the elements and discuss how the simulated equivalent to each element is chosen. Each simulated equivalent is marked by a $\{\cdot\}^s$. $\texttt{VIEW}_\mathcal{A}$ consists of the following values:

$$\begin{aligned} &s_i, r_i, sk_i, \{r_j(i)\}_{j\in\mathcal{N}_C}, && i \in \mathcal{A} \\ &\{m_i, R(i)\}, pk_i, \{[r_j(i)]\}_{j\in\mathcal{N}_C}, && i \in \mathcal{N}_C \\ &R, y_C = \sum_{i\in\mathcal{N}_C} s_i \end{aligned} \tag{11}$$

The simulator starts by choosing R^s uniformly from $\mathbb{F}_p$ since R (from (7)) is distributed in this way. Then it uses $\{R^s(i)\}_{i\in\mathcal{N}_C} = \texttt{share}(R^s, t, \mathcal{N}_C)$. $\{m_i\}_{i\in\mathcal{N}_C}$ (from (8)) are uniformly distributed on $\mathbb{F}_p$ with the condition that $\sum_{i\in\mathcal{N}_C} m_i = y_C + R$, thus $\{m_i^s\}_{i\in\mathcal{N}_C}$ are simulated according to this. $\{r_i^s\}_{i\in\mathcal{N}_C}$ are simulated by drawing uniformly random values from $\mathbb{F}_p$, with the condition that $\sum_{i\in\mathcal{N}_C} r_i^s = R$, see (7). Based on $\{r_i^s\}_{i\in\mathcal{N}_C}$, the simulator can use the steps in the pre-processing phase of Protocol 1 to simulate the remaining elements of $\texttt{SIM}_\mathcal{A}(\mathcal{S}_\mathcal{A}, \mathbb{F}_p, t, y_C)$. ■

A. Leakage of Information from Output

As noted earlier, information can be gained from the output of the protocol.To study this in detail, we consider the sum,

$$z_N = \sum_{i=1}^{N} s_i, \tag{12}$$

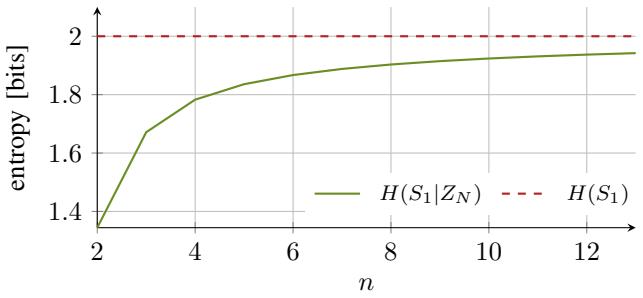

Fig. 2. Comparison between the entropy of $\boldsymbol{S_1}$ (red dashed line) with the entropy of $\boldsymbol{S_1}$ conditioned on $\boldsymbol{Z_N}$ (green line).

where $s_i \in [0, K]$ for $K \in \mathbb{F}_p$ corresponds to the secret value of node i, and $N > 1$ is the number of terms in the sum. Particularly, we will investigate the amount of information z_N leaks about a particular s_i, say s_1. To do this, z_N and $\{s_i\}_{i\in\mathcal{N}}$ are viewed as outcomes of the random variable Z_N and uniformly distributed variables $\{S_i\}_{i\in\mathcal{N}}$, respectively. We use the uniform distribution for each S_i since this will be true from the view of the adversary given that he has no prior knowledge.

We start the discussion by considering the mutual information between S_1 and Z_N, given as

$$I(S_1, Z_N) = \sum_{s_1, z_N} p(s_1, z_N) \log_2 \left(\frac{p(s_1, z_N)}{p(s_1)p(z_N)} \right), \tag{13}$$

where $p(x)$ is the probability mass function of the random variable X, and $p(x, y)$ is the joint probability mass function of the random variables X and Y. Intuitively, $I(S_1, Z_N)$ is the reduction in uncertainty about S_1 gained from Z_N. By the data processing inequality (see for instance [13]),

$$I(S_1, Z_{N-1}) \geq I(S_1, Z_N). \tag{14}$$

Hence, the mutual information is non-increasing as N is increased. To explore this result further, consider the conditional entropy of S_1 conditioned on Z_N, which is given as

$$H(S_1|Z_N) = H(S_1) - I(S_1, Z_N). \tag{15}$$

This is a measure of the uncertainty about S_1 after Z_N is given. The uncertainty is measured in bits and the more bits, the more uncertainty there is about the variable.

Finding a closed form expression for $H(S_1|Z_N)$ is still an open question. Hence, to illustrate it, we calculate $H(S_1|Z_N)$ numerically for small values of N and K since the combinatorics starts to be intractable for larger values. Fig. 2 depicts $H(S_1)$ and $H(S_1|Z_N)$ for $N = 2, \ldots, 13$ and $K = 4$. The figure compares the uncertainty of S_1 to the uncertainty of S_1 conditioned on Z_N. As seen, increasing N decreases the leakage about S_1. This means that the more neighbors a node has, the less information it will gain on the private values of its neighbors.

We conclude this section, by studying the probability of the adversary guessing $S_1 = s_1$ after learning $Z_N = z_N$ or in other words; the probability of leaking the secret. To establish a closed form expression for this probability, we assume that each S_i is uniformly distributed on $[0, z_N]$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2020.3041611, IEEE Control Systems Letters

Proposition 1: Let $S_1, \dots, S_n$ be independent uniformly distributed on $[0, K]$ and let $Z_N = \sum_{i=1}^{N} S_i$. Then the conditional probability of S_1 conditioned on Z_N is given by

$$P(S_1|Z_N) = \frac{(z_N - s_1 + N - 2)! z_N! (N-1)!}{((z_N - s_1)!(N-2)!(z_N + (N-1))!} \tag{16}$$

for $z_N \in \{0, 1, \dots, K\}$.

Proof: The conditional probability can be written as

$$P(S_1|Z_N) = \frac{P(S_1, Z_N)}{P(Z_N)}. \tag{17}$$

By counting the number of combinations of $s_1, \dots, s_N$ that satisfies (12) with given z_N, where each s_j can take values in the interval $[0, z_N]$, the probability mass function of Z_N yields

$$P(Z_N) = \frac{(z_N + N - 1)!}{z_N!(N-1)!} \frac{1}{T}, \tag{18}$$

where T is the total number of outcomes of Z_N.

Similarly, by counting the number of combinations of $s_2, \dots, s_N$ that satisfies (12) with given z_N and s_1, where each s_j can take values in the interval $[0, z_N]$, the joint probability mass function between S_1 and Z_N, yields

$$P(S_1, Z_N) = \frac{(z_N - s_1 + N - 2)!}{(z_N - s_1)!(N-2)!} \frac{1}{T}, \tag{19}$$

which concludes the proof. ∎

VI. Application to Dist. Optimization

In the following, we consider the minimization problem in equation (1) assuming: 1) a centralized setting, and 2) a decentralized setting. We study two already existing distributed optimization algorithms and use Protocol 1 to achieve privacy in each of them.

A. Centralized Optimization

Solving (1) in the scenario where each node communicates with an untrusted central unit, can be achieved by using a modified version of the ADMM algorithm. Such an algorithm is presented in [14] by the following steps in each iteration $k \geq 0$:

$$\boldsymbol{d}^{k+1} = \frac{1}{N} \sum_{j=1}^{N} \boldsymbol{B}_j \boldsymbol{x}_j^k - \boldsymbol{c}_j \tag{20a}$$

$$\boldsymbol{x}_i^{k+1} \in \underset{\boldsymbol{x}_i \in X_i}{\operatorname{argmin}} \Big\{ f_i(\boldsymbol{x}_i) + {\boldsymbol{\lambda}^k}^\top \boldsymbol{B}_i \, x_i + \frac{\rho}{2} ||\boldsymbol{B}_i \boldsymbol{x}_i - \boldsymbol{B}_i \boldsymbol{x}_i^k + \boldsymbol{d}^{k+1}||^2 \Big\} \tag{20b}$$

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho \boldsymbol{d}^{k+1}, \tag{20c}$$

using the initial values $\boldsymbol{x}_i^0 \in X_i$, and $\boldsymbol{\lambda}^0 \in \mathbb{R}^m$. Since the nodes cannot communicate with each other, (20a) will be computed by the central unit. The algorithm in (20) is related to the traditional ADMM algorithm presented in [15], where the steps (20b) and (20c) can be identified as the primal and dual updates, respectively. However, there is an important distinction which accounts for the differences; in the traditional ADMM, the primal update is separated into two parts that are updated sequentially, while the primal update here is separated into N parts that are updated simultaneously. [16] proves convergence of (20) given that the solution set to the problem in (1) is nonempty. To preserve privacy, we propose to compute $\sum_{i=1}^{N} \boldsymbol{B}_i \boldsymbol{x}_i^k - \boldsymbol{c}_i$ using Protocol 1. We refer to the following two steps as privacy preserving (PP) parallel ADMM:

1) The nodes uses Protocol 1 to compute $\boldsymbol{d}^{k+1} = \sum_{i=1}^{N} \boldsymbol{B}_i \boldsymbol{x}_i^k - \boldsymbol{c}_i$, where the nodes $j \in \mathcal{N}$ takes the roles of nodes $i \in \mathcal{N}_C$ and the central unit takes the role of node C. The central unit returns $\boldsymbol{d}^{k+1}$ to the nodes $i \in \mathcal{N}$.
2) Each node $i \in \mathcal{N}$ computes (20b) and (20c) in parallel.

B. Decentralized Optimization

For solving (1) under the assumption that each node i can only communicate with its neighboring nodes $j \in \mathcal{N}_i$, [17] presents a fully decentralized variant of the ADMM algorithm referred to as tracking-ADMM. It uses a consensus matrix, $\boldsymbol{w} \in \mathbb{R}^{n \times n}$, which is a semidefinite doubly stochastic and symmetric matrix, see [17] for details. Given, $\boldsymbol{x}_i^0 \in X_i, \boldsymbol{\lambda}_i^0 \in \mathbb{R}^M$ and $\boldsymbol{d}_i^0 = \boldsymbol{B}_i \boldsymbol{x}_i^0 - \boldsymbol{c}_i$, the following steps computed by each node $i \in \mathcal{N}$ in parallel solves (1);

$$\boldsymbol{\delta}_i^k = w_{i,i} \boldsymbol{d}_i^k + \sum_{j \in \mathcal{N}_i} w_{i,j} \boldsymbol{d}_j^k \tag{21a}$$

$$\boldsymbol{l}_i^k = w_{i,i} \boldsymbol{\lambda}_i^k + \sum_{j \in \mathcal{N}_i} w_{i,j} \boldsymbol{\lambda}_j^k \tag{21b}$$

$$\boldsymbol{x}_i^{k+1} \in \underset{\boldsymbol{x}_i \in X_i}{\operatorname{argmin}} \{ f_i(\boldsymbol{x}_i) + {\boldsymbol{l}_i^k}^\top \boldsymbol{B}_i \boldsymbol{x}_i + \frac{\rho}{2} ||\boldsymbol{B}_i \boldsymbol{x}_i - \boldsymbol{B}_i \boldsymbol{x}_i^k + \boldsymbol{\delta}_i^k||^2 \} \tag{21c}$$

$$\boldsymbol{d}_i^{k+1} = \boldsymbol{\delta}_i^k + \boldsymbol{B}_i \boldsymbol{x}_i^{k+1} - \boldsymbol{B}_i \boldsymbol{x}_i^k \tag{21d}$$

$$\boldsymbol{\lambda}_i^{k+1} = \boldsymbol{l}_i^k + \rho \boldsymbol{d}_i^{k+1}, \tag{21e}$$

where $\rho > 0$ is a penalty parameter. The algorithm in (21) is quite different from the standard ADMM due to the fully decentralized setting. The information $\sum_{i=1}^{N} \boldsymbol{B}_i \boldsymbol{x}_i^{k+1} - \boldsymbol{c}_i$ is not available to the nodes, since each node can only communicate with its neighbors. Thus, as explained in [17], steps (21a), (21b) and (21d) roughly acts as a dynamic average consensus mechanism for estimating this term. [17] proves that this algorithm converges given that each $f_i(\boldsymbol{x}_i)$ is convex and that (1) and the dual problem of (1) admits optimal solutions. To preserve privacy, we propose to use Protocol 1 to compute $\boldsymbol{\delta}_i^k$ and $\boldsymbol{l}_i^k$ for each node i. That is, (21a) and (21b) is substituted with the following steps;

1) Protocol 1 is utilized to compute $\boldsymbol{\delta}_i^k$, where node i takes the role of the central node C and the nodes $j \in \mathcal{N}_i$ takes the role of the nodes $j \in \mathcal{N}_C$. The nodes $j \in \mathcal{N}_i$ inputs $w_{i,j} \boldsymbol{d}_j^k$ to the protocol and node i learns $\bar{\boldsymbol{\delta}}_i^k = \sum_{j \in \mathcal{N}_i} w_{i,j} \boldsymbol{d}_j^k$ and computes $\boldsymbol{\delta}_i^k = w_{i,i} \boldsymbol{d}_i^k + \bar{\boldsymbol{\delta}}_i^k$.
2) Protocol 1 is utilized to compute $\boldsymbol{l}_i^k$. The nodes $j \in \mathcal{N}_i$ inputs $w_{i,j} \boldsymbol{\lambda}_j^k$ to the protocol and node i learns $\bar{\boldsymbol{l}}_i^k = \sum_{j \in \mathcal{N}_i} w_{i,j} \boldsymbol{\lambda}_j^k$ and computes $\boldsymbol{l}_i^k = w_{i,i} \boldsymbol{\lambda}_i^k + \bar{\boldsymbol{l}}_i^k$.
3) Each node $i \in \mathcal{N}$ compute in parallel the remaining steps; (21c), (21d), (21e).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2020.3041611, IEEE Control Systems Letters

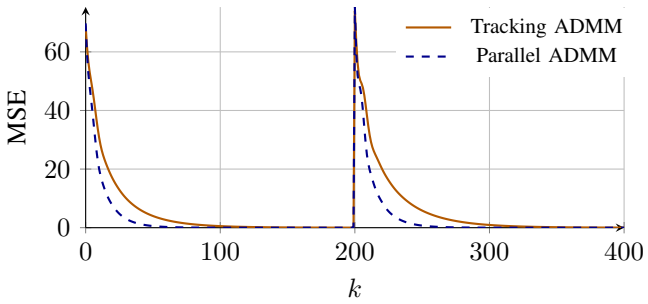

Fig. 3. Convergence of PP tracking ADMM and PP parallel ADMM with $N = 30$ nodes. After $k = 200$ iterations, 10 randomly selected nodes drop out.

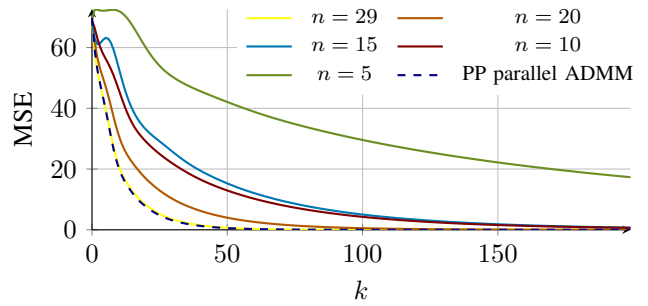

Fig. 4. Comparison of the convergence of PP tracking ADMM where each node has respectively 29, 20, 15, 10, and 5 neighbors.

We refer to these three steps as PP tracking ADMM.

C. Numerical Experiments

We simulate privacy preserving (PP) parallel ADMM and PP tracking ADMM solving the same optimization problem of the form of (1) with $q = 1$, $M = 2$, $f_i(x_i) = (x_i - i)^2$, and randomly generated $\boldsymbol{B}$ and $\boldsymbol{c}$ matrices. This problem is solved with $N = 30$ nodes and in the case of PP tracking ADMM, each node has on average 20 neighbors. After 200 iterations, we simulate the dropout of 10 randomly selected nodes. To compare the performance of PP parallel ADMM and PP tracking ADMM, Fig. 3 shows the mean squared error (MSE) of the estimate from both methods at each iteration k. As seen in Fig. 3, PP tracking ADMM has a slower convergence rate compared to PP parallel ADMM which is due to information being distributed in the network much slower. In fact, the convergence rate of PP tracking ADMM is dependent on the number of neighbors of each node. This can be observed in Fig. 4 that shows the convergence rate of PP tracking ADMM when the numerical problem is solved and each node has $n = 5, 10, 15, 20, 29$ neighbors, respectively. Note that in the case $n = 29$, the network is fully connected and the convergence rate matches with the rate of the PP parallel ADMM. That PP tracking ADMM converges faster the more neighbors each node has matches nicely with the result from section V-A stating that the more neighbors a node has, the less information is revealed by the output of the method.

VII. Conclusion

The paper presents a privacy preserving fully distributed and parallel aggregation scheme for computing the sum of private values held by individual nodes. Two straight forward applications of the proposed protocol are given in the paper, namely privacy preserving distributed optimization. We note that the protocol can be applied in many other distributed control algorithms to preserve privacy, see for instance [18].

References

[1] Z. Huang, W. Du, and B. Chen, "Deriving private information from randomized data," in *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '05. NY, USA: ACM, 2005, pp. 37–48.

[2] D. Yuan, A. Proutiere, and G. Shi, "Distributed online linear regression," 2019.

[3] G. Chen and J. Li, "A fully distributed admm-based dispatch approach for virtual power plant problems," *Applied Mathematical Modelling*, vol. 58, pp. 300 – 312, 2018.

[4] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu, "Decentralized laplacian eigenvalues estimation for networked multi-agent systems," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, Dec 2009, pp. 2717–2722.

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

[6] M. A. Will and R. K. Ko, "Chapter 5 - a guide to homomorphic encryption," in *The Cloud Security Ecosystem*, R. Ko and K.-K. R. Choo, Eds. Boston: Syngress, 2015, pp. 101 – 127.

[7] C. Dwork, "Differential privacy," in *Automata, Languages and Programming*, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12.

[8] E. Shi, T.-H. Chan, E. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," vol. 2, 01 2011.

[9] G. Danezis, C. Fournet, M. Kohlweiss, and S. Zanella-Béguelin, "Smart meter aggregation via secret-sharing," in *Proceedings of the First ACM Workshop on Smart Energy Grid Security*, ser. SEGS '13. NY, USA: Association for Computing Machinery, 2013, p. 75–80.

[10] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. NY, USA: Association for Computing Machinery, 2017, p. 1175–1191.

[11] M. Joye and B. Libert, "A scalable scheme for privacy-preserving aggregation of time-series data," in *Financial Cryptography and Data Security*, A.-R. Sadeghi, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 111–125.

[12] D. Bickson, T. Reinman, D. Dolev, and B. Pinkas, "Peer-to-peer secure multi-party numerical computation facing malicious adversaries," *Peer-to-Peer Networking and Applications*, vol. 3, 01 2009.

[13] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. USA: Wiley-Interscience, 1991.

[14] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. USA: Prentice-Hall, Inc., 1989.

[15] 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, Jan. 2011.

[16] 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.

[17] A. Falsone, I. Notarnicola, G. Notarstefano, and M. Prandini, "Tracking-admm for distributed constraint-coupled optimization," 2019, submitted to Automatica, eprint: arXiv:1907.10860.

[18] Y. Wang, Z. Huang, S. Mitra, and G. E. Dullerud, "Differential privacy in linear distributed control systems: Entropy minimizing mechanisms and performance tradeoffs," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 1, pp. 118–130, March 2017.