Privacy-Preserving Gossip Algorithms
Yang Liu, Junfeng Wu, Ian Manchester, Guodong Shi

Abstract

We propose gossip algorithms that can preserve the sum of network values (and therefore the average), and in the meantime fully protect node privacy in terms of their initial values even against external eavesdroppers possessing the entire information flow and network knowledge. At each time step, a node is selected to interact with one of its neighbors via deterministic or random gossiping. Such node generates a random number as its new state, and sends the subtraction between its current state and that random number to the neighbor. Then the neighbor updates its state by adding the received value to its current state. It turns out that this type of privacy-preserving gossiping algorithms can be used as a simple encryption step in a number of consensus-based distributed computation or optimization algorithms, so that we can fully protect the individual algebraic equations or cost functions from being observed or reconstructed throughout the processing of such algorithms. With deterministic gossiping we establish its concrete privacy-preservation guarantee by proving impossibilities for the reconstruction of the node initial values, and potential strategies of adversarial eavesdroppers are investigated in details. With randomized gossiping, the desired privacy encryption is achieved by fully distributed and self-organized node updates. In both cases, we manage to characterize the convergence limits explicitly and analytically, with clear speed of convergence being established. Finally, we attempt to illustrate that the proposed algorithms can be generalized in real-world applications for making trade-offs between resilience against node dropout or communication failure and privacy preservation capabilities.

1 Introduction

1.1 Background and Motivation

The development of distributed control, optimization, and computation algorithms has become one of the central streams in the study of complex network operations, due to the rapidly growing volume and dimension of data and information flow from a variety of applications such as social networking, smart...
grid, and intelligent transportation [3]. Celebrated results have been established for problems ranging from optimization [4,5] and learning [6] to formation [7] and localization [8], where distributed solutions provide resilience and scalability for large-scale problems through allocating information sensing and decision making over the individual agents organized via underlying communication networks. The origin of this line of research can be traced back to the 1980s from the work of distributed optimization and decision making [9] and parallel computations [10].

Consensus algorithms serve as a foundational tool for the information dissemination of distributed algorithms [11,12], where the goal is to drive the node states of a network to a common value related to the network initials, usually just the average. In such algorithms, nodes only share their state information with a few other nodes that are connected to or trusted which are called neighbors, with robustness to asynchronous node updates and intermittent communications [13,14]. Building on consensus algorithms, many algorithms [3] can be designed which utilize the distributed nature of consensus algorithms to achieve useful control and computation objectives, e.g., the recent work on network linear equation solvers [15,16] and a general optimization problem with the form of summed local objectives [5]. In particular, gossip algorithms [17] provided a simple way of self-organizing the node interactions in consensus seeking, where only a pair of nodes interacts with each other at any given time instance [18]. Protocols based on gossiping have led to new distributed networking solutions in large-scale applications [19] and new models of social network evolutions [20].

However, in consensus algorithms or distributed computations in general, nodes have to accept the fact that their privacy, at least some of it, will be inevitably lost to their neighbors during the interactions. In fact, nodes lose their privacy in terms of initial values if an attacker, a malicious user, or an eavesdropper knows the information flow and the network structure [21,22]. In fact, as long as observability holds, the whole network initial values become re-constructable with only a segment of node state updates at a few selected nodes. Several insightful privacy-preserving consensus algorithms have been presented in the literature [23–25] in the past few years, where the central idea is to inject random noise or offsets in the node communication and iterations. The literature still lacks of a solution that can on one hand resolve the privacy issues by enforcing the node initial values to be completely unreconstructable, and on the other hand deliver accurate convergence to average consensus at every node.

1.2 The Problem

A network of \( n \) nodes is indexed in the set \( V = \{1, \ldots, n\} \) and interconnected according to a set of edges \( E \), forming an underlying graph \( G = (V, E) \). Each node \( i \in V \) holds an initial state \( \beta_i \in \mathbb{R}^r \). The goal is to design an algorithm over the network so that after a recursive process of running each node \( i \) holds \( \beta_i^\# \in \mathbb{R}^r \), during which the following conditions must be satisfied.

(i) \( \text{(Graph Compliance)} \) Each node \( i \) communicates only with its neighbors in the set \( N_i := \{ j : \{i, j\} \in E \} \);
(ii) *(Local Privacy Preservation)* Each node $i$ never reveals its initial value $\beta_i$ to any other agents or a third party;

(iii) *(Global Privacy Preservation)* $(\beta_1^\top \ldots \beta_n^\top)^\top$ is non-identifiable given $(\beta_1^\top \ldots \beta_n^\top)^\top$;

(iv) *(Summation Consistency)* $\sum_{i=1}^n \beta_i^\top = \sum_{i=1}^n \beta_i$.

The condition (i) requires that the information flow of the algorithm must comply with the underlying graph; the condition (ii) says that during running of the algorithm no node will have to directly reveal its initial value $\beta_i$; the condition (iii) further suggests that even if final node states are known, it should not be possible to use that knowledge to recover $\beta_i$; the condition (iv) asks for that sum of the output of the algorithm should be consistent with that of the input. The global privacy preservation condition can be certainly strengthened to the case where the $\beta_i$ should be unidentifiable given the full information of the intermediate node states throughout the running of the algorithm.

**Definition 1.** An algorithm running over the node set $V$ is called a Privacy-Preserving-Summation-Consistent (PPSC) algorithm, which produces output $(\beta_1^\top \ldots \beta_n^\top)^\top$ from the network input $(\beta_1^\top \ldots \beta_n^\top)^\top$, if the four conditions on graph compliance, local privacy preservation, global privacy preservation, and summation consistency are all satisfied.

### 1.3 Applications

We now show that such PPSC algorithms can be used as a universal privacy preservation component in distributed computations via two basic examples in average consensus and network linear equations. To this end, we introduce $w_{ij} = w_{ji} > 0$ as the weight of the edge $\{i, j\} \in E$, and $w_{ii} > 0$ denote the self-weight of node $i \in V$. We assume

$$\sum_{j \in N_i \cup \{i\}} w_{ij} = 1$$

for all $i \in V$.

#### 1.3.1 Average Consensus

Each node $i \in V$ holds $y_i(t) \in \mathbb{R}, \ t \in \mathbb{Z}_{\geq 0}$. The standard averaging consensus algorithm is given by

$$y_i(t+1) = \sum_{j \in N_i \cup \{i\}} w_{ij} y_j(t), \ t \in \mathbb{Z}_{\geq 0}, \ i \in V. \quad (1)$$

Under this algorithm, one has

$$\lim_{t \to \infty} y_i(t) = \frac{\sum_{j=1}^n y_j(0)}/n, \ i \in V$$

if and only if $G$ is connected. It is known that with $y_1(0) = \beta_1, \ldots, y_n(0) = \beta_n$ the initial node states $(\beta_1, \ldots, \beta_n)$ can be recovered by each node $i$ from a finite number of its observations of $y_i(t), \ t \in \mathbb{Z}^+$.
under some observability conditions \cite{24}. We can now run a distributed PPSC algorithm over \((\beta_1, \ldots, \beta_n)\) and generate \(\left(\beta_1^\sharp, \ldots, \beta_n^\sharp\right)\). Letting \(y_i(0) = \beta_i^\sharp, i \in V\) in (1), there still holds \(\lim_{t \to \infty} y_i(t) = \sum_{j=1}^{n} \beta_j/n\), but \(\beta_i\) has been kept as private information for node \(i\).

### 1.3.2 Network Linear Equations

Consider a linear algebraic equation with respective to unknown \(y \in \mathbb{R}^r\)

\[
Hy = z
\]

with \(H \in \mathbb{R}^{n \times r}, z \in \mathbb{R}^n\). Let \(h_i^\top\) denote the \(i\)-th row of \(H\) and \(z_i\) denote the \(i\)-th component of \(z\). Let \(y_i(t) \in \mathbb{R}^r\) be held by \(i \in V\). Each node \(i\) only knows \(h_i\) and \(z_i\) and aims to solve the linear equation by sharing \(y_i(t)\) with its neighbors in \(N_i\). A distributed linear-equation solver has the following form:

\[
y_i(t+1) = \sum_{j \in N_i \cup \{i\}} w_{ij} \mathcal{P}_j(y_j(t)), \ t \in \mathbb{Z}^\geq 0, \ i \in V, \quad (3)
\]

where \(\mathcal{P}_i : \mathbb{R}^r \to \mathbb{R}^r\) is the projection onto the affine space \(\{y : h_i^\top y = z_i\}\). A similar analysis as in \cite{16} can be applied on (3), leading to

\[
\lim_{t \to \infty} y_i(t) = y^*, \ i \in V,
\]

where \(y^*\) is an exact solution of the given linear equation. It can be noted that the \(h_i\) and \(z_i\) known by node \(i\), in addition to the initial state \(y_i(0)\), are potentially recoverable by other nodes on the basis of a finite number of their node states under (3). In the following, we propose a modified network linear equation solver based on PPSC algorithms.

#### Privacy Preserving Linear Equation Solver

1. Set \(t \leftarrow 0\) and \(y_1(t), \ldots, y_n(t) \in \mathbb{R}^r\);
2. Run a PPSC algorithm with input \(y_1(t), \ldots, y_n(t)\) and produce output \(y_1^\sharp(t), \ldots, y_n^\sharp(t)\);
3. Run the averaging consensus algorithm with input \(y_1^\sharp(t), \ldots, y_n^\sharp(t)\) and output the agreement \(\bar{y}(t)\) at each node \(i\);
4. Each node \(i\) computes \(y_i(t+1) \leftarrow \mathcal{P}_i(\bar{y}(t))\). Set \(t \leftarrow t + 1\) and go to Step 2 unless satisfactory precision is achieved.

The above privacy preserving linear equation solver produces the following recursion:

\[
y_i(t+1) = \mathcal{P}_i\left(\frac{1}{n} \sum_{j=1}^{n} y_j(t)\right), \ t \in \mathbb{Z}^\geq 0, \ i \in V, \quad (4)
\]

which falls to the category of the projected consensus algorithms in \cite{5}. According to Proposition 2 in \cite{5}, each node state \(y_i(t)\) converges to a solution of the linear equation (2) if it admits exact solutions, i.e., \(z \in \text{span}\{H\}\). Along the proposed algorithm, it can be easily seen that \(y_i(t), t \in \mathbb{Z}^\geq 0\) is kept private to each node \(i\), which means the privacy of the linear equation \(h_i^\top y = z_i\) is also preserved.
1.3.3 Distributed Optimization

Consider a constrained optimization problem

$$
\min_{y \in \mathbb{R}^n} \sum_{i=1}^{n} f_i(y),
$$

(5)

where $f_i : \mathbb{R}^r \to \mathbb{R}$ is a differentiable convex function only known by node $i \in V$. It was shown in [5] that with an evolving state $y_i(t) \in \mathbb{R}^r$, $t \in \mathbb{Z}^\geq 0$ starting from given $y_i(0)$, each node $i$ can solve (5) by implementing the following distributed algorithm:

$$
y_i(t+1) = \sum_{j \in N_i \cup \{i\}} w_{ij} y_j(t) - \frac{1}{\sqrt{t+1}} \nabla f_i(y_i(t)).
$$

(6)

However, the knowledge of $\nabla f_i$ for each node $i$ can be potentially disclosed through $\{y_i(t)\}_{i \in V, t \in \mathbb{Z}^\geq 0}$ along (6). Inspired by (6), we provide a privacy-preserving distributed optimizer as following.

Privacy Preserving Distributed Optimizer

1: Set $t \leftarrow 0$ and $y_1(t), \ldots, y_n(t) \in \mathbb{R}^r$;
2: Each node $i$ computes $y_i^\sharp(t) \leftarrow y_i(t) - \frac{1}{\sqrt{t+1}} \nabla f_i(y_i(t))$;
3: Run a PPSC algorithm with input $y_1^\sharp(t), \ldots, y_n^\sharp(t)$ and produce output $\bar{y}_1(t), \ldots, \bar{y}_n(t)$;
4: Run the averaging consensus algorithm with input $y_1^\sharp(t), \ldots, y_n^\sharp(t)$ and output the agreement $\bar{y}(t)$ at each node $i$;
5: Each node $i$ sets $y_i(t+1) \leftarrow \bar{y}(t)$.
6: Set $t \leftarrow t + 1$ and go to Step 2 unless satisfactory precision is achieved.

The underlying dynamics of the privacy-preserving distributed optimizer described by

$$
y_i(t+1) = \sum_{j=1}^{n} \frac{y_j(t)}{n} - \frac{1}{\sqrt{t+1}} \sum_{j=1}^{n} \frac{\nabla f_j(y_j(t))}{n}.
$$

(7)

Evidently, the proposed distributed optimizer keeps the gradient information $\nabla f_i(y_i(t))$ strictly private to node $i$ even when $\{y_i(t)\}_{i \in V, t \in \mathbb{Z}^\geq 0}$ is observed, and therefore protects the privacy of node $i$ in terms of the cost function $f_i$.

1.4 Contributions

In this paper, we present a type of gossip-based PPSC algorithms that can preserve the sum of network values (and therefore the average) and fully protect node privacy even for eavesdroppers possessing the entire information flow and network knowledge. At each time step, one node is selected to initialize an interaction with its neighbor via deterministic or random gossiping [18, 26, 27]. This node generates a random number as its new state, and sends the subtraction between its current state and that random number to the neighbor. Then the neighbor updates its state by adding the received value to its current state. In this way, the entire network states will be shuffled without affecting the sum of the states in
finite time. With these privacy-preserving gossiping algorithms embedded in distributed computation or optimization, private algebraic equations or cost functions can be protected throughout the processing of such algorithms. Convergence properties and the convergence limits are also carefully studied, with concrete privacy-preservation performance characterized by proven impossibilities for the reconstruction of the node initial values. The proposed deterministic gossip algorithm reveals several key privacy-preserving properties as a theoretic benchmark; the randomized extension then adds to resilience and robustness for the practical implementation of the algorithms while retaining the privacy-preserving advantages. A preliminary version of the results are to be reported at IEEE Conference on Decision and Control in Dec. 2018 [1,2], where in the current manuscript we have provided all technical proofs and a number of refined examples.

1.5 Paper Organization

The deterministic privacy-preserving algorithm is proposed in Section 2. In Section 3, we introduce the precise definitions of network eavesdroppers and establish a universal non-identifiability theorem, and further investigates the possible strategies of the eavesdroppers and the corresponding performances, illustrated by two numerical examples. The randomized privacy-preserving algorithms are presented in Section 4, in which the trade-off between resilience and privacy preservation is also analyzed. Some concluding remarks are finally drawn in Section 5.

1.6 Notation and Preliminaries

1.6.1 Graph Theory

Consider an undirected graph \( G = (V, E) \). Nodes \( i \) and \( j \) are adjacent if \( \{i, j\} \in E \). A sequence of distinct nodes \( i_0, i_1, \ldots, i_l \) is said to be a path of length \( l \geq 1 \) between node \( i_0 \) and \( i_l \) if \( i_j \) and \( i_{j+1} \) are adjacent for all \( j = 0, 1, \ldots, l - 1 \). A graph \( G \) is connected if there exists at least one path between \( i \) and \( j \) for any \( i \neq j \in V \). A spanning subgraph of \( G \) is defined as a graph with its node set being \( V \) and its edge set being a subset of \( E \). Then we say \( T_G = (V, E_T) \) is a spanning tree of connected \( G = (V, E) \) if \( T_G \) is a spanning subgraph of \( G \) and is a tree. An orientation over the edge set \( E \) is a mapping \( o : E \rightarrow \{-1, 1\} \) with \( o(i, j) = -o(j, i) \) for all \( \{i, j\} \in E \). A directed edge, denoted by an ordered pair of nodes \((i, j)\), is generated under this orientation \( o \) if \( o(i, j) = 1 \). Particularly, \( i \) is the tail of the directed edge \((i, j)\), denoted by \( \text{Tail}(i, j) \); and \( j \) is the head denoted by \( \text{Head}(i, j) \). The graph \( G \) with an orientation \( o \) results in a directed graph \( G^o = (V, E^o) \) with \( E^o = \{(i, j) : o(i, j) = 1, \forall \{i, j\} \in E\} \), and in turn a sequence of distinct nodes \( i_0, i_1, \ldots, i_l \) is said to be a directed path of length \( l \geq 1 \) if \( (i_j, i_{j+1}) \in E^o \) for all \( j = 0, 1, \ldots, l - 1 \). We refer to [28] for more details on graph theory.
1.6.2 Conditional Entropy

Let $X$ be a continuous random variable with the PDF $f_X$ and the support set $\mathbb{X}$. Then the differential entropy of $X$ is

$$h(X) = - \int_{\mathbb{X}} f_X(x) \log f_X(x) \, dx.$$  

Let $Y$ be a continuous random variable with the support set $\mathbb{Y}$. Let $f_{X,Y}$ denote the joint PDF of $X,Y$. Then the joint differential entropy of $X,Y$ is

$$h(X,Y) = - \int_{\mathbb{X}} \int_{\mathbb{Y}} f_{X,Y}(x,y) \log f_{X,Y}(x,y) \, dx \, dy.$$  

The conditional differential entropy of $Y$ conditioned on $X$ is

$$h(Y \mid X) = h(X,Y) - h(X).$$  

This conditional differential entropy $h(Y \mid X)$ measures the average uncertainty of the outcome of $Y$ over all possible outcomes of $X$. We refer to [29] for more details on conditional entropy.

2 Deterministic PPSC Gossiping

In this section, we present a deterministic gossip-based PPSC algorithm. To carry out a deterministic gossiping process over a network, one has to arrange the sequence of node interactions, which is often a hard constraint in practical environments. However, the study of deterministic gossip protocols can eliminate the randomness of the gossiping process, and therefore allow for analysis focusing on the inherent randomness of the node updates themselves. Therefore, deterministic gossip algorithms often serve well as benchmarks for the performance of the general category of gossip algorithms [14].

2.1 The Algorithm

Without loss of generality we assume $r = 1$, i.e., each node $i$ holds $x_i(t) \in \mathbb{R}$ starting from $x_i(0) = \beta_i$. Let $T_G = (V, E_T)$ be a spanning tree of $G$. Then we assign an arbitrary orientation $o$ to $T_G$ so that an oriented tree $T_G^o = (V, E_T^o)$ is obtained. We sort the directed edges in $E_T^o$ by $E_1^o, E_2^o, \ldots, E_{n-1}^o$ with an arbitrary order. At time $t = 1, 2, \ldots, T_*$ with $T_* = n - 1$, the following steps are taken in order:

(i) Node $\text{Tail}(E_t)$ randomly and independently generates $\gamma_t \in \mathbb{R}$ according to some distribution with finite mean $\phi_\gamma$ and variance $\sigma_\gamma^2 > 0$;

(ii) Node $\text{Tail}(E_t)$ computes $\omega_t = x_{\text{Tail}(E_t)}(t - 1) - \gamma_t$ and sends $\omega_t$ to $\text{Head}(E_t)$;

(iii) Node $\text{Tail}(E_t)$ updates its state by $x_{\text{Tail}(E_t)}(t) = \gamma_t$, node $\text{Head}(E_t)$ updates its state by $x_{\text{Head}(E_t)}(t) = x_{\text{Head}(E_t)}(t - 1) + \omega_t$, and each node $i \in V \setminus \{\text{Tail}(E_t), \text{Head}(E_t)\}$ sets $x_i(t) = x_i(t - 1)$. 

7
Here $\gamma_t$, $t = 1, 2, \ldots, T_*$ are independently and identically distributed (i.i.d.) random variables. The algorithm can be re-expressed by the following equations:

$$
\begin{align*}
x_{\text{Tail}(E_t)}(t) &= \gamma_t \\
x_{\text{Head}(E_t)}(t) &= x_{\text{Head}(E_t)}(t - 1) + x_{\text{Tail}(E_t)}(t - 1) - \gamma_t \\
x_i(t) &= x_i(t - 1), \; i \in V \setminus \{\text{Tail}(E_t), \text{Head}(E_t)\}
\end{align*}
$$

(8)

for $t = 1, 2, \ldots, T_*$. It is obvious that for $\beta_i \in \mathbb{R}^r$ with $r > 1$, the algorithm (8) can run componentwise along each entry and lead to the same performance in terms of privacy preservation and summation consistency. Clearly, at each time $t$, only node $\text{Tail}(E_t)$ sends

$$
\omega_t = x_{\text{Tail}(E_t)}(t - 1) - \gamma_t
$$

to node $\text{Head}(E_t)$. Therefore, we can verify straightforwardly that the algorithm (8) satisfies the Graph Compliance, Local Privacy Preservation, and Summation Consistency conditions from our standing problem definition, leaving Global Privacy Preservation the only property to be further studied.

**Remark.** The communication content among nodes along the algorithm (8) includes offsets, which are constantly and independently generated at each discretized time. Then private initial states are evidently not perfectly recoverable given only the communication contents. Therefore, the investigation of the property Global Privacy Preservation will be based on the network state observations. In fact, if both the node communication content and the node state trajectories are exposed, it is in generally impossible to achieve full privacy protection under exact convergence guarantees.

### 2.2 Relation to Existing Work

Several privacy-preserving algorithms have been proposed in the literature for consensus seeking based on the idea of injecting random offsets to node state updates [23, 25, 30]. With plain random noise injection into the standard consensus algorithm (1), one inevitably lose accuracy in the convergence limit even in the probabilistic sense [30]. However, one can show that the output of such type of algorithms can be differentially private while maintaining a certain degree of error [23]. It was further shown in [25] that if one carries out noise injection for a finite number of times and then removes the total offsets once and for all, one can obtain convergence at the exact network average. Nodes therefore needed to maintain additional memories of each offset for the implementation of the algorithm in [25], and the offsets can be detected and reconstructed with the node states update trajectories. Recently, a comprehensive analysis was presented for the privacy protection abilities in noise-injection mechanism for average consensus algorithms with diminishing noises [24].

We remark that the use of random noise in our algorithm (8) is more than a random offset as one of the node states in the gossiping pair has been fully randomized. This leads to two advantages in terms of convergence and privacy-preservation:
(i) The algorithm [8] converges in finite time where the network sum is accurately maintained at the algorithm output. Even its randomized variations, which will be presented later, converge in finite time along every individual sample. While mean-square convergence [24] or small mean-square error [23] implies that one needs to repeat a number of samples for a single computation task to obtain practically accepted result.

(ii) Noise-injection algorithms are vulnerable against external eavesdroppers who may hold the entire network structure information and the trajectories of node state updates. Such an eavesdropper is in general equivalent to a malicious node connected to the entire network under the framework of [24,25], where the entire network initials can be disclosed even with noisy state observations [24]. By contrast, one can show that under the algorithm (8) the network initials are not identifiable even for external eavesdodgers.

2.3 Algorithm Output Statistics

Clearly the algorithm (8) yields a group of random variables as terminal network states. Those random terminal states contain information about the initial value, whose relationship and distributions would affect the further use of such states (as in network linear equation solver, for instance). We therefore establish some characteristics of those terminal states along algorithm (8).

We describes the order of two directed edges in time as \( E_s \prec E_t \) if \( s < t \) and \( E_s \preceq E_t \) if \( s \leq t \), and write \( \min(E_s,E_t) = E_\alpha \) with \( \alpha = \min(s,t) \), and \( \max(E_s,E_t) = E_\alpha \) with \( \alpha = \max(s,t) \). The following theorem proposes a necessary and sufficient condition for the dependence of two nodes’ final states in the case when the two nodes are not directly connected by a path.

**Theorem 1.** Suppose there exists no directed path in \( T_G^o \) connecting node \( i \) and node \( j \). Let \( i_0,i_1,...,i_l \) denote the unique undirected path in \( T_G \) that connects node \( i \) and node \( j \) with \( i = i_0 \) and \( j = i_l \). Then along the algorithm (8), \( x_{i_0}(T_\ast) \) and \( x_{i_l}(T_\ast) \) are dependent if and only if there exists \( 0 < p < l \) such that the following conditions hold.

(i) \( i_p,i_{p-1},...,i_0 \) and \( i_p,i_{p+1},...,i_l \) are both directed paths.

(ii) There hold \( (i_p,i_{p-1}) \prec \cdots \prec (i_1,i_0) \) and \( (i_p,i_{p+1}) \prec \cdots \prec (i_{l-1},i_l) \).

(iii) a) If \( (i_0,i^*) \in E_\pi^o, \text{ then } (i_0,i^*) \prec (i_0,i_1) \);

b) If \( (i_b,i^*) \in E_\pi^o \) for \( 0 < b < p \), then \( (i_b,i^*) \prec (i_{b+1},i_b) \) or \( (i_b,i_{b-1}) \preceq (i_b,i^*) \);

c) If \( (i_p,i^*) \in E_\pi^o \), then

\[
(i_p,i^*) \prec \min\left((i_p,i_{p-1}),(i_p,i_{p+1})\right)
\]

or

\[
\max\left((i_p,i_{p-1}),(i_p,i_{p+1})\right) \preceq (i_p,i^*) ;
\]

\[^1\]There can still be some subtle difference between an external eavesdropper or a malicious node since as a malicious node, it will know its own initial value precisely which will influence the network state evolution.
d) If \((i_b, i^*) \in E_T^o\) with \(p < b < l\), then \((i_b, i^*) \prec (i_{b-1}, i_b)\) or \((i_b, i_{b+1}) \preceq (i_b, i^*)\);
e) If \((i_l, i^*) \in E_T^o\), then \((i_l, i^*) \prec (i_{l-1}, i_l)\).

Intuitively Condition (i) and Condition (ii) in Theorem 1 constrain the directions and selection sequences of the oriented edges on the path \(i_0, \ldots, i_l\). It is worth noting that Condition (iii) further characterizes the sequential order of selecting the edges on the path \(i_0, \ldots, i_l\), and the edges with one endpoints in the path \(i_0, \ldots, i_l\). The following theorem provides a necessary and sufficient condition for the dependence of two arbitrary nodes’ final states when there exists a directed path connecting them.

**Theorem 2.** Suppose \(i_0, i_1, \ldots, i_l\) is a directed path of length \(l\) in \(T_G^o\). Then along the algorithm \(\mathbf{A}\) there holds that \(x_{i_0}(T^*)\) and \(x_{i_l}(T^*)\) are dependent if and only if the following conditions hold:

(i) \((i_0, i_1) \prec (i_1, i_2) \prec \cdots \prec (i_{l-1}, i_l)\).

(ii) a) \((i_0, i^*) \prec (i_0, i_1)\) when \((i_0, i^*) \in E_T^o\);

b) \((i_b, i^*) \prec (i_{b-1}, i_b)\) or \((i_b, i_{b+1}) \prec (i_b, i^*)\) when \((i_b, i^*) \in E_T^o\);

c) \((i_l, i^*) \prec (i_{l-1}, i_l)\) when \((i_l, i^*) \in E_T^o\).

Define \(\Sigma_t\) as the covariance matrix of random vector \(x(t) = [x_1(t) \ x_2(t) \ \ldots \ x_n(t)]^\top\). Let \([\Sigma_t]_{ij}\) denote the \(ij\)-th entry of \(\Sigma_t\). Based on \(\Sigma_t\), we provide the definition of the graphical model of a random vector \(x(t)\).

**Definition 2.** The graphical model of the random vector \(x(t) = [x_1(t) \ x_2(t) \ \ldots \ x_n(t)]^\top\) with \(t = 0, 1, \ldots, T^*\) is the undirected graph \(G_t = (V, \mathcal{E}_t)\), where the edge set is given by

\[
\mathcal{E}_t = \{\{i, j\} : i \neq j, \ [\Sigma_t]_{ij} \neq 0\}.
\]

Then the following theorem for \(G_t\) holds.

**Theorem 3.** The graphical model \(G_{T^*}\) is a tree. Moreover, \(\Sigma_{T^*}/\sigma^2\) is the Laplacian of \(G_{T^*}\).

### 2.4 Examples

We now present a few illustrative examples.

**Example 1.** Consider a 5-node undirected graph \(G = (V, E)\) as shown in Figure 1. We select an oriented spanning tree \(T_G^o = (V, E_T^o)\) as in Figure 2. Let the edges in \(E_T^o\) be sorted as

\[
\mathcal{E}_1 = (5, 2) \prec \mathcal{E}_2 = (2, 3) \prec \mathcal{E}_3 = (2, 1) \prec \mathcal{E}_4 = (3, 4).
\]
The algorithm starts with \( x_i(0) = \beta_i, \ i = 1, \ldots, 5 \) and produces
\[
    x(4) = \begin{bmatrix}
        \beta_1 + \gamma_2 - \gamma_3 \\
        \gamma_3 \\
        \gamma_4 \\
        \beta_2 + \beta_3 + \beta_4 + \beta_5 - \gamma_1 - \gamma_2 - \gamma_4 \\
        \gamma_1
    \end{bmatrix}.
\]

Evidently \( \sum_{i=1}^{5} x_i(4) = \sum_{i=1}^{5} \beta_i \), i.e., network node states sum is preserved. In addition, one can see that the following conditions hold: (i)2, 1 and 2, 3, 4 are both directed paths in \( T^o_G \); (ii) \((2, 3) \prec (3, 4)\); (iii) There exists no node \( i^* \) such that \((i, i^*) \in E^o_T\) for any \( i = 1, 2, 3, 4 \). This shows that the node state pair \((x_1(4), x_4(4))\) satisfies the dependence conditions of Theorem 1. Clearly \( x_1(4) \) and \( x_4(4) \) are dependent, which validates Theorem 1. Validation of Theorem 2 can be similarly shown from the node state pair \( x_4(4) \) and \( x_5(4) \).

The graphical model \( G_4 \) of \( x(4) \) is illustrated in Figure 3. By direct calculation, we have
\[
    \Sigma_4 = \begin{bmatrix}
        2\sigma_\gamma^2 & -\sigma_\gamma^2 & 0 & -\sigma_\gamma^2 & 0 \\
        -\sigma_\gamma^2 & \sigma_\gamma^2 & 0 & 0 & 0 \\
        0 & 0 & \sigma^2 & -\sigma_\gamma^2 & 0 \\
        -\sigma_\gamma^2 & 0 & -\sigma_\gamma^2 & 3\sigma_\gamma^2 & -\sigma_\gamma^2 \\
        0 & 0 & 0 & -\sigma_\gamma^2 & \sigma_\gamma^2
    \end{bmatrix}.
\]
Figure 3: The graphical model $G_4$ of the node states $x(4)$ over $T_G$ along the algorithm (8) with the edge selection order $(5, 2) \prec (2, 3) \prec (2, 1) \prec (3, 4)$.

It is clear that $\Sigma_4/\sigma_4^2$ is the Laplacian of $G_4$ and this validates Theorem 3.

Example 2. Consider a linear equation $H y = z$ with respect to $y \in \mathbb{R}^2$ where

$$
H = \begin{bmatrix}
2 & -1.5 \\
1 & 1 \\
-0.5 & 0.8
\end{bmatrix}, \quad z = \begin{bmatrix}
2.5 \\
3 \\
-0.2
\end{bmatrix},
$$

which has an exact solution $y^* = [2 \ 1]^\top$. Let $G_{\text{ring}}$ be a 3-node ring graph. Suppose each node $i \in \{1, 2, 3\}$ of $G_{\text{ring}}$ only has the knowledge of the $i$-th row of $H$ and the $i$-th component of $z$, and aims to obtain the solution $y^*$. Then we run the Privacy Preserving Linear Equation Solver over $G_{\text{ring}}$ with the initial values $y_1(0) = [-40 \ 60]^\top$, $y_2(0) = [0 \ 65]^\top$, $y_3(0) = [35 \ 55]^\top$. The execution of the node states computation for the first time step is plotted in Figure 4. Each node always has its state encrypted before sharing it, thus can keep the state, and further its knowledge of the linear equation private in the computation.

Example 3. Consider again the linear equation $H y = z$ in Example 2, where $H$ remains the same but $z = [1.5 \ 2 - 2]^\top$. In this case, the equation is unsolvable. The well-known least-squares solution of the unsolvable equation is defined as the optimal solution of the following problem

$$
\min_{y \in \mathbb{R}^2} \quad \frac{1}{2} \sum_{i=1}^{3} \left| h_i^\top y - z_i \right|^2
$$

(9)

with $h_i^\top$ being the $i$-th row of $H$ and $z_i$ being the $i$-th component of $z$ for $i = 1, 2, 3$. Clearly (9) is a distributed optimization problem defined in (5). The unique optimal solution is $\hat{y} = [1.29 \ 0.32]^\top$ by simple calculation. Let the privacy-preserving distributed optimizer (7) run over $G_{\text{ring}}$ with randomly selected initial states over $[0, 1]$. Then we plot the trajectories of $e(t) = \sum_{i=1}^{3} \left| y_i(t) - \hat{y} \right|^2$, $e^\delta(t) = \sum_{i=1}^{3} \left| y_i^\delta(t) - \hat{y} \right|^2$ and $e^\sharp(t) = \sum_{i=1}^{3} \left| y_i^\sharp(t) - \hat{y} \right|^2$ in Figure 5, respectively. On the one hand, both $e(t)$ and $e^\delta(t)$ go to zero, indicating that the algorithm solves the least-squares problem (9). On the other hand, $e^\sharp(t)$ fluctuates throughout the time horizon, implying the privacy preservation of the individual cost functions.
Figure 4: A geometric illustration of the privacy-preserving linear-equation solver at the first time step. The arrows in blue, green and red show the state trajectories of the node 1, 2, 3, respectively.

Figure 5: The trajectories of \( e(t) = \sum_{i=1}^{3} ||y_i(t) - \hat{y}||^2 \), \( e^b(t) = \sum_{i=1}^{3} ||y_i^b(t) - \hat{y}||^2 \) and \( e^\#(t) = \sum_{i=1}^{3} ||y_i^\#(t) - \hat{y}||^2 \). The evolution of \( e^\#(t) \) over time characterizes the encryption process, and the tendency of \( e(t) \) and \( e^\#(t) \) shows that the convergence results can be achieved.
2.5 Key Lemmas

Noting that the sum of node states remains the same over time, i.e., \( \sum_{i=1}^{n} x_i(t) = \sum_{i=1}^{n} x_i(0) \) for all \( 0 < t \leq T_s \), it is evident that each node \( i \)'s state can be expressed as

\[
x_i(t) = \sum_{j=1}^{n} c_{ij}(t) \beta_j + \sum_{j=1}^{t} d_{ij}(t) \gamma_j,
\]

where \( c_{ij}(t) \in \{0, 1\} \) and \( d_{ij}(t) \in \{-1, 0, 1\} \) represents random variable \( d_{ij}(t) \gamma_j \) appears in node state \( x_i(t) \). In a state \( x_i(t) \), \( d_{ij}(t) \gamma_j \) with \( d_{ij}(t) \neq 0 \) is a random component. The computation along (8) is a process of the \( x_i(t) \) gaining and losing these random components. We note a few basic rules for that process.

(i) For any time \( t \geq s \), \( \gamma_s \) and \( -\gamma_s \) belong to different node states, i.e., appear in the states of two different nodes.

(ii) Any random component can only be gained at a head from a tail along their directed link.

(iii) The random components do not change their signs when being gained or lost.

The following lemma illustrates the way that a random variable is passed from the state of one node to that of another.

**Lemma 1.** Suppose \( n > 2 \). Let \( t_s \in \{1, 2, \ldots, T_s - 1\} \) and \( s \in \{1, 2, \ldots, t_s\} \). Let node \( i \) and node \( j \) satisfy \( d_{is}(t_s) \neq 0 \) and \( d_{js}(t_s) = 0 \), respectively. Denote \( i_0, i_1, \ldots, i_l \) as the unique undirected path in \( T_G \) connecting node \( i \) and \( j \) with \( i = i_0 \) and \( j = i_l \). Define \( t^* \in \{t_s + 1, 2, \ldots, T_s\} \). Then \( d_{is}(t^*) = d_{is}(t_s) \) if and only if the following conditions hold:

(i) \( i_0, i_1, \ldots, i_l \) is a directed path.

(ii) \( \mathcal{E}_{t_s} < (i_0, i_1) < \cdots < (i_{l-1}, i_l) \leq \mathcal{E}_{t^*} \).

(iii) a) \( (i_0, i^*) < \mathcal{E}_{t_s} \) or \( (i_0, i_1) \leq (i_0, i^*) \) when \( (i_0, i^*) \in E_T^n \);

b) \( (i_b, i^*) < (i_{b-1}, i_b) \) or \( (i_b, i_{b+1}) \leq (i_b, i^*) \) when \( (i_b, i^*) \in E_T^n \) with \( 0 < b < l \);

c) \( (i_1, i^*) < (i_{l-1}, i_l) \) or \( \mathcal{E}_{t^*} \leq (i_1, i^*) \) when \( (i_1, i^*) \in E_T^n \).

A lemma concerning the final node states can also be established.

**Lemma 2.** \( \mathbf{x}(T_s) = [x_1(T_s) \ x_2(T_s) \ \ldots \ \ x_n(T_s)]^\top \) has the following properties.

(i) Let \( s \in \{1, 2, \ldots, T_s\} \). Then there exists unique \( i, j \) with \( i \neq j \) such that \( d_{is}(T_s) = 1 \) and \( d_{js}(T_s) = -1 \).

(ii) If \( x_i(T_s) \) and \( x_j(T_s) \) are dependent, then there exists a unique \( s \in \{1, 2, \ldots, T_s\} \) that satisfies \( d_{is}(T_s)d_{js}(T_s) = -1 \).

The results of Theorems 1 - 3 are based on Lemma 1 and Lemma 2 whose proofs are all provided in the appendices.
3 Global Privacy Preserving Performance

We aim to protect every node’s privacy from being inferred by an observer, termed an external eavesdropper in contrast to malicious nodes \[24\], who manages to physically eavesdrop on the real-time node states in the process of distributed computation. In this section, we categorize possible eavesdroppers, show the non-identifiability of node initial states under the algorithm \[8\], discuss plausible threats to the privacy of network agents, and evaluate the performance of the algorithm \[8\] for protecting privacy against node state leakage.

3.1 Network Eavesdroppers

Eavesdroppers attempt to estimate the input \(\beta = [\beta_1 \ldots \beta_n]^\top\) through observations of the output \(\beta^\# = [\beta_1^\# \ldots \beta_n^\#]^\top\) or even all intermediate network states. We assume the statistics or even distribution of the \(\gamma_t\) is public knowledge. We define the knowledge space of potential eavesdroppers as \(K = \{G, T_G, T_{G^0}, (E_1, \ldots, E_{T^*})\}\).

In other words, eavesdroppers may possess the network topology \(G\), the spanning tree \(T_G\), the oriented spanning tree \(T_{G^0}\), and even the edge sequence \((E_1, \ldots, E_{T^*})\) as a prior knowledge. As a result, we impose the following definition.

**Definition 3.** Eavesdroppers who know \(G, T_G^0\) and \((E_1, \ldots, E_{T^*})\) are called Architectural Eavesdroppers; Eavesdroppers who only know \(G\) are called Topological Eavesdroppers.

Eavesdroppers may observe the final network state \(x(T_*)\) or the entire network state flow \(x(1), \ldots, x(T_*)\). Define \(\Phi_k \in \mathbb{R}^{n \times n}\) for \(k = 1, 2, \ldots, T^*\) with

\[
[\Phi_k]_{ij} = \begin{cases} 
1 & \text{if } i = \text{Head}(E_k) \text{ and } j = \text{Tail}(E_k); \\
-1 & \text{if } i = \text{Tail}(E_k) \text{ and } j = \text{Tail}(E_k); \\
0 & \text{otherwise}.
\end{cases}
\]

Then the update rule given in the algorithm \(8\) can be written in the following compact form.

\[
x(1) = A_1 \beta + \gamma_1 v_1, \\
x(2) = A_2 x(1) + \gamma_2 v_2, \\
\vdots
\]

\[
x(T_*) = A_{T^*} x(T_* - 1) + \gamma_{T^*} v_{T^*},
\]

where \(A_i = \Phi_i + I\), and \(v_i\) are the all-zeros vectors except for its Tail(\(E_i\))-th component being one and its Head(\(E_i\))-th component being minus one with \(i = 1, 2, \ldots, T_*\). Let \(C \in \mathbb{R}^{n \times n}\) and \(D \in \mathbb{R}^{n \times T_*}\) be matrices with \([C]_{ij} = c_{ij}(T_*)\) and \([D]_{ij} = d_{ij}(T_*)\) being defined in \(10\). Introduce \(\gamma = [\gamma_1 \ldots \gamma_{T^*}]^\top\). Then from \(10\)

\[
x(T_*) = C \beta + D \gamma.
\]
We use \( P_\beta \) to denote the probability distribution of \( [x(1)^\top \ldots x(T^*)]^\top \) parametrized by \( \beta \). We also let \( P^*_\beta \) be the probability distribution of \( x(T^*) \) parametrized by \( \beta \). Then we define two statistical models by

\[
\mathcal{P} = \{ P_\beta : \beta \in \mathbb{R}^n \}
\]

\[
\mathcal{P}^* = \{ P^*_\beta : \beta \in \mathbb{R}^n \}.
\]

Let \( \ker(M) \) denote the kernel of a matrix \( M \). For \( \mathcal{P} \) and \( \mathcal{P}^* \), we have the following result.

**Theorem 4.** For Architectural Eavesdroppers, there hold

\[
P_\beta = P_\beta + \eta, \forall \eta \in \ker(A_1),
\]

(13)

\[
P^*_\beta = P^*_\beta + \eta^*, \forall \eta^* \in \ker(C).
\]

(14)

Furthermore, for Architectural Eavesdroppers, both \( \mathcal{P} \) and \( \mathcal{P}^* \) are unidentifiable.

**Proof.** We first analyze the statistical model \( \mathcal{P} \). By the chain rule, the probability density function (PDF) of \( x(1), x(2), \ldots, x(T^*) \) under \( \beta \) is

\[
f(x(1), x(2), \ldots, x(T^*); \beta) = f(x(1) | \beta) \prod_{t=2}^{T^*} f(x(t) | x(1), \ldots, x(t-1)).
\]

(15)

Recall that (11) shows \( x(t) \) is a function of \( x(t-1) \) and \( \gamma_t \) for \( t = 1, 2, \ldots, T^* \), which implies that \( f(x(t) | x(1), \ldots, x(t-1)) = f(x(t) | x(t-1)), \ t = 2, \ldots, T^*. \) According to (11), all \( \eta \in \ker(A_1) \) satisfy

\[
f(x(1) | \beta) = f(x(1) | \beta + \eta), \ \forall x(1) \in \mathbb{R}^n.
\]

(16)

Clearly (15) and (16) complete the proof of (13). It is worth noting that \( \ker(A_1) \) is nonempty, because the \( \text{Tail}(E_1) \)-th row of \( A_1 \) is zero and thus \( A_1 \) is not full rank.

A similar analysis will help us draw the desired conclusion for \( \mathcal{P}^* \). We have now completed the proof. \( \square \)

Theorem 4 shows that both \( \mathcal{P} \) and \( \mathcal{P}^* \) are unidentifiable for Architectural Eavesdroppers, which immediately implies that \( \mathcal{P} \) and \( \mathcal{P}^* \) are also unidentifiable for Topological Eavesdroppers since they possess less knowledge than Architectural Eavesdroppers. We would also like to point out that the implication of Theorem 1 for the initial-value protection power with PPSC gossip algorithms is much beyond the statement itself as it is proven under the most general terms on the eavesdroppers. By slightly generalizing it to multi-gossiping form [18] where multiple pairs of nodes can be selected at one time, or the pair sequence is only partially observed, identifiability of the network initials will be further and drastically reduced.

### 3.2 Architectural Eavesdroppers’ Estimation Strategies

Though \( \mathcal{P} \) and \( \mathcal{P}^* \) have been proved to be unidentifiable under algorithm [9], eavesdroppers can estimate the node initial states based on their observations of node states. In this section, we study the estimating behaviors of eavesdroppers with maximum likelihood estimation and maximum a posteriori estimation.
We assume \( \gamma_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2) \) for \( t = 1, 2, \ldots, T_s \). Equivalently, we have \( \gamma \sim \mathcal{N}(0, \Lambda) \) with \( \Lambda = \sigma^2 I \). Let \( d_i^T \) denote the \( i \)-th row of \( D \). Recall \( \gamma \sim \mathcal{N}(0, \Lambda) \) with \( \Lambda = \sigma^2 I \). Then \( D\gamma \sim \mathcal{N}(0, \Lambda_D) \) where

\[
[\Lambda_D]_{ij} = \begin{cases} 
\sigma^2||d_i||^2 & \text{if } i = j; \\
-\sigma^2 & \exists k \Rightarrow (D)_{i,j} = -1; \\
0 & \text{otherwise}.
\end{cases}
\]

Theorem 3 shows \( \Lambda_D/\sigma^2 \) is the Laplacian of a tree. In the following subsections, we assume that the algorithm \( 8 \) is repeatedly applied on a network independently for \( m > 0 \) times starting from the same initial value \( \beta \). Let \( x^i(t) \) with \( i = 1, 2, \ldots, m \) and \( t = 1, 2, \ldots, T_s \) denote the system state observed at time \( t \) for the \( m \)-th trial.

the algorithm \( 8 \) clearly states that for all \( i, j = 1, 2, \ldots, m, x^i_k(1) = x^j_k(1) \) for any \( k \neq \text{Head}(\mathcal{E}_i), \text{Tail}(\mathcal{E}_i) \) and \( x^\text{Tail}(\mathcal{E}_i)(1) + x^\text{Head}(\mathcal{E}_i)(1) = x^\text{Tail}(\mathcal{E}_i)(1) + x^\text{Head}(\mathcal{E}_i)(1) \). In the following we evaluate the privacy preservation ability of the algorithm \( 8 \) against Architectural eavesdroppers who use maximum likelihood estimation and maximum a posteriori estimation. Let \( \hat{\beta}_{\text{mle}}, \hat{\beta}_{\text{mle}}^* \) be the maximum likelihood estimators (MLE) of \( \beta \) in \( P \) and \( P^* \), respectively. Clearly Proposition 4 implies that \( \hat{\beta}_{\text{mle}}, \hat{\beta}_{\text{mle}}^* \) are both nonunique.

### 3.2.1 MLE with Full State Observations

We first analyze the statistical model \( P \) given the observations \( x(1), x(2), \ldots, x(T_s) \). Let \( |z| \) with \( z \) being a vector denote the element-wise absolute value of \( z \). Let \( \delta(z) \) be the Dirac delta function and \( f_{\gamma_i}(|z|) \) be the PDF of \( \gamma_i \) with \( i = 1, 2, \ldots, T_s \). Based on (11), we have for \( t = 1, 2, \ldots, T_s \),

\[
\begin{align*}
\mathcal{E}_i(t) | x(t-1) & \sim f_{\gamma_i}(|v_i|^T x(t-1) - x_{\text{Head}(\mathcal{E}_i)}(t)), \\
\mathcal{E}_i(t) | x_{\text{Head}(\mathcal{E}_i)}(t), x(t-1) & \sim \delta(x_{\text{Tail}(\mathcal{E}_i)}(t) + x_{\text{Head}(\mathcal{E}_i)}(t) - |v_i|^T x(t-1)).
\end{align*}
\]

In addition, \( x_j(t) | x(t-1) \) with \( j \neq \text{Head}(\mathcal{E}_i), \text{Tail}(\mathcal{E}_i) \) have degenerate distributions and take a single value at \( x_j(t-1) \). By (17), the PDF of \( x(t) | x(t-1) \) is given by

\[
\begin{align*}
f(x(t) | x(t-1)) &= f(x_{\text{Tail}(\mathcal{E}_i)}(t), x_{\text{Head}(\mathcal{E}_i)}(t) | x(t-1)) \prod_{i \neq \text{Tail}(\mathcal{E}_i), \text{Head}(\mathcal{E}_i)} f(x_i(t) | x(t-1)) \\
&= f(x_{\text{Head}(\mathcal{E}_i)}(t) | x(t-1)) f(x_{\text{Tail}(\mathcal{E}_i)}(t) | x_{\text{Head}(\mathcal{E}_i)}(t), x(t-1)) \prod_{i \neq \text{Tail}(\mathcal{E}_i), \text{Head}(\mathcal{E}_i)} \delta(x_i(t) - x_i(t-1)) \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{|v_i|^T x(t-1) - x_{\text{Head}(\mathcal{E}_i)}(t))^2}{2\sigma^2}\right) \delta(x_{\text{Tail}(\mathcal{E}_i)}(t) + x_{\text{Head}(\mathcal{E}_i)}(t) - |v_i|^T x(t-1)). \prod_{i \neq \text{Tail}(\mathcal{E}_i), \text{Head}(\mathcal{E}_i)} \delta(x_i(t) - x_i(t-1)).
\end{align*}
\]

17
Based on (15) and (18), an MLE of $\beta$ in $P$ is given by

$$
\hat{\beta}_{mle} \in \arg\max_{\beta} \prod_{i=1}^{m} f(x_i(1), x_i^2(2), \ldots, x_i^T; \beta)
$$

$$
= \arg\max_{\beta} \prod_{i=1}^{m} f(x_i(1) | \beta)
$$

$$
= \arg\max_{\beta} \prod_{i=1}^{m} \exp\left( -\frac{1}{2\sigma^2} (x_{\text{Head}(E_1)}^i(1) - |v_1^i| \beta)^2 \right) \delta(x_{\text{Head}(E_1)}^i(1) - |v_1^i| \beta).
$$

$$
\prod_{j \neq \text{Tail}(E_1), \text{Head}(E_1)} \delta(x_j^i(1) - \beta_j)). \tag{19}
$$

To maximize the objective function in (19), none of its Delta function factors should be zero, i.e., $\beta$ must be in the set

$$
B = \{ \beta : A_1^T A_1 \beta = A_1^T A_1 x(1) \}.
$$

When $\beta \in B$, the exponential factors in (19) become independent of $\beta$. Thus the MLE of $\beta$ in $P$ evaluated from (19) is given by $\hat{\beta}_{mle} \in B$, namely

$$
\hat{\beta}_{mle} \in \{ \beta \in \mathbb{R}^n : A_1^T A_1 \beta = A_1^T A_1 x^i(1) \}. \tag{20}
$$

It can be seen from (20) that the estimated initial states of the endpoints of $E_1$ only have to add up to the sum of their states at time $t = 1$, while other estimated node states equal their states at time $t = 1$, which is consistent with our intuition. Thus this MLE does not provide any improvements on the performance of recovering the node initial states. Let $\text{var}(M)$ denote the covariance matrix of $M \in \mathbb{R}^{n \times n}$. According to (20), we have for the bias of $\hat{\beta}_{mle}$

$$
A_1^T A_1 \text{bias}(\hat{\beta}_{mle})
$$

$$
= \mathbb{E}(A_1^T A_1 \hat{\beta}_{mle}) - A_1^T A_1 \beta
$$

$$
= \mathbb{E}(A_1^T A_1 x^i(1)) - A_1^T A_1 \beta
$$

$$
= A_1^T A_1 (A_1 - I) \beta
$$

$$
= 0,
$$

and for the covariance matrix of $\hat{\beta}_{mle}$

$$
A_1^T A_1 \text{var}(\hat{\beta}_{mle}) A_1^T A_1
$$

$$
= A_1^T A_1 (\mathbb{E}(\hat{\beta}_{mle}\hat{\beta}_{mle}^T) - \mathbb{E}(\hat{\beta}_{mle})\mathbb{E}(\hat{\beta}_{mle}^T)) A_1^T A_1
$$

$$
= \mathbb{E}(A_1^T A_1 \hat{\beta}_{mle}\hat{\beta}_{mle}^T A_1^T A_1) - \mathbb{E}(A_1^T A_1 \hat{\beta}_{mle})\mathbb{E}(\hat{\beta}_{mle}^T A_1^T A_1)
$$

$$
= A_1^T A_1 \text{var}(x^i(1)) A_1^T A_1
$$

$$
= \sigma^2 A_1^T v_1^i v_1^T A_1
$$

$$
= \sigma^2 A_1^T A_1.
$$
This explains that there exist possible values of $\hat{\beta}_{mle}$ with nonzero bias and the variance of each element in $\hat{\beta}_{mle}$ does not improve with the increasing number of experiments. Furthermore, $\sigma^2 \Lambda_1^T A_1$ is a zero matrix except for the Tail($\mathcal{E}_1$)Tail($\mathcal{E}_1$)-th, Tail($\mathcal{E}_1$)Head($\mathcal{E}_1$)-th, Head($\mathcal{E}_1$)Tail($\mathcal{E}_1$)-th and Head($\mathcal{E}_1$)Head($\mathcal{E}_1$)-th entries being $\sigma^2$.

### 3.2.2 MLE with Terminal State Observations

Now we study the MLE of $\beta$ in $\mathcal{P}^*$. Evidently the distributions of $x^i(T_*)$, $i = 1, \ldots, m$ are degenerate because $\Lambda D$ is singular. Let $y = [x_2(T_*) \ldots x_n(T_*)]^T$. Correspondingly, we define $y^i = [x^i_2(T_*) \ldots x^i_n(T_*)]^T$ for $i = 1, \ldots, m$. Define $\mu_1, \lambda_{11} \in \mathbb{R}$, $\mu_y, \lambda_{1y} \in \mathbb{R}^{n-1}$, $c_1 \in \mathbb{R}^n$, $\Lambda_{yy} \in \mathbb{R}^{(n-1) \times (n-1)}$, $C_y \in \mathbb{R}^{(n-1) \times n}$ with

$$C/\beta = \begin{bmatrix} \mu_1 \\ \mu_y \end{bmatrix}, \quad \Lambda_D = \begin{bmatrix} \lambda_{11} & \lambda^1_y \\ \lambda_{1y} & \Lambda_{yy} \end{bmatrix}. \quad (21)$$

$\Lambda_{yy}$ is noted to be nonsingular by Cauchy’s interlacing theorem [32] and the fact that $\Lambda D$ is the Laplacian of a connected graph. From (12), we know $x_1(T_*) | y$ and $y$ are both normally distributed as follows [33]:

$$x_1(T_*) | y \sim N(\mu_1 + \phi_{1y}(y - \mu_y), \psi_{1y}) \quad (22)$$

$$y \sim N(\mu_y, \Lambda_{yy}). \quad (23)$$

where

$$\phi_{1y} = \Lambda_{yy}^{-1} \lambda_{1y}$$

$$\psi_{1y} = \lambda_{11} - \lambda_{1y}^\top \Lambda_{yy}^{-1} \lambda_{1y}.$$ 

Further by $\Lambda D 1 = 0$, one has

$$\phi_{1y} = -1, \quad \psi_{1y} = 0. \quad (24)$$

Then (22) and (24) imply

$$x_1(T_*) | y \sim \delta(1^\top x(T_*) - 1^\top \beta). \quad (25)$$

Based on (21), (23) and (25), one can find

$$\hat{\beta}_{mle}^* \in \arg\max_{\beta} \prod_{i=1}^m f(x^i(T_*); \beta)$$

$$\quad = \arg\max_{\beta} \prod_{i=1}^m \delta(1^\top x^i(T_*) - 1^\top \beta) \exp\left(-\frac{1}{2}(x^i(T_*) - C\beta)^\top \hat{\Lambda}(x^i(T_*) - C\beta)\right), \quad (26)$$

where

$$\hat{\Lambda} = \begin{bmatrix} 0 & 0 \\ 0 & \Lambda_{yy}^{-1} \end{bmatrix}.$$ 

Evidently the unconstrained optimization problem (26) is equivalent to the constrained one

$$\min_{\beta} \sum_{i=1}^m (x^i(T_*) - C\beta)^\top \hat{\Lambda}(x^i(T_*) - C\beta)$$

$$\text{s.t.} \quad 1^\top \beta = 1^\top x^1(T_*). \quad (27)$$
It is clear that Slater’s condition holds for (27) because the only constraint is an affine equality. Then the \( \beta \)s satisfying the following two KKT conditions \cite{34}

\[
\begin{align*}
0 &= \sum_{i=1}^{m} (C^\top \hat{A} C \beta - C^\top \hat{A} x^i(T_*)) + \lambda 1, \\
0 &= 1^\top (\beta - x^1(T_*)),
\end{align*}
\]

are all the solutions of (27). This gives us

\[
\hat{\beta}_{\text{mle}}^* \in \{ \beta \in \mathbb{R}^n : \text{E}_{\text{mle}}^* \left[ \beta \right] = b_{\text{mle}}^*, \, \lambda \in \mathbb{R} \},
\]

where

\[
\text{E}_{\text{mle}}^* = \begin{bmatrix} mC^\top \hat{A} C & 1 \\ 1^\top & 0 \end{bmatrix}, \quad b_{\text{mle}}^* = \begin{bmatrix} C^\top \hat{A} \sum_{i=1}^{m} x^i(T_*) \\ 1^\top x^1(T_*) \end{bmatrix}.
\]

Clearly \( \text{E}_{\text{mle}}^* \) is nonsingular. The analysis above shows that \( \hat{\beta}_{\text{mle}}, \hat{\beta}_{\text{mle}}^* \) are both nonunique for Architectural Eavesdroppers, which is consistent with Proposition 4.

### 3.2.3 MAP Estimation

For MAP Estimation, we let the eavesdroppers have a normal distribution assumption of the initial state for the simplicity of our analysis, which is also a good approximation of distributions in practical conditions. Assume \( \beta \sim N(\mu_\beta, \Lambda_\beta) \) with \( \mu_\beta \in \mathbb{R}^n \) and \( \Lambda_\beta \) being an \( n \)-by-\( n \) positive definite matrix. Let \( \hat{\beta}_{\text{map}}, \hat{\beta}_{\text{map}}^* \) be the max a posteriori (MAP) estimators of \( \beta \) in \( P \) and \( P^* \), respectively. According to Bayes’ rule

\[
f(\beta | x(1), \ldots, x(T_*)) = \frac{f(\beta)(f(x(1), \ldots, x(T_*) \mid \beta))}{f(x(1), \ldots, x(T_*)^1)}. \tag{28}
\]

Recall the assumption \( \beta \sim N(\mu_\beta, \Lambda_\beta) \). Then \cite{15} and \cite{28} yield

\[
\begin{align*}
f(\beta \mid x(1), \ldots, x(T_*)) & \propto f(\beta)f(x(1), \ldots, x(T_*) \mid \beta) \\
& \propto f(\beta)f(x(1) \mid \beta) \\
& \propto \exp\left(-\frac{1}{2}(\beta - \mu_\beta)^\top \Lambda_\beta^{-1}(\beta - \mu_\beta)\right) \exp\left(-\frac{1}{2\sigma^2}(|v_1|^\top \beta - x_{\text{Head}(E_1)}(1))^2\right). \\
& \delta(x_{\text{Tail}(E_1)}(1) + x_{\text{Head}(E_1)}(1) - |v_1|^\top \beta) \prod_{i \neq \text{Tail}(E_1),\text{Head}(E_1)} \delta(x_i(1) - \beta_i). \tag{29}
\end{align*}
\]

Similar to \cite{19} and based on \cite{29}, we know the optimization problem

\[
\arg\max_{\beta} f(\beta \mid x(1), \ldots, x(T_*))
\]

is equivalent to the following constrained optimization problem

\[
\begin{align*}
\arg\min_{\beta} & \quad (\beta - \mu_\beta)^\top \Lambda_\beta^{-1}(\beta - \mu_\beta) \\
\text{s.t.} & \quad A_1^\top A_1 \beta = A_1^\top A_1 x^1(1). \tag{30}
\end{align*}
\]
The Slater’s condition clearly holds for (30) and the KKT conditions imply that the MAP estimator \( \hat{\beta}_{\text{map}} \) of \( \beta \) in \( \mathcal{P} \) is the the \( \beta \)-part of the solution of

\[
\mathbf{E}_{\text{map}} \begin{bmatrix} \beta \\ \lambda \end{bmatrix} = \mathbf{b}_{\text{map}}, \quad \beta \in \mathbb{R}^n, \quad \lambda \in \mathbb{R}^n,
\]

where

\[
\mathbf{E}_{\text{map}} = \begin{bmatrix} \mathbf{I} & \lambda \beta \mathbf{A}_1^\top \mathbf{A}_1 \\ \mathbf{A}_1^\top \mathbf{A}_1 & 0 \end{bmatrix}, \quad \mathbf{b}_{\text{map}} = \begin{bmatrix} \mu \beta \\ \mathbf{A}_1^\top \mathbf{A}_1 \mathbf{x}(1) \end{bmatrix}.
\]

It can be easily verified that \( \mathbf{E}_{\text{map}} \mathbf{z} = 0 \) implies \( \mathbf{z} = 0 \). Thus \( \hat{\beta}_{\text{map}} \) is unique. Analogously, the unique MAP estimator of \( \beta \) in \( \mathcal{P}^* \) can be obtained by solving the following constrained optimization problem:

\[
\arg\min_{\beta} \sum_{i=1}^m (x^i(T_s) - \mathbf{C} \beta)^\top \mathbf{A}_1^\top (x^i(T_s) - \mathbf{C} \beta) + (\beta - \mu \beta)^\top \Lambda_{\beta}^{-1} (\beta - \mu \beta)
\]

\[
s.t. \quad 1^\top \beta = 1^\top \mathbf{x}(T_s).
\]

As a result, the MAP estimator \( \hat{\beta}_{\text{map}}^* \) of \( \beta \) in \( \mathcal{P}^* \) is the \( \beta \)-part of the solution to

\[
\mathbf{E}_{\text{map}}^* \begin{bmatrix} \beta \\ \lambda \end{bmatrix} = \mathbf{b}_{\text{map}}^*, \quad \beta \in \mathbb{R}^n, \quad \lambda \in \mathbb{R},
\]

where

\[
\mathbf{E}_{\text{map}}^* = \begin{bmatrix} m \mathbf{C}^\top \Lambda \mathbf{C} + \Lambda_{\beta}^{-1} & 1 \\ 1^\top & 0 \end{bmatrix}, \quad \mathbf{b}_{\text{map}}^* = \begin{bmatrix} \mathbf{C}^\top \Lambda \sum_{i=1}^m x^i(T_s) + \Lambda_{\beta}^{-1} \mu \beta \\ 1^\top \mathbf{x}(T_s) \end{bmatrix}.
\]

Clearly \( \mathbf{E}_{\text{map}}^* \) is nonsingular, leading to the uniqueness of \( \hat{\beta}_{\text{map}}^* \).

### 3.3 Topological Eavesdroppers

Clearly the knowledge of Topological Eavesdroppers is strictly less than the Architectural Eavesdroppers, and thus the estimators given by Topological Eavesdroppers would be still nonunique and no better in terms of accuracy. Under the algorithm [8], it is evident that only two nodes’ states are updated at each time, while the other states remain unchanged. Thus for Topological Eavesdroppers, with the observations \( \mathbf{x}(1), \ldots, \mathbf{x}(T_s) \) in the model \( \mathcal{P} \), an undirected edge sequence of length \( T_s - 1 \) can be given by \{\( u_2, v_2 \), \ldots, \( u_{T_s}, v_{T_s} \)\}, where \( u_t, v_t \) are the indices of two nonzero components of \( \mathbf{x}(t) - \mathbf{x}(t-1) \) for \( t = 2, \ldots, T_s \). This undirected edge sequence with certain orientations is clearly \( \mathcal{E}_2, \ldots, \mathcal{E}_{T_s} \). However, Topological Eavesdroppers can only restrict the choice of \( \{u_1, v_1\} \) to the set \( \mathcal{E} \setminus \bigcup_{t=2}^m \{u_t, v_t\} \). These inference actions on the edge sequence evidently gains the knowledge of Topological Eavesdroppers.

In the model \( \mathcal{P}^* \), however, such actions cannot be taken because no state information is given except for \( \mathbf{x}(T_s) \). Therefore, \( T_G^o \) becomes another parameter that affects the outcome of \( \mathbf{x}(T_s) \) in the model. In order to obtain the estimators in \( \mathcal{P}^* \), Topological Eavesdroppers must assume some prior probability assumption of \( T_G^o \), and then the likelihood becomes

\[
f(\mathbf{x}(T_s) \mid \beta) = \sum_{G^o \in S(T_G^o)} f(\mathbf{x}(T_s) \mid \beta, T_G^o = G^o) \mathbb{P}(T_G^o = G^o), \quad (31)
\]
where \( S(T^o_G) \) denotes the set of all possible choices of \( T^o_G \). It is worth noting that the likelihood (31) is a linear combination of a group of exponential functions of \( \beta \), maximizing which does not yield an explicit optimal solution as the MLE. With certain prior probability assumptions of \( T^o_G \), not only the MAP estimator, but the MLE can be unique.

3.4 Numerical Examples

Two examples are provided in this section to make comparison between the MLEs and the MAP estimators given by different Eavesdroppers. In these examples, we consider the graph \( G \) in Figure 1 and select the oriented spanning tree \( T^o_G = (V, E^o_T) \) in Figure 2. Let the edges in \( E^o_T \) be sorted as

\[(5, 2) \prec (2, 3) \prec (2, 1) \prec (3, 4).\]

Suppose \( \gamma \sim \text{N}(0, 1) \), \( t = 1, \ldots, T_\star \). The evolution of all node states along (8) starting with \( x_i(0) = \beta_i \), \( i = 1, \ldots, 5 \) can be shown by the following table.

| Time | \( t = 0 \) | \( t = 1 \) | \( t = 2 \) | \( t = 3 \) | \( t = 4 \) |
|------|-------------|-------------|-------------|-------------|-------------|
| \( x_1(t) \) | \( \beta_1 \) | 3.2 | 3.2 | 2.5 | 2.5 |
| \( x_2(t) \) | \( \beta_2 \) | 1.2 | 1 | 1.7 | 1.7 |
| \( x_3(t) \) | \( \beta_3 \) | -1 | -0.8 | -0.8 | 0.3 |
| \( x_4(t) \) | \( \beta_4 \) | 1.5 | 1.5 | 1.5 | 0.4 |
| \( x_5(t) \) | \( \beta_5 \) | 0.8 | 0.8 | 0.8 | 0.8 |

It is worth mentioning that the true initial state \( \beta \) can be any vector in

\[ S \triangleq \{ [3.2 \ \beta_2 - 1.5 \ \beta_3]^\top : \beta_2 + \beta_5 = 2 \}. \]

Example 3. (Architectural Eavesdropper) Based on the knowledge of \( T^o_G \), an Architectural Eavesdropper obtains

\[
A_1 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}, \quad v_1 = \begin{bmatrix}
0 \\
-1 \\
0 \\
0 \\
1
\end{bmatrix},
\]

\[
C = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}, \quad D = \begin{bmatrix}
0 & 1 & -1 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 \\
-1 & -1 & 0 & -1 \\
1 & 0 & 0 & 0
\end{bmatrix}.
\]
By our definition, this eavesdropper observes $x(1), x(2), x(3)$ in the model $P$, and $x(3)$ in the model $P^*$. This eavesdropper can compute the MLEs of $\beta$ in $P$ and $P^*$ as

$$\hat{\beta}_{\text{mle}} \in \{[3.2 \beta_2 - 1 1.5 \beta_5]^\top : \beta_2 + \beta_5 = 2\},$$

$$\hat{\beta}_{\text{mle}}^* \in \{[4.2 \beta_2 \ldots \beta_5]^\top : \beta_2 + \ldots + \beta_5 = 1.5\}.$$ 

In $P$, with the knowledge of the edge $E_1$ and the node states $x(1)$, the eavesdropper assumes the prior distribution $N([3.2 1 - 1 1.5 1]^\top, I)$. In $P^*$, the eavesdropper can exactly recover the sum of node states $5.7$, and thus has the prior distribution assumption $N([1.14 \ldots 1.14]^\top, I)$, the MAP estimators of $\beta$ in $P$ and $P^*$ are given by

$$\hat{\beta}_{\text{map}} = [3.2 1 - 1 1.5 1]^\top,$$

$$\hat{\beta}_{\text{map}}^* = [2.5 0.8 0.8 0.8 0.8]^\top.$$ 

As can be seen from the computed estimators, the MLEs are nonunique while the MAP estimators are unique, which is consistent with the theoretical analysis.

**Example 4.** (Topological Eavesdropper) In the model $P$, by calculating $x(t) - x(t-1)$ for $t = 2, 3, 4$ and based on $E_T$, the eavesdropper infers two undirected edge sequences $\{5, 2\}, \{2, 3\}, \{2, 1\}, \{3, 4\}$ and $\{5, 4\}, \{2, 3\}, \{2, 1\}, \{3, 4\}$. Thus $T_G$ has two possible values. For each possible $T_G$, the uncertain edge orientations gives $2^4$ random choices of $T_G^\circ$. Let the eavesdroppers assume that all possible entries of $T_G^\circ$ have equal probabilities. Then the eavesdropper gives the MLE and the MAP estimator of $\beta$ with an arbitrary prior distribution assumption of $\beta$ in $P$ as

$$\hat{\beta}_{\text{mle}} = \hat{\beta}_{\text{map}} = [3.2 1.2 - 1 1.5 0.8]^\top.$$ 

In the model $P^*$, $T_G$ has 4 possible entries and $T_G^\circ$ has $4! \cdot 2^4$ possible values for each $T_G$. The eavesdropper assumes all possible values of $T_G^\circ$ for $T_G$ have equal chances to appear. Then the eavesdropper can obtain the MLE and the MAP estimator of $\beta$ with the prior assumption $N([1.14 \ldots 1.14]^\top, I)$ in $P^*$ as

$$\hat{\beta}_{\text{mle}} = [3.80 - 1.10 1.17 0.36 1.47]^\top,$$

$$\hat{\beta}_{\text{map}} = [1.27 1.21 1.12 0.98 1.12]^\top.$$ 

It is worth noting that $\hat{\beta}_{\text{mle}}^*$ is now unique because the eavesdropper has a prior probability assumption of $T_G^\circ$.

### 4 Randomization and Resilience/Privacy Preservation Trade-off

#### 4.1 Randomized PPSC Gossiping

In this section, we propose a randomized PPSC algorithm based on classical random gossiping [18]. To this end, let $P$ be a stochastic matrix [32], i.e., a matrix with non-negative entries possessing a sum one along each row. The matrix $P$ complies with the structure of the graph $G$ in the sense that $[P]_{ij} > 0$ if and only if $\{i, j\} \in E$. Compared to deterministic gossiping, randomized gossip algorithms are ideal solutions
for distributed systems, where nodes are self-organized with asynchronous clocks \[18\]. Independently at each time step, a node \(i\) is selected with probability \(\frac{1}{n}\), and then this node \(i\) randomly selects a neighbor \(j \in N_i\) with probability \([P]_{ij}\). This defines a standard random gossip algorithm \[18\].

When node \(i\) is selected to meet with its neighbor \(j\) at time \(t\), node \(i\) randomly generates \(\gamma_t \in \mathbb{R}\) and sends \(x_i(t - 1) - \gamma_t\) to \(j\). Then the nodes \(i, j\) update their states by

\[
x_i(t) = \gamma_t \\
x_j(t) = x_i(t - 1) + x_j(t - 1) - \gamma_t,
\]

where \(\gamma_t, t = 1, 2, \ldots\) are i.i.d. The states remain unchanged for the rest of the nodes in \(V\), i.e.,

\[
x_k(t) = x_k(t - 1), \ k \in V \setminus \{i, j\}.
\]

### 4.1.1 Convergence Limits

We let \(q_1\) denote the unique left Perron vector of \(P\) with \(1^\top q_1 = 1\). Let \((q_2, q'_2), \ldots, (q_n, q'_n)\) denote the left-right generalized eigenvector pairs of \(P\) satisfying \(q_i^\top q'_i = 1\) for all \(i = 2, \ldots, n\).

**Proposition 1.** Let \(\mu_\gamma\) denote the expected value of \(\gamma_t\). Then along the randomized PPSC gossip algorithm, there holds

\[
\lim_{t \to \infty} E(x(t)) = q_1 1^\top x(0) + \mu_\gamma \sum_{i=2}^{n} q_i q'_i 1.
\]

**Proof.** In a compact form, the update of network node states can be written as

\[
x(t) = A(t)x(t - 1) + \gamma_t v(t), \ t = 1, 2, \ldots,
\]

where \(A(t)\) is a random matrix and \(v(t)\) is a random vector. By the structure of the proposed algorithm, one easily know

\[
E(A(t)) = I + \frac{1}{n}(P^\top - I), \tag{33}
\]

\[
E(v(t)) = \frac{1}{n}(I - P^\top)1. \tag{34}
\]

From \(33\), it is worth noting that \(E(A(t))\) is a primitive stochastic matrix with left Perron vector being \(q_1\). Hence \(32\)

\[
\lim_{t \to \infty} (E(A(t)))^t = q_1 1^\top. \tag{35}
\]

By performing Jordan decomposition on \(P^\top\), one can easily obtain from \(33\) and \(34\)

\[
\lim_{t \to \infty} \sum_{i=0}^{t-1} (E(A(t)))^i E(v(t)) = \sum_{i=2}^{n} q_i q'_i 1. \tag{36}
\]

Then the desired conclusion can be obtained noticing the independence of the node updates. \(\square\)
4.1.2 Convergence Rate

Introduce $\mathcal{H}_t$ as the event that all nodes have altered their states at least once during the time $s \in [0, t]$. Then $\mathcal{H}_t$ holding true implies that the entire network states have been encrypted by the randomized PPSC algorithm. Define

$$\xi = \frac{1}{n}(P + P^T)1.$$  

Let $\xi_i$ denote the $i$-th component of $\xi$. Let $2^S$ denote the power set of a set $S$. Recall that an independent set of a graph is a subset of the graph vertex set, in which two arbitrary nodes are not adjacent in the graph $^{25}$. Based on this, a result regarding the convergence rate is shown in the following.

**Proposition 2.** Consider an undirected and connected graph $G = (V, E)$. Let the node set $V$ be partitioned into $\kappa > 1$ mutually disjoint independent vertex sets $W_1, \ldots, W_\kappa$, which satisfy $\bigcup_{i=1}^\kappa W_i = V$. Let $\pi_1, \ldots, \pi_\kappa$ denote some arbitrary elements in $W_1, \ldots, W_\kappa$, respectively. Then there holds

$$\mathbb{P}(\mathcal{H}_t) \geq 1 - \kappa + \sum_{i=1}^\kappa \sum_{U \subseteq 2^{W_i \setminus \{\pi_i\}}} (-1)^{|U|}(1 - \sum_{j \in U} \xi_j)^t - (1 - \xi_{\pi_i} - \sum_{j \in U} \xi_j)^t).$$

**Proof.** Now we define $n$ event sequences

$$\{\mathcal{F}^t_i\}_{t=1,2,\ldots}, \{\mathcal{F}^t_j\}_{t=1,2,\ldots},$$

where $\mathcal{F}^t_i$ represents the event that the state of node $i$ at time $t$ is unequal to its initial state. Clearly $\mathcal{H}_t = \bigcap_{i=1}^\kappa \mathcal{F}^t_i$, $t = 1, 2, \ldots$. We have by the Fréchet inequalities

$$\mathbb{P}(\mathcal{H}_t) \geq \sum_{i=1}^\kappa \mathbb{P}\left( \bigcap_{j \in W_i} \mathcal{F}^t_j \right) - (\kappa - 1).$$

(37)

For $i = 1, \ldots, \kappa$, it is clear by the Inclusion–exclusion theorem $^{35}$

$$\mathbb{P}\left( \bigcap_{j \in W_i} \mathcal{F}^t_j \right) = \sum_{U \subseteq 2^{W_i \setminus \{\pi_i\}}} (-1)^{|U|}\mathbb{P}\left( \left( \bigcap_{j \in U} \mathcal{F}^t_j \right) \cap \mathcal{F}_{\pi_i}^t \right) = \sum_{U \subseteq 2^{W_i \setminus \{\pi_i\}}} (-1)^{|U|}\mathbb{P}\left( \bigcap_{j \in U} \mathcal{F}^t_j \right) \cdot \mathbb{P}\left( \mathcal{F}_{\pi_i}^t | \bigcap_{j \in U} \mathcal{F}^t_j \right).$$

(38)

It can be noted from the definition of $\xi$ that $\xi_i \in (0, 1]$ with $i = 1, \ldots, n$ represents the probability of the event that node-to-node communication involves node $i$ in a time slot. For each $U \subseteq 2^{W_i \setminus \{\pi_i\}}$, it is known that any two nodes in $U$ are not adjacent, and thus

$$\mathbb{P}\left( \bigcap_{j \in U} \mathcal{F}^t_j \right) = (1 - \sum_{j \in U} \xi_j)^t.$$  

(39)

Similarly, there holds

$$\mathbb{P}\left( \left( \bigcap_{j \in U} \mathcal{F}^t_j \right) \cap \mathcal{F}_{\pi_i}^t \right) = (1 - \xi_{\pi_i} - \sum_{j \in U} \xi_j)^t.$$  

(40)

Then (39) and (40) yield

$$\mathbb{P}\left( \mathcal{F}_{\pi_i}^t | \bigcap_{j \in U} \mathcal{F}^t_j \right) = 1 - \mathbb{P}\left( \mathcal{F}_{\pi_i}^t | \bigcap_{j \in U} \mathcal{F}^t_j \right) = 1 - \frac{\mathbb{P}\left( \left( \bigcap_{j \in U} \mathcal{F}^t_j \right) \cap \mathcal{F}_{\pi_i}^t \right)}{\mathbb{P}\left( \bigcap_{j \in U} \mathcal{F}^t_j \right)} = 1 - \frac{1 - (1 - \sum_{j \in U} \xi_j)^t}{1 - \sum_{j \in U} \xi_j^t}. $$

(41)

The proof is completed by (37), (38), (39) and (41).  

□
We can also quantify the rate of convergence by the following $\epsilon$-encryption time.

**Definition 4.** For any $\epsilon \in (0, 1)$, the $\epsilon$-encryption time for an undirected and connected graph $G = (V, E)$ with $n$ nodes and a randomized PPSC gossiping associated with the edge selection probability given in $P \in \mathbb{R}^{n \times n}$ is defined by

$$T_\epsilon(G, P) = \inf \{ t : 1 - P(\mathcal{H}_t) \leq \epsilon \}.$$  

Define $\xi_m = \min_i \xi_i$. For the $\epsilon$-encryption time, we have the following proposition.

**Proposition 3.** For any $\epsilon \in (0, 1)$, the $\epsilon$-encryption time associated with graph $G$ and matrix $P$ for the random PPSC algorithm satisfies

$$\log \frac{\epsilon}{\log(1 - \xi_m)} \leq T_\epsilon(G, P) \leq \log \frac{\epsilon - \log n}{\log(1 - \xi_m)}.$$  

**Proof.** Let $F^t_i$, $i = 1, \ldots, n$, $t = 1, 2, \ldots$ be as defined in the proof of Proposition 2. Then it can be concluded

$$P(F^t_i) = 1 - P(\overline{F^t_i}) = 1 - (1 - \xi_i)^t.$$  

(42)

By (42) and the Fréchet inequalities [36], we have

$$P(\mathcal{H}_t) \geq \sum_{i=1}^n P(F^t_i) - (n - 1) = 1 - \sum_{i=1}^n (1 - \xi_i)^t \geq 1 - n(1 - \xi_m)^t.$$  

(43)

By the Fréchet inequalities [36], one also has

$$P(\mathcal{H}_t) \leq \min_i \{P(F^t_i)\} = 1 - (1 - \xi_m)^t.$$  

(44)

Clearly (43) and (44) complete the proof. \hfill $\square$

We would like to point out that techniques from optimizing the structure of the work and the selection of the neighbors [20,37,38] might significantly accelerate the convergence rate of the algorithm.

**4.2 Resilience vs Privacy Preserving Trade-off**

Throughout the running of the algorithm [8] with deterministic or randomized edge selection over a network, a circumstance may occur that a node chooses to drop out of the network autonomously at a random time. Let us assume that independently at each time step, each node of the network has a probability $p_d > 0$ of dropping out. We focus on a particular time instance $t_d > 0$. Let $G_t = (V_t, E_t)$ denote the random network at time $t = 0, 1, 2, \ldots$. It is clear that

$$V_t \subset V_{t-1},$$

$$E_t = E_{t-1} \setminus \left\{ \{i, j\} : i \in V_{t-1} \setminus V_t, j \in V_{t-1} \right\}.$$
On one hand, from the point of view of Tail($\mathcal{E}_{td}$), nodes in $V_{td}$ should have their states sum only yielding a small change compared to that at time $t_{td} - 1$ due to the node dropout, i.e.,

$$\mathbb{E}\left| \sum_{i \in V_{td}} x_i(t_{td}) - \sum_{i \in V_{td}} x_i(t_{td} - 1) \right|^2$$

is preferred to be as small as possible, where $\mathbb{E}(\cdot)$ is with respect to the randomness from node dropouts. By direct calculation we can see that

$$\mathbb{E}\left| \sum_{i \in V_{td}} x_i(t_{td}) - \sum_{i \in V_{td}} x_i(t_{td} - 1) \right|^2 = 2p_d(1 - p_d)\mathbb{E}\left| x_{\text{Tail}}(\mathcal{E}_{td})(t_{td} - 1) - \gamma_{td} \right|^2.$$  

It worth emphasizing that $\omega_{td} = x_{\text{Tail}}(\mathcal{E}_{td})(t_{td} - 1) - \gamma_{td}$ is also the packet of communication during the node pair interaction at time $t_{td}$, which will be the error added into the network sum if the communication fails at the receiving node Head($\mathcal{E}_{td}$). Therefore,

$$\mathcal{R}_R = 2p_d(1 - p_d)\mathbb{E}(|x_{\text{Tail}}(\mathcal{E}_{td})(t_{td} - 1) - \gamma_{td}|^2)$$

serves as a natural network resilience metric. On the other hand, by receiving the packet $\omega_{td}$, Head($\mathcal{E}_{td}$) or a third party can possibly recover the state $x_{\text{Tail}}(\mathcal{E}_{td})(t_{td} - 1)$. In that case, Tail($\mathcal{E}_{td}$) would hope

$$h(x_{\text{Tail}}(\mathcal{E}_{td})(t_{td} - 1) \mid \omega_{td})$$

i.e., the entropy of $x_{\text{Tail}}(\mathcal{E}_{td})(t_{td} - 1)$ given $\omega_{td}$, to be as large as possible. As a result,

$$\mathcal{P}_P = h(x_{\text{Tail}}(\mathcal{E}_{td})(t - 1) \mid \omega_{t})$$

can be a good privacy preservation metric for any time $t$.

With normal distribution assumptions on both the $\beta_i$s and the $\gamma_i$s, $x_{\text{Tail}}(\mathcal{E}_{td})(t - 1)$ is normally distributed with its mean and the variance denoted as $\bar{\mu}$ and $\bar{\sigma}^2$, respectively. We can now conclude that

$$\mathcal{R}_R = 2p_d(1 - p_d)((\bar{\mu} - \bar{\sigma} \gamma)^2 + \bar{\sigma}^2 + \bar{\sigma}^2),$$

$$\mathcal{P}_P = \frac{1}{2} \log 2 \pi e - \frac{1}{2} \log \left( \frac{1}{\bar{\sigma}^2} + \frac{1}{\sigma^2} \right).$$

Thus, a tradeoff between network resilience and privacy preservation can be characterized by

$$\arg\min_{\bar{\sigma}, \sigma^2} \mathcal{R}_R - \nu \mathcal{P}_P,$$

where $\nu \in \mathbb{R}^+$ is a parameter that weights the importance of the resilience and the privacy preservation capability. With (47) and (48), (49) yields a unique solution

$$\bar{\sigma} = \bar{\mu},$$

$$\sigma^2 = \frac{2}{\nu} \sqrt{\bar{\sigma}^2 + \frac{\nu}{p_d(1 - p_d)} - \bar{\sigma}}.$$

The relation (50) provides an inspiration on how we can generalize the algorithm (8) to the adaptively generated noise sequence $\gamma_t$. Letting the random variable $\gamma_t$ have a state-dependent mean and variance related to the state of the node that generates it, one can achieve a degree of a balance between resilience and privacy preserving.
5 Conclusions

In this paper, we adapted the idea of the gossip protocol and proposed privacy-preserving algorithms with consistent summation of node states. We established necessary and sufficient conditions on the dependence of two arbitrary nodes’ final states, and characterized the their pairwise dependence with stochastic graphical models. We classified possible categories of network privacy eavesdroppers from their knowledge about the network structure and information flow. It was shown that even the eavesdroppers with full knowledge are unable to reconstruct the network initial values from a non-identifiability theorem, and thus the proposed algorithms can be used to protect each node’s private algebraic equations and cost functions in distributed computation and optimization. The strategies of the eavesdroppers for estimating the node initial values by MLE or MAP estimators were also discussed. Some simple but illustrative examples were presented as well. In addition, we proposed a privacy-preserving algorithm with randomized edge selection, whose convergence limit and convergence rate for full-network encryption were established. As an extension, the trade-off between resilience and privacy preservation was studied. Future work includes the design of optimal network structure for information preservation, and study of the fundamental limits between privacy preservation and computation efficiency in distributed algorithms.

Appendicies

Appendix A. Proof of Lemma [1]

The conclusion $d_{i_0,s}(t^*) = d_{i_0,s}(t^*) \neq 0$ describes that the random component $d_{i_0,s}(t^*) \gamma_s$ is passed from node $i_0$ to node $i_l$ during the time interval $[t^*, t^*]$. Before presenting the proof, we provide some intuitive explanation on the three conditions. Condition (i) confirms the orientations of edges in the path $i_0, \ldots, i_l$, while the order of the edge selection is given in Condition (ii). Condition (iii) prevents the random component $d_{i_0,s}(t^*) \gamma_s$ from being passed to the nodes that are not in the path $i_0, \ldots, i_l$. Next the sufficiency and necessity of the conditions are proved, respectively.

**Sufficiency.** Let $\mathcal{E}_{ik} = (i_k, i_{k+1})$ for $k = 0, 1, \ldots, l - 1$. Then Condition (ii) is equivalent to

$$t_s < t_0 < t_1 < \cdots < t_{l-1} \leq t^*.$$

$d_{i_0,s}(t_s) \neq 0$ states that the random variable $d_{i_0,s}(t_s) \gamma_s$ appears in the node state $x_i(t_s)$. Condition (iii) a) describes that $i_0$ is not the tail of any edges $\mathcal{E}_t$ with $t_s \leq t < t_0$, and hence guarantees that the random components in $x_{i_0}(t_s)$, including $d_{i_0,s}(t_s) \gamma_s$, will be kept in $x_{i_0}(t)$ for all $t_s \leq t < t_0$. At time $t_0$, node $i_0$ and node $i_1$ are chosen to mutually communicate and get their states updated according to the algorithm (8). As a result, $x_{i_0}(t_0)$ loses the random component $d_{i_0,s}(t_s) \gamma_s$, which now becomes a component of $x_{i_1}(t_1)$. The similar analysis applied at time $t_1, \ldots, t_{l-1}$ recursively shows the random component $d_{i_0,s}(t_s) \gamma_s$ appears in $x_{i_l}(t_{l-1})$ under Condition (i), (ii) and (iii) b). Finally, Condition (iii) c) tells that random component $d_{i_0,s}(t_s) \gamma_s$ will be kept in $x_{i_l}(t)$ for $t_{l-1} \leq t \leq t^*$, which completes the proof of sufficiency.

**Necessity.** In the following, we prove the necessity of the three conditions. Assume the random component
\(d_{i_0s}(t^*)\gamma_s\) is passed from node \(i_0\) to node \(i_l\) during the time interval \([t_s, t^*]\). Recall that the transition of the random variable \(d_{i_0s}(t^*)\gamma_s\) occurs from the tail of an edge to the head at each time step. Thus one has \((i_0, i_1), \ldots, (i_{l-1}, i_l) \in E_T\) due to the uniqueness of the undirected path \(i_0, i_1, \ldots, i_l\) in \(T_G\), and completes the proof of the necessity of Condition (i).

According to the orientations of \((i_k, i_{k+1})\), there exist time \(\tau_k\) when \(d_{i_0s}(t^*)\gamma_s\) appears in the node state \(x_{i_k}(\tau_k)\) for \(k = 0, 1, \ldots, l\). By the uniqueness of the directed path from \(i_0\) to \(i_l\), \(\tau_{ks}\) can be arranged in the following order
\[
\tau_0 < \tau_1 < \cdots < \tau_l.
\]

The definition of \(\mathcal{E}_{\tau_k}\) directly implies that for \(k = 0, 1, \ldots, l - 1\)
\[
\tau_k < t_k \leq \tau_{k+1}.
\]

It is clear \(51\) and \(52\) yield
\[
t_0 < t_1 < \cdots < t_{l-1}.
\]

It is evident that if \(\mathcal{E}_{t_0} \leq \mathcal{E}_{t_1}\), the random component \(d_{i_0s}(t^*)\gamma_s\) can never appear in the state of node \(i_1\), and thereby the states of \(i_2, \ldots, i_l\). Thus one has \(\mathcal{E}_{t_1} \prec \mathcal{E}_{t_0}\). In addition, node \(i_l\) cannot gain the random component \(d_{i_0s}(t^*)\gamma_s\) when the time \(t < t_{l-1}\), which implies \(\mathcal{E}_{t_{l-1}} \leq \mathcal{E}_{t^*}\). Then the necessity of Condition (ii) can be seen from \(53\), \(\mathcal{E}_{t_l} \prec \mathcal{E}_{t_0}\) and \(\mathcal{E}_{t_{l-1}} \leq \mathcal{E}_{t^*}\).

Suppose Condition (iii) a) does not hold for contradiction. Then there exists a node set \(\Gamma^* \subset V \setminus \{i_0, \ldots, i_l\}\) such that \((i_0, i^*) \in T_G^*\) and \((i_0, i^*) \prec \mathcal{E}_{t_0}\) for all \(i^* \in \Gamma^*\). As a result, the random component \(d_{i_0s}(t^*)\gamma_s\) will be passed to some node in \(\Gamma^*\), which is impossible to be passed to \(i_1\) again. Thus Condition (iii) a) must hold. Similarly, due to the uniqueness of the path between two arbitrary nodes in spanning trees, the random component \(d_{i_0s}(t^*)\gamma_s\) must be always held along path \(i_0, i_1, \ldots, i_l\) during time \(t_s \leq t \leq t^*\), which proves the necessity of Condition (iii).

**Appendix B. Proof of Lemma 2**

(i) For the endpoints of an arbitrary edge \(\mathcal{E}_s\) with \(s \in \{1, 2, \ldots, T_s\}\), the algorithm \(8\) yields at time \(\hat{t}\)
\[
\begin{align*}
\text{x}_{\text{Tail}(\mathcal{E}_s)}(\hat{t}) &= \gamma_s \\
\text{x}_{\text{Head}(\mathcal{E}_s)}(\hat{t}) &= \text{x}_{\text{Tail}(\mathcal{E}_s)}(s - 1) + \text{x}_{\text{Head}(\mathcal{E}_s)}(s - 1) - \gamma_s.
\end{align*}
\]

From \(54\), we see that a pair of random variables \(\gamma_s, -\gamma_s\) are added to the states of \(\text{Tail}(\mathcal{E}_s)\) and \(\text{Head}(\mathcal{E}_s)\) respectively at time \(s\). Next we analyze the random variable \(\gamma_s\) in \(\text{x}_{\text{Tail}(\mathcal{E}_s)}(t)\) for \(t > s\). If \(\text{Tail}(\mathcal{E}_s) \neq \text{Tail}(\mathcal{E}_t)\) and \(\text{Tail}(\mathcal{E}_s) \neq \text{Head}(\mathcal{E}_t)\) for all \(t > s\), then it is clear that \(\gamma_s\) remains in \(\text{x}_{\text{Tail}(\mathcal{E}_s)}(t)\) for all \(t > s\). For any time \(t > s\) with \(\text{Tail}(\mathcal{E}_s) = \text{Head}(\mathcal{E}_t)\), the random variables held by \(\text{x}_{\text{Tail}(\mathcal{E}_s)}(t - 1)\) are still kept in \(\text{x}_{\text{Tail}(\mathcal{E}_s)}(t)\). Therefore, the only way that the state of node \(\text{x}_{\text{Tail}(\mathcal{E}_s)}\) loses \(\gamma_s\) is to let node \(\text{Tail}(\mathcal{E}_s)\) be the tail of edge \(\mathcal{E}_t\) for some \(t > s\). Suppose there exists a nonempty set \(\mathcal{T}\) such that \(\text{Tail}(\mathcal{E}_s) = \text{Tail}(\mathcal{E}_t)\) for all \(t \in \mathcal{T}\). Define \(\bar{t} = \min\{t : t \in \mathcal{T}\}\). Then by the algorithm \(8\)
\[
\begin{align*}
\text{x}_{\text{Tail}(\mathcal{E}_s)}(\bar{t}) &= \gamma_{\bar{t}} \\
\text{x}_{\text{Head}(\mathcal{E}_s)}(\bar{t}) &= \text{x}_{\text{Tail}(\mathcal{E}_s)}(\bar{t} - 1) + \text{x}_{\text{Head}(\mathcal{E}_t)}(\bar{t} - 1) - \gamma_{\bar{t}}.
\end{align*}
\]
from which we see that the random variable $\gamma_s$ transfers to $x_{\text{Head}(E_s)}(\bar{t})$ without changing its sign. It can be concluded by applying the same analysis that $\gamma_s$ exists in one and only one of all node states for all $t \geq s$. Analogously, we can easily know that $-\gamma_t$ has the same properties. This completes the proof of (i).

(ii) Since $x_i(T_s)$ and $x_j(T_s)$ are dependent, there exists a set $S \subset \{1, 2, \ldots, T_s\}$ such that $d_{is}(T_s)\gamma_s$ appears in $x_i(T_s)$ and $d_{js}(T_s)\gamma$ appears in $x_j(T_s)$ for all $s \in S$. Let $i_0, i_1, \ldots, i_l$ denote the unique undirected path in $T_G$ that connects node $i$ and node $j$ with $i_0 = i$ and $i_l = j$. Thus we only need to prove $|S| = 1$. For every $s \in S$, under the algorithm (8), $\gamma_s$ is held by $x_{\text{Tail}(E_s)}(s)$ and $-\gamma_s$ is held by $x_{\text{Head}(E_s)}(s)$. Without loss of generality, we assume $x_{i_0}(T_s)$ holds $\gamma_s$ and $x_{i_1}(T_s)$ holds $-\gamma_s$. If $s = T_s$, then it is necessary $\text{Tail}(E_s) = i_0$ and $\text{Head}(E_s) = i_l$. If $s \neq T_s$, at most one of $\text{Tail}(E_s) = i_0$ and $\text{Head}(E_s) = i_l$ holds. When $\text{Tail}(E_s) = i_0$ and $\text{Head}(E_s) \neq i_l$, $-\gamma_s$ has to transfer from to $x_{\text{Head}(E_s)}(s)$ from $x_{i_1}(T_s)$. By Lemma 1, the process of transfer requires the path that starts from $\text{Head}(E_s)$ and ends at $i_l$ is a directed path. In this case, $i_0, i_1, \ldots, i_l$ is a directed path and $E_s = (i_0, i_1)$. Similarly, $i_l, i_{l-1}, \ldots, i_0$ is a directed path and $E_s = (i_l, i_{l-1})$ when $\text{Tail}(E_s) \neq i_0$ and $\text{Head}(E_s) = i_l$. In a general case in which $\text{Tail}(E_s) \neq i_0, \text{Head}(E_s) \neq i_l$, the paths that connect $\text{Tail}(E_s)$ and $i_0$, $\text{Head}(E_s)$ and $i_l$ are both directed paths by Lemma 1 leading to that $E_s$ is in $i_0, i_1, \ldots, i_l$. In conclusion, $E_s$ must be in path $i_0, i_1, \ldots, i_l$ and the cases studied above can be summarized as follows.

(i) $\text{Tail}(E_s) = i_0, \text{Head}(E_s) = i_l \Rightarrow E_s = (i_0, i_l)$ is an edge in $T_G \Rightarrow E_s$ is unique;

(ii) $\text{Tail}(E_s) = i_0, \text{Head}(E_s) \neq i_l \Rightarrow i_0, i_1, \ldots, i_l$ is directed path and $E_s = (i_0, i_1) \Rightarrow E_s$ is unique;

(iii) $\text{Tail}(E_s) \neq i_0, \text{Head}(E_s) = i_l \Rightarrow i_l, i_{l-1}, \ldots, i_0$ is directed path and $E_s = (i_l, i_{l-1}) \Rightarrow E_s$ is unique;

(iv) $\text{Tail}(E_s) \neq i_0, \text{Head}(E_s) \neq i_l \Rightarrow$ the paths from $\text{Tail}(E_s)$ to $i_0$ and from $\text{Head}(E_s)$ to $i_l$ are directed paths and $E_s = (i_l, i_{l-1}) \Rightarrow E_s$ is unique.

This completes the proof of (ii).

Appendix C. Proof of Theorem 1

We start the proof by showing the sufficiency of Condition (i), (ii) and (iii). Without loss of generality, we assume $(i_p, i_{p+1}) \prec (i_p, i_{p-1})$. Let $E_{i_p} = (i_p, i_{p+1})$. According to Algorithm (8), the node states $x_{i_p}(t_p)$ and $x_{i_{p+1}}(t_p)$ are given by

\[
x_{i_p}(t_p) = \gamma_{t_p} \\
x_{i_{p+1}}(t_p) = x_{i_p}(t_p - 1) + x_{i_{p+1}}(t_p - 1) - \gamma_{t_p}.
\]

As specified by Condition (i), $i_p, i_{p-1}, \ldots, i_0$ is a directed path, which satisfies Condition (i) of Lemma 1. Condition (ii) gives $E_{i_p} \prec (i_p, i_{p-1}) \prec \cdots \prec (i_1, i_0) \preceq E_{i_0}$, satisfying Condition (ii) of Lemma 1. In addition, Condition (iii) is equivalent to Condition (iii) of Lemma 1 for path $i_p, i_{p-1}, \ldots, i_0$. Thus Lemma 1 shows that random variable $\gamma_{t_p}$ appears in node state $x_{i_0}(T_s)$,
i.e.,
\[ d_{i_0t_p}(T_s) = 1. \] (55)

Analogously, three conditions of Lemma [1] are met for path \( i_{p+1}, i_{p+2}, \ldots, i_l \), which yields
\[ d_{i_{p}t_p}(T_s) = -1. \] (56)

Evidently, (55) and (56) make it sufficient for node states \( x_{i_0}(T_s) \) and \( x_{i_l}(T_s) \) to be dependent. In the following, we prove the necessity of Condition (i), (ii) and (iii).

Necessity of (i). Without loss of generality, we assume (55) and (56) that appear in node states \( x_{i_0}(T_s) \) and \( x_{i_l}(T_s) \), respectively. Algorithm (8) gives
\[
\begin{align*}
{x}_{\text{Tail}}(E_{t_p})(t_p) &= \gamma_{t_p} \\
{x}_{\text{Head}}(E_{t_p})(t_p) &= {x}_{\text{Tail}}(E_{t_p})(t_p - 1) + {x}_{\text{Head}}(E_{t_p})(t_p - 1) - \gamma_{t_p}.
\end{align*}
\]

We suppose that \( \gamma_{t_p} \) and \( -\gamma_{t_p} \) transfer to \( x_{\text{Tail}}(E_{t_p})(T_s) \) and \( x_{\text{Head}}(E_{t_p})(T_s) \), respectively, i.e., \( d_{i_0t_p} = 1 \) and \( d_{i_{p}t_p} = -1 \). Due to the uniqueness of paths in spanning trees, nodes \( \text{Tail}(E_{t_p}) \) and \( \text{Head}(E_{t_p}) \) are in path \( i_0, i_1, \ldots, i_l \). Let \( i_p = \text{Tail}(E_{t_p}) \). Condition (i) of Lemma [1] shows that \( i_p, i_{p-1}, \ldots, i_0 \) and \( i_p, i_{p+1}, \ldots, i_l \) are directed paths. In addition, \( i_p \neq i_0 \) and \( i_p \neq i_l \) because there exists no directed path that connects node \( i \) and node \( j \). The necessity of Condition (i) can be similarly proved, provided that \( -\gamma_{t_p} \) and \( \gamma_{t_p} \) transfer to \( x_{\text{Tail}}(E_{t_p})(T_s) \) and \( x_{\text{Head}}(E_{t_p})(T_s) \), respectively.

Necessity of (ii). Without loss of generality, we assume \( (i_p, i_{p+1}) \prec (i_p, i_{p-1}) \). Now we prove the necessity of Condition (ii). Since the random variable \( d_{i_0t_p}(T_s)\gamma_{t_p} \) transfers from \( x_{i_{p}}(t_p) \) to \( x_{i_0}(T_s) \), Lemma [1] provides
\[ (i_p, i_{p-1}) \prec \cdots \prec (i_1, i_0). \] (57)

Similarly, random variable \( d_{i_{p}t_p}(T_s)\gamma_{t_p} \) transfers from \( x_{i_{p+1}}(t_p) \) to \( x_{i_{l}}(T_s) \) assures
\[ (i_p, i_{p+1}) \prec \cdots \prec (i_{l-1}, i_l). \] (58)

Clearly (57) and (58) shows the necessity of Condition (ii). Necessity of (iii). We finally prove Condition (iii) is necessary for the dependence result. We have shown above that if the node states \( x_{i_0}(T_s) \) and \( x_{i_l}(T_s) \) are dependent, the random variables \( d_{i_0t_p}(T_s)\gamma_{t_p} \) and \( d_{i_{p}t_p}(T_s)\gamma_{t_p} \) transfer to \( x_{i_0}(T_s) \) and \( x_{i_l}(T_s) \), respectively. Thus Condition (iii) of Lemma [1] are necessarily met on both path \( i_p, i_{p-1}, \ldots, i_0 \) and path \( i_{p+1}, i_{p+2}, \ldots, i_l \), which is equivalent to Condition (iii). Now the necessity of all three conditions is proved.

Appendix D. Proof of Theorem [2]

First we focus on proving the sufficiency part of the statements. Let \( E_{t_k} = (i_k, i_{k+1}) \) for \( k = 0, 1, \ldots, l - 1 \). According to the algorithm (8), information transmission occurs on edge \((i_0, i_1)\) at time \( t_0 \). Then it follows
\[
\begin{align*}
x_{i_0}(t_0) &= \gamma_{t_0} \\
x_{i_1}(t_0) &= x_{i_0}(t_0 - 1) + x_{i_1}(t_0 - 1) - \gamma_{t_0}.
\end{align*}
\] (59)
It is seen from (59) that the random variables $\gamma_{t_0}$ and $-\gamma_{t_0}$ are held by the node states $x_{i_0}(t_0)$ and $x_{i_1}(t_0)$, respectively. Evidently, Condition (ii) specifies that the endpoints of all edges not equal to $(i_0, i_1)$ with their tail being node $i_0$ exchange information according to the algorithm (8) prior to $(i_0, i_1)$. Thus

$$d_{i_0t_0}(T_*) = 1.$$  

(60)

Since $i_0, i_1, \ldots, i_l$ is a directed path, Lemma 4 provides that Condition (i) and (ii) guarantees that $-\gamma_{t_0}$ transfers to $x_{i_l}(T_*)$, i.e.,

$$d_{i_l t_0}(T_*) = -1.$$  

(61)

(60) and (61) clearly show that node states $x_{i_0}(T_*)$ and $x_{i_l}(T_*)$ are dependent.

Now we prove the necessity of these two conditions. Suppose $x_{i_0}(T_*)$ and $x_{i_l}(T_*)$ are dependent. Then there exist random variables $d_{i_0 t_p}(T_*) \gamma_{t_p}$ and $d_{i_l t_p}(T_*) \gamma_{t_p}$ that appear in $x_{i_0}(T_*)$ and $x_{i_l}(T_*)$, respectively. It is clear that the nodes Tail($E_{t_p}$) and Head($E_{t_p}$) are in the path $i_0, i_1, \ldots, i_l$ because of the path uniqueness in spanning trees. However, it is impossible that Tail($E_{t_p}$) = $i_k$ and Head($E_{t_p}$) = $i_{k+1}$ for any $k \in \{1, \ldots, l-1\}$, because $i_k, i_{k-1}, \ldots, i_0$ is necessarily a directed path by Lemma 4. Hence Tail($E_{t_p}$) = $i_0$ and Head($E_{t_p}$) = $i_1$. In addition, $d_{i_0 t_p} = 1$ and $d_{i_1 t_p} = -1$. Finally the random variable $-\gamma_{t_p}$ becomes a component of $x_{i_l}(T_*)$. Thus by Lemma 4 it is necessary for Condition (i) and (ii) to hold, which completes the proof.

Appendix E. Proof of Theorem 3

Lemma 2 (i) implies that at most $T_*$ pairs of node states in $x_1(T_*)$, $x_2(T_*)$, $\ldots$, $x_n(T_*)$ are dependent. Since it is described in Lemma 2 (ii) that every pair of dependent final node states possess only one of $\pm \gamma_1, \ldots, \pm \gamma_{T_*}$, there are exactly $T_*$ pairs of dependent final node states. In addition, it can be calculate the covariance of two dependent states is $-\sigma^2$. Thus $|E_{T_*}| = T_*$. Second, we complete the proof that $G_{T_*}$ is a tree by showing $G_{T_*}$ is connected. We know for an arbitrary node $i \in V$, there exists $t \in \{1, 2, \ldots, T_*\}$ such that $i$ is an endpoint of $E_t$. Thus the algorithm (8) guarantees that each $x_i(T_*)$ holds at least one of $\gamma_1, \ldots, \gamma_{T_*}$ and thereby $G_{T_*}$ has no isolated nodes. Assume, for contradiction, that $G_{T_*}$ has $r$ connected components with $r > 1$. Let $G_{S_1}$ and $G_{S_2}$ denote two of the connected components of $G_{T_*}$ on $S_1, S_2 \subset V$, respectively. Then there exists $s_0 \in \{1, 2, \ldots, T_*\}$ such that edge $E_{s_0}$ has one of its endpoints in $S_1$ and the other in $S_2$. Without loss of generality, we assume Head($E_{s_0}$) $\in S_1$ and Tail($E_{s_0}$) $\in S_2$. By the definition of connected components, there exists $S \in V$ such that $G_S$ is a connected component of $G_{T_*}$ on $S$ and there exist node $i, j \in S$ such that $x_i(T_*)$ holds $\gamma_{s_0}$ and $x_j(T_*)$ holds $-\gamma_{s_0}$, resulting in $\{i, j\} \in E_{T_*}$. Similarly to the proof of Lemma 2 (ii), the dependence of $x_i(T_*)$ and $x_j(T_*)$ gives

(i) The path from Head($E_{s_0}$) to $i$ is a directed path and Tail($E_{s_0}$) = $j$;

(ii) The path from Tail($E_{s_0}$) to $j$ is a directed path and Head($E_{s_0}$) = $i$;

(iii) The paths from Head($E_{s_0}$) to $i$ and from Tail($E_{s_0}$) to $j$ are both directed paths.

In case (i), conditions in Theorem 2 for Head($E_{s_0}$) and $i$ are still satisfied as a result of the dependence of $x_i(T_*)$, $x_j(T_*)$ by Theorem 1 or Theorem 2. As a consequence, $x_{\text{Head}(E_{s_0})}(T_*)$ and $x_i(T_*)$ are dependent.
and \( \{\text{Head}(E_0), i\} \in E_T \). Clearly, the edges \( \{\text{Head}(E_0), i\} \) and \( \{i, \text{Tail}(E_0)\} \) makes neither \( G_S_1 \) nor \( G_S_2 \) not connected components. Hence \( G_{T_1} \) has one connected component, i.e., \( G_{T_1} \) is connected. Therefore, \( G_{T_1} \) is a tree in case (i). The same conclusion can be drawn for case (ii) and (iii). Thus \( G_{T_1} \) is a tree.

Next we show \( \Sigma_{T_1} \) is the Laplacian of \( G_{T_1} \) by proving the following properties of \( \Sigma_{T_1} \).

(i) For any \( i \neq j \)
\[
[\Sigma_{T_1}]_{ij} = \begin{cases} 
-\sigma^2 \gamma & \text{if } \{i, j\} \in E_{T_1}; \\
0 & \text{otherwise.}
\end{cases}
\]

(ii) For any \( i \in \{1, 2, \ldots, n\} \)
\[
[\Sigma_{T_1}]_{ii} = -\sum_{j \neq i} [\Sigma_{T_1}]_{ij}.
\]

Proof of (i). If \( \{i, j\} \in E_{T_1} \), by Lemma 2 (ii), the covariance of \( x_i(T_1) \) and \( x_j(T_1) \) can be calculated
\[
[\Sigma_{T_1}]_{ij} = \text{cov}(\gamma_s, -\gamma_s) = -\text{var}(\gamma_s) = -\sigma^2 \text{for some } s \in \{1, 2, \ldots, T_1\}.
\]
If \( \{i, j\} \notin E_{T_1} \), \( [\Sigma_{T_1}]_{ij} = 0 \) by the Definition 2. Then (i) has been proved.

Proof of (ii). Suppose node \( i \) has \( r_i \) random variables forming a subset of \( \{\pm \gamma_1, \ldots, \pm \gamma_{T_1}\} \). Then \( [\Sigma_{T_1}]_{ii} = r_i \sigma^2 \). By Lemma 2 (i) and (ii), there exists \( r_i \) node states that are dependent of \( x_i(T_1) \) with covariance \( -\sigma^2 \). This completes the proof of (ii).

It is clear (i) and (ii) show \( \Sigma_{T_1} \) is the Laplacian of \( G_{T_1} \).

References

[1] Y. Liu, J. Wu, I. Manchester, and G. Shi, “Gossip algorithms that preserve privacy for distributed computation Part I: the algorithms and convergence conditions,” IEEE Conference on Decision and Control, to be presented, 2018.

[2] Y. Liu, J. Wu, I. Manchester, and G. Shi, “Gossip algorithms that preserve privacy for distributed computation Part II: performance against eavesdroppers,” IEEE Conference on Decision and Control, to be presented, 2018.

[3] M. Mesbahi and M. Egerstedt. Graph Theoretic Methods in Multiagent Networks. Princeton University Press, 2010.

[4] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” IEEE Transactions on Automatic Control, vol. 54, no. 1, pp. 48–61, 2009.

[5] A. Nedic, A. Ozdaglar, and P. A. Parrilo, “Constrained consensus and optimization in multi-agent networks,” IEEE Transactions on Automatic Control, vol. 55, no. 4, pp. 922–938, 2010.

[6] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al., “Distributed optimization and statistical learning via the alternating direction method of multipliers,” Foundations and Trends in Machine Learning, vol. 3, no. 1, pp. 1–122, 2011.
[7] M. Egerstedt and X. Hu, “Formation constrained multi-agent control,” *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 947–951, 2001.

[8] U. A. Khan, S. Kar, and J. M. Moura, “Distributed sensor localization in random environments using minimal number of anchor nodes,” *IEEE Transactions on Signal Processing*, vol. 57, no. 5, pp. 2000–2016, 2009.

[9] J. Tsitsiklis, D. Bertsekas, and M. Athans, “Distributed asynchronous deterministic and stochastic gradient optimization algorithms,” *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.

[10] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc. San Francisco, 1996.

[11] A. Jadbabaie, J. Lin, and A. S. Morse, “Coordination of groups of mobile autonomous agents using nearest neighbor rules,” *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[12] L. Xiao and S. Boyd, “Fast linear iterations for distributed averaging,” *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[13] M. Mehyar, D. Spanos, J. Pongsajapan, S. H. Low, and R. M. Murray, “Asynchronous distributed averaging on communication networks,” *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 512–520, 2007.

[14] G. Shi, B. D. O. Anderson, and K. H. Johansson, “Consensus over random graph processes: Network Borel-Cantelli lemmas for almost sure convergence,” *IEEE Transactions on Information Theory*, vol. 61, no. 10, pp. 5690-5707, 2015.

[15] S. Mou, J. Liu, and A. S. Morse, “A distributed algorithm for solving a linear algebraic equation,” *IEEE Transactions on Automatic Control*, vol. 60, no. 11, pp. 2863–2878, 2015.

[16] G. Shi, B. D. Anderson, and U. Helmke, “Network flows that solve linear equations,” *IEEE Transactions on Automatic Control*, vol. 62, no. 6, pp. 2659–2674, 2017.

[17] D. Kempe, A. Dobra, and J. Gehrke, “Gossip-based computation of aggregate information,” in *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pp. 482-491, 2003.

[18] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Randomized gossip algorithms,” *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[19] D. Shah, “Gossip algorithms,” *Foundations and Trends® in Networking*, vol. 3, no. 1, pp. 1-125, 2008.

[20] B. Doerr, M. Fouz, and T. Friedrich, “Why rumors spread so quickly in social networks,” *Communications of the ACM*, vol. 55, no. 6, pp. 70-75, 2012.

[21] S. Sundaram and C. Hadjicostis, “Finite-time distributed consensus in graphs with time-invariant topologies,” *American Control Conference*, pp. 711-716, 2007.
[22] Y. Yuan, G.-B. Stan, L. Shi, M. Barahona, and J. Goncalves, “Decentralised minimum-time consensus,” *Automatica*, vol. 49, no. 5, pp. 1227-1235, 2013.

[23] Z. Huang, S. Mitra, and G. Dullerud, “Differentially private iterative synchronous consensus,” in *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, pp. 81–90, ACM, 2012.

[24] Y. Mo and R. M. Murray, “Privacy preserving average consensus,” *IEEE Transactions on Automatic Control*, vol. 62, no. 2, pp. 753–765, 2017.

[25] N. E. Manitara and C. N. Hadjicostis, “Privacy-preserving asymptotic average consensus,” in *European Control Conference*, pp. 760–765, IEEE, 2013.

[26] A. G. Dimakis, S. Kar, J. M. Moura, M. G. Rabbat, and A. Scaglione, “Gossip algorithms for distributed signal processing,” *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, 2010.

[27] F. Bullo, R. Carli, and P. Frasca, “Gossip coverage control for robotic networks: Dynamical systems on the space of partitions,” *SIAM J. Control Optimization*, vol. 50, no. 1, pp. 419-447, 2012.

[28] C. Godsil and GF. Royle. *Algebraic Graph Theory*. Springer Science & Business Media, 2013.

[29] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

[30] M. Kefayati, M. S. Talebi, B. H. Khalaj and H. R. Rabiee, “Secure consensus averaging in sensor networks using random offsets,” *STelecommunications and Malaysia International Conference on Communications, IEEE International Conference on*, pp. 556-560, 2007.

[31] Barber. David. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[32] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.

[33] C. E. Rasmussen, “Gaussian processes in machine learning,” in *Advanced lectures on Machine Learning*, pp. 63–71, Springer, 2004.

[34] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[35] R.B.J.T. Allenby and A. Slomson. *How to Count: An Introduction to Combinatorics*. CRC Press, 2011.

[36] M. Fréchet, “Généralisation du théoreme des probabilités totales,” *Fundamenta Mathematicae*, vol. 1, no. 25, pp. 379–387, 1935.

[37] G. Shi, B. Li, M. Johansson, and K. H. Johansson, “Finite-Time convergent gossiping,” *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2782-2794, 2016.

[38] S. Boyd, P. Diaconis, and L. Xiao, “Fastest mixing Markov chain on a graph,” *SIAM Review*, vol. 46, no. 4, pp. 667-689, 2004.