Simple, Private, and Accurate Distributed Averaging

Israel Donato Ridgley
Electrical & Computer Engineering
Northwestern University
Evanston, IL USA
israelridgley2023@u.northwestern.edu

Randy A. Freeman
Electrical & Computer Engineering
Northwestern University
Evanston, IL USA
freeman@ece.northwestern.edu

Kevin M. Lynch
Mechanical Engineering
Northwestern University
Evanston, IL USA
kmlynch@northwestern.edu

Abstract—Some distributed optimization applications require privacy, meaning that the values of certain parameters local to a node should not be revealed to other nodes in the network during the joint optimization process. A special case is the problem of private distributed averaging, in which a network of nodes computes the global average of individual node reference parameters in a distributed manner while preserving the privacy of each reference. We present simple iterative methods that guarantee high accuracy (i.e. the exact asymptotic computation of the global average) and high privacy (i.e. no node can estimate another node’s reference value to any meaningful degree). To achieve this, we assume that the digraph modeling the communication between nodes satisfies certain topological conditions. Other related methods in the literature also achieve high accuracy and privacy, but under topological conditions more restrictive than ours. Moreover, our method is simpler because it does not require any initial scrambling phase, it does not inject any noise or other masking signals into the distributed computation, it does not require any random switching of edge weights, and it does not rely on homomorphic encryption.

Index Terms—distributed averaging, privacy, consensus

I. INTRODUCTION

Privacy in distributed computation is becoming increasingly important as more systems become decentralized. Within systems such as the smart grid, smart transportation, and smart healthcare, there is an obvious need to protect user information from being revealed to other agents. In other systems such as sensor networks deployed in sensitive environments, there is a need to keep the information on each node private from other (potentially compromised) nodes. The particular example of distributed averaging studied in this paper is a basic component of many distributed computations, and the problem of maintaining privacy during distributed averaging has received significant recent attention [1]–[12].

II. BACKGROUND AND RELATED WORK

In this work, we consider networks of honest-but-curious nodes [6], [12] that cooperatively compute the global average of their reference values. This means all nodes in the network participate faithfully in the averaging algorithm but would like to reconstruct the reference values of other nodes. We do allow the possibility of collusion, i.e., nodes sharing data not specified by the algorithm, possibly over side channels.

One popular approach to private distributed averaging is based on the concept of differential privacy from the database literature (see [4], [9], [13] and the references therein). These methods use added noise to obfuscate an individual’s contribution to the computation in a manner that produces a modest impact on the desired result. This approach offers the potential to ensure some level of privacy against any amount of collusion [13]; however, it also requires a trade-off between accuracy and privacy: the more accurate the computation, the less private the input data. In particular, for distributed averaging with differential privacy, it is impossible to achieve both high privacy and high accuracy [4]. Thus the differential privacy approach might not be suitable for applications in which high accuracy is required.

Approaches to distributed averaging that achieve both high privacy and high accuracy put topological restrictions on the digraph modeling the communication between nodes. Ours is one such approach, but our topological restrictions are significantly more relaxed than those appearing in other work. For example, if each node has at least two out-neighbors and at least one of these is not colluding with other nodes, then our method provides both high privacy and high accuracy. To achieve privacy under our relatively weak topological restrictions, we employ a method that is similar to the state expansion method in [12] but that instead uses the concept of the line digraph of a given digraph [14]. In particular, the nodes of the line digraph are precisely the edges of the original digraph.

Our approach achieves privacy by using the unobservable subspace of a discrete-time LTI system in state-space form. Specifically, we show that other nodes’ reference values are completely unobservable from the data received at a given node. In particular, no estimator running on a given node during the computation can produce any meaningful estimates of other nodes’ reference values.

Many recent results are based on the standard averaging protocol introduced in [15]. This standard protocol is not private, as it requires nodes to communicate their reference values to their neighbors at the initial update step. Several methods have been proposed to modify this standard protocol so that it preserves privacy. Some are based on the fact that the protocol computes the correct average whenever the sum of the initial internal states is equal to the sum of the reference values, not just when each node sets its initial state equal to its own reference value. Thus if an initial phase of the protocol allows the nodes to scramble their initial states while maintaining both privacy and the initial
sum condition, then the standard protocol applied after this initial phase is complete will still compute the desired average. This approach is developed in [6], [7], [11], and a related method in [12] involves the scrambling of initial edge weights rather than initial states. Other approaches employ masking, that is, nodes transmit not their actual states to their neighbors but rather masked or noisy versions of their states [1]–[3], [8], [10]. In these approaches, the masking signals are chosen carefully so that they guarantee privacy while ensuring that their effect on the calculation of the average vanishes in the limit. None of these methods are hot-pluggable: if nodes enter or leave the network, if they change their reference values at some point in time, or if there is some fault in communication or computation, then these algorithms must start the computation over again from the beginning (and must somehow coordinate to do so).

Still other approaches employ homomorphic encryption to maintain privacy. These methods typically rely on a trusted node [16], add computation and communication overhead that may not be acceptable for low power systems [5], [17], and are difficult to extend to more general distributed optimization problems due to the restrictive nature of the homomorphic cryptosystem and the complexity of homomorphic computing.

In contrast to all of this previous work, our method is hot-pluggable, it does not require any initial scrambling phase, it does not inject any noise or other masking signals into the computation, it does not require any random switching of the edge weights, and it does not use homomorphic encryption. It is based on a simple method described in [18] for dynamic average consensus.

III. Distributed Averaging with Privacy
A. Notation and terminology

We model a network of $n$ agents participating in distributed computation as a digraph $G = (V, E)$, where $V = \{1, \ldots, n\}$ is the set of nodes (or vertices) and $E$ is the set of directed edges, where each such edge is an ordered pair of distinct nodes. There exists an edge $(i, j) \in E$ if and only if node $i$ can send information to node $j$, so that the edge direction corresponds to the communication direction. The digraph has no self-loops: even though a node can communicate with itself, this internal communication is not modeled by the digraph. The sets $N^\text{in}_i, N^\text{out}_i \subset V$ denote the sets of in- and out-neighbors of node $i$ (respectively).

We let $a_{ij}$ denote the weight on edge $(i, j)$, with $a_{ij} > 0$ if $(i, j) \in E$ and $a_{ij} = 0$ otherwise. An unweighted digraph has $a_{ij} = 1$ for all $(i, j) \in E$. The weighted in- and out-degrees of node $i$ are $d^\text{in}_i = \sum_{j \in \mathbb{N}^\text{in}_i} a_{ji}$ and $d^\text{out}_i = \sum_{j \in \mathbb{N}^\text{out}_i} a_{ij}$ (respectively). The weighted in- and out-degree matrices $D^\text{in}$ and $D^\text{out}$ are the $n \times n$ diagonal matrices whose $i$th diagonal entries are $d^\text{in}_i$ and $d^\text{out}_i$ (respectively). The digraph is weight-balanced when $D^\text{in} = D^\text{out}$. The $n \times n$ in- and out-Laplacians are given by

$$L^\text{in} = D^\text{in} - [a_{ij}]_{n \times n}$$

$$L^\text{out} = D^\text{out} - [a_{ij}]_{n \times n}$$

where $[a_{ij}]_{n \times n}$ is the $n \times n$ weighted adjacency matrix. Note that $L^\text{in}$ has zero column sums and $L^\text{out}$ has zero row sums. Also, $L^\text{in} = L^\text{out}$ when the digraph is weight-balanced.

We let $\mathbf{e}_i$ denote the vector of all ones (with size determined by context). We let $e_i$ denote the $i$th column of the identity matrix $I$ (again with size determined by context).

B. Assumptions and results

We suppose each node $i$ in the digraph $G$ has a constant vector reference $r_i \in \mathbb{R}^p$, and we let $r_{\text{ave}} = \frac{1}{n} \sum_{i=1}^n r_i$ denote the global average. All nodes are honest but curious [6], [12], which means the following:

1) they all want to compute $r_{\text{ave}}$ accurately and thus they participate faithfully in the averaging algorithm, and

2) they all want to discover the reference values of other nodes.

We suppose there are two types of nodes, conformists and colluders. Conformists wish to preserve the privacy of their reference values and thus they only send and receive data as specified by the averaging algorithm. Colluders don’t care about the privacy of their reference values and will share all information they have with other colluders (possibly over side channels not modeled by $G$). Our main assumptions are as follows:

(A1) The digraph $G$ is strongly connected, has at least three nodes, and has at least two conformist nodes.

(A2) Each node $i$ knows its out-neighbor set $N^\text{out}_i$.

(A3) Nodes can send individual messages to each of their out-neighbors, without the other out-neighbors hearing.

(A4) For each conformist node $i$, either

a) $i$ has at least two out-neighbors, and at least one of these is conformist, or

b) for all $j \neq i$ there exists $k \in \{i\} \cup N^\text{in}_i$ such that:

i) if $j$ is a conformist node then $k \not\in \{j\} \cup N^\text{in}_i$, and

ii) if $j$ is a colluder node then $k \not\in \{i\} \cup N^\text{in}_i$ for all colluder nodes $\ell$.

Without assumption (A1) it is either trivial or impossible to achieve both privacy and accuracy. Assumptions (A4)(a) and (A4)(b) describe two different local topological restrictions on the digraph $G$, but only one of these needs to be satisfied for a given conformist node $i$ (there is an “or” between these conditions, not an “and”). Also, some conformist nodes might satisfy (A4)(a) while others satisfy (A4)(b), i.e., the choice between these two options need not be uniform. An example of a digraph $G$ satisfying (A4)(a) for each conformist node is any complete digraph with $n \geq 3$ having at least two conformist nodes. An example of a digraph $G$ satisfying (A4)(b) for each conformist node is any directed cycle graph with $n \geq 3$ for which every conformist node is either in- or out-adjacent to another conformist node, such as the digraph shown in Fig. 1.

We will present a distributed, synchronous, discrete-time algorithm to be run on each node such that under these assumptions (A1)–(A4) we achieve the following:
Fig. 1. A digraph with blue conformist nodes and red colluder nodes such that each conformist node satisfies (A4)(b).

Accuracy: each node’s output signal converges exponentially to the exact global average $r_{\text{ave}}$.

Privacy: if $i$ is a conformist node and $j$ is some other node, then for every vector $\delta \in \mathbb{R}^p$ there exist two different overall algorithm trajectories that produce the same data at node $j$ but for which the associated reference values $r_i$ differ by $\delta$.

We will actually present two different algorithms, a simple base algorithm that only guarantees privacy when each conformist node satisfies (A4)(b) (like in Fig. 1), and a slightly more complex expanded algorithm that requires more communication between nodes but that allows both options (A4)(a) and (A4)(b).

Remark 1. We have similar results for the case in which (A2)–(A3) are not satisfied, that is, when nodes don’t know who their out-neighbors are and simply broadcast their data to any nodes that can hear them. In this case, however, the digraph $G$ must be weight-balanced, and we disallow option (A4)(a) and instead always require option (A4)(b). For details, see Remark 6 below.

Remark 2. Our algorithms are hot-pluggable and robust in the following sense: if nodes enter or leave the network, if they change their reference values at some point in time, or if there is some fault in communication or computation, then the algorithms automatically recover to produce the correct result without having to reinitialize (provided such changes do not occur on a persistent basis).

C. The base algorithm

All nodes in the digraph $G$ choose their outgoing weights so that the weighted out-degrees satisfy the following:

\[ d_{i_{\text{out}}}^i \leq 1 \text{ for each node } i. \]

These weights need not be private. Each node $i$ will run a local discrete-time system with input $u_i$, output $y_i$, and internal state $x_i$, all taking values in $\mathbb{R}^q$ for some $q$ (to be specified later). These will all be functions of the discrete time value $t = 0, 1, 2, \ldots$. At each time $t$, node $i$ sends the vector $a_i x_i[t]$ individually to each of its out-neighbors $j$; likewise, it receives the vector $a_k x_k[t]$ from each of its in-neighbors $k$. It then uses this received data to update its local signals as follows:

\[ y_i[t] = u_i[t] - d_i^{\text{out}} x_i[t] + \sum_{k \in X_i^+} a_k x_k[t] \quad (3) \]
\[ x_i[t+1] = x_i[t] + y_i[t]. \quad (4) \]

If we stack the vector signals by defining

\[ u[t] = \begin{bmatrix} u_1[t] \\ \vdots \\ u_n[t] \end{bmatrix}, \quad y[t] = \begin{bmatrix} y_1[t] \\ \vdots \\ y_n[t] \end{bmatrix}, \quad x[t] = \begin{bmatrix} x_1[t] \\ \vdots \\ x_n[t] \end{bmatrix}, \quad (5) \]

then we can write the collection of all such updates (3)–(4) as

\[ y[t] = u[t] - (L^T \otimes I)x[t] \quad (6) \]
\[ x[t+1] = x[t] + y[t]. \quad (7) \]

Here $L$ denotes the out-Laplacian in (2). A block diagram of the global update equations (6)–(7) is shown in Fig. 2. Before we specify how we choose the inputs $u_i$, we give a generic convergence result for this algorithm in Theorem 3. In the statement of this result, the vector $v \in \mathbb{R}^n$ denotes the unique vector satisfying $L^T v = 0$ and $1^T v = 1$. The Perron-Frobenius theorem together with (A1) guarantee both the existence and uniqueness of $v$ and the property that all entries in $v$ are positive.

Theorem 3. If $u[t] \to u^*$ as $t \to \infty$, then

\[ \lim_{t \to \infty} y[t] = (v \otimes 1^T \otimes I)u^*. \quad (8) \]

Furthermore, if the convergence of $u[\cdot]$ is exponential, then so is the convergence of $y[\cdot]$.

Proof. The matrix $I - L^T$ has a simple eigenvalue at 1 with eigenvector $v$. It follows from (A5) and the Gershgorin circle theorem that all other eigenvalues of $I - L^T$ lie strictly inside the unit circle in the complex plane. Thus there exist a matrix $Q \in \mathbb{R}^{n \times (n-1)}$ and a Schur matrix $F \in \mathbb{R}^{(n-1) \times (n-1)}$ such that

\[ T^{-1}(I - L^T)T = \begin{bmatrix} 1 & 0 \\ 0 & F \end{bmatrix}, \quad (9) \]

where $T = [v \ Q]$. We can change coordinates $w = (T^{-1} \otimes I)x$ to obtain the update

\[ w[t+1] = \begin{bmatrix} I & 0 \\ 0 & F \otimes I \end{bmatrix}w[t] + (T^{-1} \otimes I)u[t]. \quad (10) \]

If we let $\hat{w}$ denote the bottom $(n - 1)q$ entries of $w$, then because $F$ is Schur there exists $\hat{w}^*$ such that $w[t] \to \hat{w}^*$ as $t \to \infty$. It follows that $(L^T \otimes I)x[t] \to \xi^*$ as $t \to \infty$, where

\[ \xi^* = (L^T Q \otimes I)\hat{w}^*. \]

It then follows from (6) that $y[t] \to y^*$ as $t \to \infty$, where

\[ y^* = u^* - \xi^*. \]

Next, we see from (6)–(7) that

\[ y[t+1] = y[t] - (L^T \otimes I)y[t] + u[t+1] - u[t]. \quad (11) \]

Therefore because both $y$ and $u$ converge we must have

\[ (L^T \otimes I)y^* = 0, \]

and hence there exists a vector $\eta^*$ such
that \( y^* = v \otimes \eta^* \). Now it is clear that \( (1^T \otimes I)\xi^* = 0 \) which means \( \eta^* = (1^T \otimes I)y^* = (1^T \otimes I)u^* \), and the result follows (because the system is LTI, we get exponential convergence when the input converges exponentially).

Now suppose each node \( i \) chooses a constant input as

\[
u_i[\cdot] \equiv \begin{bmatrix} r_i \\ 1 \end{bmatrix} \in \mathbb{R}^{p+1},
\]

so that \( q = p + 1 \). Then we conclude from Theorem 3 that its output \( y_i \) will converge exponentially to the constant

\[
y^{*}_i = \begin{bmatrix} v_i \sum_{j=1}^n r_j \\ v_i n \end{bmatrix},
\]

where \( v_i \) denotes the \( i \)th component of the vector \( v \) (which satisfies \( v_i > 0 \)). Thus by taking the ratio of the appropriate values in \( y_i^* \), node \( i \) can compute the global average \( r_{ave} \). This approach is related to the “push-sum” or “ratio consensus” methods for distributed averaging over directed graphs (see [19] and the references therein).

**Remark 4.** According to Theorem 3, the final value of the output \( y \) is independent of the initial state \( x[0] \). This is a key feature of the algorithm and is the mechanism by which privacy is possible without using any noise/masking, weight switching, initial scrambling, or homomorphic encryption. This feature also makes the algorithm hot-pluggable and robust in the sense of Remark 2.

**Remark 5.** The base algorithm is not internally stable: even though the output \( y \) converges, the state \( x \) does not. Indeed, we see from the update equation (7) that \( x \) grows linearly in \( t \) as \( y \) converges to a constant. Such linear growth may cause numerical problems if the algorithm is run over long time periods. As explained in [18], it is possible to modify this base algorithm so that the output \( y \) converges to the same value while keeping \( x \) bounded, but at the price of slower convergence.

**Remark 6.** If each node assigns its incoming weights instead of its outgoing weights, then assumptions (A2) and (A3) are no longer necessary (nodes can just communicate their unweighted states). In this case, however, the digraph \( G \) must be weight-balanced for accurate averaging. Note that for a weight-balanced digraph we have \( v = \frac{1}{n} \mathbb{1} \), which means nodes can compute the global average \( r_{ave} \) directly without having to append the reference value \( r_i \) with the constant \( 1 \) as in (12) (and without having to take a ratio). Because distributed weight-balancing can be a challenge for directed graphs without assumption (A2), this scenario is best suited for undirected graphs (i.e., graphs \( G \) representing two-way communication and having symmetric weights \( a_{ij} = a_{ji} \)). Because the expanded algorithm described below generally results in directed graphs, this scenario will also disallow option (A4)(a) and instead always require option (A4)(b).

**Remark 7.** The base algorithm is cascadable [18], meaning it allows the exact asymptotic tracking (with no delay) of time-varying reference signals that are polynomial functions of \( t \) up to some known degree. For example, to track averages of reference signals that are quadratic functions of \( t \), we simply cascade three of the systems shown in Fig. 2.

### D. Privacy for the base algorithm

In this section we show that the base algorithm guarantees the privacy of the conformist nodes' reference values when each conformist node satisfies (A4)(b) (like in Fig. 1).

Each node can choose an arbitrary initial state \( x[0] \) in the base algorithm, but this choice should not be made public and it should be difficult for other nodes to obtain a good a priori estimate of it. For example, it might be drawn at random from a distribution having infinite or undefined moments.

To analyze the privacy of the base algorithm, we rewrite the dynamics (6)–(7) as

\[
\begin{bmatrix} x[t+1] \\ u[t+1] \end{bmatrix} = A \begin{bmatrix} x[t] \\ u[t] \end{bmatrix} + B\Delta u[t]
\]

where \( A \) and \( B \) are the matrices

\[
A = \begin{bmatrix} (I - L^T) \otimes I & I \\ 0 & I \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ I \end{bmatrix}
\]

and \( \Delta u \) is the finite difference signal \( \Delta u[t] = u[t+1] - u[t] \). It is easy to show that for any \( k \) and any \( \varphi \in \mathbb{R}^q \), the vector

\[
\begin{bmatrix} e_k \otimes \varphi \\ L^e_k \otimes \varphi \end{bmatrix}
\]

is an eigenvector of \( A \) associated with the unit eigenvalue. In what follows, we let \( L_{ij} \) denote the entry of \( L \) in row \( i \) and column \( j \), so that \( L = [L_{ij}]_{n \times n} \). Note that (A1) implies \( L_{ii} = a_{iout} > 0 \) for each \( i \). In particular, \( L_{ij} \neq 0 \) if and only if \( i \in \{j\} \cup \mathcal{N}^\text{in}_j \).

Suppose \( i \) is a conformist node, let \( j \) be some other node, and suppose (A4)(b) holds. Let \( \xi_j \) be the collection of all signals available to \( j \). We now construct a matrix \( C_j \) such that

\[
\xi_j[t] = C_j \begin{bmatrix} x[t] \\ u[t] \end{bmatrix}
\]

for all \( t \). First suppose \( j \) is conformist; then because \( j \) can measure its own state \( x_j \) and input \( u_j \), the top part of \( C_j \) looks like

\[
\begin{bmatrix} e_j^T \otimes I & 0 \\ 0 & e_j^T \otimes I \end{bmatrix}
\]

Also, if \( h \in \mathcal{N}^\text{in}_j \) then \( C_j \) includes blocks of the form

\[
\begin{bmatrix} a_{hj} e_j^T \otimes I & 0 \\ 0 & 0 \end{bmatrix}
\]

stacked underneath each other and underneath (18). Let \( k \) be as in (A4)(b) so that \( k \in \{i\} \cup \mathcal{N}^\text{in}_j \) and \( k \notin \{j\} \cup \mathcal{N}^\text{in}_j \), i.e., so that \( L_{ki} \neq 0 \) but \( L_{kj} = 0 \). It is straightforward to show that the eigenvector (16) is in the null spaces of the matrices in (18) and (19) and thus also in the unobservable subspace of the pair \( (A, C_j) \). In additional, this eigenvector satisfies

\[
\begin{bmatrix} 0 \\ e_j^T \otimes I \end{bmatrix} \begin{bmatrix} e_k \otimes \varphi \\ L^e_k \otimes \varphi \end{bmatrix} = L_{ki} \varphi.
\]
We also have
\[ u_i[t] = \begin{bmatrix} 0 & e_i^\top \otimes I \\ e_i^\top \otimes I & 0 \end{bmatrix} x[t] \]
and it follows that there exist two trajectories of (14) that produce the same measurements \( \zeta[j][\cdot] \) but for which the inputs \( u_i[\cdot] \) differ by the constant offset \( L_k \varphi \). Because \( L_{ki} \neq 0 \) and \( \varphi \) is arbitrary, we conclude that node \( i \) keeps its input \( u_i \) private from node \( j \).

Next suppose \( j \) is a colluder node; then the matrix \( C_j \) has vertically stacked blocks of the form
\[ \begin{bmatrix} e_j^\top \otimes I & 0 \\ 0 & e_j^\top \otimes I \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} a_{i\ell} e_h^\top \otimes I & 0 \end{bmatrix} \]
for any colluder node \( \ell \) (including \( \ell = j \)) and any \( h \in N_j^{in} \). Let \( k \) be as in (A4)(b) so that \( k \in \{i\} \cup N_j^{in} \) and \( k \notin \{\ell\} \cup N_j^{in} \) for every colluder node \( \ell \), i.e., so that \( L_{ki} \neq 0 \) but \( L_{k\ell} = 0 \) for every colluder node \( \ell \). Again, it is straightforward to show that the eigenvector (16) is in the null space of matrices of the form (22), and the proof of privacy follows using the same argument as above.

**Remark 8.** Because the privacy argument is based on a standard unobservability analysis for the LTI system (14), it is clear that knowledge of the matrices \( A \) and \( B \) or of the finite difference input \( \Delta u \) provides no advantage to curious nodes wishing to reconstruct the unobservable state. In particular, the privacy result still holds when all nodes know the entire weighted Laplacian \( L \).

**E. The expanded algorithm**

In the expanded algorithm, we first use \( G \) to construct an expanded digraph \( G_e \) and then apply the base algorithm to \( G_e \). We refer to the nodes and edges of \( G_e \) as virtual nodes and virtual edges as they will exist within the nodes and edges of \( G \) itself. We construct the expanded digraph \( G_e \) from \( G \) as follows.

1) The virtual node set of \( G_e \) is the same as the edge set \( E \) of \( G \). In other words, \((i,j)\) is a virtual node of \( G_e \) if and only if it is an edge of \( G \).
2) \( G_e \) has a virtual edge from \((i,j)\) to \((k,\ell)\) if and only if either \( j = k \) or both \( i = k \) and \( j \neq \ell \).

Because virtual nodes of \( G_e \) are edges of \( G \), each such virtual node \((i,j)\) has a tail \( i \) and a head \( j \). Thus \( G_e \) is the digraph we obtain by taking the line digraph of \( G \) [14] and adding in all possible virtual edges between distinct virtual nodes that have the same tail. All virtual nodes having tail \( i \) are internal to \( i \) in the sense that their updates will run on \( i \) itself. Thus the virtual edges of \( G_e \) connecting two virtual nodes internal to \( i \) are also implemented internally within node \( i \). On the other hand, the virtual edges of \( G_e \) between virtual nodes having different tails are implemented externally over an existing edge in \( G \), so that the expanded algorithm uses precisely those communication links modeled by \( G \) itself.

Fig. 3 shows an example of how the expanded digraph \( G_e \) fits within \( G \). Note that \( G_e \) remains strongly connected.

We propose a simple weighting scheme that guarantees that (A5) holds for the expanded digraph \( G_e \). For each node \( i \) of \( G \), we let \( m_i = |N_i^{out}| \) denote its unweighted out-degree, which means \( m_i = \sum_{i=1}^n |m_i| \) is the total number of edges in \( G \) (as well as the total number of nodes in \( G_e \)). Note that (A1) implies \( m_i > 1 \) for each \( i \). In this weighting scheme, each node \( i \) of \( G \) chooses positive internal and external gains \( y_i^{int} \) and \( y_i^{ext} \) such that \( y_i^{int} + y_i^{ext} < 1 \). Next, suppose \( j \) is an out-neighbor of \( i \) in \( G \). Then \( i \) chooses internal out-going weights \( a_{i,j}(i,k) \) for each \((i,k) \neq (i,j)\) and external out-going weights \( a_{i,j}(j,\ell) \) for each \((j,\ell) \) as follows:

\[
a_{i,j}(i,k) = \frac{y_i^{int}}{m_j - 1}, \quad a_{i,j}(j,\ell) = \frac{y_i^{ext}}{m_j}.
\]

There is no division by zero in (23) because if \( m_j = 1 \) then there are no other virtual nodes \((i,k) \neq (i,j)\). Also, node \( i \) does not need to know \( m_j \) to implement these edge weights; it simply sends \( y_i^{ext} x_{i,j}(\ell)[t] \) to node \( j \) at each time \( t \), and then node \( j \) splits up this value equally among all of its internal virtual nodes \((j,\ell)\).

Next, we assign a reference value to each virtual node of \( G_e \) so that the base algorithm running on \( G_e \) produces the correct global average \( r_{ave} \). In the expanded algorithm, each reference value will be time varying but will converge exponentially to a constant. The reference value \( r_{i,j}[\cdot] \) that we assign to a virtual node \((i,j)\) of \( G_e \) is

\[
r_{i,j}[t] = \frac{\hat{m}_{ave,i}[t]}{m_i} r_i,
\]

where \( \hat{m}_{ave,i}[t] \) is node \( i \)'s estimate at time \( t \) of the global average \( m_{ave} = \frac{1}{n} \sum_{i=1}^n m_i \). Note that the right-hand side of (24) does not depend on \( j \), so all virtual nodes internal to a given node in \( G \) have the same reference value. The nodes compute these estimates \( \hat{m}_{ave,i}[\cdot] \) by running a parallel base algorithm on \( G \), so that each of these estimates converges exponentially to \( m_{ave} \) (according to Theorem 3). Thus in the limit we obtain

\[
r_{i,j}[\infty] = \lim_{t \to \infty} r_{i,j}[t] = \frac{m_{ave}}{m_i} r_i.
\]
Because there are $m_i$ copies of each reference value $r(i,j)$ in $G_e$, we can use (25) to compute the global average of all the reference values for $G_e$ in the limit as $t \to \infty$ as follows:

$$\frac{1}{ne} \sum_{(i,j) \in E} r(i,j)[\infty] = \frac{m_{\text{ave}}}{ne} \sum_{i=1}^n r_i = r_{\text{ave}}. \quad (26)$$

In other words, the expanded algorithm produces the correct global average. Note that we don’t care about privacy in the parallel computation of $m_{\text{ave}}$ because the out-degrees $m_i$ have nothing to do with the reference values $r_i$. Also, it is important to note that we do not employ an initial phase to compute $m_{\text{ave}}$; instead, each virtual node $(i,j)$ of $G_e$ uses the time-varying input

$$u(i,j)[t] = \begin{bmatrix} r(i,j)[t] \\ 1 \end{bmatrix}. \quad (27)$$

for each $t$. It follows that the expanded algorithm remains hot-pluggable and robust in the sense of Remark 2. We can still use Theorem 3 to guarantee the convergence of the expanded algorithm because these inputs (27) converge exponentially to constant values.

**Remark 9.** In the base algorithm, each node at each time step sends a vector in $\mathbb{R}^{p+1}$ to each of its out-neighbors. In the expanded algorithm there is the additional parallel computation of the estimates of $m_{\text{ave}}$, so each node at each time step sends a vector in $\mathbb{R}^{p+3}$ to each of its out-neighbors. If $p$ is large then this is not much of a difference, and if $p$ is small then there may be enough extra room in each communication packet for this difference not to matter.

**F. Privacy for the expanded algorithm**

In this section we show that the expanded algorithm guarantees the privacy of the conformist nodes’ reference values when each conformist node satisfies either (A4)(a) or (A4)(b).

The expanded algorithm again has dynamics of the form (14) with $A$ and $B$ as in (15), but now $L$ is the out-Laplacian of the expanded digraph $G_e$ and we label the stacked vector components of the signals $x[]$ and $u[]$ as $x(i,j)[·]$ and $u(i,j)[·]$ using the virtual node index $(i,j)$.

Suppose $i$ is a conformist node, let $j$ be some other node, and let $ξ_j$ in (17) be the collection of all signals available to $j$ in the expanded algorithm. First suppose $j$ is conformist; then the matrix $C_j$ consists of vertically stacked blocks of the form

$$\begin{bmatrix} e_{(j,i)}^T \otimes I & 0 \\ 0 & e_{(j,s)}^T \otimes I \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} y_h^{\text{ext}} e_{(j,i)}^T \otimes I & 0 \end{bmatrix} \quad (28)$$

for each virtual node $(j,s)$ of $G_e$ and for all $h \in N_j^{\text{in}}$. Suppose (A4)(a) holds; then $i$ has at least two out-neighbors, which means there exists a node $k \in N_j^{\text{out}}$ such that $k \neq j$. It follows that $(i,k) \notin \{(j,s) \cup N_j^{\text{out}}\}$ for any virtual node $(j,s)$ of $G_e$. Like before, for any $φ \in \mathbb{R}^q$, the vector

$$\begin{bmatrix} e_{(i,k)}^T \otimes φ \\ L^T e_{(i,k)} \otimes φ \end{bmatrix} \quad (29)$$

is an eigenvector of $A$ associated with the unit eigenvalue. It is straightforward to show that (29) is in the null spaces of the matrices in (28) and thus also in the unobservable subspace of the pair $(A,C_j)$. In addition, this eigenvector satisfies

$$\begin{bmatrix} 0 & e_{(i,j)}^T \otimes I \\ L^T e_{(i,k)} \otimes φ \end{bmatrix} = L_{i,(i,j)}(i,s) \cdot φ \quad (30)$$

for all virtual nodes $(i,j)$ of $G_e$ (including $σ = k$). Because the virtual nodes internal to $i$ form a complete subgraph of $G_e$, each such $L_{i,(i,j)}$ is nonzero. Moreover, because $(i,k)$ has at least one external virtual edge into $k$, the sum of all such $L_{i,(i,j)}$ over $σ$ is also nonzero. Thus $j$ cannot determine any of the inputs $u(i,σ)$ individually, nor their sum over $σ$, to within any arbitrary offset. It is important that we also consider the sum over $σ$ here, because if $j$ knows this sum then it can determine $r_i$ using (25) and (27) together with its knowledge of $m_{\text{ave}}$ in the limit. Next suppose (A4)(b) holds; then there exists $k \in \{(j) \cup N_j^{\text{in}}\}$ such that $k \notin \{(j) \cup N_j^{\text{in}}\}$. Then for any $φ \in \mathbb{R}^q$ and any virtual node $(k,τ)$ of $G_e$, the vector

$$\begin{bmatrix} e_{(k,τ)}^T \otimes φ \\ L^T e_{(k,τ)} \otimes φ \end{bmatrix} \quad (31)$$

is an eigenvector of $A$ associated with the unit eigenvalue, and again it is straightforward to show that (31) is in the null spaces of the matrices in (28) and thus also in the unobservable subspace of the pair $(A,C_j)$. In addition, this eigenvector satisfies

$$\begin{bmatrix} 0 & e_{(i,j)}^T \otimes I \\ L^T e_{(k,τ)} \otimes φ \end{bmatrix} = L_{(i,j)}(i,σ) \cdot φ \quad (32)$$

for all virtual nodes $(i,j)$ of $G_e$. If $k \neq i$ then we can choose $τ = i$ so that $L_{(i,i)}(i,σ) < 0$ for each $σ$, and it follows that the sum of $L_{(i,i)}(i,σ)$ over $σ$ is nonzero. If $k = i$ then we choose $τ$ arbitrarily, and because the virtual nodes internal to $i$ form a complete subgraph of $G_e$, each such $L_{(i,i)}$ is nonzero; moreover, their sum over $σ$ is also nonzero because $i$ has at least one external virtual edge into $τ$. Thus again $j$ cannot determine any of the inputs $u(i,σ)$ individually, nor their sum over $σ$, to within any arbitrary offset.

Next suppose $j$ is a colluder node; then the matrix $C_j$ has vertically stacked blocks of the form

$$\begin{bmatrix} e_{(ℓ,s)}^T \otimes I & 0 \\ 0 & e_{(ℓ,s)}^T \otimes I \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} y_h^{\text{ext}} e_{(h,ℓ)}^T \otimes I & 0 \end{bmatrix} \quad (33)$$

for any colluder node $ℓ$ (including $ℓ = j$), for any virtual node $(ℓ,s)$ of $G_e$, and for any $h \in N_j^{\text{out}}$. Suppose (A4)(a) holds; then $i$ has a conformist out-neighbor $k$, and it follows that $(i,k) \notin \{(j,s) \cup N_j^{\text{in}}\}$ for any virtual node $(j,s)$ of $G_e$ internal to any colluder node $ℓ$. Like before, it is straightforward to show that the eigenvector (29) is in the null spaces of the matrices in (33) and that it again satisfies (30), and the proof of privacy follows using the same argument as above. Finally, suppose (A4)(b) holds; then there exists $k \in \{(j) \cup N_j^{\text{in}}\}$ such that $k \notin \{(j) \cup N_j^{\text{in}}\}$ for every colluder node $ℓ$. It is straightforward to show that the eigenvector (31) is in the null spaces of the matrices in (33), and the proof of privacy follows using the same argument as those following (32).
IV. Summary and future work

In this paper, we showed that a simple dynamic average consensus algorithm based on those presented in [18] guarantees accuracy and preserves privacy under relatively weak topological restrictions on the communication digraph. In contrast to previous work, our method is hot-pluggable, it does not require any initial scrambling phase, it does not inject any noise or other masking signals into the computation, it does not require any random switching of the edge weights, and it does not use homomorphic encryption. We have yet to study how the line digraph expansion we employ affects the performance relative to the base algorithm.

References