Privacy Preserving Distributed Summation in a Connected Graph

Tjell, Katrine ; Wisniewski, Rafal

Published in:
IFAC-PapersOnLine

DOI (link to publication from Publisher):
10.1016/j.ifacol.2020.12.1677

Creative Commons License
CC BY-NC-ND 4.0

Publication date:
2020

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Tjell, K., & Wisniewski, R. (2020). Privacy Preserving Distributed Summation in a Connected Graph. *IFAC-PapersOnLine*, *53*(2), 3445-3450. https://doi.org/10.1016/j.ifacol.2020.12.1677

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 53-2 (2020) 3445–3450

Privacy Preserving Distributed Summation in a Connected Graph [★]

Katrine Tjell [*] **Rafael Wisniewski** [*]

[*] *Aalborg University, Department of Control and Automation, Fredrik Bajers Vej 7C, Aalborg, Denmark (e-mail: {kst},{raf}@es.aau.dk).*

Abstract: Most decentralized algorithms for multi-agent systems used in control, signal processing and machine learning for example, are designed to fit the problem where agents can only communicate with immediate neighbors in the network. For instance, decentralized and distributed optimization algorithms are based on the fact that every agent in a network will be able to influence every other agent in the network even if each agent only communicates with its immediate neighbors (given that the network is connected). That is, a distributed optimization problem can be solved in a decentralized manner by letting the agents exchange messages with their neighbors iteratively. In many algorithms that solve this kind of problem, agents in the network does not need individual values from their neighbors, rather they need a function of the values from its neighbors. This observation makes it interesting to consider privacy preservation in such algorithms. By privacy preservation, we mean that raw data from individual agents will not be exposed at any time during calculations.
This paper is concerned with decentralized algorithms, where each agent must learn the *sum* of its neighbors values, and we propose a privacy preserving method to compute this sum. Employing this method in corresponding decentralized algorithms makes the whole algorithm privacy preserving. The only restriction we make on the graph topology of the network is that each agent must have at least two neighbors. We provide simulations of the proposed method, which illustrates the scalability of it.

Keywords: Privacy, multi-agent systems, decentralized control, decentralized and distributed control, cyber-physical systems.

1. INTRODUCTION

Multi-agent systems with a decentralized graph topology appear in many areas such as; formation control, distributed resource allocation, workload balancing and energy optimization. Thus, many decentralized algorithms for solving different problems in a distributed and decentralized fashion has been proposed. For some of these algorithms, each agent are only required to communicate with immediate neighbors and moreover; each agent does not necessarily need to learn individual values from neighboring agents, rather they need to learn the sum of neighboring agents values. Such algorithms can for instance be seen in the work by Banjac et al. (2019), Chang et al. (2015), and Ma et al. (2018). Other examples include the work by Franceschelli et al. (2009) that considers decentralized estimation of Laplacian eigenvalues in multi-agent systems and the work by Liu et al. (2019) that proposes a communication efficient algorithm for resource-aware exact decentralized optimization.

This paper considers privacy preservation of agents in a multi-agent network where each agent needs to learn the sum of its neighboring agents values in order to carry out required local computations. Specifically, we are interested in the case where agents are unwilling to share raw data, perhaps because the data is sensitive from a business perspective or because it leaks personal information. Motivated by the many decentralized algorithms only requiring agents to learn the sum of neighboring agents values, we investigate the privacy preserving calculation of

$$\sum_{j\in\mathcal{N}_i} x_j, \tag{1}$$

where $\mathcal{N}_i$ is the set of indices of neighbors to agent a_i and x_j is a value known only to agent a_j.

To give thorough motivation for the problem, consider the minimization problem

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

where $\boldsymbol{x} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$, $\boldsymbol{x}_i$ is a local variable, f_i is a local objective and $\boldsymbol{B}_i$ and $\boldsymbol{c}$ are constraints. For solving the problem distributed and decentralized, Chang et al. (2015) proposes the following steps:

[★] This work is supported by SECURE project funded by AAU.

$$\boldsymbol{p}_i^{(k)} = \boldsymbol{p}_i^{(k-1)} + \rho \sum_{j\in\mathcal{N}_i} \left(\boldsymbol{v}_i^{(k-1)} - \boldsymbol{v}_j^{(k-1)}\right), \tag{3}$$

$$\boldsymbol{x}_i^{(k)} = \arg\min_{\boldsymbol{x}_i} \Bigg\{ f_i(\boldsymbol{x}_i) + \frac{\rho}{4|\mathcal{N}_i|} \Bigg\| \frac{1}{\rho}(\boldsymbol{B}_i\boldsymbol{x}_i - \frac{1}{N}\boldsymbol{c}) - \frac{1}{\rho}\boldsymbol{p}_i^{(k)} + \sum_{j\in\mathcal{N}_i} \left(\boldsymbol{v}_i^{(k-1)} + \boldsymbol{v}_j^{(k-1)}\right) \Bigg\|_2^2 \Bigg\}, \tag{4}$$

$$\boldsymbol{v}_i^{(k)} = \frac{1}{2|\mathcal{N}_i|} \Bigg(\sum_{j\in\mathcal{N}_i} \left(\boldsymbol{v}_i^{(k-1)} + \boldsymbol{v}_j^{(k-1)}\right) - \frac{1}{\rho}\boldsymbol{p}_i^{(k)} + \frac{1}{\rho}\left(\boldsymbol{B}_i\boldsymbol{x}_i - \frac{1}{N}\boldsymbol{c}\right) \Bigg), \tag{5}$$

that are performed locally by each agent. As can be seen, values of neighboring agents appear in all three equations. In (3), the values appear as

$$\sum_{j\in\mathcal{N}_i} \left(\boldsymbol{v}_i^{(k-1)} - \boldsymbol{v}_j^{(k-1)}\right) = |\mathcal{N}_i|\boldsymbol{v}_i^{(k-1)} - \sum_{j\in\mathcal{N}_i} \boldsymbol{v}_j^{(k-1)}, \tag{6}$$

and in (4) and (5) as

$$\sum_{j\in\mathcal{N}_i} \left(\boldsymbol{v}_i^{(k-1)} + \boldsymbol{v}_j^{(k-1)}\right) = |\mathcal{N}_i|\boldsymbol{v}_i^{(k-1)} + \sum_{j\in\mathcal{N}_i} \boldsymbol{v}_j^{(k-1)}. \tag{7}$$

Evidently, if $\sum_{j\in\mathcal{N}_i} \boldsymbol{v}_j^{(k-1)}$ can be computed without leaking individual $\boldsymbol{v}_j$ values, the algorithm is privacy preserving. Hence, the aim of this paper is to, for each agent in a multi-agent system, compute the sum of its neighboring agents values without individual terms of the sum being exposed.

Related work. Computing the sum of private values among a set of participants, without leaking the private values, is a well-known problem within the field of cryptography. Most secret sharing schemes (see the work by Cramer et al. (2015)) such as Shamir's secret sharing scheme and the additive secret sharing scheme are able to compute *shares* of the sum of secret input values from *shares* of the input values. The same goes for more or less all the homomorphic encryption schemes, see Will and Ko (2015). All though our proposed solution is based both on secret sharing and encryption, the paper distinguishes it self by being applied to agents in a graph network. Thus, we do not assume that all agents can communicate or that there exist a central node which all agents can communicate with, which is the general assumption in this kind of work. Our work also distances it self from traditional secret sharing and homomorphic encryption based approaches, as it includes a preprocessing phase which speed up efficiency at the actual execution time.

Many works (also outside cryptography) are occupied with the privacy preserving summation of values, for instance Mehnaz et al. (2017), Clifton et al. (2002) and Sheikh et al. (2009). The work by Mehnaz et al. (2017) proposes a secure sum protocol for computing a sum of N private values among N participants. Their solution relies on an (untrusted) moderator to ensure privacy. Our proposed method does not require a moderator and for the setup considered in this paper, it is not a viable solution to adopt moderators in the network.

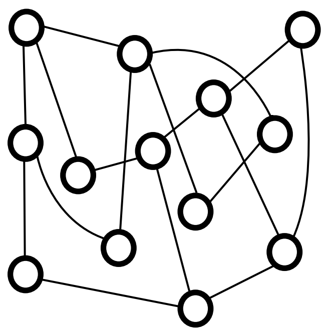

Fig. 1. A multi-agent network, where agents are depicted as vertices and the communication channels are depicted as edges.

Structure. Section 2 states the problem of the paper in detail and gives the necessary preliminaries. The main content of the paper is in sections 3 and 4, where the problem is solved with two different assumptions on the graph topology. Section 5 gives an illustration of the scale-ability of the proposed method and finally, section 6 concludes the paper.

2. PROBLEM FORMULATION

Consider a multi-agent network consisting of N agents, $a_1, \ldots, a_N$, with a certain communication network linking them. Fig. 1 depicts such a multi-agent network, where the vertices are the agents and the edges illustrates the communication network. We define $\mathcal{N}_i$ as the neighbors to agent a_i; that is, $\mathcal{N}_i$ is the set of indices j of agents a_j where there is an edge between a_i and a_j. Note that we consider a_i to be a neighbor to itself.

Furthermore, each agent a_i has a private value x_i. The aim is to compute the sum

$$y_i = \sum_{j\in\mathcal{N}_i} x_j, \tag{8}$$

for each agent a_i in the multi-agent network. As each agent can communicate with all of their neighbors, the problem is trivially solved be letting all agents a_j for $j \in \mathcal{N}_i$ send their value x_j to agent a_i, who can then compute the sum. However, in this paper we are interested in the case where agents are unwilling to hand over raw data, for instance because the data is considered corporate secrets or simply because it leaks private information. Thus, the goal is that agent a_i learns only y_i and not the individual terms x_j in the sum. For this to be possible, we assume that all agents has at least two neighbors besides from itself, that is $|\mathcal{N}_i| \geq 3 \forall i$. Furthermore, we assume that agents will follow the protocol, however they may collude in attempting to disclose the secret values of other agents. To this end, we assume that a majority of an agents neighbors are not colluding, which will ensure the privacy of the honest agents.

We consider two different scenarios with respect to graph topology, we will refer to these as problem 1 (P1) and problem 2 (P2). The difference between P1 and P2 is that in P1, we assume that in each neighborhood $\mathcal{N}_i$, the agents $\{a_j\}_{j\in\mathcal{N}_i}$ form one or more *cliques*. A clique is a fully connected subset of a graph, thus in other words, we assume in P1, that each a_j, $j \in \mathcal{N}_i$ is connected to at least

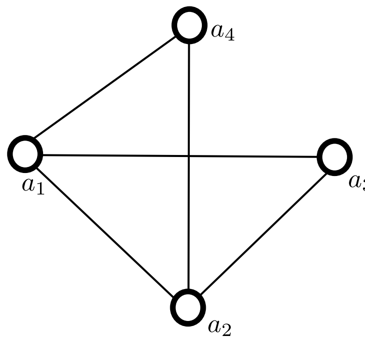

Fig. 2. An example of a multi-agent network considered in P1.

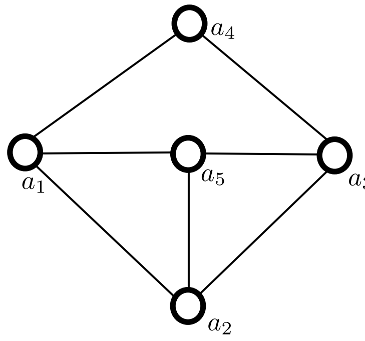

Fig. 3. An example of a multi-agent network considered in P2.

one other agent $a_{j'}$, $j' \in \mathcal{N}_i$, where $j \neq \{i, j'\}$. On the other hand, in P2, we make no assumptions on cliques in the neighborhood of a_i. In the following, P1 and P2 are more formally presented.

Problem 1. (P1). *Let $a_1, \ldots, a_N$ be agents in a multi-agent network and assume that the agents in $\mathcal{N}_i$ form one or more cliques with cardinality at least three, with all of them including a_i. Then, the problem is to calculate* (8) *for all agents a_i.*

Fig. 2 is a simple example of the graph topology P1 is investigating. As seen, all agents has at least two neighbors (besides itself) and the neighbors of each agent forms at least one clique. For instance, for agent a_1, (a_1, a_2, a_3) forms a clique and so does (a_1, a_2, a_4).

For P2, we relax the constraint on the graph topology and accepts the case where not all neighbors to an agent is part of a clique.

Problem 2. (P2). *For all agents, a_i, in an arbitrary connected multi-agent network, the problem is to calculate* (8).

Fig. 3 is a simple example of the graph topology P2 is investigating. It can be seen that agent a_4 does not form a clique with other agents that are in the neighborhood of either a_1 or a_3.

The aim of the paper is to solve P1 and P2 without leaking the agents private values. For P1, we propose to use secret-sharing and two rounds of communication. This approach is evident under the assumption of cliques in the neighborhood. In the case of P2, where cliques in the graph are not assumed, we introduce a so-called virtual clique between agents in the same neighborhood, which is done with the use of encryption. In this way, we use the same method in P2 as in P1 but with the cost of additional communication rounds. This we see as a generalization of the proposed method, making it independent of the graph topology. Remark, P1 is a special case of P2, and is as such covered by the solution to P2. However, we believe that solving them one at a time will improve readability.

Notation. Let $\mathcal{N}_i$ denote the neighbors of agent a_i and $C_1, \ldots, C_m$ are the cliques in the multi-agent network, specifically, C_k is a set consisting of the indices j of agents a_j in the k'th clique. Moreover, Ci is the cliques agent a_i is part of; that is, $k \in Ci$ iff $a_i \in C_k$. Lastly, for each agent a_i, we will need to define the sets $C_{i,j}, j \in \mathcal{N}_i$, where for all $k \in Ci$, $k \in C_{i,j}$ iff $a_j \in C_k$. Note that $C_{i,j} = C_{j,i}$.

3. PRIVACY PRESERVING SOLUTION TO P1

For solving P1 without leaking data, the idea is to compute the sum of values in each clique in the neighborhood of each agent. The sum is computed using secret sharing techniques, which will be key to preserve privacy in the protocol.

3.1 Secret Sharing

The aim of secret sharing is to share a secret among a set of participants such that none of the participants learn the secret and only by recombining the information of each participant, can the secret be reconstructed. In this way, an adversary attempting to learn the secret has to attack several entities instead of just one.

Dividing a secret into shares can be done in many ways. A simple scheme is called *additive secret sharing* and it has the property that the shares $s_1, \ldots, s_n$ of the secret s satisfies

$$s = s_1 + \cdots + s_n \mod p,$$

where p is a prime large enough that the probability of guessing s_i is negligible.

The modulo operator is used to make sure that each individual share does not leak information about s. Specifically, the shares are uniformly distributed on $0, \ldots, p-1$ and $p > s$. In our method, we will use a sharing of zero to achieve privacy.

3.2 Proposed method: Solution to P1

To present the idea in the method, consider a clique C_k, $k \in \{1, \ldots, m\}$. The approach is for each agent a_j for $j \in C_k$ to add a uniformly random number r_j to x_j before sending it to the other agents in the k'th clique. In this way, x_j is not revealed to the neighbors of a_j, since $x_j + r_j$ is a uniformly random number. Designing the random r_j values such that $\sum_{j \in C_k} r_j = 0$, ensures that the sum of the values in the cliques can still be learned by the agents in the clique.

To improve efficiency, we introduce a two-phase algorithm. The first phase is a *preprocessing phase* where the described uniformly random numbers are chosen; hence, this phase can be carried out in advance of the actual execution which boost efficiency at run-time. The second phase is the *execution phase*, where the private values are involved.

We explain the preprocessing phase with a numerical example, see Example 1.

Example 1. Let a clique consist of the agents a_1, a_2 and a_3 and let $p = 23$. Table 1 illustrates the steps in the preprocessing phase. The first step is that each agent chooses 3 random numbers which must sum to zero; this is shown in the first column. Then each agent sends one random number to each agent (including itself), the random number received by each agent is shown in the second column. The last column shows for each agent the sum of received numbers modulo 23, which is the result of the preprocessing phase. Note that the sum of the last column modulo 23 is zero.

	Choose uniformly random numbers	**Recieved numbers**	**Sum of rec. numbers**
Agent 1	0 = 15+5+3 mod 23	15,10,8	15+10+8 mod 23 = 10
Agent 2	0=10+6+7 mod 23	5,6,9	5+6+9 mod 23 = 20
Agent 3	0=8+9+6 mod 23	3,7,6	3+7+6 mod 23 = 16

Table 1. Overview of the steps each agent take per clique in the preprocessing phase. In this example there is one clique consisting of three agents.

Example 2 continues Example 1, and shows the steps the agents take in the execution phase. As seen, all agents learn the sum of private values without exposing them.

Example 2. This example is a continuation of Example 1. Assume the agents have secret values, $x_1 = 5, x_2 = 2, x_3 = 10$. The goal is to find the sum of secret values in the clique, thus all agents should end up with learning the number 17, as $5 + 2 + 10 = 17$. Table 2 illustrates the execution phase that would follow the preprocessing phase in Example 1. The first step is for each agent to add their secret value to the random number generated in the preprocessing phase; this is shown in the first column. Then each agent sends this sum to all other agents; the second column shows the numbers each agent receives. The last colunmn shows the computed sum, which is the result of the execution phase.

	Add x_i to s_{i_1}	**Received numbers**	**Sum of rec. numbers % 23**
Agent 1 $x_1 = 5$	5+10 mod 23 = 15	15,22,3	17
Agent 2 $x_2 = 2$	2 +20 mod 23 = 22	15,22,3	17
Agent 3 $x_3 = 10$	10 + 16 mod 23 = 3	15,22,3	17

Table 2. Overview of the steps each agent take per clique in the execution phase, which would follow the steps in the preprocessing phase shown in Example 1.

There is still one thing that must be taken into account in the algorithm. If agent $a_j, j \in \mathcal{N}_i$ is part of more than one clique that includes a_i, then x_j would be added to the sum, (8), more than one time. To see this, consider the graph depicted in Fig. 2. Attempting to compute $y_1 = x_1 + x_2 + x_3 + x_4$ by the described method would result in the computation of $(x_1 + x_2 + x_3) + (x_1 + x_2 + x_4) \neq y_1$. As seen, x_1 and x_2 are added twice. x_1 can be subtracted since a_1 knows this value, however, a_1 cannot subtract x_2. To circumvent this, a_2 must add only half of x_2 in each of the two cliques.

In Protocol 1, the privacy preserving solution to P1 is formally presented from the view of agent a_i, where we use the notation introduced in section 2.

Protocol 1 Privacy Preserving Solution to P1

```
$p > \max_i(\sum_{j\in\mathcal{N}_i} x_j)$ is a public prime and $D = \{0, \dots, p-1\}$.
1: For each $a_j$, $j \in \mathcal{N}_i$, $a_i$ chooses $\lambda_{i,j,k}, k \in C_{i,j}$ such that
       $$\left(\sum_{k\in C_{i,j}} \lambda_{i,j,k}\right) \mod p = 1.$$
   Preprocessing
2: For each $k \in Ci$, $a_i$ draws from $D$ a uniformly distributed number $r_{i,j,k}$ for each agent $j \in C_k$ (including itself), such that
       $$\left(\sum_{j\in C_k} r_{i,j,k}\right) \mod p = 0, \quad k \in Ci.$$
3: $a_i$ sends $r_{i,j,k}$ to agent $a_j$ for $j \in C_k$ and $k \in Ci$.
4: Upon receiving $r_{j,i,k}$ from each agent $a_j$, $j \in C_k$, $k \in Ci$, agent $a_i$ computes the following sum for each clique $C_k$, $k \in Ci$,
       $$s_{i,k} = \left(\sum_{j\in C_k} r_{j,i,k}\right) \mod p, \quad k \in Ci.$$
   Execution
5: For each $k \in Ci$, and each $j \in C_k$, $a_i$ computes
       $$p_{i,j,k} = (\lambda_{i,j,k}x_i + s_{i,k}) \mod p.$$
6: $a_i$ sends $p_{i,j,k}$ to each agent $a_j, j \in C_k$ for each $k \in Ci$.
7: Upon receiving $p_{j,i,k}$ for each $j \in C_k, k \in Ci$, $a_i$ computes
       $$y'_k = \sum_{j\in C_k} p_{j,i,k} \mod p, \quad k \in Ci.$$
8: $a_i$ computes
       $$y_i = \left(\sum_{k\in Ci} y'_k\right) \mod p. \qquad (9)$$
```

We now provide a sketch of the proof of correctness and privacy of Protocol 1.

Sketch of Proof. Correctness. The protocol gives the correct result if a_i learns

$$\left(\sum_{j\in\mathcal{N}_i} x_j\right) \mod p, \qquad (10)$$

which is equal to the sum in (8), since we choose $p > \sum_{j\in\mathcal{N}_i} x_j$. Consider the following rewrite of (9), where each equation is modular p even though we omit it to improve notation;

$$
\begin{aligned}
y_i &= \left(\sum_{k\in Ci} y'_k\right) = \sum_{k\in Ci}\sum_{j\in C_k} p_{j,i,k}\\
&= \sum_{k\in Ci}\sum_{j\in C_k} (\lambda_{j,i,k}x_j + s_{j,k})\\
&= \sum_{k\in Ci}\left(\sum_{j\in C_k}\lambda_{j,i,k}x_j + \sum_{j\in C_k}\sum_{i\in C_k} r_{i,j,k}\right)\\
&= \sum_{k\in Ci}\left(\sum_{j\in C_k}\lambda_{j,i,k}x_j + \sum_{i\in C_k}\underbrace{\sum_{j\in C_k} r_{i,j,k}}_{=0}\right) \qquad (11)\\
&\overset{(*)}{=} \sum_{j\in\mathcal{N}_i}\sum_{k\in C_{i,j}}\lambda_{j,i,k}x_j\\
&= \sum_{j\in\mathcal{N}_i} x_j \underbrace{\sum_{k\in C_{i,j}}\lambda_{j,i,k}}_{=1} = \sum_{j\in\mathcal{N}_i} x_j,
\end{aligned}
$$

where $(*)$ is because summing over the cliques a_i is in ($k \in Ci$) and all the agents in each of those cliques ($j \in C_k$) is equivalent to summing over all the neighbors a_j of a_i ($j \in \mathcal{N}_i$) and the cliques both a_i and a_j are in ($k \in C_{i,j}$) since we assume that all neighbors to a_i is part of a clique that includes a_i. This proves the correctness of the protocol.

Privacy. We consider the execution phase as the preprocessing phase does not involve any private values. For each $k \in Ci$, a_i sends $(\lambda_{i,j,k}x_i + s_{i,k}) \mod p$ to each a_j, $j \in C_k$. This communication is privacy preserving since $s_{i,k} = \sum_{j\in C_k} r_{j,i,k} \mod p$, is a uniformly random number, known only by a_i. For a_j receiving $v = (\lambda_{j,i,k}x_i + s_{i,k}) \mod p$, the information a_j gets is

$$
\underbrace{\lambda_{j,i,k}}_{\text{known to } a_j} x_i = \underbrace{v}_{\text{known to } a_j} - \underbrace{r_{j,i,k}}_{\text{known to } a_j} - \underbrace{\sum_{j'\in C_k, j'\neq j} r_{j',i,k}}_{\text{unknown to } a_j}, \qquad (12)
$$

Since $|C_k| \geq 3$, $\sum_{j'\in C_k,j'\neq j} r_{j',i,k}$ will at least consist of two uniformly random numbers each of a probability of $\frac{1}{p}$. Hence, by choosing p large, the probability of a_j guessing the last term in (12) is negligible.

□

4. PRIVACY PRESERVING SOLUTION TO P2

The algorithm for solving P2 without leaking private data is based on Protocol 1. Actually, the only difference is for agents that are not part of a clique in a given neighborhood. To clarify, consider Fig. 3, where a_4 is a neighbor to a_1 but a_4 does not form a clique with a_1 and another agent in $\mathcal{N}_1$. For a_1 to learn $x_2 + x_5 + x_4$ without learning individual values in the sum, we need to extend Protocol 1. Our propose is to create a so-called *virtual clique* between a_j, a_i and one other agent, $a_{j'}$, in $\mathcal{N}_i$. In Definition 1, we define what is meant by a virtual clique.

Definition 1. (Virtual Clique). *Let a_1, a_2 and a_3 be agents in a multi-agent network and let there be a communication link between a_1 and a_2 and between a_1 and a_3. A virtual clique between a_1, a_2 and a_3 is made by letting a_2 encrypt its messages to a_3 such that only a_3 can decrypt them. a_2 sends the encrypted messages to a_1 who forwards to a_3 and vice versa for messages from a_3 to a_2.*

In continuation, let $V_m, \ldots, V_v$, be the virtual cliques in the multi-agent network, where m is the number of cliques, thus the indices of the virtual cliques starts from the last index of the regular cliques. Let Vi be defined similarly to Ci and define for each agent, a_i, the set $V_{i,j}, j \in \mathcal{N}_i$, such that for all $k \in Ci \cup Vi$, $k \in V_{i,j}$ iff $a_j \in C_k \cup V_k$. For encrypting messages, one can chose among many schemes, see for instance the study of encryption algorithms by Singh and Supriya (2013).

The protocol for solving $P2$ is formally written in Protocol 2 from the perspective of agent a_i.

Protocol 2 Privacy Preserving Solution to P2

```
    $p$ and $D$ are as in Protocol 1.
 1: $a_i$ determines, for each $a_j, j \in \mathcal{N}_i$, the values $\{\lambda_{i,j,k}\}, k \in V_{i,j}$ such that
        $\sum_{k\in V_{i,j}} \lambda_{i,j,k} \mod p = 1, \quad j \in \mathcal{N}_i.$   (13)
 2: $a_i$ executes Protocol 1 to compute the sum,
        $r1_i = \sum_{k\in Ci}\sum_{j\in C_k} x_j,$
    using the $\lambda_{i,j,k}$ values in (13).
    Preprocessing
 3: Through a secret key generation process, $a_i$ determines the secret keys $g_k$ for $k \in Vi$ for encrypted communication in the virtual cliques.
 4: For each $k \in Vi$, $a_i$ chooses two uniformly distributed random numbers $r_{i,k,0}$ and $r_{i,k,1}$ from $D$ such that
        $(r_{i,k,0} + r_{i,k,1}) \mod p = 0.$
 5: For each $k \in Vi$, $a_i$ encrypts $r_{i,k,1}$ using $g_k$ and sends $enc(r_{i,k,1})_{g_k}$ to agent $a_j$, $j \in V_k \cap \mathcal{N}_i$.
 6: Upon receiving $enc(r_{j,k,1})_{g_k}$, $a_i$ decrypts and computes the sum
        $s_{i,k} = (r_{i,k,0} + r_{j,k,1}) \mod p.$
    Execution
 7: For each $k \in Vi$, $a_i$ computes
        $p_{i,j,k} = \lambda_{i,j,k}x_i + s_{i,k} \mod p, \quad j \in V_k \cap \mathcal{N}_i, j \neq i.$
 8: $a_i$ sends $p_{i,j,k}$ to $a_j$, $j \in V_k \cap \mathcal{N}_i, j \neq i$, $k \in Vi$.
 9: Upon receiving $p_{j,i,k}$ $a_i$ computes
        $y'_{i,k} = \sum_{j\in V_k, j\neq i} p_{j,i,k} \mod p, \quad k \in Vi.$
10: Finally, $a_i$ adds the result from the cliques with the result from the virtual cliques,
        $y_i = r1_i + \sum_{k\in Vi} y'_{i,k} \mod p.$   (14)
```

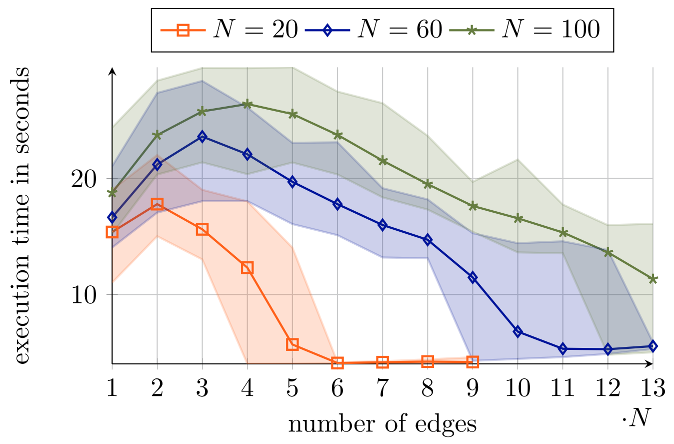

Fig. 4. Execution time versus number of edges in the graph. Note that the x-axis is scaled by N.

The correctness of Protocol 2 follows from the correctness of Protocol 1. Considering only the execution phase, showing the privacy of Protocol 2 is equivalent to showing the privacy of Protocol 1. The distinction lies in the creation of the random numbers in the preprocessing phase since in Protocol 2 agents in a virtual clique with no comunication link between them needs to send messages through their common neighbor. However, since we encrypt these messages, we make sure that the common neighbor cannot learn the messages which would otherwise break the privacy.

5. SCALABILITY

To illustrate the scalability of Protocol 2, we have conducted simulations showing execution time as a function of a specific graph constellation. The simulations are carried out on a 2.70 GHz laptop, where one thread is created for each agent in the simulated network. For this reason, the absolute execution times may be misleading, as one would expect the execution time to be lower if each agent were given individual machines. However, the execution times are comparable to each other, thus providing an illustration of scalability.

Fig. 4 shows how the execution time is affected by the number of edges in the graph. The orange line shows an average simulation of the protocol, where the number of agents $N = 20$ and the execution time is measured with the number of edges in the graph being equal to $k \cdot N$ for $k = 1, \ldots, 13$. It may be counter-intuitive, that the execution time decreases as edges are added to the network. However, this is explained by the fact that as edges are added, the number of virtual cliques in the graph is decreasing, resulting in a faster computation time.

The same goes for the blue line, where the number of agents is 60 and the green line where the number of agents is 100.

6. CONCLUSION

The paper presents privacy preserving protocols for calculating a sum function among neighbors in a connected graph where each agent can only communicate with their immediate neighbors. The result of the paper can be directly applied in existing decentralized protocols (where agents need the sum of its neighbors values) for achieving privacy. The protocols has a constant number of communication rounds and simulations show great scalability. For future work, it will be interesting to consider other functions of neighbors values than the sum function. To this end, the simple additive secret sharing protocol can be substituted by Shamir's secret sharing scheme.

REFERENCES

Banjac, G., Rey, F., Goulart, P., and Lygeros, J. (2019). Decentralized resource allocation via dual consensus admm. *2019 American Control Conference (ACC)*. doi: 10.23919/acc.2019.8814988.

Chang, T., Hong, M., and Wang, X. (2015). Multi-agent distributed optimization via inexact consensus admm. *IEEE Transactions on Signal Processing*, 63(2), 482–497. doi:10.1109/TSP.2014.2367458.

Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., and Zhu, M. (2002). Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations Newsletter*, 4. doi: 10.1145/772862.772867.

Cramer, R., Damgaard, I.B., and Nielsen, J.B. (2015). *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 1 edition. ISBN: 978-1-107-04305-3.

Franceschelli, M., Gasparri, A., Giua, A., and Seatzu, C. (2009). 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*, 2717–2722. doi:10.1109/CDC.2009.5400723.

Liu, C., Li, H., and Shi, Y. (2019). Resource-aware exact decentralized optimization using event-triggered broadcasting. URL `https://arxiv.org/abs/1907.10179v2`. In press.

Ma, M., Nikolakopoulos, A.N., and Giannakis, G.B. (2018). Hybrid admm: a unifying and fast approach to decentralized optimization. *EURASIP Journal on Advances in Signal Processing*, 2018(1), 73. doi:10.1186/s13634-018-0589-x.

Mehnaz, S., Bellala, G., and Bertino, E. (2017). A secure sum protocol and its application to privacy-preserving multi-party analytics. In *Proceedings of the 22Nd ACM on Symposium on Access Control Models and Technologies*, SACMAT '17 Abstracts, 219–230. ACM, New York, NY, USA. doi:10.1145/3078861.3078869. URL `http://doi.acm.org/10.1145/3078861.3078869`.

Sheikh, R., Beerendra, K., and Mishra, D. (2009). Privacy-preserving k-secure sum protocol. *International Journal of Computer Science and Information Security*, 6.

Singh, G. and Supriya (2013). Article: A study of encryption algorithms (rsa, des, 3des and aes) for information security. *International Journal of Computer Applications*, 67(19), 33–38.

Will, M.A. and Ko, R.K. (2015). Chapter 5 - a guide to homomorphic encryption. In R. Ko and K.K.R. Choo (eds.), *The Cloud Security Ecosystem*, 101 – 127. Syngress, Boston. doi:10.1016/B978-0-12-801595-7.00005-7. URL `https://doi.org/10.1016/B978-0-12-801595-7.00005-7`.