Privacy-Preserving Distributed Averaging via Homomorphically Encrypted Ratio Consensus

Christoforos N. Hadjicostis, *Fellow, IEEE*, and Alejandro D. Domínguez-García, *Senior Member, IEEE*

***Abstract*—In this article, we develop distributed iterative algorithms that enable the components of a multicomponent system, each with some integer initial value, to asymptotically compute the average of their initial values, without having to reveal to other components the specific value they contribute to the average calculation. We assume a communication topology captured by an arbitrary strongly connected digraph, in which certain nodes (components) might be curious but not malicious (i.e., they execute the distributed protocol correctly, but try to identify the initial values of other nodes). We first develop a variation of the so-called ratio consensus algorithm that operates exclusively on integer values and can be used by the nodes to asymptotically obtain the average of their initial (integer) values, by taking the ratio of two integer values they maintain and iteratively update. Assuming the presence of a trusted node (i.e., a node that is not curious and can be trusted to set up a cryptosystem and not reveal any decrypted values of messages it receives), we describe how this algorithm can be adjusted using homomorphic encryption to allow the nodes to obtain the average of their initial values while ensuring their privacy (i.e., without having to reveal their initial value). We also extend the algorithm to handle situations where multiple nodes set up cryptosystems and privacy is preserved as long as one of these nodes can be trusted (i.e., the ratio of trusted nodes over the nodes that set up cryptosystems decreases).**

***Index Terms*—Average consensus, distributed algorithms, homomorphic encryption, privacy preservation.**

I. Introduction

This article addresses the topic of privacy-preserving asymptotic average consensus, which has recently received attention by the control community (we review related literature at the end of this section). The distributed algorithms we develop and analyze are based on homomorphic encryption and enable the components of a distributed system, each with a certain initial value, to calculate the average of these initial values, without loss of privacy, i.e., by preventing certain *curious but not malicious* components that might be present to determine their exact initial value. Curious but not malicious components are assumed to have full knowledge of the proposed protocol and are allowed to collaborate arbitrarily among themselves, but do *not* interfere in the computation of the average value of the network in any other way (in other words, these nodes behave normally but they may use their observations to infer the private information of other nodes). For brevity, we refer to such nodes as "curious nodes" in the remainder of the article.

The backbone of the privacy-preserving schemes we propose in this article is a ratio consensus (more generally, a push sum) iteration (see, for example, [2]) extended, however, to exclusively involve integer operations (i.e., integer variables and integer weights). The proposed privacy-preserving distributed algorithms rely on one or more homomorphic cryptosystems (e.g., Pailler encryption [3]), the public keys of which are assumed to be known by all components of the distributed system. In the first algorithm, we propose, we assume that there is a trusted node, i.e., a node that is trusted by all nodes in the network to 1) generate a correct cryptosystem, with a publicly available key, and 2) not to reveal the decrypted values of any of the messages it receives. In the basic version of the algorithm, the key for decryption is held by the trusted node, which is in charge of announcing the result of the computation (after a large enough number of iterations), so that components learn what the average is (e.g., via a flooding operation). Compared to centralized algorithms that rely on a trusted node, the main advantage of this algorithm is that it allows in-network processing of encrypted values (by all nodes in the system with no routing or other setup costs), while at the same time preventing curious nodes from gathering information about the private (initial) values of other nodes.

The second algorithm we develop relaxes the assumption that there is a single node that is trusted by all nodes in the distributed system. In particular, the algorithm allows for two or more nodes to issue public keys for their own homomorphic cryptosystems (and be in charge of decrypting and announcing the result of the iteration associated with their cyptosystem). All nodes can participate in homomorphic iterations that involve these available cryptosystems. We argue that if at least one of these cryptosystems is set up by a trusted node, then privacy is preserved (i.e., the requirements on the ratio of the number of trusted nodes over the number of nodes that set up cryptosystems are relaxed). An extreme version of this second scheme is when each node sets up a cryptosystem and issues its own public key (and is in charge of decrypting and announcing the result of the iteration associated with its cryptosystem). In such case, the requirement that each node can find at least one trusted cryptosystem is trivially satisfied.

A. Literature Review

An anonymization transform using random offsets on the initial values was proposed for a cooperative wireless network in [4]. This method relies on the fact that the random initial offsets chosen by each node following the protocol are i.i.d. random variables with zero mean; thus, if an infinite number of nodes add a random offset, their net effect will be zero and the average calculation will not be affected. In real networks of finite size, however, this method fails to converge

Manuscript received October 4, 2019; accepted January 15, 2020. Date of publication January 23, 2020; date of current version August 28, 2020. This paper was presented in part at the IEEE Conference on Decision and Control [1]. This paper generalizes the results in [1] by presenting a privacy-preserving distributed average strategy that relaxes the requirements on the number of trusted nodes. *(Corresponding author: Christoforos N. Hadjicostis.)*

C. N. Hadjicostis is with the Department of Electrical and Computer Engineering, University of Cyprus, Nicosia 1678, Cyprus, and also with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: chadjic@ucy.ac.cy).

A. D. Domínguez-García is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: aledan@illinois.edu).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2020.2968876

to the true average and introduces a random offset with zero mean and some finite variance. Differential privacy techniques like the above have also appeared in the context of distributed averaging (see, for example, [5]–[7]). Unlike these works, the work proposed in this article leads to the *exact* average of the initial values.

An alternative to differential privacy techniques are privacy-preserving techniques that aim at calculating the *exact* average of the initial values. For example, the work in [8] describes a privacy-preserving protocol that is a variation of an asymptotic average consensus protocol that runs, in a distributed manner, a linear iteration with weights that form a doubly stochastic matrix. The main enhancement is that, at each time-step, each node following the protocol adds an arbitrary offset value to the result of its update, in an effort to avoid revealing its own initial value as well as the initial values of other nodes. What is important for each node is to ensure that the total (accumulated sum of) offsets that it adds cancel themselves out in the end. Similarly, the work in [9] proposed a strategy in which nodes asymptotically subtract their initial offsets, and characterized the mean square convergence rate and the covariance matrix of the maximum likelihood estimate on the initial state. Gupta *et al.* [10] proposed a distributed privacy mechanism that preserves the privacy of noncurious nodes by masking their inputs in a structured manner. Wang [11] proposed a protocol in which the state of a node is randomly decomposed into two substates, such that the mean remains the same and only one of the substates is revealed to neighboring nodes. Finally, Gao *et al.* [12] achieved privacy-preserving average consensus in digraphs by adding randomness on the edge weights used in a ratio consensus iteration. All of the above works lead to the exact value of the average, but privacy guarantees rely on topological conditions on the structure of the underlying communication topology. In contrast, the homomorphically encrypted protocols proposed in this article offer great advantages in that nodes that may overhear communications intended for other nodes cannot interpret them, which decouples the effectiveness of the scheme from the structure of the underlying communication topology (as long as the digraph describing it is strongly connected).

The approaches in [13] and [14] preserve privacy via homomorphic encryption over undirected topologies for a gossip-based consensus scheme and an average consensus scheme, respectively. In both cases, the nodes use homomorphic encryption to perform a pairwise exchange of the values (without revealing them). Freris and Patrinos [15] discussed how Paillier cryptography can be used to achieve consensus in systems that operate over finite fields. The distributed algorithms proposed in this article essentially extend these ideas over the set of integers (and over directed communication topologies) by introducing integer ratio consensus and constructing appropriate choices of weight matrices.

II. Mathematical Background and Notation

A. Digraphs and Distributed Averaging

A distributed system, the components of which can exchange information via (possibly directed) links, can conveniently be captured by a digraph (directed graph). A digraph of order N ($N \geq 2$) is defined as $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} - \{(v_j, v_j) \mid v_j \in \mathcal{V}\}$ is the set of edges. A directed edge from node v_i to node v_j is denoted by $(v_j, v_i) \in \mathcal{E}$, and indicates that node v_i can send information to node v_j.

A digraph is called *strongly connected* if for each pair of vertices $v_j, v_i \in \mathcal{V}$, $v_j \neq v_i$, there exists a directed *path* from v_i to v_j, i.e., we can find a sequence of vertices $v_i =: v_{l_0}, v_{l_1}, \ldots, v_{l_t} := v_j$ such that $(v_{l_{\tau+1}}, v_{l_\tau}) \in \mathcal{E}$ for $\tau = 0, 1, \ldots, t-1$. All nodes that can directly send information to node v_j are said to be in-neighbors of node v_j and the set comprising them is denoted by $\mathcal{N}_j^- = \{v_i \in \mathcal{V} \mid (v_j, v_i) \in \mathcal{E}\}$. The cardinality of $\mathcal{N}_j^-$ is called the *in-degree* of v_j and is denoted by $\mathcal{D}_j^-$. The nodes that can directly receive information from node v_j are referred to as its out-neighbors and the set comprising them is denoted by $\mathcal{N}_j^+ = \{v_l \in \mathcal{V} \mid (v_l, v_j) \in \mathcal{E}\}$. The cardinality of $\mathcal{N}_j^+$ is called the *out-degree* of v_j and is denoted by $\mathcal{D}_j^+$.

Consider a distributed system, captured by a directed graph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$, in which each node $v_j \in \mathcal{V}$ has an initial value V_j. Ratio consensus is a distributed algorithm that allows the nodes to asymptotically calculate the average $\frac{1}{N}\sum_{l=1}^{N} V_l$ by performing two linear iterations. The algorithm does not require the nodes to have any knowledge of the global structure of the network or the total number of nodes. The only requirement is for each node v_j to be aware of the local structure of the network (i.e., its in-neighbor and out-neighbor sets, $\mathcal{N}_j^-$ and $\mathcal{N}_j^+$); in fact, in some variations of ratio consensus, node v_j does not need explicit knowledge of the set of out-neighbors $\mathcal{N}_j^+$ but only knowledge of $\mathcal{D}_j^+$. Running-sum ratio consensus with retransmissions [16] does not require knowledge of the out-neighbors or the out-degree.

In the simplest version of ratio consensus, each node v_j maintains two state variables, $y_j[k]$ and $z_j[k]$, and updates them, at iterative step k ($k \geq 0$), as follows:

$$y_j[k+1] = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} y_i[k]/(1+\mathcal{D}_i^+) \tag{1}$$

$$z_j[k+1] = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} z_i[k]/(1+\mathcal{D}_i^+) \tag{2}$$

where $y_j[0] = V_j$, and $z_j[0] = 1$, for $v_j \in \mathcal{V}$. The protocol assumes that each node v_j is aware of its out-degree $\mathcal{D}_j^+$ and transmits the values $\overline{y}_j[k] := y_j[k]/(1+\mathcal{D}_j^+)$ and $\overline{z}_j[k] := z_j[k]/(1+\mathcal{D}_j^+)$ to all of its out-neighbors; each receiving node simply adds the values it receives from all of its in-neighbors. Compactly, the above iterations can be written as

$$y[k+1] = P_c y[k] \tag{3}$$

$$z[k+1] = P_c z[k] \tag{4}$$

where $y[k] = [y_1[k], \ \ldots, \ y_N[k]]^{\mathrm{T}}$, $z[k] = [z_1[k], \ \ldots, \ z_N[k]]^{\mathrm{T}}$, and P_c is a column stochastic matrix such that $P_c(l,j) = \frac{1}{1+\mathcal{D}_j^+}$ if $v_l \in \mathcal{N}_j^+ \cup \{v_j\}$ (zero otherwise).

At each time step k, each node v_j can calculate the ratio $r_j[k] := y_j[k]/z_j[k]$; under the assumption that the digraph describing the exchange of information is strongly connected [2], it can be shown that $r_j[k]$ asymptotically converges to the average of the initial values. Specifically, with the chosen initial conditions, we have

$$\lim_{k\to\infty} r_j[k] = \frac{\sum_l y_l[0]}{\sum_l z_l[0]} = \frac{\sum_l V_l}{N} \quad \forall v_j \in \mathcal{V}. \tag{5}$$

It turns out that ratio consensus works with any primitive column stochastic matrix P (whose zero/nonzero structure—excluding the diagonal elements—necessarily reflects the given communication topology, so that the distributed protocol conforms to the communication constraints) [2]. In fact, ratio consensus iterations can also take the time-varying form

$$y[k+1] = P[k]y[k] \tag{6}$$

$$z[k+1] = P[k]z[k] \tag{7}$$

where $P[k]$, $k = 0, 1, 2, \ldots$, are column stochastic $N \times N$ matrices that vary at each time step. Subject to some joint properties on the sequence of matrices $P[k]$, $k = 0, 1, 2, \ldots$, the convergence result in (5) still holds, although the proof is more complex [16]–[19]. More specifically, the following are *sufficient* conditions for reaching asymptotic average consensus [as in (5)].

C1) At each iteration k, the matrix $P[k]$ is a column stochastic matrix with nonzero entries bounded away from zero.

C2) There exists a finite window K, such that

$$P[\tau K + K - 1] \ldots P[\tau K + 1] P[\tau K] \ , \ \tau = 0, 1, 2, \ldots$$

form primitive column stochastic matrices. A sufficient condition for **C2** to hold is for matrices $P[k]$ to have positive elements on the diagonal and the union graphs

$$\begin{aligned}&\mathcal{G}_d[\tau K + K - 1] \cup \ldots \cup \mathcal{G}_d[\tau K + 1] \cup \mathcal{G}_d[\tau K] \\ &\quad := (\mathcal{V}, \mathcal{E}[\tau K + K - 1] \cup \ldots \cup \mathcal{E}[\tau K + 1] \cup \mathcal{E}[\tau K])\end{aligned}$$

$\tau = 0, 1, 2, \ldots$, to be strongly connected (here, $\mathcal{G}_d[k]$ is the digraph corresponding to the zero/nonzero structure of $P[k]$).

B. Homomorphic Encryption

An encryption scheme is captured by an encryption mechanism E and a decryption mechanism D. The encryption mechanism takes a message m, typically treated as a non-negative integer in some range, and encrypts it into a (possibly non-unique) ciphertext c, also treated as a non-negative integer in some range; similarly, the decryption mechanism D takes the ciphertext c and generates the original message. In other words, we have $m = D(E(m))$. In most encryption schemes, knowing E (the encyption mechanism) implies knowledge of D (the decryption mechanism); public key cryptography schemes (e.g., RSA [20]), however, are such that knowledge of E depends on a public key (which is available to everybody), but knowledge of D depends on a private key (which is not available to everybody and cannot be inferred easily based on the public key). Thus, everybody knows E but only one entity (which presumably designed the cryptosystem) knows D.

Homomorphic encryption (see, for example, [21]) has the property that $E(m_1 \circ m_2) = E(m_1) \odot E(m_2)$ where $\circ$ and $\odot$ are binary operations defined on the message and ciphertext spaces, respectively. Effectively, this allows certain types of processing of the data without necessarily having direct access to them. Naturally, with the emergence of cloud storage and the need for big data analytics, homomorphic encryption has recently generated a lot of interest [22].

Among popular homomorphic encryption algorithms, the one we are interested in is the Paillier cryptosystem [3]. Due to space limitations, we do not provide the full details of the scheme, but only the features that are necessary for our development. (Below, $gcd(a, b)$ denotes the greatest common divisor of a pair of integers a and b, and $\lfloor r \rfloor$ denotes the floor of a real number r, i.e., the largest integer that is smaller or equal to r.) The public key is given by (n, g) and the private key is given by (λ, μ), where n, g, λ, and μ are non-negative integers chosen according to the code design. Encryption of a message m (viewed as an integer in $[0, n-1]$) is $c = E(m, r)$ where c is obtained by choosing a random integer r, $0 < r < n$ such that $gcd(r, n) = 1$ (any random r that satisfies this can be chosen), and setting $c = g^m r^n \mod n^2$. Decryption of a ciphertext c is defined as $m' = D(c)$, where m' is obtained by setting $L' = \lfloor \frac{(c^\lambda \mod n^2) - 1}{n} \rfloor$ and $m' = L' \mu \mod n$.

An important feature of the Paillier cryptosystem is that it has the following homomorphic property:

$$m_1 + m_2 \mod n = D(E(m_1, r_1) E(m_2, r_2) \mod n^2)$$

and the following (semi-homomorphic) property:

$$\begin{aligned} m_1 m_2 \mod n &= D(E(m_1, r_1)^{m_2} \mod n^2) \\ &= D(E(m_2, r_2)^{m_1} \mod n^2) \end{aligned}$$

(notice that one of the two messages remains unecrypted). In particular, this means that the multiplication of a message m by a constant integer weight w satisfies

$$wm \mod n = D(E(m, r)^w \mod n^2)$$

where the weight w is unencrypted. Also, note that the random numbers (r_1, r_2, r above) used to encrypt are not important in the decoding operations.

III. Integer Distributed Averaging

In this section, we present variations of ratio consensus and push sum that operate on integer values and use integer weights, so that all operations involve integer values. The main motivation for this is to allow for encyption of the data (using a public key homomorphic cryptosystem, such as the Paillier cryptosystem), which is done in subsequent sections.

We consider a distributed system, captured by a strongly connected digraph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$, in which each node $v_j \in \mathcal{V}$ has an initial *integer* value V_j. The goal is to devise an algorithm to calculate the average $\frac{1}{N} \sum_{l=1}^N V_l$. The proposed algorithm operates as follows: each node v_j maintains two (integer-valued) state variables $y_j[k]$ and $z_j[k]$, and updates them, at iterative step k ($k \geq 0$), as follows:

$$y_j[k+1] = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} w_{ji}[k] y_i[k] \tag{8}$$

$$z_j[k+1] = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} w_{ji}[k] z_i[k] \tag{9}$$

where $y_j[0] = V_j$, and $z_j[0] = 1$, for $v_j \in \mathcal{V}$. The time-varying weights $w_{lj}[k]$ are required to take non-negative integer values such that, for each k, $\sum_{v_l \in \mathcal{N}_j^+ \cup \{v_j\}} w_{lj}[k] = c[k]$ for all $v_j \in \mathcal{V}$ for some *positive* integer (time-varying) value $c[k]$.

Using matrix-vector notation, the above iterations can be captured concisely via

$$y[k+1] = W[k] y[k] \tag{10}$$

$$z[k+1] = W[k] z[k] \tag{11}$$

where the matrices $W[k]$ satisfy the requirements in the following lemma.

Lemma 1: Consider a strongly connected digraph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$ where the nodes execute the iterations in (10)–(11) so that the chosen weight matrices $W[k]$, $k = 0, 1, 2, \ldots$, satisfy the following.

1) Each entry of $W[k]$ is a non-negative integer; in particular, $w_{lj}[k] = 0$ if $v_l \notin \mathcal{N}_j^+ \cup \{v_j\}$ and $w_{lj}[k] \geq 0$ otherwise.

2) The column sums of each $W[k]$ are equal to $c[k] \geq 1$ where $c[k] \leq c_{\max}$ for some maximum value $c_{\max}$.

3) Condition **C2** is satisfied on the sequence of column stochastic matrices $P[k] = \frac{1}{c[k]} W[k]$.

Then

$$\lim_{k \to \infty} r_j[k] = \frac{\sum_l y_l[0]}{\sum_l z_l[0]} = \frac{\sum_l V_l}{N}.$$

Proof: Let $P[k] := \frac{1}{c[k]} W[k]$, where $P[k]$ is a column stochastic matrix with zero/nonzero structure that adheres to the given digraph $\mathcal{G}_d$. Furthermore, the nonzero entries of $P[k]$ are bounded below by $\frac{1}{c[k]} \geq \frac{1}{c_{\max}}$.

We can unwrap the iterations for y as follows:

$$\begin{aligned} y[k+1] &= \left(\Pi_{\tau=0}^k W[\tau]\right) y[0] \\ &= \left(\Pi_{\tau=0}^k c[\tau]\right) \left(\Pi_{\tau=0}^k P[\tau]\right) y[0]. \end{aligned}$$

Similarly, we obtain $z[k+1] = (\Pi_{\tau=0}^{k} c[\tau])(\Pi_{\tau=0}^{k} P[\tau])z[0]$.

Matrices $P[k]$, $k = 0, 1, 2, \ldots$, satisfy Condition **C1** (since the entries are bounded below by $\frac{1}{c_{\max}}$). Thus, the ratio $r'_j[k] = \frac{y'_j[k]}{z'_j[k]}$ (where $y'[k+1] = \frac{1}{\Pi_{\tau=0}^{k} c[\tau]} y[k+1]$ and $z'[k+1] = \frac{1}{\Pi_{\tau=0}^{k} c[\tau]} z[k+1]$) converges to the average according to (5). We conclude that

$$\lim_{k\to\infty} r_j[k] = \frac{\sum_l y_l[0]}{\sum_l z_l[0]} = \frac{\sum_l V_l}{N} \quad \forall v_j \in \mathcal{V}$$

which completes the proof of the lemma. ■

We next discuss some possible weight choices that lead to matrices $W[k]$ that satisfy the above requirements.

A. Transmission to a Single Out-Neighbor

Suppose that each node v_j can select which node to transmit to. Furthermore, suppose that each node v_j does the following at each k.

1) it sets its self-weight to $w_{jj}[k] = 1$;
2) it chooses (in a round robin fashion, in some arbitrary order) one of its out-neighbors $v_l, v_l \in \mathcal{N}_j^+$, and sets the weight on that out-going link to $w_{lj}[k] = 1$;
3) it sets the weights on all of its remaining out-going links to zero, i.e., $w_{l'j}[k] = 0$ for $l' \neq l, l' \neq j$.

If all nodes do this in a synchronized manner, the resulting matrix $W[k]$ will have columns that sum to $c[k] = 2$, because each column j has exactly two nonzero entries, the diagonal entry $W(j,j) = 1$ and $W(l,j)[k] = 1$. Since node v_j has $\mathcal{D}_j^+$ out-neighbors, the jth column of matrix $W[k]$ can take $\mathcal{D}_j^+$, different forms (depending on the out-neighbor node v_j chooses to transmit to). In total, there are $K := \Pi_{\ell=1}^{N} \mathcal{D}_\ell^+$ different possible matrices $W[k]$ (depending on the out-neighbor that is chosen by each node). However, it is possible that not all such matrices materialize at some specific iterative step k because of common factors among the node degrees $\{\mathcal{D}_1^+, \mathcal{D}_2^+, \ldots, \mathcal{D}_N^+\}$. More specifically, node v_1 will cycle through its out-neighbors every $\mathcal{D}_1^+$ iterations, node v_2 will cycle through its out-neighbors every $\mathcal{D}_2^+$ iterations, and so forth, implying that all nodes will cycle through the same matrices every $K' := \text{lcm}(\mathcal{D}_1^+, \mathcal{D}_2^+, \ldots, \mathcal{D}_N^+)$ iterations (leading to K', $K' \leq K$, different matrices), where $\text{lcm}(i_1, i_2, \ldots, i_N)$ is the least common multiple of the integers in its argument. In any case, since we are guaranteed that all links are active at least once every K (or K') time steps (and the diagonal elements are positive, and the graph is strongly connected), conditions **C1** and **C2** will be satisfied on matrices $P[k] := \frac{1}{2} W[k]$, and the ratios will converge to the average of the initial values.

B. Transmission to All Out-Neighbors

The above scheme would not necessarily work if we allowed nodes to send their values to all of their out-neighbors, i.e., if we had $W[k] = A_d + I_N$, where A_d is the adjacency[1] matrix of the digraph and I_N is the $N \times N$ identity matrix. The problem is that nodes might have different out-degrees, which means that column sums are not necessarily equal (note that the ratios still converge, but to a weighted average [2]). The following variation, however, would work: set $W[k] = A_d + D =: W$, where $D = \text{diag}(1 + \mathcal{D}_{\max}^+ - \mathcal{D}_j^+)$ (i.e., a diagonal matrix such that $D(j,j) = 1 + \mathcal{D}_{\max}^+ - \mathcal{D}_j^+$), with $\mathcal{D}_{\max}^+ = \max_{v_j \in \mathcal{V}} \mathcal{D}_j^+$ being the maximum out-degree among all nodes in the graph. Note that as long as the graph is strongly connected, we are guaranteed that $P := \frac{1}{1+\mathcal{D}_{\max}^+} W$ is a primitive column stochastic matrix (in this case $c[k] = 1 + \mathcal{D}_{\max}^+$).

[1]The adjacency matrix of a digraph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$ satisfies $A_d(l,j) = 1$ if $(v_l, v_j) \in \mathcal{E}$ (and is zero otherwise).

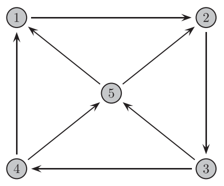
Example 1: Consider the (strongly connected) digraph on the left of Fig. 1 (with $n = 5$ nodes) with initial values taken to be $V_j = j$, $j = 1, 2, 3, 4, 5$ (so that the average is 3). Consider first the scheme in which each node selects, at each time step k, a single out-neighbor, assuming the following order in which nodes choose their out-neighbors: node v_3 chooses v_4, then v_5 (and repeat); node v_4 chooses v_1, then v_5 (and repeat); node v_5 chooses v_1, then v_2 (and repeat). Note that nodes v_1 and v_2 only have one out-neighbor. The middle of Fig. 1 shows the two possible update matrices, W_0 and W_1. In particular, at iterative step $k = 0$, the update matrix is $W[0] = W_0$; at iterative step at $k = 1$, the update matrix is $W[1] = W_1$; then, $W[2] = W_0$, $W[3] = W_1$, and so forth. Note that, despite the fact that we could potentially have $K = \Pi_{\ell=1}^{5} \mathcal{D}_\ell^+ = 8$, different possibilities for the values that matrices $W[k]$ could take (depending on the out-neighbor that it is chosen by each node), only two instantiations materialize in this example. This is in agreement with our earlier discussions since in this case $K' = \text{lcm}(\mathcal{D}_1^+, \mathcal{D}_2^+, \ldots, \mathcal{D}_5^+) = \text{lcm}(1, 1, 2, 2, 2) = 2$. Moreover, we are guaranteed that all links are active at least once every $K' = 2$ time steps and conditions **C1** and **C2** are satisfied on matrices $P[k] := \frac{1}{2} W[k]$, so ratios converge to the average of the initial values asymptotically.

Consider next the scheme in which each node selects all out-neighbors (with weight 1) and adjusts its self-weight so that the sum of each column of the weight matrix is $1 + \mathcal{D}_{\max}^+$. The right of Fig. 1 shows the (constant) weight matrix W_{all} used in the updates. Again, convergence of the ratios to the average of the initial values is asymptotic, but the ratios get close to the average of the initial values relatively quickly (after approximately ten iterative steps). ■

Remark 1: Since the integer matrices used are scaled versions of column stochastic matrices, rates of convergence can be obtained using the techniques applicable to column stochastic matrices. For example, if the same matrix is used for all iterations (namely, transmission to all out-neighbors), then the rate of convergence is governed by $|\lambda_2(P)|$, where $\lambda_2(P)$ is the second largest in magnitude eigenvalue of matrix $P := \frac{1}{1+\mathcal{D}_{\max}^+} W$ [23]. If transmission to a single out-neighbor is used at a time, then the rate of convergence is governed by $|\lambda_2(\overline{P})|^{\frac{1}{K'}}$, where $\overline{P} := \frac{1}{2^{K'}} W_1 W_2 \ldots W_{K'}$ with $K' := \text{lcm}(\mathcal{D}_1^+, \mathcal{D}_2^+, \ldots, \mathcal{D}_N^+)$, capturing the number of different matrices W that are realized (this bound can be obtained by looking over the K' iterations that involve all matrices $W[k]$ that are realized). Bounds on the rate of convergence can also be easily obtained based on the smallest nonzero entry in any of the $\frac{1}{c[k]} W[k]$ matrices or using coefficients of ergodicity [23]. Rate of convergence analysis could be used to determine what would constitute a large enough number of iterations for all of the nodes to reach values that are close to their asymptotic limits; alternatively, distributed stopping techniques like the ones proposed for ratio consensus in [24] can be used. ■

IV. Homomorphically Encrypted Ratio Consensus

Assume that there is a trusted node (without loss of generality taken to be node v_1), which devises a homomorphic encryption scheme using the Paillier cryptosystem and issues a public key (n, g) that gets distributed (by flooding or other means) to all the nodes. Node v_1 is trusted in the sense that it will generate a correct cryptosystem and will not reveal the decrypted value of any of the messages it receives, at least not before a large enough number of iterations has been executed. We use $\tilde{y} = E(y)$ to denote the encrypted version of integer y (where



$$W_0 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad W_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} \quad W_{all} = \begin{bmatrix} 2 & 0 & 0 & 1 & 1 \\ 1 & 2 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Fig. 1. Digraph used in Example 1 (left); weight matrices W_0 and W_1 for the case when each node transmits to only one out-neighbor at each iterative step (middle); weight matrix W_{all} for the case when each node transmits to all of its out-neighbors (right).

$0 \leq y < n$). (This value is nonunique as it depends on the random integer r used by the encryption mechanism, but any value of $\tilde{y}$ that is used will have the same properties.)

The first observation is that the linear iterations in (8)–(9) involve non-negative integers (at all time steps, at all nodes). If the integer n is large enough (so that these values do not exceed n for the number of iterative steps we are interested in), then performing the linear iterations modulo n does not change anything in the process. Note that this is not a restrictive assumption given that encryption schemes typically require n to be a very large integer. Thus, we can think of the iteration in (8), as an iteration where the operations are modulo n

$$\begin{aligned} y_j[k+1] &= \left(\sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} w_{ji}[k] y_i[k] \right) \mod n \\ &= \left(\sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} (w_{ji}[k] y_i[k] \mod n) \right) \mod n \end{aligned} \quad (12)$$

with $y_j[0] = V_j \mod n$, for $v_j \in \mathcal{V}$. The time-varying weights $w_{lj}[k]$ remain exactly the same as before: in particular, they take positive integer values such that $\sum_l w_{lj}[k] = c[k]$ for all $v_j \in \mathcal{V}$ for some positive integer value $c[k]$.

Note that each node v_j is in charge of selecting $w_{lj}[k]$ and sending to its neighbor v_l, $v_l \in \mathcal{N}_j^+$, the values $w_{lj}[k]y_j[k]$ and $w_{lj}[k]z_j[k]$. Assuming all nodes are aware of the public key (n, g), we can perform the y iteration in (8) modulo n [i.e., the iteration in (12)] in the encrypted space. Specifically, using the homomorphic properties at the end of Section II-B, we can execute the following y-iteration in the ciphertext space

$$\begin{aligned} \tilde{y}_j[k+1] &= \left(\Pi_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} E(w_{ji}[k] y_i[k]) \right) \mod n^2 \\ &= \Pi_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} \left((\tilde{y}_i[k])^{w_{ji}[k]} \mod n^2 \right) \mod n^2 \end{aligned}$$

where $\tilde{y}_j[0] = E(y_j[0])$.

Effectively, the proposed protocol does the following: each node v_j encrypts its initial value to $\tilde{y}_j[0] = E(y_j[0])$; then, the nodes perform (in a synchronized manner) the above iteration for $\tilde{y}$, as well as the unencrypted iteration for z in (9).

At some iterative step k_0 (where k_0 is sufficiently large), node v_1 (which holds the private key) can decrypt $\tilde{y}_1[k_0]$ via $y_1[k_0] = D(\tilde{y}_1[k_0])$, calculate the ratio $r_1[k_0] = y_1[k_0]/z_1[k_0]$, and then forward this value (or forward both $y_1[k_0]$ and $z_1[k_0]$) to all other nodes (via flooding or other means).

We consider the privacy of a noncurious node v_j to be preserved if the curious node(s) cannot determine its *exact* initial value. Note that, as long as node v_1 is trustworthy, the above strategy will preserve privacy. In fact, this is easy to see if decrypting is done by node v_1 only once (say, at time step k_0) to let other nodes know what the ratio $r_1[k_0]$ (i.e., the average of the initial values) is. In such case, the only value that becomes known to curious nodes is the average of the initial values. If N is also known, the curious nodes will be in position to calculate the sum of the initial values of all other nodes (by multiplying $r_1[k_0]$ by N and, then, subtracting their own initial values). Notice, however, that regardless of how the average is calculated, this information will necessarily be available to curious nodes that are colluding (in particular, we need at least two nodes to be noncurious for privacy preservation to be possible).

V. Homomorphically Encrypted Iterations

One concern regarding the strategy discussed in the previous section is that it relies heavily on the (universally) trusted node v_1 that is supposed to eventually (at some large enough iterative step k_0) decrypt the $\tilde{y}_1[k_0]$-value and provide the ratio $y_1[k_0]/z_1[k_0]$ (or, alternatively provide $y_1[k_0]$ and $z_1[k_0]$) so that all nodes can obtain the average. In this section, we relax this assumption by allowing the average to be calculated based on the result of K ($K \leq N$) homomorphically encrypted iterations, which requires that at least one node out of a set of K nodes can be trusted (but not necessarily all).

We first describe the unencrypted version of the protocol and then analyze its encrypted version.

A. Average Consensus via Multiple Ratio Consensus Iterations

We assume that we are given a distributed system, captured by a strongly connected directed graph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$, in which each node $v_j \in \mathcal{V}$ has a *non-negative integer* V_j as an initial value. We will execute K iterations, each of which is associated with one of the nodes in the set $\{v_1, v_2, \ldots, v_K\}$. We will denote the y-values and z-values held at node v_j for the ℓth iteration at time step k by $y_j^{(\ell)}[k]$ and $z_j^{(\ell)}[k]$, and we will update them as in (8)–(9). Letting

$$y^{(\ell)}[k] = [y_1^{(\ell)}[k], y_2^{(\ell)}[k], \ldots, y_N^{(\ell)}[k]]^T, \quad \ell = 1, 2, \ldots, K$$

$$z^{(\ell)}[k] = [z_1^{(\ell)}[k], z_2^{(\ell)}[k], \ldots, z_N^{(\ell)}[k]]^T, \quad \ell = 1, 2, \ldots, K$$

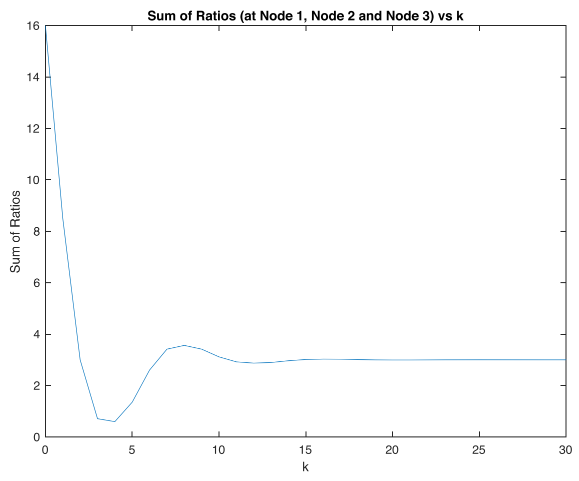
we can use matrix-vector notation to concisely capture the K y-iterations and z-iterations as

$$y^{(\ell)}[k+1] = W[k] y^{(\ell)}[k], \quad \ell = 1, 2, \ldots, K \quad (13)$$

$$z^{(\ell)}[k+1] = W[k] z^{(\ell)}[k], \quad \ell = 1, 2, \ldots, K. \quad (14)$$

As in the previous section, the weight matrices, $W[k]$, $k = 0, 1, 2, \ldots$, are time-varying but identical for all iterations at a particular time step. Furthermore, we assume that the $W[k]$'s satisfy the requirements in Lemma 1. Notice that $z^{(\ell)}[0] = \mathbf{1}_N$, $\ell = 1, 2, \ldots, K$ (where $\mathbf{1}_N$ is the N-dimensional all-ones vector). Thus, the vectors, $z^{(\ell)}[k]$, $\ell = 1, 2, \ldots, K$, evolve identically and only one of them, which we denote simply by $z[k]$, needs to be maintained by the nodes. As a result, (14) becomes

$$z[k+1] = W[k] z[k]. \quad (15)$$



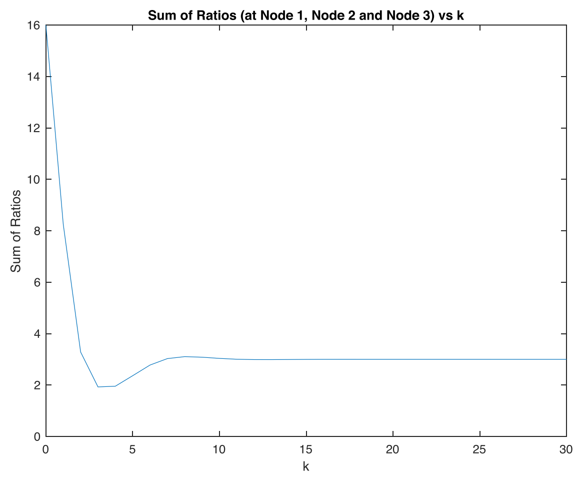


Fig. 2. Sum of ratios at nodes v_1, v_2, and v_3, as a function of k: transmission to single out-neighbor (left); transmission to all out-neighbors (right).

Note that we use $z_j[k]$ to denote the scalar z-value held at node v_j at step k.

Lemma 2: Consider a strongly connected directed graph $\mathcal{G}_d = (\mathcal{V}, \mathcal{E})$, where each node $v_j \in \mathcal{V}$ has as an initial value a non-negative integer V_j. Suppose that the nodes execute in parallel the K iterations in (13) and the iteration in (15). If each node v_j chooses the initial values $y_j^{(\ell)}[0]$ for $\ell = 1, 2, \ldots, K$, to be integers that satisfy $\sum_{\ell=1}^{K} y_j^{(\ell)}[0] = V_j$, then we have

$$\lim_{k\to\infty} \sum_{\ell=1}^{K} r_\ell^{(\ell)}[k] = \frac{\sum_l V_l}{N} \tag{16}$$

where $r_\ell^{(\ell)}[k] = \frac{y_\ell^{(\ell)}[k]}{z_\ell[k]}$.

Proof: If we consider any node v_j and any iteration ℓ, we know from the discussion in Section II that

$$r_j^{(\ell)} := \lim_{k\to\infty} r_j^{(\ell)}[k] = \frac{\sum_l y_l^{(\ell)}[0]}{N} \quad \forall v_j \in \mathcal{V}$$

where $r_j^{(\ell)}[k] := \frac{y_j^{(\ell)}[k]}{z_j[k]}$. In particular, for every iteration ℓ, $\ell = 1, 2, \ldots, K$, we have $r_\ell^{(\ell)} = \sum_l y_l^{(\ell)}[0]/N$. If we sum up all $r_\ell^{(\ell)}$ (over the K iterations), we have

$$\sum_{\ell=1}^{K} r_\ell^{(\ell)} = \sum_{\ell=1}^{K} \frac{\sum_l y_l^{(\ell)}[0]}{N} = \frac{1}{N} \sum_l \left(\sum_{\ell=1}^{K} y_l^{(\ell)}[0] \right) = \frac{\sum_l V_l}{N}. \tag{17}$$

■

Example 2: Consider again the strongly connected digraph on the left of Fig. 1. Assume that $K = 3$ and that the nodes execute three iterations with initial values $y^{(1)}[0] = [4, -4, -1, -2, 4]$, $y^{(2)}[0] = [0, 5, -3, -2, -3]^{\mathrm{T}}$, $y^{(3)}[0] = [-3, 1, 7, 8, 4]^{\mathrm{T}}$. The values in vectors $y^{(1)}[0]$ and $y^{(2)}[0]$ were chosen to be random integers in the interval $[-4, 5]$, with uniform probability and independently between different entries of the vectors; then, $y^{(3)}[0] = [1, 2, 3, 4, 5]^{\mathrm{T}} - (y^{(1)}[0] + y^{(2)}[0])$.

As in Example 1, we consider two schemes, one in which each node selects a single out-neighbor, and one in which each node transmits to all out-neighbors. Consider first the scheme in which each node selects a single out-neighbor, assuming (for simplicity of presentation) that the order in which each node chooses its out-neighbors is exactly the same as in Example 1. In such case, the resulting update matrices are W_0 and W_1 as given in the middle of Fig. 1, i.e., at even k the weight matrix is given by W_0 and at odd k by W_1.

On the left of Fig. 2, we plot the sum of the ratios $r_1^{(1)}[k] + r_2^{(2)}[k] + r_3^{(3)}[k]$ as a function of k. Convergence of the sum of ratios to the average of the initial values is asymptotic, but we see that this sum becomes close to the average (as expected) after approximately 15 iterative steps. Note that the ratios that the nodes have for each iteration converge asymptotically to the average of the initial values for that iteration (i.e., the ratios for iteration 1 converge to 1/5, the ratios for iteration 2 converge to -3/5, and the ratios for iteration 3 converge to 17/5).

Consider next the scheme in which each node selects all out-neighbors (with weight 1) and adjusts its self-weight so that the sum of each column of the weight matrix is $1 + \mathcal{D}^+_{\max}$ (the resulting weight matrix is as on the right of Fig. 1). In this case, we also observe that the ratios converge relatively quickly (after approximately ten iterative steps) to the average of the initial values. On the right-hand side of Fig. 2, we plot the sum of the ratios $r_1^{(1)}[k] + r_2^{(2)}[k] + r_3^{(3)}[k]$ as a function of k. Again, we see that the sum of the ratios converges (as expected) after approximately 15 iterative steps to the average. ■

B. Privacy Preservation via Multiple Encrypted Ratio Consensus Iterations

Assume that there is a set of K nodes ($K \leq N$), each of which devises a homomorphic encryption scheme (using the Paillier cryptosystem). Without loss of generality, we can take these K nodes to be nodes v_1, v_2,..., v_K and assume that each node v_ℓ, $\ell = 1, 2, \ldots, K$, issues a public key (n_ℓ, g_ℓ) that gets distributed (by flooding or other means) to all other nodes. We use $\tilde{y}_\ell = E_\ell(y)$, $\ell = 1, 2, \ldots, K$, to denote the (nonunique) encrypted version of integer y (where $0 \leq y < n_\ell$) using the key issued by node v_ℓ (again, any value of $\tilde{y}_\ell$ will have the same result). We propose a scheme that allows nodes to calculate the average of the initial values in a privacy-preserving manner. More specifically, node v_j can be certain that its initial value cannot be inferred exactly as long as it can trust at least one node in the set $\{v_1, v_2, \ldots, v_K\}$. A special case would be when $K = N$, in which case each node is guaranteed that its privacy is preserved (because each node can presumably trust itself). As discussed earlier, we need at least two noncurious nodes to ensure privacy preservation in any averaging scheme. As in the case of a single iteration, we assume for simplicity that all initial values are non-negative, so that the ℓ linear iterations in (13) involve non-negative integers (at all time steps, at all nodes). Let $n = \min_\ell\{n_\ell\}$ and assume that n is large enough so that these values do not exceed n for the number of iterations we are interested in.

Under the above assumptions, we can think of each iteration in (13) for $\ell = 1, 2, \ldots, K$ as an iteration where the operations are modulo n [refer to (12)] with $y_j^{(\ell)}[0]$ chosen for each iteration as described in the previous section. The time-varying weights $w_{lj}[k]$ remain exactly the same as before: in particular, they take positive integer values such that $\sum_l w_{lj}[k] = c[k]$ for all $v_j \in \mathcal{V}$ for some positive integer value $c[k]$.

Note that each node v_j is in charge of selecting $w_{lj}[k]$ and sending to its neighbor v_l, $v_l \in \mathcal{N}_j^+$, the K y-values $w_{lj}[k]y_j^{(\ell)}[k]$, $\ell = 1, 2, \ldots, K$, and the z-value $w_{lj}[k]z_j[k]$. Assuming all nodes are aware of the public key (n_ℓ, g_ℓ) for each iteration, we can perform each y iteration in (13) modulo n using the homomorphic encryption scheme devised by the corresponding node. Specifically, using the homomorphic properties at the end of Section II-B, we can replace the iteration in (13) (for each ℓ in the set $\{1, 2, \ldots, K\}$) by the following $\tilde{y}$-iteration in the ciphertext space:

$$\begin{aligned}\tilde{y}_j^{(\ell)}[k+1] &= \left(\Pi_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} E(w_{ji}[k] y_i^{(\ell)}[k])\right) \mod n^2 \\ &= \Pi_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} (\tilde{y}_i^{(\ell)}[k])^{w_{ji}[k]} \mod n^2 \end{aligned} \tag{18}$$

where $\tilde{y}_j^{(\ell)}[k] = E_\ell(y_j^{(\ell)}[k])$ and the initial values satisfy $\tilde{y}_j^{(\ell)}[0] = E(y_j^{(\ell)}[0])$.

Effectively, the proposed approach executes the K average consensus iterations described in the previous section, each encrypted according to the homomorphic encryption scheme devised by one of the nodes in the set $\{v_1, v_2, \ldots, v_K\}$. If we assume that at some step k_0 (where k_0 is sufficiently large), these K nodes (which hold the private keys needed to decrypt) provide the values $y_\ell^{(\ell)}[k_0] = D_\ell(\tilde{y}_\ell^{(\ell)}[k_0])$, for $\ell = 1, 2, \ldots, K$, and their corresponding $z_l[k_0]$ value, then the nodes can compute the quantity $\sum_{\ell=1}^K \frac{y_\ell^{(\ell)}[k_0]}{z_\ell[k_0]}$, which, as argued in the previous section, asymptotically reaches the desirable average of their initial values, at least for large values of k_0 [see (17)].

We now discuss the privacy-preservation capabilities of the proposed scheme. To do that, we assume for simplicity that k_0 is large enough (so that all ratios have effectively reached values that are adequately close to their asymptotic values) and that decryption by at least one of the nodes v_1, v_2, ..., and v_K occurs only once (at time step k_0). For simplicity, let us consider the case when $K = 2$; assuming (as a worst-case assumption) that curious nodes can collaborate arbitrarily, what becomes available to them at time step k_0 (when decryption takes place by nodes v_1 and v_2) are the values of the following two sums, namely $X_1 = \sum_{v_j \in \mathcal{V}_{nc}} y_j^{(1)}[0]$ and $X_2 = \sum_{v_j \in \mathcal{V}_{nc}} y_j^{(2)}[0]$, where $\mathcal{V}_{nc}$ denotes the remaining nodes (i.e., the nodes that are not in the set of curious colluding nodes). In general, the curious nodes have no way of isolating the values $y_j^{(\ell)}[0]$, $v_j \in \mathcal{V}_{nc}$, $\ell \in \{1, 2\}$, because there are only two equations and $2|\mathcal{V}_{nc}|$ unknowns. This remains true even if one of the two nodes, say node v_2, is curious, in which case (depending on the position of node v_2) the curious node(s) may have access to the (unencrypted) values $y_{j,2}[0]$ for some (or all) of the nodes v_j in $\mathcal{V}_{nc}$. Note, however, that curious nodes can use the data that are available to them to form an estimate of the initial values of nodes in the set $\mathcal{V}_{nc}$. This could be formulated, for instance, as an estimation problem where V_j needs to be estimated based on the values of X_1, X_2, and the statistics that are known about the variables $y_j^{(\ell)}[0]$, $v_j \in \mathcal{V}_{nc}$, $\ell \in \{1, 2\}$ (e.g., one could use mean square error estimation).

Remark 2: Note that, even if we ignore the complexity of devising cryptosystems and performing the encryption/decryption computations (one-time costs), the computational complexity of the proposed scheme increases by a factor of K (where K is the number of homomorphically encrypted iterations). Also, each addition/multiplication is replaced by addition/multiplication modulo a large number. ■

VI. Conclusion and Future Work

In this article, we have considered the problem of privacy-preserving asymptotic average consensus in the presence of curious (but not malicious) nodes. Specifically, we described how nodes in a distributed system can use homomorphic encryption to perform integer operations (additions only or additions and multiplications) in order to asymptotically reach consensus to the average of their integer initial values in a privacy-preserving manner.

In our future work, we will study possible extensions to real initial values (i.e., take into account finite precision effects) and to schemes that guarantee privacy preservation (in the sense that they involve all nodes in the encryption), but do not require the execution of N iterations. Other interesting future directions include: 1) An analysis of information leakage (i.e., partial information about the initial values) in the context of homomorphically encrypted iterations, particularly for the case when the trusted node(s) decrypt their ratio value(s) at multiple time instants). 2) Handling of possibly malicious nodes, e.g., by combining some of the techniques proposed here with some of the techniques in [25]–[27], which describe ways to precisely or approximately compute, in a distributed manner, functions of the initial values (including the average).

References

[1] C. N. Hadjicostis, “Privacy preserving distributed average consensus via homomorphic encryption,” in *Proc. IEEE Conf. Decis. Control*, 2018, pp. 1258–1263.

[2] C. N. Hadjicostis, A. D. Domínguez-García, and T. Charalambous, “Distributed averaging and balancing in network systems, with applications to coordination and control,” *Found. Trends Syst. Control*, vol. 5, nos. 3/4, pp. 99–292, 2018.

[3] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1999, pp. 223–238.

[4] M. Kefayati, M. S. Talebi, B. H. Khalaj, and H. R. Rabiee, “Secure consensus averaging in sensor networks using random offsets,” in *Proc. IEEE Int. Conf. Telecommun. Malaysia Int. Conf. Commun.*, 2007, pp. 556–560.

[5] Z. Huang, S. Mitra, and G. Dullerud, “Differentially private iterative synchronous consensus,” in *Proc. ACM Workshop Privacy Electron. Soc.*, 2012, pp. 81–90.

[6] S. S. Kia, J. Cortés, and S. Martinez, “Dynamic average consensus under limited control authority and privacy requirements,” *Int. J. Robust Nonlinear Control*, vol. 25, no. 13, pp. 1941–1966, 2015.

[7] J. Cortés, G. E. Dullerud, S. Han, J. Le Ny, S. Mitra, and G. J. Pappas, “Differential privacy in control and network systems,” in *Proc. IEEE Conf. Decision Control*, 2016, pp. 4252–4272.

[8] N. E. Manitara and C. N. Hadjicostis, “Privacy-preserving asymptotic average consensus,” in *Proc. Eur. Control Conf.*, 2013, pp. 760–765.

[9] Y. Mo and R. M. Murray, “Privacy preserving average consensus,” *IEEE Trans. Autom. Control*, vol. 62, no. 2, pp. 753–765, Feb. 2017.

[10] N. Gupta, J. Katz, and N. Chopra, “Privacy in distributed average consensus,” *IFAC Papers OnLine*, vol. 50, no. 1, pp. 9515–9520, 2017.

[11] Y. Wang, “Privacy-preserving average consensus via state decomposition,” *IEEE Trans. Autom. Control*, vol. 64, no. 11, pp. 4711–4716, Nov. 2019.

[12] H. Gao, C. Zhang, M. Ahmad, and Y. Wang, “Privacy-preserving average consensus on directed graphs using push-sum,” in *Proc. IEEE Conf. Commun. Netw. Secur.*, May 2018, pp. 1–9.

[13] R. Lazzeretti, S. Horn, P. Braca, and P. Willett, “Secure multi-party consensus gossip algorithms,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2014, pp. 7406–7410.

[14] M. Ruan, M. Ahmad, and Y. Wang, “Secure and privacy-preserving average consensus,” in *Proc. Workshop Cyber-Phys. Syst. Secur. Privacy*, 2017, pp. 123–129.

[15] N. M. Freris and P. Patrinos, “Distributed computing over encrypted data,” in *Proc. Allerton Conf. Commun., Control, Comput.*, 2016, pp. 1116–1122.

[16] C. N. Hadjicostis, N. H. Vaidya, and A. D. Domínguez-García, "Robust distributed average consensus via exchange of running sums," *IEEE Trans. Autom. Control*, vol. 61, no. 6, pp. 1492–1507, Jun. 2016.

[17] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proc. Annu. IEEE Symp. Found. Comput. Sci.*, 2003, pp. 482–491.

[18] F. Bénézit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in *Proc. IEEE Int. Symp. Inf. Theory*, 2010, pp. 1753–1757.

[19] C. N. Hadjicostis and T. Charalambous, "Average consensus in the presence of delays in directed graph topologies," *IEEE Trans. Autom. Control*, vol. 59, no. 3, pp. 763–768, Mar. 2014.

[20] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[21] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Found. Secure Comput.*, vol. 4, no. 11, pp. 169–180, 1978.

[22] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2010, pp. 24–43.

[23] E. Seneta, *Non-Negative Matrices and Markov Chains*. New York, NY, USA: Springer, 2006.

[24] N. E. Manitara and C. N. Hadjicostis, "Distributed stopping for average consensus in digraphs," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 3, pp. 957–967, Sep. 2018.

[25] S. Sundaram and C. N. Hadjicostis, "Distributed function calculation via linear iterative strategies in the presence of malicious agents," *IEEE Trans. Autom. Control*, vol. 56, no. 7, pp. 1495–1508, Jul. 2011.

[26] N. H. Vaidya, L. Tseng, and G. Liang, "Iterative approximate Byzantine consensus in arbitrary directed graphs," in *Proc. ACM Symp. Principles Distrib. Comput.*, 2012, pp. 365–374.

[27] H. J. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram, "Resilient asymptotic consensus in robust networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 4, pp. 766–781, Apr. 2013.