Securely Solving Linear Algebraic Equations in a Distributed Framework Enhanced With Communication-Efficient Algorithms

Bo Yin, *Student Member, IEEE*, Wenlong Shen, *Student Member, IEEE*, Xianghui Cao, *Senior Member, IEEE*, Yu Cheng, *Senior Member, IEEE*, and Qing Li, *Member, IEEE*

***Abstract*—Solving linear algebraic equations (a.k.a., an LAE problem) distributedly in a network with multiple agents has wide applications in distributed control, estimation, and signal processing. A consensus-based distributed computing framework is studied in this paper. Specifically, each agent knows only a subproblem of the LAE, i.e., a subset of all equations, and then all agents apply a consensus-based algorithm to update their estimates of the correct solution of the LAE problem iteratively. Under certain conditions, it has been shown that all the estimates converge to the exact solution exponentially fast. However, such a distributed paradigm is vulnerable to malicious behaviors in an adversarial environment. In this paper, we indicate a number of security threats in this process, and thus develop robust computing solutions against those attacks. With particular attention to low storage overhead, we develop a new alternating projection method to enhance the consensus algorithm. Furthermore, we design an innovative misbehavior detection mechanism by exploiting the homomorphic signature technique. Our method can detect misbehaving agents without the common, yet sometimes infeasible, assumption of local good majority. Another significant contribution presented in this paper is an original component dropping approach for mitigating the communication overhead during the consensus process. From both theoretical and engineering perspectives, we study how the consensus can still be reached when a big portion of elements in exchanged message vectors are dropped. In our preliminary numerical results, roughly 80% data transmission can be reduced per epoch at the expense of 35% extra epochs to reach the consensus, implying 73% reduction in terms of the communication overhead.**

***Index Terms*—Distributed consensus, linear algebraic equations, security.**

Manuscript received June 6, 2018; revised January 23, 2019; accepted February 18, 2019. Date of publication February 27, 2019; date of current version September 2, 2020. This work was supported in part by the NSF of USA under Grant ECCS-1610874, in part by the National Natural Science Foundation of China under Grant 61573103, and in part by the Outstanding Youth Fund of Jiangsu Province under Grant BK20180012. Recommended for acceptance by S. Xu. *(Corresponding author: Yu Cheng.)*

B. Yin, W. Shen, and Y. Cheng are with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL 60616, USA (e-mail: byin@hawk.iit.edu; wshen7@hawk.iit.edu; cheng@iit.edu).

X. Cao is with the School of Automation, Southeast University, Nanjing 210096, China (e-mail: xhcao@seu.edu.cn).

Q. Li is with Symantec Corporation, Mountain View, CA 94043, USA (e-mail: qing_li@symantec.com).

Digital Object Identifier 10.1109/TNSE.2019.2901887

I. Introduction

SOLVING linear algebraic equations (LAE) $\mathbf{Ax} = \mathbf{b}$ is a fundamental mathematical problem, which frequently arises in a large variety of real-world engineering and scientific computations. In recent years, due to their potential application in distributed tracking, network localization, parameter estimation and page ranking, distributed algorithms for solving LAE have evoked much research attention [1]–[5]. Many of such algorithms exploit a projected consensus framework, the basic idea of which stems from the seminal studies in [6], [7]. The projected consensus framework allows multiple agents in a networked system to align their estimates to a consensus value which lies in the intersection of constraint sets imposed by individual agents. Specifically, within a consensus-based distributed framework, each agent is assigned a subproblem of the LAE formed by one row (or possibly multiple rows) of $\mathbf{A}$ and $\mathbf{b}$. Starting with a feasible solution to its subproblem, each agent iteratively updates its local estimate based on estimates received from the neighboring agents. It is shown that the final consensus that all local estimates reach is the exact solution of the original LAE and the convergence process is exponentially fast over static connected network topology [1]. The results have been extended to other network topologies and communication patterns, e.g., time-varying repeatedly connected graph and asynchronous communications [2], [8].

While distributed computing allows exploiting the computation power of multiple agents in a collaborative manner [9], a common concern is the robustness of the system when misbehaving agents exist. In fact, consensus-based algorithms are vulnerable to erroneous updates. For example, false estimates, even if occasional, could result in an incorrect final consensus. In addition, a malicious agent can easily disrupt the progress of consensus by continuously misreporting its local estimate, which prevents normal agents from converging to the correct solution. The focus of this paper is to develop a robust consensus based computing framework that can solve the LAE problem securely in a distributed manner.

Securely solving the LAE problem has been studied in the context of outsourcing the problem to a remote centralized cloud [10]–[12]. Those solutions mainly focus on preventing cloud from probing private information contained in the outsourced LAE, rather than on vulnerabilities in a distributed setting. There are studies on the performance of consensus-based distributed algorithms under adversarial agents in a variety of scenarios such as parameter estimation, time synchronization, and distributed optimization [13]–[17]. However, these approaches cannot be easily adapted for solving an LAE problem. For example, some of the aforementioned studies target on consensus of a scalar variable while the LAE problem usually encounters the consensus of vectors. Moreover, some algorithms only assure that the final agreement lies in the convex hull of the inputs of normal agents; such algorithms are hard to be applied for distributedly solving LAE, to be shown later in this paper.

In [18], we conducted some pioneering studies on developing a distributed outsourcing scheme for solving an LAE problem securely in a consensus-based framework. While the work in [18] established some general principles for developing a secure distributed framework for solving LAE problem, it did not fully address some important issues. 1) The proposed algorithm in [18] considers a special case that each agent handles only one row of the LAE problem. The case in which each node stores multiple rows is not well investigated. 2) The monitoring algorithm in [18] for detecting malicious agents requires local good majority and suffers from large communication overhead. 3) The scheme in [18] does not address the communication overhead in the consensus procedure. Note that agents need to exchange hundreds of high dimensional vectors for solving a large scale LAE problem.

In this paper, we develop a distributed framework where multiple agents collaborate to securely solve an LAE problem. The framework consists of both a robust consensus-based algorithm and a misbehavior detection mechanism, which together protect the integrity and availability of the final solution. Comparing to our initial work [18], this paper considers a general heterogeneous model that each agent may handle different number of rows in the consensus framework. By leveraging a homomorphic signature scheme, we develop a communication-efficient monitoring mechanism that removes the assumption of the local good majority. Shifting and scaling operations are used to ensure the applicability of the homomorphic signature scheme. We also develop a component dropping technique that can significantly mitigate the communication overhead in the consensus process. The main contributions of this paper are summarized as follows:

1) We enhance the consensus algorithm in [18] for a general setting where each agent may be associated with multiple rows of the original LAE and different agents may have different number of rows. In particular, the new consensus algorithm at each round updates the local estimates according to an alternating projection method, which not only perform robustly against malicious local results but also brings storage savings.
2) We design an innovative misbehavior detection mechanism by leveraging a homomorphic signature primitive. The detection mechanism circumvents the assumption of local good majority and allows normal agents to monitor the progress of the consensus process with moderate communication overhead.
3) We develop a component dropping method for mitigating communication overhead. With such an algorithm, each element of a message vector is transmitted (or equivalently dropped) with a probability. That is, some randomly selected elements are transmitted and others are dropped. From both theoretical and engineering perspectives, we study how the consensus can still be reached with random dropping applied.

The remainder of this paper is organized as follows. Section II overviews the related work. Section III describes preliminaries of solving LAE in a distributed framework and presents threat models and the system model. Section IV presents the fault-tolerant consensus-based algorithm and Section V develops the misbehavior detection mechanism. Section VI studies the random dropping technique and numerical results are provided in Section VII. Finally, Section VIII concludes the whole paper.

II. Related Work

Developing efficient methods to solve LAE has received sustained research efforts in the past few decades. One credited philosophy is to decompose the original problem into smaller ones which can be solved in parallel [19], [20]. Many parallel algorithms, however, either are only valid on LAE with special structure or require a central controller to coordinate the results of decomposed problems. Such requirements hinder the applicability of conventional parallel approaches in fully distributed scenarios, which arise naturally in ad-hoc networks or sensor networks. The seminal work in [7] presents a distributed computing framework to address constrained consensus and optimization problem in multi-agent networks. Considering LAE as a special case, several projected consensus algorithms have been proposed to distributedly solve an LAE [1]–[4]. Those works focus on developing algorithms that work properly under different assumptions about network topologies, e.g., static connected graph [1], [4] or time-varying repeatedly connected graph [2], as well as communication patterns, e.g., synchronous [1] or asynchronous [2]. Potential failures of the proposed distributed algorithms in the presence of malicious nodes are not considered in these works.

Existing studies on how to solve LAE securely concentrate on computation outsourcing, where privacy preserving is the primary concern. To this end, various schemes have been designed, enabling a resource-constrained user to outsource the computation involved in solving LAE to powerful cloud without disclosing the information of the LAE [10]–[12], [21]. The work in [21] introduces a series of mechanisms for the secure outsourcing of scientific computations, among which is a disguising scheme targeting LAE. The secure outsourcing scheme for LAE proposed in [10] applies homomorphic encryption scheme to protect user's privacy. The work in [11] hides the problem information through perturbation-based approach, by adding a random noise to the original problem. The work in [12] develops a generic transformation method to disguise the original problem. It is worth noting that schemes proposed in these exiting studies work in centralized manners as the LAE is outsourced to a single cloud server. Endeavors to develop privacy preserving approaches in those works are orthogonal to our work in this paper, where we assume the information of the LAE is well protected and focus on mitigating the vulnerabilities caused by node misbehaviors in a distributed setting. In fact, privacy-preserving outsourcing and secure distributed computing can be integrated in a complementary manner, as demonstrated in [18].

Efforts to investigate the resilient issues of distributed consensus algorithms have been made for a long time [22]–[24]. The recent work [13] and [14] study resilient consensus protocols under different threat models, exploiting traditional graph theoretic properties to characterize the resilience of algorithms. The work in [15] designs a threshold-based parameter checking mechanism to dynamically constrain the misbehaving space of malicious nodes by leveraging the two-hop neighboring information. Although achieving asymptotic consensus, resilient solutions in these works are not suitable to constrained consensus problem, in which the final agreement makes little sense if specific constraints are violated. An impossibility result on the performance of any distributed optimization algorithm is derived in [25] and [16], claiming that "the price paid for resilience is a loss in optimality". The local filtering algorithm proposed in these two works provides provable guarantee with respect to the final agreement of non-adversarial nodes in terms of converging to the convex hull of local optimal points. Nevertheless, the applicability of this method is restricted to scalar optimization problems. The state of each node in such problem is characterized by a scalar variable, which is not the case for solving an LAE. Byzantine vector consensus problems are studied in [17] and [26], where each agent has a multi-dimensional state. The objective of such problems is to drive non-adversarial agent to agree on a point lies in the convex hull of normal inputs, which does not apply in the case of solving LAE.

Our previous work [18] studies a distributed secure outsourcing scheme which is based on the consensus-based distributed algorithm in [1] for solving LAE in ad-hoc clouds. In addition to capability of privacy preserving, that outsourcing scheme is robust against the identified security attacks. Two disadvantages of that scheme motivate this work. For one thing, the scheme in [18] focuses on the setting that each row of the LAE is assigned to one node. The algorithm that can handle multiple rows has yet to be explored. For another, the straightforward checking mechanism for misbehavior detection needs two-hop neighbor estimates to conduct verification, which will incur huge communication overhead. In the work of [5], improvement is made to the LAE-solving algorithm presented in [2] such that less communication between agents is required. However, method in [5] is only applicable to LAE problems with special structure. Specifically, it assumes that zero blocks exist in the matrix $\mathbf{A}$. Inspired by our observation that the misbehavior detection can be considered as integrity verification, we employ a signature-based technique to avoid the requirement for two-hop neighbor estimates. Homomorphic signature scheme has been used in conjunction with network coding to prevent malicious modification of data in the literature [27]–[29]. This cryptographic primitive allows any node to sign a linear space, e.g., linear combination of a set of data, directly without knowing the signature key. In this paper, we will exploit this property to design a monitoring scheme which incurs lower communication overhead. In addition, we also explore the potential of dropping technique for mitigating the communication overhead in the consensus procedure, which imposes no requirement on the structure of the LAE problem.

III. Problem Statement

In this paper, we consider the scenario that m agents form a connected network and collaboratively compute the solution $\mathbf{x}^*$ of an LAE problem in a distributed manner. Throughout this paper, we use boldface letters to represent column vectors and matrices. We denote an LAE problem $\mathbf{Ax} = \mathbf{b}$ by $\mathbf{\Phi} \triangleq [\mathbf{A}\ \mathbf{b}]$, where $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{b} \in \mathbb{R}^{n\times 1}$. This paper focuses on the situation that the LAE problem has a unique solution. That is, $\mathbf{A}$ is an $n \times n$ non-singular matrix. In our system, the LAE parameter $\mathbf{\Phi}$ is fully partitioned into m disjoint subsets of rows which are respectively stored in m agents. More precisely, node i stores r_i out of n rows of $\mathbf{\Phi}$, denoted by $\mathbf{\Phi}_i \triangleq [\mathbf{A}_i\ \mathbf{b}_i]$, where $\mathbf{A}_i \in \mathbb{R}^{r_i\times n}$, $\mathbf{b} \in \mathbb{R}^{r_i\times 1}$ and $\sum_{i=1}^{m} r_i = n$. Note that such a setting is more generic than that of considered in [18] where each agent is assigned with only one row of $\mathbf{\Phi}$. The network topology of the m agents is characterized by an undirected graph $G = \{V, E\}$, where V and E denote the node set[1] and edge set, respectively. Node i and node j can communicate with each other if they have an edge in E. We assume that the connectivity information is known by the nodes, e.g., all nodes can launch a secure neighbor discovery process to obtain the information of their neighborhood [30], [31].

A. Preliminaries on Consensus

The work in [1] proposed a consensus-based algorithm which allows multiple agents to collaboratively solve an LAE. Specifically, each agent maintains a local estimate of the solution and iteratively updates this estimate based on the latest estimates of its neighbors. The updating rule of agent i can be expressed as

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) - \mathbf{P}_i\left(\mathbf{x}_i(t) - \frac{1}{d_i}\sum_{j\in\mathcal{N}_i}\mathbf{x}_j(t)\right) \tag{1}$$

where $\mathcal{N}_i$ denotes the set of neighbor agents of agent i (including agent i itself) and d_i is the number of neighbors of agent i. $\mathbf{P}_i \in \mathbb{R}^{n\times n}$ is the computable orthogonal projection on the kernel of $\mathbf{A}_i$ such that $\mathbf{A}_i\mathbf{P}_i\mathbf{y} = \mathbf{0}, \forall \mathbf{y} \in \mathbb{R}^{n\times 1}$. The projection matrix can be computed as $\mathbf{P}_i = \mathbf{I} - \mathbf{A}_i^{\mathrm{T}}(\mathbf{A}_i\mathbf{A}_i^{\mathrm{T}})^{-1}\mathbf{A}_i$, where $\mathbf{I}$ is the identity matrix. Each agent chooses an initial estimate $\mathbf{x}_i(0)$ such that $\mathbf{A}_i\mathbf{x}_i(0) = \mathbf{b}_i$. According to the updating rule, it can be seen that $\mathbf{A}_i\mathbf{x}_i(t) = \mathbf{b}_i$ always holds during the iterations. Intuitively, by the computation (1), each node tries to drag its local estimate in the kernel space of $\mathbf{A}_i$ to the average of all the estimates within its neighborhood. It has been proved in [1] that, as long as the network topology is connected, local estimates of all agents would converge exponentially fast to the exact solution of problem $\mathbf{\Phi}$.

In practice, the termination condition for node i could be that $\max_{j\in\mathcal{N}_i}\|\mathbf{x}_j(t) - \mathbf{x}_i(t+1)\|_\infty \leq \epsilon$ holds for consecutive L steps, where $\|\cdot\|_\infty$ denotes the infinity norm. A

[1]In this paper, we use node and agent interchangeably.

conservative choice of L is the diameter of the underlying graph. Such stopping criteria is based on the bounded convergence time of maximum and minimum consensus protocols (see [32], [33] for details).

B. Threat Model

The distributed nature of the above algorithm makes it highly vulnerable to various malicious attacks, e.g., the false data injection attacks [34]. Consider the complexities associated with adversarial behaviors, we assume that the problem parameters $\mathbf{\Phi}_i$ that are stored in each node are tamper-resistant. For example, subproblems of the LAE problem in the nodes are assigned by a trusted third party (TTP), e.g., a crowdsourcing platform. In this case, the TTP can sign all the subproblems properly such that a malicious node cannot modify the parameters of the LAE problem. We also assume that the malicious nodes cannot collude with each other. In this paper, we focus on the vulnerabilities of the basic consensus algorithm in [1] to erroneous updates performed by malicious nodes. We consider that the normal nodes strictly follow the algorithm to do the required computation. The malicious nodes, however, tend to update their local estimates in an arbitrary fashion to interrupt the consensus procedure. The effects of the false estimates can be categorized into two types:

- **Preventing the consensus:** Malicious nodes can easily prevent the consensus, sabotaging the convergence process. For example, a malicious node can act as a stubborn agent which never updates its local estimate, instead of updating according to the algorithm (1). It can be seen that a consensus will not be reached if this stubborn local estimate is not the exact solution of the LAE problem.
- **Manipulating the solution:** Besides preventing the consensus, malicious nodes also have the capability to manipulate the final result through error injection. Specifically, a malicious node can modify its intermediate local estimates during the consensus process, and thus trick all nodes to agree on an incorrect solution. The effect of such attack is further explained in the following lemma.

Lemma 1. A finite number of error injections will not prevent the consensus but can deviate the final solution.

Proof. Without loss of generality, we first analyze how a one-time error injection can affect the final result. Consider that at a certain moment, say the t-th iteration, the local estimate of node i is modified from $\mathbf{x}_i(t)$ to $\mathbf{x}'_i(t) = \mathbf{x}_i(t) + \mathbf{q}$, where $\mathbf{q}$ is in the row space of $\mathbf{A}_i$ such that $\mathbf{P}_i\mathbf{q} = \mathbf{0}$ and $\mathbf{A}_i\mathbf{q} \neq \mathbf{0}$. Otherwise, we consider the orthogonal projection of $\mathbf{q}$ onto row space of $\mathbf{A}_i$ as the error vector since its orthogonal complement vanishes in the iteration. In the following iterations, node i strictly obey the updating rule. According to (1), the local estimate of node i at the $(t+1)$-th iteration becomes

$$\begin{aligned}\mathbf{x}'_i(t+1) &= \mathbf{x}'_i(t) - \mathbf{P}_i\left(\mathbf{x}'_i(t) - \frac{1}{d_i}\sum_{j\in\mathcal{N}_i}\mathbf{x}_j(t) - \frac{1}{d_i}\mathbf{q}\right)\\ &= \mathbf{x}_i(t+1) + \mathbf{q} - \left(\frac{d_i-1}{d_i}\right)\mathbf{P}_i\mathbf{q}\\ &= \mathbf{x}_i(t+1) + \mathbf{q}\end{aligned} \quad (2)$$

Thus $\mathbf{A}_i\mathbf{x}'_i(t+1) = \mathbf{b}_i + \mathbf{A}_i\mathbf{q} = \mathbf{b}'_i$. The value of $\mathbf{x}'_i(t+1)$ can be considered as the valid local estimate of node i with respect to LAE problem $\mathbf{\Phi}' \triangleq [\mathbf{A}\ \mathbf{b}']$, where $\mathbf{b}'_i = \mathbf{b}_i + \mathbf{A}_i\mathbf{q}$ and $\mathbf{b}'_j = \mathbf{b}_j, \forall j \neq i$, if node i follows the algorithm (1) in all the remain iterations. If no further error is introduced in the system in the following computations, estimates of all nodes can still reach a consensus and the final result will be the exact solution of $\mathbf{\Phi}'$. If a node launch error injection for multiple (but a finite number) rounds or there are multiple misbehaving nodes, it is not difficult to see that the impact is just that a certain b'_i is changed for multiple rounds or multiple b_is (i can take different index values) are changed to b'_is, respectively. After the last-time modification and when every node resumes to the normal algorithm (1), the local estimates of all nodes will converge to the solution of the modified LAE problem. ■

Based on Lemma 1, a malicious node can change the consensus by sending one-time (or multiple) false update and then behaving normally. In this paper, we term this kind of misbehavior as camouflage attack.

C. System Overview

Regarding the threats indicated above, although a series of countermeasures were developed in [18] to ensure the availability and correctness of the final result, some fundamental issues are yet to be considered. First, the scheme in [18] mainly focuses on the setting that one row of $\mathbf{\Phi}$ is assigned to each node. A more general and practical setting is that each node in the network may be assigned with different number of rows fitting its computation capability. Second, the proposed misbehavior detection algorithm in [18] requires the assumption of local good majority. Such assumption is strong and may be invalid in practice. Due to the random distribution of misbehaving nodes, the number of malicious nodes may exceed the number of normal nodes in a certain neighborhood. Third, the communication overhead was not considered in [18] for neither the consensus procedure nor the misbehavior detection scheme. The communication overhead is in fact an important issue, especially when the size of the LAE is large. In this paper, we present a set of innovative techniques to address such three aspects of fundamental issues.

The robust distributed computing framework to be developed is illustrated in Fig. 1. The basic computing model is that each agent is assigned with a subproblem that may have different rows from the original LAE problem $\mathbf{\Phi}$. The central computing component is a *robust consensus-based algorithm*. It can protect the correctness of the final solution, if any, under camouflage attack. Such a property can avoid incurring unnecessary computation and communication overhead, compared to a straightforward design trying to defending against possible attacks at each step. The *misbehavior detection mechanism* provides another layer of protection, considering that some persistent attackers may continue injecting false estimates without stopping. In such a situation, the robust consensus algorithm itself cannot ensure the availability of a correct final result. Our security design philosophy here is to explicitly detect such persistent attackers. We are to present an innovative design based on a homomorphic signature, which does not require the strong assumption of local

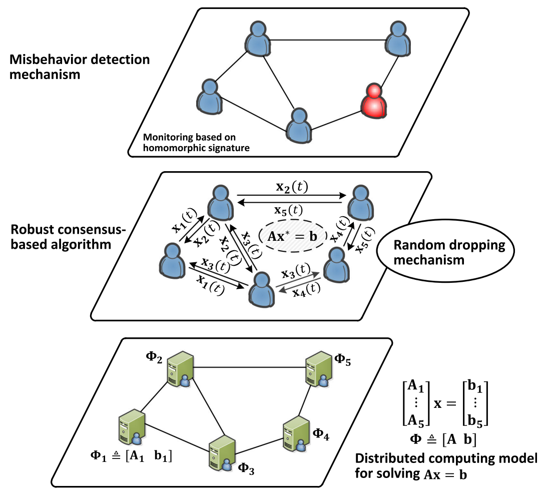

Misbehavior detection mechanism
Monitoring based on homomorphic signature
Robust consensus-based algorithm
$\mathbf{x}_2(t)$
$\mathbf{x}_5(t)$
$\mathbf{x}_1(t)$
$\mathbf{x}_2(t)$
$\mathbf{x}_2(t)$
$\mathbf{x}_3(t)$
$\mathbf{A}\mathbf{x}^* = \mathbf{b}$
$\mathbf{x}_4(t)$
$\mathbf{x}_5(t)$
$\mathbf{x}_3(t)$
$\mathbf{x}_1(t)$
$\mathbf{x}_3(t)$
$\mathbf{x}_4(t)$
Random dropping mechanism
$\mathbf{\Phi}_2$
$\mathbf{\Phi}_5$
$\mathbf{\Phi}_4$
$\mathbf{\Phi}_1 \triangleq [\mathbf{A}_1 \ \ \mathbf{b}_1]$
$\mathbf{\Phi}_3$
$\begin{bmatrix}\mathbf{A}_1\\ \vdots \\ \mathbf{A}_5\end{bmatrix}\mathbf{x} = \begin{bmatrix}\mathbf{b}_1\\ \vdots \\ \mathbf{b}_5\end{bmatrix}$
$\mathbf{\Phi} \triangleq [\mathbf{A} \ \ \mathbf{b}]$
Distributed computing model for solving $\mathbf{A}\mathbf{x} = \mathbf{b}$

Fig. 1. An illustration of the distributed computing framework for solving LAE.

good majority. The proposed computing framework also incorporates a *random dropping mechanism*, which can greatly mitigate the communication overhead in both the consensus procedure and misbehavior detection algorithm while maintaining the performance.

IV. Robust Consensus-Based Algorithm

In this section, we first analyze the root reason of the vulnerability of the original consensus algorithm in [1], which makes it suffering from the camouflage attack. We then describe a new version of the updating rule which is robust against such attack. The robust updating rule is further implemented with an alternating projection method, which reduces the storage overhead required in computing.

A. Robust Updating Rule

Analysis in Section III-B demonstrates that false updates can mislead all nodes converging to a wrong solution. Such vulnerability stems from the updating rule (1), where the current local estimate $\mathbf{x}_i(t)$ (at a certain node i) is explicitly propagated to next round value $\mathbf{x}_{i+1}(t)$, with an updating value in the kernel space of $\mathbf{A}_i$; note that the local updating in the kernel space has no way to correct any errors or malicious attacks directly applied on $\mathbf{x}_i(t)$.

In order to overcome the vulnerability, we first reformulate the updating rule (1) as follows,

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) - \mathbf{P}_i\mathbf{x}_i(t) + \mathbf{P}_i\bar{\mathbf{x}}_i(t) \tag{3}$$

where $\bar{\mathbf{x}}_i(t) = \frac{1}{d_i}\sum_{j\in\mathcal{N}_i}\mathbf{x}_j(t)$ represents the average estimate of neighbors of node i. Let $\mathbf{z}_i = \mathbf{x}_i(t) - \mathbf{P}_i\mathbf{x}_i(t)$. The projection matrix has idempotent property that $\mathbf{P}_i^2 = \mathbf{P}_i$. Thus, we can get that

$$\mathbf{P}_i\mathbf{z}_i = \mathbf{P}_i\mathbf{x}_i(t) - \mathbf{P}_i^2\mathbf{x}_i(t) = \mathbf{0} \tag{4}$$

With $\mathbf{A}_i\mathbf{P}_i = \mathbf{0}$, we then have

$$\mathbf{A}_i\mathbf{z}_i = \mathbf{A}_i\mathbf{x}_i(t) - \mathbf{A}_i\mathbf{P}_i\mathbf{x}_i(t) = \mathbf{b}_i \tag{5}$$

Note that $\mathbf{P}_i = \mathbf{I} - \mathbf{A}_i^{\mathrm{T}}(\mathbf{A}_i\mathbf{A}_i^{\mathrm{T}})^{-1}\mathbf{A}_i$. Plugging in this result to the left hand side (LHS) of (4) and then further applying (5), we obtain

$$\mathbf{z}_i = \mathbf{A}_i^{\mathrm{T}}(\mathbf{A}_i\mathbf{A}_i^{\mathrm{T}})^{-1}\mathbf{b}_i \tag{6}$$

With the steps above, the updating rule is transformed to

$$\mathbf{x}_i(t+1) = \mathbf{z}_i + \mathbf{P}_i\bar{\mathbf{x}}_i(t) \tag{7}$$

In the right hand side (RHS) of updating rule (7), $\mathbf{z}_i$ is determined only by problem parameter $\mathbf{\Phi}_i$, and independent of local estimates. The updating at each round is the sum of a constant vector $\mathbf{z}_i$ with the updating part in the kernel space of $\mathbf{A}_i$. With updating rule (7), injected errors will be always confined in the kernel space of $\mathbf{A}_i$. The influence of erroneous update is thus limited to impacting the initial value for starting the consensus procedure, refer to the proof of theorem 1 below.

Lemma 2. With updating rule (7), each node can start the consensus algorithm with an arbitrary initial value.

Proof. In updating rule (7), the error introduced by $\mathbf{x}_i(t)$ is confined in the kernel space of $\mathbf{A}_i$. Thus, $\mathbf{A}_i\mathbf{x}_i(t) = \mathbf{b}_i$ always holds for $t > 0$, regardless of the value of $\mathbf{x}_i(0)$. Since $\mathbf{x}_i(1)$ satisfies the requirement for the initial value imposed by the original consensus algorithm, updating rule (7) has the same convergent property as the algorithm (1). ■

Theorem 1. The proposed updating rule (7) is robust against the camouflage attack. That is, as long as all nodes (including malicious nodes) update their local estimates according to (7) from a certain moment on, these local estimates will not converge to an incorrect solution.

Proof. Consider that all the malicious nodes stop injecting false updates by time t. Afterward, the dynamic of node i's local estimate is equivalent to its counterpart with initial value $\mathbf{x}_i(t)$. According to Lemma 2, all the local estimates will converge to the exact solution with updating rule (7). ■

Remark. Compared to the updating rule (1), the updating rule (7) has no requirement on the $\mathbf{x}_i(0)$. Based on Theorem 1, an algorithm is robust against the camouflage attack if it works properly with arbitrary initialization. Basically, the algorithm with updating rule (7) is the plain generalization to the robust algorithm proposed in [18], which achieves the aforementioned condition by confining the injected errors in the kernel space of $\mathbf{A}_i$. In parallel with [18], we find that the work in [35] also developed an initialization-free algorithm by adding one additional term to (1). With appropriate expression manipulation, the approach proposed in [35] coincides with the updating rule (7).

B. Alternating Projection Based Algorithm

When examining the computational complexity of updating rule (7), it can be seen that the average of neighbors' latest estimates can be calculated easily and the computational complexity of (7) is dominated by the projection operation $\mathbf{P}_i\bar{\mathbf{x}}_i(t)$. If $\mathbf{P}_i$ is precomputed and stored in node i, the straightforward matrix-vector multiplication takes time $\mathcal{O}(n^2)$. Note that storing $\mathbf{P}_i$ incurs the storage overhead $\mathcal{O}(n^2)$. The projection may be calculated more efficiently by exploiting the orthogonal decomposition. That is, $\mathbf{P}_i\mathbf{y} = \mathbf{y} - \mathbf{A}_i^{\mathrm{T}}(\mathbf{A}_i\mathbf{A}_i^{\mathrm{T}})^{-1}(\mathbf{A}_i\mathbf{y})$. If the $n \times r_i$ matrix $\mathbf{A}_i^{\mathrm{T}}(\mathbf{A}_i\mathbf{A}_i^{\mathrm{T}})^{-1}$ is computed in advance, the projection operation has time complexity $\mathcal{O}(r_i n)$. However, such a method still incurs $\mathcal{O}(r_i n)$ storage overhead to store the $n \times r_i$ matrix $\mathbf{A}_i^{\mathrm{T}}(\mathbf{A}_i\mathbf{A}_i^{\mathrm{T}})^{-1}$. Such a size is comparable to the problem parameter $\mathbf{\Phi}_i$.

We develop a more efficient local updating algorithm, which is executed on the fly without extra storage overhead. Our inspiration comes from special case where $r_i = 1$. Let $\mathbf{a}^{\mathrm{T}} \in \mathbb{R}^{1\times n}$ be a row vector and $\mathbf{P}_a$ be the corresponding orthogonal projection matrix on the kernel of $\mathbf{a}^{\mathrm{T}}$. Through orthogonal decomposition, for an arbitrary vector $\mathbf{y}$, $\mathbf{P}_a\mathbf{y} = \mathbf{y} - \mathbf{a}(\mathbf{a}^{\mathrm{T}}\mathbf{a})^{-1}\mathbf{a}^{\mathrm{T}}\mathbf{y}$. It can be observed that the projection on the kernel of a row vector only incurs computation time complexity $\mathcal{O}(n)$. With this point in mind, we next develop an efficient fault-tolerant algorithm based on the alternating projection technique [36]. The algorithm requires no precomputed results and thus incurs no extra storage overhead.

The basic idea of alternating projection based algorithm is to replace the projection on the kernel of $\mathbf{A}_i$ in (7) by projecting $\bar{\mathbf{x}}_i(t)$ alternately onto the kernel of each row in $\mathbf{A}_i$. For ease of presentation, $\mathbf{\Phi}_i$ is represented as

$$[\mathbf{A}_i \quad \mathbf{b}_i] = \begin{bmatrix} \mathbf{a}_{i,1}^{\mathrm{T}} & b_{i,1} \\ \vdots & \vdots \\ \mathbf{a}_{i,r_i}^{\mathrm{T}} & b_{i,r_i} \end{bmatrix}, \tag{8}$$

where $\mathbf{a}_{i,j}^{\mathrm{T}}$ and $b_{i,j}$ are the j-th row and component of $\mathbf{A}_i$ and $\mathbf{b}_i$, respectively. Let $\mathbf{P}_{i,j}$ be the orthogonal projection matrix on the kernel of $\mathbf{a}_{i,j}^{\mathrm{T}}$. Thus, in the alternating projection based algorithm, $\mathbf{P}_i$ is replaced by $\tilde{\mathbf{P}}_i = \mathbf{P}_{i,r_i}\mathbf{P}_{i,r_i-1}\cdots\mathbf{P}_{i,1}$. In a nutshell, the updating rule of alternating projection-based algorithm is

$$\mathbf{x}_i(t+1) = \tilde{\mathbf{z}}_i + \tilde{\mathbf{P}}_i\bar{\mathbf{x}}_i(t), \tag{9}$$

where the invariant vector $\mathbf{z}_i$ is adapted correspondingly to $\tilde{\mathbf{z}}_i = (\mathbf{I} - \tilde{\mathbf{P}}_i)\mathbf{x}^*$, which is analogous to $\mathbf{z}_i = (\mathbf{I} - \mathbf{P}_i)\mathbf{x}^*$ in (7). While $\tilde{\mathbf{z}}_i$ is defined over $\mathbf{x}^*$, we are to show that it in fact can be computed without the knowledge of $\mathbf{x}^*$. In the following, we will first study the computation complexity of (9) and then theoretically analyze the convergence property and robustness under false updates.

It can be seen that calculating $\tilde{\mathbf{P}}_i\bar{\mathbf{x}}_i(t)$ takes time $\mathcal{O}(r_i n)$ with alternating projection through $\mathbf{P}_{i,1}$ to $\mathbf{P}_{i,r_i}$. Next, we are to show that $\tilde{\mathbf{z}}_i$ can also be computed with computational complexity $\mathcal{O}(r_i n)$. Since $\mathbf{x}^*$ is the exact solution, $\mathbf{a}_{i,j}^{\mathrm{T}}\mathbf{x}^* = b_{i,j}$ holds for each row. Thus, we have

$$\tilde{\mathbf{z}}_{i,j} = (\mathbf{I} - \mathbf{P}_{i,j})\mathbf{x}^* = \frac{\mathbf{a}_{i,j}^{\mathrm{T}}\mathbf{x}^*}{\mathbf{a}_{i,j}^{\mathrm{T}}\mathbf{a}_{i,j}}\mathbf{a}_{i,j} = \frac{b_{i,j}}{\mathbf{a}_{i,j}^{\mathrm{T}}\mathbf{a}_{i,j}}\mathbf{a}_{i,j}. \tag{10}$$

Note that the computation in (10) requires only the information in $\mathbf{\Phi}_i$ and has time complexity $\mathcal{O}(n)$.

Consider further that $\mathbf{I} - \tilde{\mathbf{P}}_i$ can be represented as

$$\begin{aligned} \mathbf{I} - \tilde{\mathbf{P}}_i &= \mathbf{I} - \mathbf{P}_{i,r_i}\mathbf{P}_{i,r_i-1}\cdots\mathbf{P}_{i,1} \\ &= \mathbf{I} - \mathbf{P}_{i,r_i} + \mathbf{P}_{i,r_i} - \mathbf{P}_{i,r_i}\mathbf{P}_{i,r_i-1} \\ &\quad + \mathbf{P}_{i,r_i}\mathbf{P}_{i,r_i-1} - \mathbf{P}_{i,r_i}\mathbf{P}_{i,r_i-1}\mathbf{P}_{i,r_i-2} \\ &\quad + \cdots \\ &\quad + \mathbf{P}_{i,r_i}\mathbf{P}_{i,r_i-1}\cdots\mathbf{P}_{i,2} - \mathbf{P}_{i,r_i}\mathbf{P}_{i,r_i-1}\cdots\mathbf{P}_{i,1} \\ &= (\mathbf{I} - \mathbf{P}_{i,r_i}) + \mathbf{P}_{i,r_i}(\mathbf{I} - \mathbf{P}_{i,r_i-1}) \\ &\quad + \mathbf{P}_{i,r_i}\mathbf{P}_{i,r_i-1}(\mathbf{I} - \mathbf{P}_{i,r_i-2}) \\ &\quad + \cdots \\ &\quad + \mathbf{P}_{i,r_i}\mathbf{P}_{i,r_i-1}\cdots\mathbf{P}_{i,2}(\mathbf{I} - \mathbf{P}_{i,1}) \end{aligned} \tag{11}$$

thus, $\tilde{\mathbf{z}}_i = \tilde{\mathbf{z}}_{i,r_i} + \mathbf{P}_{i,r_i}\tilde{\mathbf{z}}_{i,r_i-1} + \cdots + \mathbf{P}_{i,r_i}\mathbf{P}_{i,r_i-1}\cdots\mathbf{P}_{i,2}\tilde{\mathbf{z}}_{i,1}$. We introduce auxiliary vectors $\mathbf{u}(r), r \in [1, 2, \ldots, r_i]$, to demonstrate the implicit recursive structure of above equation. Let

$$\begin{aligned} &\mathbf{u}(1) = \tilde{\mathbf{z}}_{i,1} \\ &\mathbf{u}(2) = \tilde{\mathbf{z}}_{i,2} + \mathbf{P}_{i,2}\tilde{\mathbf{z}}_{i,1} \\ &\quad\cdots \\ &\mathbf{u}(r_i) = \tilde{\mathbf{z}}_{i,r_i} + \mathbf{P}_{i,r_i}\tilde{\mathbf{z}}_{i,r_i-1} + \cdots + \mathbf{P}_{i,r_i}\mathbf{P}_{i,r_i-1}\cdots\mathbf{P}_{i,2}\tilde{\mathbf{z}}_{i,1} \end{aligned}$$

It can be seen that $\mathbf{u}(r)$ satisfies

$$\mathbf{u}(j+1) = \tilde{\mathbf{z}}_{i,j+1} + \mathbf{P}_{i,j+1}\mathbf{u}(j) \tag{12}$$

Computing $\tilde{\mathbf{z}}_{i,j+1}$ has time complexity $\mathcal{O}(n)$, and the projection with respect to $\mathbf{P}_{i,j+1}$ takes time $\mathcal{O}(n)$. Thus, given $\mathbf{u}(j)$, calculating $\mathbf{u}(j+1)$ takes time $\mathcal{O}(n)$. According to (10) and (11), it is worth noting that $\tilde{\mathbf{z}}_i = \mathbf{u}(r_i)$. Therefore, the invariant part $\tilde{\mathbf{z}}_i$ can be calculated by iteratively updating $\mathbf{u}(r)$, with time complexity $\mathcal{O}(r_i n)$. Recall that computing the alternating projection part $\tilde{\mathbf{P}}_i\bar{\mathbf{x}}_i(t)$ also takes time $\mathcal{O}(r_i n)$, the computational complexity for updating $\mathbf{x}_i(t)$ based on (11) and (12) is $\mathcal{O}(r_i n)$.

In summary, the alternating projection based algorithm involves two-level of iterations. The outer level is responsible for updating local estimates, i.e., updating $\mathbf{x}_i(t)$ to $\mathbf{x}_i(t+1)$ iteratively. For ease of presentation, hereinafter, we use the term "epoch" to describe one outer iteration. Within an epoch, the inner level of iteration runs according to (12) to obtain the value $\tilde{\mathbf{z}}_i$. By the end of each epoch, nodes emit their local estimates to their neighbors for next round update.

Definition 1. (Mixed Matrix Norm [2]) Given a block matrix $\mathbf{Q} \in \mathbb{R}^{mn\times mn}$, the mixed matrix norm of $\mathbf{Q}$, denoted by $\|\mathbf{Q}\|$, is defined as

$$\|\mathbf{Q}\| = |\langle\mathbf{Q}\rangle|_\infty, \tag{13}$$

where $\langle\mathbf{Q}\rangle$ represents a matrix in $\mathbb{R}^{m\times m}$ whose ijth entry is $|\mathbf{Q}_{ij}|_2$. $|\cdot|_2$ and $|\cdot|_\infty$ denotes the induced two norm and infinity norm, respectively.

The consensus algorithm over the whole system, with updating rule (9) at each agent, can still converge exponentially fast with arbitrary initial values at each agent. We have the following Theorem.

Theorem 2. Suppose each node i updates the local estimate $\mathbf{x}_i(t)$ according to (9). If the network topology is connected, then there exists a positive constant $\lambda < 1$ such that all $\mathbf{x}_i(t)$ converges to the $\mathbf{x}^*$ as $t \to \infty$, as fast as λ^t converges to 0.

Proof. The exponential convergence property of alternating projection based algorithm can be derived by applying the proof techniques in [2]. Suppose that the local estimate at node i, after $t+1$ iterations, deviates from $\mathbf{x}^*$ with a vector of $\mathbf{e}_i(t+1)$. According to (9), we have

$$\begin{aligned}\mathbf{e}_i(t+1) &= \mathbf{x}_i(t+1) - \mathbf{x}^* \\ &= (\mathbf{I} - \tilde{\mathbf{P}}_i)\mathbf{x}^* + \tilde{\mathbf{P}}_i \bar{\mathbf{x}}_i(t) - \mathbf{x}^* \\ &= \tilde{\mathbf{P}}_i \frac{\sum_{j \in \mathcal{N}_i} (\mathbf{x}_j(t) - \mathbf{x}^*)}{d_i} \\ &= \tilde{\mathbf{P}}_i \frac{\sum_{j \in \mathcal{N}_i} \mathbf{e}_j(t)}{d_i}.\end{aligned} \tag{14}$$

Let $\mathbf{e}(t) = [\mathbf{e}_1^{\mathrm{T}}(t), \mathbf{e}_2^{\mathrm{T}}(t), \ldots, \mathbf{e}_m^{\mathrm{T}}(t)]^{\mathrm{T}}$ and $\mathbf{D}$ be the adjacency matrix of network topology. According to (14),

$$\mathbf{e}(t+1) = \tilde{\mathbf{P}}(\mathbf{H} \otimes \mathbf{I})\mathbf{e}(t), \tag{15}$$

where

$$\tilde{\mathbf{P}} = \begin{pmatrix} \tilde{\mathbf{P}}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \tilde{\mathbf{P}}_m \end{pmatrix}, \qquad \mathbf{H} = \begin{pmatrix} \frac{1}{d_1} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \frac{1}{d_m} \end{pmatrix} \mathbf{D},$$

with $\otimes$ denoting the Kronecker product.

The basic idea of the proof is to show that an infinite sequence of matrix products of the form $(\tilde{\mathbf{P}}(\mathbf{H} \otimes \mathbf{I}) \cdots \tilde{\mathbf{P}}(\mathbf{H} \otimes \mathbf{I}))$ converge to zero matrix exponentially fast. Let $\mathbf{M} = \tilde{\mathbf{P}}(\mathbf{H} \otimes \mathbf{I}) \cdots \tilde{\mathbf{P}}(\mathbf{H} \otimes \mathbf{I})$. $\mathbf{M} \in \mathbb{R}^{mn \times mn}$ is a $m \times m$ block matrix with the ijth entry $\mathbf{M}_{ij} \in \mathbb{R}^{n \times n}$. In alternating projection based algorithm, $\tilde{\mathbf{P}}_i$ is the product matrix of a series of projection matrix, which is a projection matrix polynomial. Since the set of projection matrix polynomials is closed under matrix addition and multiplication, it can be checked that $\mathbf{M}_{ij}$ is also a projection matrix polynomial.

By leveraging the Lemma 2 in [2], similar result to the Proposition 1 in [2] can be derived. That is, when the network topology is a connected graph, a sufficiently long sequence of matrix product in the form of $\mathbf{M}$ is a contraction in the mixed matrix norm, which implies the exponential convergence of the alternating projection based algorithm. ■

Remark. Recall that, with the help of matrix caching, the computational complexity of updating rule (7) is $\mathcal{O}(r_i n)$. The alternating projection based algorithm has the same per epoch computational complexity without the extra $\mathcal{O}(r_i n)$ space. Intuitively, projecting $\bar{\mathbf{x}}_i(t)$ in an alternating manner degrades the convergent performance of the consensus algorithm. It will be shown in Section VII-B that such degradation is negligible for certain types of LAE problem. In other words, compared to the algorithm with updating rule (7), the alternating projection based algorithm can bring storage savings with commensurate computational overhead.

Corollary 1. The proposed updating rule (9) is robust against the camouflage attack.

Proof. We have shown that, with updating rule (9), the deviation of $\mathbf{x}_i$ from $\mathbf{x}^*$ will converge to $\mathbf{0}$. Each node can start the alternating projection based algorithm with arbitrary initial value. Thus, the robustness of (9) can be proved by the same reasoning in Theorem 1. ■

V. Misbehavior Detection Mechanism

The algorithm proposed in Section IV is robust against the camouflage attack. However, a malicious agent may fully prevent the consensus process by refusing to update its local estimate or keep injecting arbitrary local values at every epoch. In this section, we investigate how to defend the consensus algorithm against such kind of divergence attack. The basic idea is to enhance the consensus algorithm with a monitoring mechanism that can detect those persistent injection attackers. Specifically, each node will monitor its neighbors' updates by double checking whether they obey the updating rule in (9). Notice that at $(t+1)$-th epoch, for node j to verify the local estimate $\mathbf{x}_i(t+1)$ of node i, node j requires the knowledge of $\mathbf{A}_i^{\mathrm{T}}$, b_i, and $\bar{\mathbf{x}}_i(t)$. $\mathbf{A}_i^{\mathrm{T}}$ and b_i can be acquired by node j during the network discovery or the problem distribution phase, in a secure manner (e.g., protected with digital signatures). If the correctness of $\bar{\mathbf{x}}_i(t)$ is also verifiable, node j can verify the integrity of $\mathbf{x}_i(t+1)$ by directly evaluating the equation (9). Therefore, the key problem in monitoring is to make sure that node i reports an authentic $\bar{\mathbf{x}}_i(t)$ in its $(t+1)$th update.

A. The Monitoring Structure

In the distributed computing framework, each agent will be monitored by all its neighbors. To facilitate the detection mechanism, a μTESLA-like protocol [37] is considered. All the monitoring neighbors share a one-way key chain[2], $MK = [s(T) \to s(T-1) \to \cdots \to s(0)]$, where $s(t-1)$ can be computed through $s(t)$ with a one-way function. The elements in the chain will be used in homomorphic signature operation as monitoring keys. We assume that a key distribution infrastructure is available for safely distributing the keys for information security related operations in the distributed computing framework. The assignment of key chains is illustrated in Fig. 2. All the neighbors of node i share the key chain MK_i, i.e., node j, k, l, m, n share MK_i. In this way, each node keeps a chain for each of its neighbors, i.e., node i keeps MK_j, MK_k, MK_l, MK_m, MK_n.

It is worth noting that each node plays double roles in the monitoring structure: while a node is being monitored by its neighbors, it holds monitoring keys to monitor those neighbors too. Let $s_i(t)$ to denote a homomorphic signature key for monitoring node i at epoch t. To facilitate the verification, along with $\bar{\mathbf{x}}_i(t)$, node i is supposed to broadcast a proof that the $\bar{\mathbf{x}}_i(t)$ is computed correctly. We here use node i as a tagged node to explain our monitoring methodology. The local estimates that node i receives from its neighbors will be signed by $s_i(t)$ according to the homomorphic signature. After collecting the updates,

[2]We assume that the system has a predefined maximum number of epochs, denoted by T.

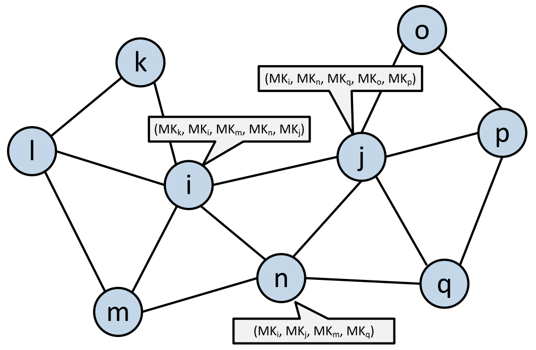

o
k
(MK_i, MK_n, MK_q, MK_o, MK_p)
p
(MK_k, MK_l, MK_m, MK_n, MK_j)
l
j
i
n
q
m
(MK_i, MK_j, MK_m, MK_q)

Fig. 2. Signature key structure.

node i can compute the summation of its neighbors' latest estimates and the corresponding signature without knowing $s_i(t)$. For monitoring, node i needs to report both the summation and the corresponding signature to all the monitoring neighbors. The protection provided by homomorphic signature is that if node i does not honestly compute the summation of local estimates from neighbors (which is required to obtain $\bar{\mathbf{x}}_i(t)$), it will not be able to forge the corresponding signature as node i does not have the current monitoring key. On the other hand, node i needs to have the capability to check whether the local estimates from its neighbors are signed properly. To this end, node i's neighbors are required to disclose the monitoring key used in previous epoch $s_i(t-1)$ to node i. Based on the idea of μTESLA protocol [38], node i can easily verify that $s_i(t-1)$ is a part of the key chain MK_i. Furthermore, with $s_i(t-1)$, node i can verify the validity of the signatures received in previous epoch. In this way, nodes that do not follow the signature scheme will be detected by a normal neighboring node. In the next subsection, we assume that all the signatures are legitimate and present the verification scheme with respect to the local estimates.

B. Monitoring by Homomorphic Signature

The proposed misbehavior detection mechanism is built upon a homomorphic signature algorithm introduced in [39]. By that algorithm, the signature of message addition equals to the multiplication of the individual signatures, say, $\text{Sig}(m_1 + m_2) = \text{Sig}(m_1) \times \text{Sig}(m_2)$. Specifically, the homomorphic signature key for monitoring node i is $s_i = (g, g_{i1}, g_{i2}, \ldots, g_{in})$. Note that g is a public number known to every node for the modular operation. With $s_i(t)$, node i's neighbor, say node j, can sign its local estimate $\mathbf{x}_j(t)$ as follows[3]

$$\text{Sig}_i(\mathbf{x}_j(t)) = \prod_{l=1}^{n} g_{il}^{x_j^l(t)} \mod g, \tag{16}$$

where $x_j^l(t)$ is the l-th component of $\mathbf{x}_j(t)$. Actually, node j needs to generate totally $d_j - 1$ signatures for its neighbors, where the signature for a certain neighbor $k \in \mathcal{N}_j$ is generated with the monitoring key $s_k(t)$. All these signatures will be sent out by node j at the end of the t-th epoch.

[3]For ease of presentation, we omit the index t in the parameters of the monitoring key since one-shot update is presented here.

At node i, let $\boldsymbol{\sigma}_i(t) = \sum_{j\in\mathcal{N}_i\setminus i} \mathbf{x}_j(t)$. At epoch $t+1$, after receiving $\text{Sig}_i(\mathbf{x}_j(t)), j \in \mathcal{N}_i$, node i can generate the signature of $\boldsymbol{\sigma}_i(t)$ by

$$\text{Sig}_i(\boldsymbol{\sigma}_i(t)) = \prod_{j\in\mathcal{N}_i\setminus i} \text{Sig}_i(\mathbf{x}_j(t)) \mod g \tag{17}$$

The integrity of $\boldsymbol{\sigma}_i(t)$ is assured by $\text{Sig}_i(\boldsymbol{\sigma}_i(t))$ and can be verified by any neighbor of node i that knows the monitoring key $s_i(t)$. To this end, the broadcast message of node i at each epoch, e.g, epoch $t+1$, contains its latest update, the signatures of this update, $\boldsymbol{\sigma}_i(t)$ it uses to update the local estimate, and the signature of $\boldsymbol{\sigma}_i(t)$ generated according to Eq. (17). We denote such a message by $\Theta_i(t+1)$,

$$\begin{aligned}\Theta_i(t+1) = \{&\mathbf{x}_i(t+1), \boldsymbol{\sigma}_i(t), \text{Sig}_i(\boldsymbol{\sigma}_i(t)),\\ &\{\text{Sig}_j(\mathbf{x}_i(t+1))||s_j(t), \forall j \in \mathcal{N}_i\setminus i\}\}.\end{aligned} \tag{18}$$

Upon receiving $\theta_i(t+1)$, the correctness of $\mathbf{x}_i(t+1)$ can be verified by the neighbors of node i via a two-step verification process:

(i) Check whether or not $\boldsymbol{\sigma}_i(t)$ has been honestly computed according to $\sum_{j\in\mathcal{N}_i\setminus i} \mathbf{x}_j(t)$ via $\text{Sig}_i(\boldsymbol{\sigma}_i(t))$. More precisely, $\boldsymbol{\sigma}_i(t)$ can pass this step if

$$\text{Sig}_i(\boldsymbol{\sigma}_i(t)) = \prod_{l=1}^{n} g_{il}^{\boldsymbol{\sigma}_i^l(t)} \mod g \tag{19}$$

holds, where $\boldsymbol{\sigma}_i^l(t)$ denotes the l-th component of $\boldsymbol{\sigma}_i(t)$.

(ii) If the integrity of $\boldsymbol{\sigma}_i(t)$ is validated, verify the correctness of $\mathbf{x}_i(t+1)$ by testing

$$\mathbf{x}_i(t+1) \overset{?}{=} \tilde{\mathbf{z}}_i + \tilde{\mathbf{P}}_i((\boldsymbol{\sigma}_i(t) + \mathbf{x}_i(t))/d_i). \tag{20}$$

Note that condition (19) holds if $\boldsymbol{\sigma}_i(t)$ is honestly computed, since condition (19) is equivalent to

$$\begin{aligned}\text{Sig}_i(\boldsymbol{\sigma}_i(t)) &= \prod_{j\in\mathcal{N}_i\setminus i} \text{Sig}_i(\mathbf{x}_j(t)) \mod g\\ &= \prod_{j\in\mathcal{N}_i\setminus i}\prod_{l=1}^{n} g_{il}^{x_j^l(t)} \mod g\\ &= \prod_{l=1}^{n} g_{il}^{(\sum_{j\in\mathcal{N}_i\setminus i} x_j^l(t))} \mod g\\ &= \prod_{l=1}^{n} g_{il}^{\boldsymbol{\sigma}_i^l(t)} \mod g.\end{aligned}$$

On the other hand, according to [39], the hardness of finding two different messages with the same signature is based on the hardness of the discrete logarithm in groups of prime order.

If $\Theta_i(t+1)$ does not pass the verification procedure launched by node j, then any future update from node i will be discarded by node j. Node j will also take the workload of node i to ensure the convergence of the distributed algorithm. That is, in addition to rows in $\boldsymbol{\Phi}_j$, rows in $\boldsymbol{\Phi}_i$ will be used by node j in the alternating projection operation to calculate its subsequent updates. In this way, estimates of normal nodes will converge to the correct solution as long as the induced subgraph formed by the normal nodes is connected. To pass the future verifications from its neighbors, node j is supposed to send the information of $\mathbf{\Phi}_i$ to its neighbors via a local broadcasting procedure. It is worth noting that while we mainly focus on algorithm design in this paper, a full development of all implementation details will be an interesting study to pursue.

Consider the robustness of the proposed consensus algorithm, it is unnecessary to find out all erroneous estimates, which indicates that the verification of $\mathbf{x}_i(t)$ can be conducted in a probabilistic manner. Therefore, the computation overhead incurred by the estimate validation can be further mitigated. Formally, a node will verify a received message from its neighbors with a preset verification probability p at each epoch. Let F denote the number of incorrect updates a malicious node can inject, indicating that, after updating F false estimates, messages from a malicious node will be blocked by all its normal neighbors. The cumulative distribution function of F is manifested as

$$\Pr(F \leq t) = (1-(1-p)^t)^d \tag{21}$$

where d is the number of normal neighbors. As t goes to infinity, the impact of a malicious node will be eliminated almost surely. Consider that the system works in a finite-time manner, T is supposed to be a large number such that not only the malicious nodes will be detected with high probability but also the nodes that behave normally have enough time to derive a solution with acceptable accuracy. In addition, the discrepancy between theoretical analysis and practical application can be mitigated by introducing an extension scheme. Roughly speaking, when the system runs out of time without obtaining an acceptable solution, another collection of key chains will be generated. All the normal nodes can then continue the consensus process with their current estimates for another T epochs.

Remark. The proposed detection mechanism is based on neighbor monitoring, which implicitly assumes that the neighboring malicious nodes cannot collude with each other. Otherwise, a malicious node i can obtain MK_i from its neighbor and forge the local estimates as well as signatures without being detected. A promising idea to cope with the collusion attacks is the common neighbor cross-validation [40], [41], where honest common neighbors are in charge of the legitimacy check of the data. However, incorporating such idea into the signature-based detection mechanism is a non-trivial task. We leave the misbehavior detection mechanism that can defend against collusion attacks as our future work.

C. Operation With Real Numbers

The applicability of existing homomorphic signature schemes are constrained to specific domains, e.g., finite field. With a sufficiently large field size, they can also be used to sign positive integer values. In practical applications, however, the computation results are normally real numbers. To cope with this, our method is to apply appropriate shifting and scaling to related real numbers, termed as mapping operation, so that the homomorphic signature computations are still conducted over nonnegative integers which are mapped from the original real values.

We assume that the values of numbers involved in the LAE are finite, falling in the range $[-M, M]$, with M being a large enough positive integer value. For the signature generation and verification, we can map all numbers to the nonnegative interval $[0, 2M]$ by adding M to each number. For the whole system, each node just needs to do such mapping for related values when it initially starts the consensus process, and then all following steps just normally follow the updating rule (9). When the consensus process converge, the original solution value can be obtained by subtracting M from the converged value. To handle the real values, we can employ the scaling approach [10], [42]–[44]. Specifically, let s be a positive integer and 10^s be the scaling factor. Given a real number x (after the M-mapping in our context), $\tilde{x} = \lfloor 10^s \times x \rfloor$ is used in the signature-related operations. It can be seen that the value s indicates the number of digits behind the decimal points that will be mapped to the integer value. With the scaling operation, each node is supposed to broadcast following message at each epoch

$$\begin{aligned}\tilde{\mathbf{\Theta}}_i(t+1) = \{&\mathbf{x}_i(t+1), \tilde{\boldsymbol{\sigma}}_i(t), \mathrm{Sig}_i(\tilde{\boldsymbol{\sigma}}_i(t)),\\ &\{\mathrm{Sig}_j(\tilde{\mathbf{x}}_i(t+1))||s_j(t), \forall j \in \mathcal{N}_i \backslash i\}\}\end{aligned} \tag{22}$$

where $\tilde{\boldsymbol{\sigma}}_i(t) = \sum_{j\in\mathcal{N}_i\backslash i} \tilde{\mathbf{x}}_j(t)$.

When one neighbor node receives the message, it will conduct the two-step verification operations as discussed in section V-B. It is not difficult to see that the integrity of $\tilde{\boldsymbol{\sigma}}$ can be checked by the homomorphic signature, as it just involves summation operation. Since the consensus updating is still based on the real numbers, for second-step verification, we need to scale down $\tilde{\boldsymbol{\sigma}}_i(t)$ by 10^{-s} to recover the real value. Here we need to consider the value deviation induced by the flooring operation in the scaling. Specifically, letting $\hat{\mathbf{x}}_i(t+1)$ denote the consensus updating value after scaling down $\tilde{\boldsymbol{\sigma}}_i(t)$, we have,

$$\hat{\mathbf{x}}_i(t+1) = \tilde{\mathbf{z}}_i + \tilde{\mathbf{P}}_i\left(\frac{(10^{-s})\tilde{\boldsymbol{\sigma}}_i(t) + \mathbf{x}_i(t)}{d_i}\right) \tag{23}$$

For normal nodes, the gap between $\mathbf{x}_i(t+1)$ and $\hat{\mathbf{x}}_i(t+1)$ is bounded. It can be seen that the difference between $(10^{-s})\tilde{x}$ and x is bounded above by 10^{-s}. By generalizing this result, we have

$$\left\|\frac{(10^{-s})\tilde{\boldsymbol{\sigma}}_i(t)}{d_i} - \frac{\boldsymbol{\sigma}_i(t)}{d_i}\right\|_\infty \leq 10^{-s} \tag{24}$$

Since orthogonal projection is a non-expansive operation, the above result leads to

$$\|\mathbf{x}_i(t+1) - \hat{\mathbf{x}}_i(t+1)\|_\infty \leq 10^{-s} \tag{25}$$

Condition (25) is supposed to be satisfied by messages from normal nodes, which therefore can be used as the criteria for the second step of the verification procedure. Condition (25) also indicates that a malicious node misreporting its local estimates with an error range of 10^{-s} will pass the verification procedure without being detected. However, the missed detection ratio can be well controlled by s.

VI. Consensus With Component Dropping

Recall that, within each epoch of the consensus algorithm, each node needs to receive several local estimates from neighbors to update its own local estimate. And by the end of that epoch, each node will broadcast its latest local estimate. When the dimension of the LAE grows, the communication overhead incurred by the algorithm may become too huge. A natural improvement to reduce the communication overhead is to let neighbors only exchange the most effective information. For instance, if the estimate vector is sparse, only transmitting its nonzero components along with their corresponding indices can lower the size of transmitted data. Such an intuition cannot be extended to non-sparse situation. In this section, however, we demonstrate that we can still exchange information in a sparse manner, even if most of the components of an estimate vector are non-zero.

A. Dropping Strategy

We consider a dropping strategy with which each node intentionally discards a portion of components when broadcasting its estimate. When we incorporate such dropping strategy into consensus, we need extra caution. Consider a tagged node with d_i neighbors. For the component k of $\bar{\mathbf{x}}_i(t)$, the tagged node receives information from $d_{ik}(\leq d_i)$ neighbors (as some neighbors dropped the k-th component of their local estimate). For distinct j and k, the values of d_{ij} and d_{ik} are probably different. In this situation, computing $\bar{\mathbf{x}}_i(t)$ with a standard manner gives an incorrect value and may ultimately damage the convergence of the consensus algorithm. More precisely, $\bar{\mathbf{x}}_i(t)$ which node i uses to update its local estimate at t-th epoch should be the component-wise average of the received estimates.

Formally, the dropping behavior of node j at epoch t can be characterized by a diagonal matrix $\mathbf{V}_j(t)$ with the k-th diagonal entry defined as follows:

$$v_{jk}(t) = \begin{cases} 0 & \text{if node } j \text{ drops its } k\text{th component at epoch } t \\ 1 & \text{otherwise} \end{cases}$$

When receiving the local estimate from its neighbor j with dropped components, node i can approximately reconstruct $\mathbf{x}_j(t)$ by padding zeros at the missing indices. In this way, node i receives $\mathbf{V}_j(t)\mathbf{x}_j(t)$ at epoch t. Let $d_{ik}(t) = \sum_{j\in\mathcal{N}_i\setminus i} v_{jk}(t) + 1$ denote the number of the kth components that node i receives at epoch t (including the one of its own). The calculation of $\bar{\mathbf{x}}_i(t)$ should be calibrated as follows,

$$\bar{\mathbf{x}}_i(t) = \mathbf{W}_i(t)\left(\sum_{j\in\mathcal{N}_i\setminus i} \mathbf{V}_j(t)\mathbf{x}_j(t) + \mathbf{x}_i(t)\right), \tag{26}$$

where $\mathbf{W}_i(t) = \mathrm{diag}(\frac{1}{d_{i1}(t)}, \dots, \frac{1}{d_{in}(t)})$.

Consider the toy LAE example of $\mathbf{Ix} = \mathbf{1}$, where $\mathbf{I} \in \mathbb{R}^3$ and $\mathbf{1}$ are identity matrix and all-ones vector, respectively. Obviously, $\mathbf{x}^* = \mathbf{1}$. Three agents with a complete graph topology solve this problem distributedly. Without loss of generality, let $\mathbf{x}_1(0) = [1, 1/2, 2]^T$, $\mathbf{x}_2(0) = [1/3, 1, 1/4]^T$, and $\mathbf{x}_3(0) = [10, 5, 1]^T$. While exchanging estimates, agent i only sends out the i-th component of its estimate. That is, $\mathbf{V}_1 = \mathrm{diag}(1, 0, 0)$, $\mathbf{V}_2 = \mathrm{diag}(0, 1, 0)$, and $\mathbf{V}_3 = \mathrm{diag}(0, 0, 1)$. With respect to the calculation of $\bar{\mathbf{x}}_i(t)$, agent 1 can exploit the received fragmented estimates as follows

$$\bar{\mathbf{x}}_1(t) = \begin{pmatrix} 1 & & \\ & \frac{1}{2} & \\ & & \frac{1}{2} \end{pmatrix} (\mathbf{V}_2\mathbf{x}_2(t) + \mathbf{V}_3\mathbf{x}_3(t) + \mathbf{x}_1(t)) \tag{27}$$

The other two agents can also apply similar approach to perform the consensus operation and then update their estimates according to the algorithm. One can check that, with this kind of dropping strategy, $\mathbf{x}^*$ can still be reached while only one third information is exchanged at each epoch. In general case, the agents may change their dropping patterns periodically, or even at each epoch, in order to converge to $\mathbf{x}^*$.

B. Theoretical Analysis on Dropping

Similar to the analysis in Theorem 2, we analyze the consensus algorithm with dropping by characterizing the dynamics of the deviation vector $\mathbf{e}(t)$. For node i,

$$\begin{aligned} \mathbf{e}_i(t+1) &= \mathbf{x}_i(t) - \mathbf{x}^* = \tilde{\mathbf{P}}_i(\bar{\mathbf{x}}_i(t) - \mathbf{x}^*) \\ &= \tilde{\mathbf{P}}_i\mathbf{W}_i(t)\left(\sum_{j\in\mathcal{N}_i\setminus i} \mathbf{V}_j(t)\mathbf{e}_j(t) + \mathbf{e}_i(t)\right) \end{aligned} \tag{28}$$

Therefore, we have $\mathbf{e}(t+1) = \tilde{\mathbf{P}}\mathbf{G}'(t)\mathbf{e}(t)$. Here $\mathbf{G}'(t) = [\mathbf{g}_{ij}(t)]_{m\times m}$ is an $mn \times mn$ matrix, with block entry $\mathbf{g}_{ij}(t)$ being an $n \times n$ matrix:

$$\mathbf{g}_{ii}(t) = \mathbf{W}_i(t); \quad \mathbf{g}_{ij}(t) = D_{ij}\mathbf{W}_i(t)\mathbf{V}_j(t) \quad \forall i \neq j$$

where D_{ij} is the corresponding entry in the adjacency matrix $\mathbf{D}$. $\mathbf{W}_i(t)$ represents the weights node i uses to average the estimates from its neighbors, which is determined by the dropping patterns of its neighbors.

With a fixed network topology, $\mathbf{G}'(t)$ is determined by the dropping patterns of the nodes. For arbitrary dropping patterns, $\mathbf{G}'(t)$ is a right stochastic matrix. Consider that $\|\mathbf{PG}'(t)\|$ is bounded above by a constant, say α. Ideally, if there exists an oracle controller which has the knowledge about α and assigns each node $(\mathbf{I} - \frac{\tilde{\mathbf{P}}_i}{\alpha'})\mathbf{x}^*$ as its invariant vector $\tilde{\mathbf{z}}_i$ in (9), where $\alpha' \geq \alpha$. Then all the nodes can obey the following rule to update their estimates

$$\mathbf{x}_i(t+1) = \tilde{\mathbf{z}}_i + \frac{1}{\alpha}\tilde{\mathbf{P}}_i\bar{\mathbf{x}}_i(t), \tag{29}$$

With (29), the deviation vector of node i can be expressed as

$$\begin{aligned}\mathbf{e}_i(t+1) &= \tilde{\mathbf{z}}_i + \frac{1}{\alpha}\tilde{\mathbf{P}}_i\bar{\mathbf{x}}_i(t) - \mathbf{x}^* \\ &= \frac{1}{\alpha}\tilde{\mathbf{P}}_i\bar{\mathbf{x}}_i(t) - \frac{1}{\alpha'}\tilde{\mathbf{P}}_i\mathbf{x}^* \\ &= \frac{1}{\alpha}\tilde{\mathbf{P}}_i(\bar{\mathbf{x}}_i(t) - \mathbf{x}^*) - \frac{\alpha'-\alpha}{\alpha'\alpha}\tilde{\mathbf{P}}_i\mathbf{x}^* \\ &= \frac{1}{\alpha}\tilde{\mathbf{P}}_i\mathbf{W}_i(t)\left(\sum_{j\in\mathcal{N}_i\setminus i}\mathbf{V}_j(t)\mathbf{e}_j(t) + \mathbf{e}_i(t)\right) - \mathbf{c}_i\end{aligned} \tag{30}$$

where $\mathbf{c}_i = \frac{\alpha'-\alpha}{\alpha'\alpha}\tilde{\mathbf{P}}_i\mathbf{x}^*$ is a constant vector. And let $\mathbf{c} = [\mathbf{c}_1^{\mathrm{T}}, \mathbf{c}_2^{\mathrm{T}}, \ldots, \mathbf{c}_m^{\mathrm{T}}]^{\mathrm{T}}$, we have

$$\mathbf{e}(t+1) = \frac{\mathbf{PG}'(t)}{\alpha}\mathbf{e}(t) - \mathbf{c} \tag{31}$$

Let $\lambda = \|\mathbf{PG}'(t)\|/\alpha$ and $\lambda < 1$, then

$$\limsup_{t\to\infty}\|\mathbf{e}(t)\| \leq \frac{1}{1-\lambda}\|\mathbf{c}\| \tag{32}$$

This implies that all the estimates will converge to a neighborhood of the exact solution, which can be controlled by the oracle through adjusting α'. Particularly, if $\alpha' = \alpha$, the final consensus will be $\mathbf{x}^*$.

C. Random Dropping

According to above analysis, driven by the initialization-free updating rule (29), estimates of all the nodes can converge to the exact solution in the scenario of components dropping. In practice, such an oracle controller is not available since each node cannot obtain $\tilde{\mathbf{z}}_i$ without explicitly knowing $\mathbf{x}^*$. If there exists a series of dropping patterns $\{\mathbf{V}_1(t), \mathbf{V}_2(t), \ldots, \mathbf{V}_m(t)\}, t = 1, 2, \ldots, k$, such that the spectral radius $\rho(\prod_{t=1}^{k}(\mathbf{PG}'(t))) < 1$, then each node can employ its corresponding pattern series cyclically to guarantee a final agreement via updating rule (9). However, finding out feasible series, if any, is nontrivial, particularly in distributed network settings. We notice through numerical calculations that random dropping is a promising approach. That is, each component in the estimate vector will be dropped independently with probability p_d. In our cases of numerical calculations, the number of epochs needed to reach the consensus will not increase very much even a big portion, to some extent, of elements in exchanged message vectors are dropped. This implies that such a random dropping strategy has the potential to maintain the convergence property of the consensus algorithm while reducing the communication overhead.

Remark. In our cases of numerical calculations, the consensus algorithm also converges when the missing components are padded with their most recent values and $\bar{\mathbf{x}}_i$ is calculated as usual. However, the empirical convergence performance of this approach is inferior to that achieved by approach of calibrating $\bar{\mathbf{x}}_i$, i.e., Eq. (26). Therefore, we focus on the later approach in this work.

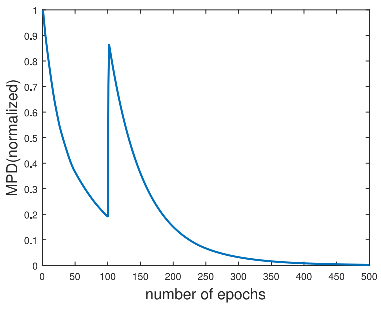
Regarding the random dropping mechanism, the calibrated $\bar{\mathbf{x}}_i$ can be considered as the noise-corrupted $\bar{\mathbf{x}}_i$ in the case that the whole estimates are exchanged. Intuitively, driven by the consensus algorithm, estimates of neighboring nodes get closer after an epoch. As the estimates get closer, the noise introduced by component dropping diminishes, which vanishes if all the estimates are exactly the same. To theoretically characterize the effects of random dropping, however, is very challenging. For one thing, the model that can bound the magnitudes of the noises is complicated, depending on not only the closeness of the estimates but also the dropping probability. For another, the dynamic equations with respect to the estimate errors, e.g., Eq. (14), need to be adapted carefully such that the dynamics of the noises are taken into account. We plan to explore the full analysis in our future work.

D. Rethinking the Detection Mechanism

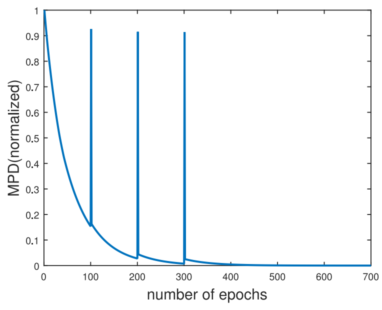
With the dropping strategy, $\bar{\mathbf{x}}_i(t)$ is evaluated in a component-wise manner according to Eq. (26). The misbehavior detection mechanism needs to be tailored accordingly to be compatible with the dropping strategy. Let $\mathbf{d}_i(t) = [d_{i1}(t), \ldots, d_{in}(t)]$. To verify the correctness of $\bar{\mathbf{x}}_i(t)$, not only the sum of neighbors' estimates $\sum_{j\in\mathcal{N}_i}\mathbf{V}_j(t)\mathbf{x}_j(t)$ but also the weight vector $\mathbf{d}_i(t)$ should be protected via the homomorphic signature scheme. For the former, at the end of an epoch, each node can directly apply the signature scheme to sign the fragment (due to component dropping) of its estimate and then broadcasts it out. The signature of $\sum_{j\in\mathcal{N}_i}\mathbf{V}_j(t)\mathbf{x}_j(t)$ should be equal to the product of the received fragments' signatures. The applicability of the homomorphic signature scheme described in Section V is based on the observation that padding zeros onto the missing components does not change the signature. Let $\mathbf{v}_j(t) = [v_{j1}(t), \ldots, v_{jn}(t)]$ be the indicator vector corresponding to the dropping pattern of node j at epoch t. Obviously, $\mathbf{d}_i(t)$ is a linear combination of $\mathbf{v}_j(t), \forall j \in \mathcal{N}_i$, i.e., $\mathbf{d}_i(t) = \sum_{j\in\mathcal{N}_i}\mathbf{v}_j(t)$. Along with the estimate, the nodes can broadcast the indicator vector at the end of each epoch and $\mathbf{d}_i(t)$ can be protected by the signature scheme similarly as above. It is worth noting that, although $\mathbf{v}_i(t)$ is a n-dimension vector, its size can be considered as negligible compared with a n-dimension estimate. This is because each component in $\mathbf{v}_i(t)$ is a binary value which is less than 2% of the size of a double-precision value, typically used to represent the component in the estimate vector. In this sense, the extra communication overhead introduced to the detection mechanism is negligible.

VII. Numerical Results

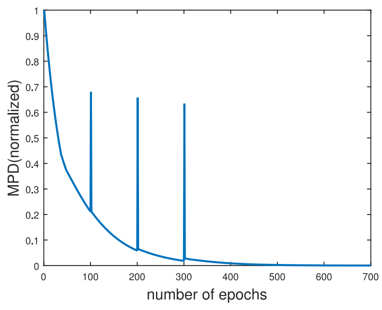
In this section, we evaluate the performance of the proposed schemes in terms of both robustness and efficiency using MATLAB simulations. We employ a PC with Intel i7 CPU of 3.6 GHz and 8 GB memory to run the consensus-based LAE solving process. Due to the hardware limit, we build a parallel implementation of the proposed algorithm in the case of only 5 nodes via the MATLAB parallel computing toolbox. The results are consistent with the serial implementation and we omit them due to space limit. In this sense, serial



(a) original algorithm

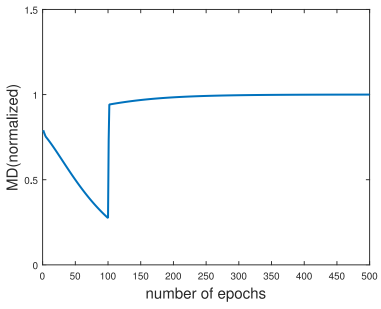


(b) plain robust consensus algorithm

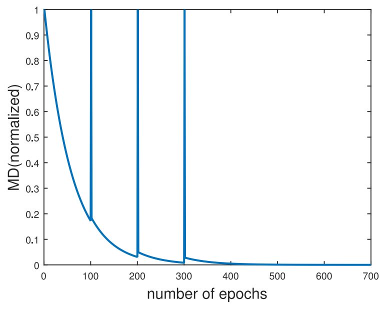


(c) alternating projection based algorithm

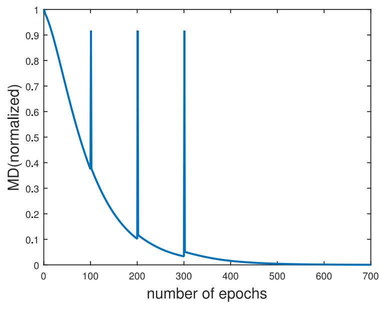
Fig. 3. MPD of all local estimates with different algorithms.



(a) original algorithm



(b) plain robust consensus algorithm



(c) alternating projection based algorithm

Fig. 4. MD of all local estimates with different algorithms.

implementations are applied to evaluate the performance of the proposed algorithm in solving different LAEs with different number of nodes. Recall that the algorithm with updating rule (7) is the plain generalization to the algorithm proposed in [18], we use it as a baseline for evaluating the alternating projection based algorithm in this section. We call it plain robust consensus algorithm. Throughout our calculations, rows of the LAE are distributed to the nodes evenly.

Theoretically, the consensus algorithm converges as long as $\mathbf{A}$ is nonsingular and the network topology is connected. The condition number of $\mathbf{A}$ and the network topology are two factors which influence the convergence rate. In our simulations, we notice that a randomly generated nonsingular $\mathbf{A}$ probably yields poor convergence rate. Diagonally dominant matrix is a type of matrix on which the consensus algorithm can achieve an acceptable convergence performance. It is worth noting that this type of matrix is also the target of the well-known Jacobi method [45] and arises in many real-world applications. In [18], we study the impact of network topology on the convergence performance of the proposed consensus algorithm using Erdős-Rényi (ER) random graphs. The alternating projection based algorithm produces similar results. Roughly speaking, for ER random graphs, the convergence performance will not degrade significantly as long as the adjacency probability is not too small, e.g., less than 0.2. Consider that this work focuses on the security issues of the consensus-based algorithm, analyzing which types of matrix are suitable to the proposed algorithm or an in-depth investigation of the impact of network topology on the convergence performance is out of scope of this work. In this section, all the algorithms are evaluated with diagonally dominant matrix of $\mathbf{A}$ and complete network graph. Random diagonally dominant matrices are generated via the method in [10] to construct the LAE. The size of an LAE problem is defined to be n.

A. *Robust Convergence*

As discussed in Section III-B, one-time false update which occurs on one node is able to deviate the outcome of all nodes to an incorrect one. In order to illustrate such kind of vulnerability as well as the robustness property of our proposed algorithms, we introduce two metrics to characterize the convergence process. The first is maximum pairwise difference (MPD), which is expressed as

$$\mathrm{MPD} = \max_{i \neq j} \|\mathbf{x}_i - \mathbf{x}_j\|_\infty \tag{33}$$

The second is maximum deviation (MD), which is expressed as

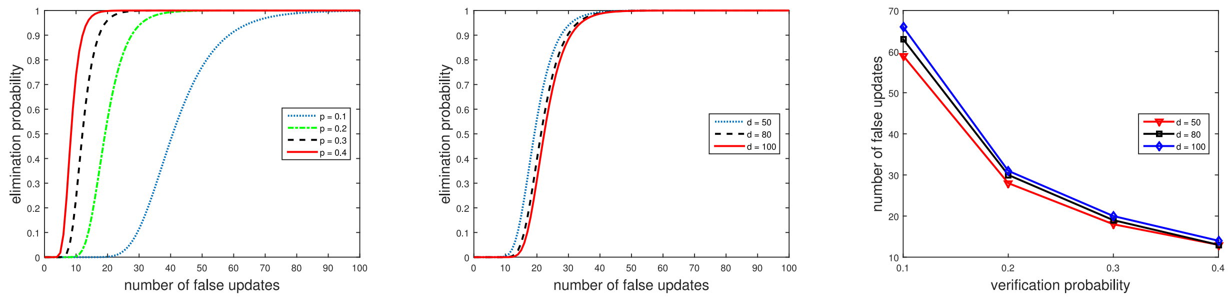
$$\mathrm{MD} = \max_{i} \|\mathbf{x}_i - \mathbf{x}^*\|_\infty \tag{34}$$

where $\mathbf{x}^*$ is the exact solution of the LAE.

We run different algorithms, including the original one in [1], the plain robust consensus algorithm and the alternating projection based algorithms, on an LAE with size 1000 over a 50-node complete network. False update is modeled as a zero vector. Normalized MPD and MD of all local estimates with these three algorithms are illustrated in Fig. 3 and Fig. 4, respectively. For the original algorithm, an aforementioned false update occurs at

TABLE I
NUMBER OF EPOCHS NEEDED TO REACH CONSENSUS

	problem size n	1000	2000	3000	4000	5000	8000	10000
plain robust consensus algorithm	50 nodes	270	270	286	289	301	324	336
	100 nodes	519	546	579	590	612	637	656
alternating projection based algorithm	50 nodes	270	271	288	289	301	324	336
	100 nodes	523	547	581	593	617	639	658



(a) elimination probability versus number of false updates under different verification probability (b) elimination probability versus number of false updates under different number of normal neighbors (c) times of false updates a malicious agent can conduct such that it will be eliminated with probability 0.9

Fig. 5. Performance of the misbehavior detection mechanism.

the 100-th epoch. As shown in Fig. 3(a), the algorithm can still reach consensus in the presence of that fault as the MPD decreases to zero eventually. However, the MD with original algorithm shown in Fig. 4(a) indicates that all local estimates converge to an incorrect solution other than $\mathbf{x}^*$. For the other two robust algorithms, three false updates are introduced by different nodes at different epochs. Fig. 3 and Fig. 4 show that our proposed algorithms tolerate these faults successfully. All local estimates agree on the same vector which is the exact solution.

In Fig. 3(a), although the false update cannot prevent nodes from reaching consensus, the injected error influences the progress of the consensus process. In contrast, the values of MPD decrease significantly after the attacks in the cases of robust algorithms, implying that effects of the attacks on the convergence process can be alleviated. This is because that errors of the estimates on node i can only influence $\bar{\mathbf{x}}_j, j \in \mathcal{N}_i$ in the robust algorithms. After averaging and the following projection operation, errors that propagate to the subsequent estimates decay significantly.

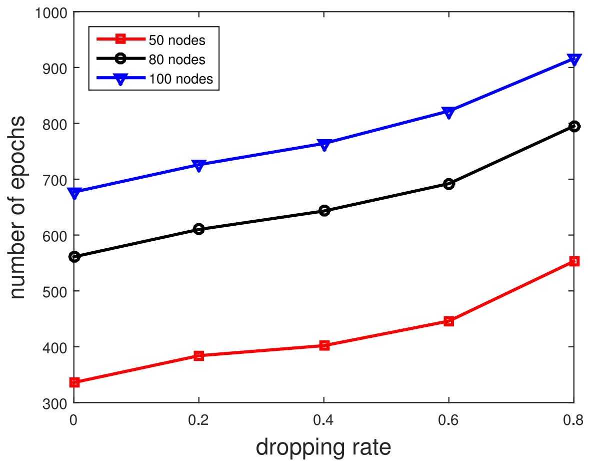
B. Algorithm Comparison

We employ the fault-tolerant algorithms described in Section IV to solve LAE with different size via different number of nodes. The number of epochs needed for each algorithm to reach consensus under different settings are summarized in Table I. Since the types of LAE (diagonally dominant matrix $\mathbf{A}$) and the network topology are fixed, the number of epochs required for the plain robust consensus algorithm is mainly determined by the number of participating nodes. It experiences a slow growth from 270 to 336 when the problem size increases from 1000 to 10000 in the 50-node case. These results are also consistent with the analysis with respect to the original algorithm in [2] that the upper bound of the convergence rate approaches to 1 as the number of agents increases.[4] As shown in Table I, although extra number of epochs are required to reach consensus, the alternating projection does not degrade the convergent performance significantly. For these LAE problems, the alternating projection based algorithm achieves commensurable convergent performance to the plain robust consensus algorithm while bringing extra storage savings.

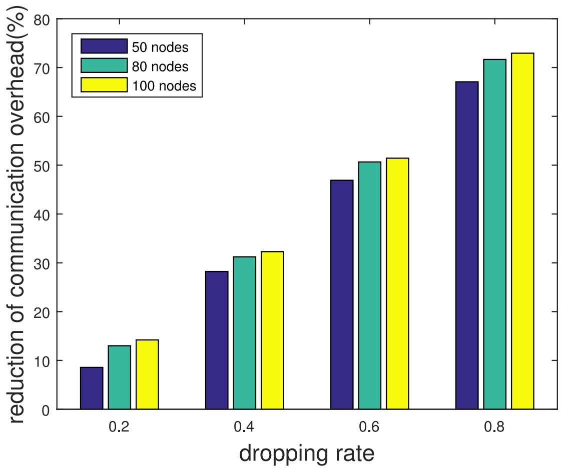
C. Misbehavior Detection

We evaluate the performance of our misbehavior detection mechanism by investigating the relationship between elimination probability of a malicious node and the number of false updates it can inject. Since a malicious node can be considered as being eliminated from the network if its message is blocked by all normal neighbors, the aforementioned relationship is characterized by Eq. (21). The results are shown in Fig. 5. That relationship under different verification probability is illustrated in Fig. 5(a), where the number of normal neighbors is fixed at 50. The performance in the case of $p = 0.1$ is significantly inferior to other cases. When $p = 0.2$, a malicious node would be eliminated almost surely after conducting 40 times of attack. For a higher detection probability, e.g. $p = 0.3$ and $p = 0.4$, almost no malicious node is able to update more than 20 times of false estimates. Fig. 5(b) shows the relationship under different number of normal neighbors, where the verification probability is fixed at 0.2. Performances in three cases are quite close. This result implies that, if the numbers of normal neighbors of malicious nodes do not vary too much, e.g., within the range from 50 to 100, the numbers of attacks the malicious nodes can conduct are almost the same. Results in Fig. 5(c) further validates this

[4]Without false updates, convergent dynamics of the original algorithm and the plain robust consensus algorithm are exactly the same.



(a) convergence performacne



(b) reduction of communication overhead

Fig. 6. Impact of random dropping.

conjecture. It illustrates the times of false updates a malicious node can conduct such that it will be eliminated with probability 0.9. The elimination probability is mainly determined by the verification probability. For a fixed p, the detection mechanism achieves similar performance. Therefore, although a malicious node can degrade the detection performance by increasing d, if possible, such advantage has marginal impact on the monitoring mechanism.

D. *Impact of Random Dropping*

We simulate the case where an LAE with size 10000 is solved by multiple nodes and homogeneous random dropping strategy is applied by these nodes. Epochs required for the alternating projection based algorithm under different dropping rate is shown in Fig. 6(a). It shows that applying the random dropping strategy will not slow down the consensus process significantly. For example, in the case of 100 nodes, number of epochs that the algorithm takes to reach consensus increases from 677 to 916 as the dropping rate increases from 0 to 0.8. With dropping rate 0.8, roughly 80% data transmission can be reduced per epoch at the expense of 35% extra epochs to reach the consensus. As shown in Fig. 6(b), more communication overhead can be reduced with higher dropping rate. An approximate 73% reduction in terms of communication overhead can be achieved when the dropping each component with probability 0.8. It is worth noting that the benefit of random dropping vanishes when the dropping components too aggressively. For example, when the dropping rate equals to 0.9, the convergence performance of the algorithm degrades significantly, i.e., an agreement cannot be reached after 10000 epochs, which results in higher communication overhead than the case without dropping.

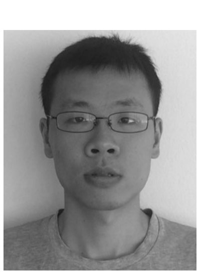
VIII. Conclusion

In this paper, we have proposed a distributed computing framework to securely solve LAE in a networked system which consists of multiple agents. The proposed framework consists of fault-tolerant consensus-based algorithm and misbehavior detection mechanism. An efficient robust consensus algorithm based on alternating projection are developed to guarantee the correctness of final consensus while a cooperative verification mechanism is designed to detect the false updates from malicious nodes. In addition, a random dropping strategy is introduced to reduce the communication overhead involved in the consensus process. The efficiency and robustness of our proposed framework are validated by performance analysis as well as numerical results.

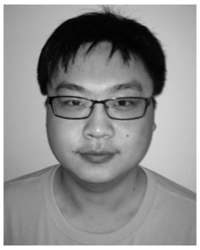
References

[1] S. Mou and A. Morse, "A fixed-neighbor, distributed algorithm for solving a linear algebraic equation," in *Proc. IEEE Eur. Control Conf.*, 2013, pp. 2269–2273.
[2] S. Mou, J. Liu, and A. S. Morse, "A distributed algorithm for solving a linear algebraic equation," *IEEE Trans. Autom. Control*, vol. 60, no. 11, pp. 2863–2878, Nov. 2015.
[3] K. You, S. Song, and R. Tempo, "A networked parallel algorithm for solving linear algebraic equations," in *Proc. IEEE Conf. Decis. Control*, 2016, pp. 1727–1732.
[4] G. Shi, B. D. Anderson, and U. Helmke, "Network flows that solve linear equations," *IEEE Trans. Autom. Control*, vol. 62, no. 6, pp. 2659–2674, Jun. 2017.
[5] S. Mou, Z. Lin, L. Wang, D. Fullmer, and A. S. Morse, "A distributed algorithm for efficiently solving linear equations and its applications (special issue JCW)," *Syst. Control Lett.*, vol. 91, pp. 21–27, 2016.
[6] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multiagent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.
[7] A. Nedic, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Trans. Autom. Control*, vol. 55, no. 4, pp. 922–938, Apr. 2010.
[8] J. Liu, S. Mou, and A. S. Morse, "An asynchronous distributed algorithm for solving a linear algebraic equation," in *Proc. IEEE Conf. Decis. Control*, 2013, pp. 5409–5414.
[9] D. M. Shila, W. Shen, Y. Cheng, X. Tian, and X. S. Shen, "Amcloud: Toward a secure autonomic mobile ad hoc cloud computing system," *IEEE Wireless Commun.*, vol. 24, no. 2, pp. 74–81, Apr. 2017.
[10] C. Wang, K. Ren, J. Wang, and Q. Wang, "Harnessing the cloud for securely outsourcing large-scale systems of linear equations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1172–1181, Jun. 2013.
[11] S. Salinas, C. Luo, X. Chen, and P. Li, "Efficient secure outsourcing of large-scale linear systems of equations," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 1035–1043.
[12] X. Chen, X. Huang, J. Li, J. Ma, W. Lou, and D. S. Wong, "New algorithms for secure outsourcing of large-scale systems of linear equations," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 1, pp. 69–78, Jan. 2015.

[13] H. J. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram, "Resilient asymptotic consensus in robust networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 4, pp. 766–781, Apr. 2013.
[14] H. J. LeBlanc and F. Hassan, "Resilient distributed parameter estimation in heterogeneous time-varying networks," in *Proc. ACM Int. Conf. High Confidence Netw. Syst.*, 2014, pp. 19–28.
[15] J. He, P. Cheng, L. Shi, and J. Chen, "SATS: Secure average-consensus-based time synchronization in wireless sensor networks," *IEEE Trans. Signal Process.*, vol. 61, no. 24, pp. 6387–6400, Dec. 2013.
[16] S. Sundaram and B. Gharesifard, "Distributed optimization under adversarial nodes," 2016, *arXiv:1606.08939*.
[17] N. H. Vaidya and V. K. Garg, "Byzantine vector consensus in complete graphs," in *Proc. ACM Symp. Princ. Distrib. Comput.*, 2013, pp. 65–73.
[18] W. Shen, B. Yin, X. Cao, Y. Cheng, and X. S. Shen, "A distributed secure outsourcing scheme for solving linear algebraic equations in ad hoc clouds," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 415–430, Apr.-Jun. 2019.
[19] P. J. Eberlein and H. Park, "Efficient implementation of jacobi algorithms and jacobi sets on distributed memory architectures," *J. Parallel Distrib. Comput.*, vol. 8, no. 4, pp. 358–366, 1990.
[20] J. Götze, "On the parallel implementation of jacobi and kogbetliantz algorithms," *SIAM J. Scientific Comput.*, vol. 15, no. 6, pp. 1331–1348, 1994.
[21] M. J. Atallah, K. N. Pantazopoulos, J. R. Rice, and E. E. Spafford, "Secure outsourcing of scientific computations," *Advances Comput.*, vol. 54, pp. 215–272, 2002.
[22] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.
[23] F. Pasqualetti, A. Bicchi, and F. Bullo, "Distributed intrusion detection for secure consensus computations," in *Proc. IEEE Conf. Decis. Control*, 2007, pp. 5594–5599.
[24] S. Sundaram and C. N. Hadjicostis, "Distributed function calculation via linear iterative strategies in the presence of malicious agents," *IEEE Trans. Autom. Control*, vol. 56, no. 7, pp. 1495–1508, Jul. 2011.
[25] S. Sundaram and B. Gharesifard, "Consensus-based distributed optimization with malicious nodes," in *Proc. IEEE Annu. Allerton Conf. Commun., Control, Comput.*, 2015, pp. 244–249.
[26] N. H. Vaidya, "Iterative byzantine vector consensus in incomplete graphs," in *Proc. Int. Conf. Distrib. Comput. Netw.*, 2014, pp. 14–28.
[27] D. Charles, K. Jain, and K. Lauter, "Signatures for network coding," in *Proc. IEEE Annu. Conf. Inf. Sci. Syst.*, 2006, pp. 857–863.
[28] D. Boneh, D. Freeman, J. Katz, and B. Waters, "Signing a linear subspace: Signature schemes for network coding," in *Proc. Int. Workshop Public Key Cryptography*, 2009, pp. 68–87.
[29] R. Gennaro, J. Katz, H. Krawczyk, and T. Rabin, "Secure network coding over the integers," in *Proc. Int. Workshop Public Key Cryptography*, 2010, pp. 142–160.
[30] R. Stoleru, H. Wu, and H. Chenji, "Secure neighbor discovery and wormhole localization in mobile ad hoc networks," *Ad Hoc Netw.*, vol. 10, no. 7, pp. 1179–1190, 2012.
[31] M. Fiore, C. E. Casetti, C.-F. Chiasserini, and P. Papadimitratos, "Discovery and verification of neighbor positions in mobile ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 2, pp. 289–303, Feb. 2013.
[32] V. Yadav and M. V. Salapaka, "Distributed protocol for determining when averaging consensus is reached," in *Proc. IEEE Annu. Allerton Conf.*, 2007, pp. 715–720.
[33] J. Cortés, "Distributed algorithms for reaching consensus on general functions," *Automatica*, vol. 44, no. 3, pp. 726–737, 2008.
[34] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, 2011, Art. no. 13.
[35] X. Wang, S. Mou, and D. Sun, "Improvement of a distributed algorithm for solving linear equations," *IEEE Trans. Ind. Electron.*, vol. 64, no. 4, pp. 3113–3117, Apr. 2017.
[36] R. Escalante and M. Raydan, *Alternating Projection Methods*. Philadelphia, PA, USA: SIAM, 2011.
[37] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "Spins: Security protocols for sensor networks," *Wireless Netw.*, vol. 8, no. 5, pp. 521–534, 2002.
[38] A. Perrig, D. Song, R. Canetti, J.D Tygar, and B. Briscoe, "Timed efficient stream loss-tolerant authentication (TESLA): Multicast source authentication transform introduction," IETF RFC4082, June 2005.
[39] M. Bellare, O. Goldreich, and S. Goldwasser, "Incremental cryptography: The case of hashing and signing," in *Proc. Annu. Int. Cryptology Conf.*, 1994, pp. 216–233.
[40] H. Chan, A. Perrig, and D. Song, "Secure hierarchical in-network aggregation in sensor networks," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2006, pp. 278–287.
[41] Q. Yan, M. Li, T. Jiang, W. Lou, and Y. T. Hou, "Vulnerability and protection for distributed consensus-based spectrum sensing in cognitive radio networks," in *Proc. IEEE Conf. Comput. Commun.*, 2012, pp. 900–908.
[42] P.-A. Fouque, J. Stern, and G.-J. Wackers, "Cryptocomputing with rationals," in *Financial Cryptography*. New York, NY, USA: Springer, 2003, pp. 136–146.
[43] C. Orlandi, A. Piva, and M. Barni, "Oblivious neural network computing via homomorphic encryption," *EURASIP J. Inf. Secur.*, vol. 7, pp. 1–11, 2007.
[44] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Manual for using homomorphic encryption for bioinformatics," *Proc. IEEE*, vol. 105, no. 3, pp. 552–567, Mar. 2017.
[45] Y. Saad, *Iterative Methods for Sparse Linear Systems*, vol. 82. Philadelphia, PA, USA: SIAM, 2003.



Bo Yin (S'14) received the B.E. degree in electronic information engineering and the M.E. degree in electronic science and technology from Beihang University, Beijing, China, in 2010 and 2013, respectively. He is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA. His research interests include edge computing, wireless networking, and network security.



Wenlong Shen (S'13) received the B.E. degree in electrical engineering from Beihang University, Beijing, China, in 2010, and the M.S. degree in telecommunication from the University of Maryland, College Park, MD, USA, in 2012. He is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA. His current research interests include wireless networking, cloud computing, and network security.

Xianghui Cao (S'08–M'11–SM'16) received the B.S. and Ph.D. degrees in control science and engineering from Zhejiang University, Hangzhou, China, in 2006 and 2011, respectively. From July 2012 to July 2015, he was a Senior Research Associate with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA. Currently, he is an Associate Professor with the School of Automation, Southeast University, Nanjing, China. His research interests include cyber-physical systems, wireless network performance analysis, wireless networked control, and network security. He is the Organization Co-Chair for The Youth Academic Annual Conference of Chinese Association of Automation (YAC 2018), the Symposium Co-Chair for ICNC 2017, the Publicity Co-Chair for ACM MobiHoc 2015 and IEEE/CIC ICCC 2015, and the TPC Member for a number of conferences. He is also an Associate Editor for *ACTA Automatica Sinica*, the IEEE/CAA JOURNAL OF AUTOMATICA SINICA, and the *International Journal of Ad Hoc and Ubiquitous Computing*. He was the recipient of the Best Paper Runner-Up Award from ACM MobiHoc in 2014 and the First Prize of Natural Science Award of Ministry of Education of China in 2017.

Yu Cheng (S'01–M'04–SM'09) received the B.E. and M.E. degrees in electronic engineering from Tsinghua University, Beijing, China, in 1995 and 1998, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2003. He is now a Full Professor with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA. His research interests include wireless network performance analysis, network security, big data, and cloud computing. He was the Symposium Co-Chair for the IEEE ICC and the IEEE GLOBECOM, and the Technical Program Committee Co-Chair for WASA 2011 and ICNC 2015. He is a founding Vice Chair of the IEEE ComSoc Technical Subcommittee on Green Communications and Computing. He was an IEEE ComSoc Distinguished Lecturer during 2016–2017. He is an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. He was the recipient of the Best Paper Award at QShine 2007, IEEE ICC 2011, and a Runner-Up Best Paper Award at ACM MobiHoc 2014. He was also the recipient of the National Science Foundation CAREER Award in 2011 and IIT Sigma Xi Research Award in the junior faculty division in 2013.

Qing Li (M'02) is a Vice President of Engineering with the Network Protection Products business unit, Symantec, Mountain View, CA, USA. He is an Industry Veteran with more than 20 years of experience, with 24 issued patents and many more pending. Prior to the acquisition by Symantec, he was the Chief Scientist with Blue Coat Systems, where he is a well-known V1.0 technology and product innovator. He is a published author of five first-of-its-kind books, by Springer-Verlag, Morgan Kaufmann, and Wiley. His books have been translated into multiple foreign languages and are serving as reference texts in universities and in the industry around the world.