UNIVERSITY OF CALIFORNIA

Los Angeles

Towards Resilience in Cyber-Physical Systems

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Electrical and Computer Engineering

by

Yanwen Mao

2022

ABSTRACT OF THE DISSERTATION

Towards Resilience in Cyber-Physical Systems

by

Yanwen Mao
Doctor of Philosophy in Electrical and Computer Engineering
University of California, Los Angeles, 2022
Professor Paulo Tabuada, Chair

In this dissertation, we consider the problem of collaborating a network of nodes to accomplish an objective, which could be monitoring the state of a dynamical system or training a machine learning model, where each node only has access to partial data. Some nodes in the network are assumed to be attacked by an adversary. The attacked nodes may release incorrect information or, in the worst case, deviate from the prescribed rule and behave in an unexpected manner. This problem lies in the intersection of control theory, signal processing, machine learning, and network coding theory.

This problem is of great interest due to its wide applications in real life, such as monitoring the energy flow of a power grid, localizing a drone with multiple cameras, and machine learning in a decentralized network. Also, with the growing size of networked systems, it is getting harder and harder to physically shield network entities from adversarial attacks, which calls for a solution to the aforementioned problem even in the presence of adversaries.

In the first part of the dissertation, we study the problem of how to reconstruct the state of a linear system using partially corrupted state observations from heterogeneous sensors. In this part, we first discuss the computational complexity of this problem. We point out that

although this problem is, in general, NP-hard, it allows a polynomial-time solution if certain conditions are met. We then extend our study on the same state-reconstruction problem, but in a decentralized network, where we focus on how the network topology and the dynamics of the linear systems affect the maximum number of correctable attacked nodes and attacked communication channels. In the end, we propose a novel approach, based on source coding and dynamic average consensus algorithms, that enables each node in the network to track the state of a linear system using minimal communications.

In the second part of the dissertation, we switch to the decentralized machine learning problem. We consider a collection of nodes connected through a network, each equipped with a local data set. The objective for all the nodes is to collectively train a machine learning model that minimizes the empirical loss, in a decentralized manner, i.e., each node can only use its local function and messages exchanged with nodes it is connected to. Moreover, each node is to agree on the said minimizer despite an adversary that can arbitrarily change the local functions of a fraction of the nodes. To solve this problem, we propose a novel decentralized learning algorithm that enables all nodes to reach consensus on the optimal model, by identifying attacked nodes and filtering out erroneous messages.

The dissertation of Yanwen Mao has been approved.

Shreyas Sundaram

Christina Panagio Fragouli

Suhas N. Diggavi

Paulo Tabuada, Committee Chair

University of California, Los Angeles

2022

To my family

To my parents, Lixin and Yanping,

To those who I loved . . .

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The past five years has witnessed my growth, from a bold and ambitious student, to a sober and steady researcher. I would like to express my gratitude to everyone who helped me and supported me through this challenging part of my life.

I am deeply indebted to my advisor, Professor Paulo Tabuada, for his knowledge, patience, and support throughout this period, and without whom this dissertation would not have been possible. I would also like to express my deepest gratitude to Professor Suhas Diggavi, for his guidance and persistent help in this long journey.

My gratitude should also go to Professor Christina Fragouli and Professor Shreyas Sundaram for serving on my doctoral committee, for leading me through my first steps in my doctoral research, and whose feedback was imperative to building this dissertation.

Many thanks to my lab mates and collaborators: Luigi Pannocchi, Carlos Murguia, Marcus Lucas, Tzanis Anevlavis, Lucas Fraile, Alimzhan Sultangazin, Jonathan Bunton, Aritra Mitra, and Deepesh Data. I would like to extend my thanks to Jing Liu and Nirupam Gupta for their insightful discussions.

Last but not least, I would like to thank my family: my parents, Lixin and Yanping, for their unwavering support and belief in me.

VITA

2013-2017 B.S., Electrical Engineering
Shanghai Jiao Tong Univeristy

2017-2022 Research Assistant, Electrical and Computer Engineering
University of California, Los Angeles

PUBLICATIONS

Yanwen Mao, Paulo Tabuada, *Decentralized Secure State-Tracking in Multi-Agent Systems*, conditionally accepted by IEEE Transactions on Automatic Control.

Yanwen Mao, Aritra Mitra, Shreyas Sundaram, Paulo Tabuada, *On the Computational Complexity of the Secure State-Reconstruction Problem*, Automatica.

Yanwen Mao, Deepesh Data, Suhas Diggavi, Paulo Tabuada, *Decentralized Learning Robust to Data Poisoning Attacks*, Accepted by 61th Conference on Decision and Control (CDC'22).

Yanwen Mao, Paulo Tabuada *Decentralized resilient state-tracking*, 60th Conference on Decision and Control (CDC'21).

Yanwen Mao, Suhas Diggavi, Christina Fragouli, Paulo Tabuada *Secure State-Reconstruction Over Networks Subject to Attacks*, Joint publication in the IEEE Control Systems Letters and 59th Conference on Decision and Control (CDC'20).

Yanwen Mao, Aritra Mitra, Shreyas Sundaram, Paulo Tabuada, *When is the Secure State-Reconstruction Problem NP-hard?*, 58th Conference on Decision and Control (CDC'19).

CHAPTER 1

Introduction

Cyber-Physical Systems (CPSs), which merge computing and networking with physical entities, bring numerous advantages to human life, such as safer and more efficient systems and lower cost in interacting with the physical world. However, the complication in the nature of a CPS makes it vulnerable to adversarial attacks. Notorious examples include the StuxNet computer worm and the most recent credential stuffing attack reported by state farm keeps reminding people the importance of the security aspect of CPSs.

However, compared with the significance of CPS security, it is usually impractical to physically shield CPS entities from adversarial attacks, both because of the heterogeneity of CPS entities and their nature of being geographically dispersed over large areas. This motivates people to design a more robust algorithm, which, in the presence of adversarial attacks, still enables all attack-free entities in a CPS to collectively perform an assigned task.

In this dissertation, we provide a systematic analysis on how to design fault-tolerance algorithms for large-scale networked systems and a more fundamental characterization on whether a system can operate properly in the presence of a certain number of attacks. We first illustrate some key notions in this dissertation.

1.1 Key Notions

1.1.1 Dynamical Systems and Cyber-Physical Systems

In this dissertation, a dynamical system refers to a system whose state evolution over time and whose state observation are governed by explicit functions. In particular, if such functions are linear, the dynamical system is referred to as a linear dynamical system. Such systems have found numerous applications in real life, from inverted pendulum to social network. Studying dynamical systems is always the focus of control theory and has brought countless benefits to the human society, for example, autonomous driving and controlling the spread of disease.

Cyber-Physical Systems merge computing and networking with physical entities, and is thus modeled as dynamical systems in this dissertation. However, different from ordinary dynamical systems, CPSs are usually geographically dispersed and subject to attacks, and hence it is of great interest to develop fault-tolerant algorithms specifically for different kinds of CPSs.

1.1.2 Attacks

In this dissertation, attacks refer to the inferences from a third-party agent on a networked system (or a CPS) which render attacked nodes to misbehave. There are two kinds of attacks: benign and malicious. A benign attack is usually conducted unintentionally or even enforced by nature power, such as corrosion and aging. On the contrary, a malicious attack is conducted intentionally, which is much harder to handle due to the following three reasons: the attacker will likely exploit his knowledge of the system to design the attack signal, the attack will last much longer, and all attacked nodes will likely to cooperate. In this dissertation, We study the malicious attack when the attacker has full knowledge of the system and is able to attack a fixed subset of nodes in the network. An attacked node

may provide incorrect information, or may, in the worst case, totally deviate from prescribed rules. We point out here that any algorithm developed in this dissertation also applies to the benign case.

1.1.3 Machine Learning in the Presence of Data Poisoning Attacks

Machine learning has wide applications in human life. However, this modern technology is facing more challenges nowadays, and one of them is the need of designing machine learning algorithms that is robust to partially corrupted data samples, as machine learning is applied to increasingly sensitive tasks.

Consider a scenario where fifty hospitals in a state are collectively training a model to diagnose with radiograph if a certain patient has lung cancer. However, one hospital uploaded radiographs associated with break bones by mistake. Such mistakes are not easy to identify but will ruin the training process.

Such kind of scenarios call for robust machine learning algorithms which makes the system resilient against attacks. In response for this, we study how to train a nearly-optimal machine learning model in the presence of attacks is discussed in this dissertation.

1.2 Dissertation Outline and Contributions

This dissertation is composed of four research topics. The contribution of this dissertation is summarized as follows:

1.2.1 On the Computational Complexity of the Secure State-Reconstruction Problem

In Chapter 2, we reconstruct the state of a linear system at a central node using partially corrupted state observations from heterogeneous sensors geographically dispersed over large

areas. In this part, we systematically characterize the computational complexity of the problem mentioned above, which is usually referred to as the secure state-reconstruction problem. The first result establishes that this problem is, in general, NP-hard. We then identify classes of subproblems that can be solved in polynomial time. When there are at most s malicious sensors, the problem can be solved in polynomial time when each eigenvalue is observable by at least $2s + 1$ sensors. When each eigenvalue has geometric multiplicity one, this condition is equivalent to the system being $2s$-sparse observable. We also give a detailed discussion on the more nuanced case when each eigenvalue is not observable by at least $2s + 1$ sensors.

1.2.2 Secure State-Reconstruction Over Networks Subject to Attack

Whereas most work on secure state-reconstruction problem focuses on attacks on sensors, in Chapter 3 we consider the more challenging case where attacks occur on sensors as well as on nodes and links of a network that transports sensor measurements to a receiver. In this chapter we provide necessary and sufficient conditions for the secure state-reconstruction problem to be solvable in the presence of attacks on sensors and on the network.

1.2.3 Decentralized Secure State-Tracking in Multi-Agent Systems

Chapter 4 studies the problem of decentralized state-tracking in the presence of sensor attacks. We consider a network of nodes where each node has the objective of tracking the state of a linear dynamical system based on its measurements and messages exchanged with neighboring nodes notwithstanding some measurements being spoofed by an adversary. We propose a novel decentralized attack-resilient state-tracking algorithm based on the simple observation that a compressed version of all the network measurements suffices to reconstruct the state. This motivates a 2-step solution to the decentralized secure state-tracking problem: (1) each node tracks the compressed version of all the network measurements, and

(2) each node asymptotically reconstructs the state from the output of step (1). We prove that, under mild technical assumptions, our algorithm enables each node to track the state of the linear system and thus solves the decentralized secure state-tracking problem.

1.2.4 Decentralized Robust Optimization

In Chapter 5, we switch our focus, from reconstructing the state of a dynamical system with partially corrupted measurements from a network of sensors, to the decentralized machine learning problem in a hostile environment. In this problem, we consider a collection of nodes connected through a network, each equipped with a local function. These nodes are asked to collaboratively find out the global optimizer, i.e., the point that minimizes the aggregated local functions, using their local information and messages exchanged with their neighbors. Moreover, each node is to agree on the said minimizer despite an adversary that can arbitrarily change the local functions of a fraction of the nodes. We present RAGD, the Resilient Averaging Gradient Descent algorithm, a decentralized, consensus+outlier filtering algorithm that is resilient to attacks on local functions. We demonstrate that, as long as the portion of attacked nodes does not exceed a given threshold, RAGD guarantees that all nodes will be able to have a good estimate of the said minimizer.

1.3 Notations

In this section we introduce the notions used throughout the dissertation. Some additional notions, if necessary, will be introduced at the beginning of each chapter.

1.3.1 Basic Notions

We denote by $|S|$ the cardinality of a set S. For any two sets S and S', the set subtraction $S \backslash S'$ is the set defined by $S \backslash S' = \{s \in S | s \notin S'\}$. For matrices $Q_{i_1}, \ldots, Q_{i_p}$ over the same

field and with the same number of columns, we define the matrix $Q_{\mathcal{I}} = \left[Q_{i_1}^T | Q_{i_2}^T | \dots | Q_{i_p}^T\right]^T$ by stacking the individual matrices vertically.

Let $\mathbb{R}$, $\mathbb{N}$, $\mathbb{Q}$, and $\mathbb{C}$ denote the set of real, natural, rational, and complex numbers, respectively. A complex number $z \in \mathbb{C}$ is written in the form $z = a + b\mathbf{i}$ where $\mathbf{i}^2 = -1$. The support of $v \in \mathbb{R}^p$, denoted by $\text{supp}(v)$, is the set of indices of the non-zero entries of v, i.e., $\text{supp}(v) = \{i \in \{1, 2, \dots, p\} | v_i \neq 0\}$. For a scalar $s \in \mathbb{N}$ we say v is $s-$sparse if $|\text{supp}(v)| \leq s$. Also, we define the all-ones vector $\mathbf{1}_n = (1, 1, \dots, 1)^T$ and I_n to be the identity matrix of order n. When the dimension of the identity matrix is unambiguous, we will write I instead. Moreover, for any $p \in \mathbb{N}$, we denote by $\mathbf{E}_p = \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_p\}$ the standard basis of $\mathbb{R}^p$ with $\mathbf{e}_i \in \mathbb{R}^p$ being the vector with all entries equal to zero except the i-th entry which is 1.

Given a vector $\mathbf{b} \in \mathbb{R}^n$, we denote by $\|\mathbf{b}\|_0$ the number of non-zero entries in $\mathbf{b}$.

1.3.2 Matrix Related Notions

We denote by $A \in \mathbb{R}^{m \times n}$ a real matrix with $m \in \mathbb{N}$ rows and $n \in \mathbb{N}$ columns. We will also refer to matrices where only the number of rows or columns is specified using the notation $A \in \mathbb{R}^{m \times *}$ or $A \in \mathbb{R}^{* \times n}$.

For a real square matrix A, We denote by $\lambda_1, \dots, \lambda_r \in \mathbb{C}$ the (counted without repetition) eigenvalues of A and by $sp(A) = \{\lambda_1, \dots, \lambda_r\}$ its spectrum. The eigenvalue with the largest and smallest magnitude is denoted by $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$, respectively. The algebraic multiplicity of an eigenvalue λ_j, denoted by $\alpha(\lambda_j)$, is the number of times (counted with repetition) that λ_j is a solution of $\det(A - \lambda_j I_n) = 0$. The geometric multiplicity of an eigenvalue λ_j, denoted by $\gamma(\lambda_j)$, is the dimension of the vector space $\ker(A - \lambda_j I_n)$. We denote the space of generalized eigenvectors associated with λ_j, $\ker(A - \lambda_j I_n)^{\alpha(\lambda_j)}$, by V_j. Note that V_j has dimension $\alpha(\lambda_j)$ and $\gamma(\lambda_j)$ Jordan chains.

Also, for a real matrix B, we denote by $\sigma_{\max}(B)$ and $\sigma_{\min}(B)$ the largest and smallest singular values of matrix B, respectively. The range of a real matrix A is denoted by $\mathcal{R}(A)$,

and its kernel is denoted by $\ker(A)$. Moreover, we denote by $A \otimes B$ the Kronecker product of two real matrices A and B. Consider a set Q of indices and a matrix K, the matrix K_Q is obtained by removing any row in K not indexed by Q.

1.3.3 Graph Related Notions

Here we review some of the basic notions of graph theory. A weighted undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ is a triple consisting of a set of vertices $\mathcal{V} = \{v_1, v_2, \ldots, v_p\}$ with cardinality p, a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, and a weighted adjacency matrix $\mathbf{A} \in \mathbb{R}^{p \times p}$ which we will define in the coming paragraph. The set of neighbors of a vertex $i \in \mathcal{V}$, denoted by $\mathcal{N}_i = \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$, is the set of vertices that is connected to i by an edge. To clarify, we assume each vertex is not a neighbor of itself, i.e., $(i, i) \notin \mathcal{E}$ for any i. The weighted adjacency matrix $\mathbf{A}$ of the graph $\mathcal{G}$ is defined entry-wise. The entry in the i-th row and j-th column, a_{ij}, satisfies $a_{ij} > 0$ if $(i, j) \in \mathcal{E}$ and otherwise $a_{ij} = 0$. Since the graph is undirected, $a_{ij} = a_{ji}$ for any i, j ranging from 1 to p which results in $\mathbf{A}$ being a symmetric matrix. The degree matrix $\mathbf{D} \in \mathbb{R}^{p \times p}$ of the graph $\mathcal{G}$ is a diagonal matrix with its i-th diagonal element defined by $d_{ii} = \sum_{j=1}^{p} a_{ij}$. The Laplacian matrix $\mathcal{L}$ of the graph $\mathcal{G}$ is defined by $\mathcal{L} = \mathbf{D} - \mathbf{A}$, which is known to be symmetric, if the graph is undirected, positive semi-definite, and having $\text{span}\{\mathbf{1}_p\}$ as its kernel.

CHAPTER 2

On the Computational Complexity of the Secure State-Reconstruction Problem

2.1 Introduction

This chapter is concerned with the detection of attacks on Cyber-Physical Systems. The distributed nature of these large-scale systems often leads to increased vulnerabilities. Of particular concern are adversaries that exploit the distributed nature of CPSs to gain access to sensors and launch attacks by modifying their measurements [CAS08, GUC18]. The most notorious example is the Stuxnet malware [Lan11], which attacked numerous industrial control systems.

Over the last decade, a significant amount of research has focused on reconstructing the state in the presence of sensor attacks - we will refer to this as the Secure State-Reconstruction problem throughout the chapter. The first experimental demonstration of a stealthy attack on a control system was reported in [ALS10] and it was followed by the first theoretical results developed for special classes of systems [STJ10, GLB10]. Stealthy attacks were then formalized in [Smi11, Smi15]. An important step in the conceptual understanding of these attacks was given in [PDB12, PDB13, SH10], where the existence of such attacks was characterized by the system theoretic notion of zero-dynamics.

In addition to detecting and identifying attacks, it is important to mitigate their effect by continuing to control the plant. Hence, researchers have invested a significant effort in developing algorithms to reconstruct the state since the chapters [FTD11, FTD14]. However,

the SSR problem is intrinsically an NP-hard problem (as we show in this chapter). Based on how the NP-hardness is tackled, we classify the existing work in two classes: 1) brute force search [CWH15, LY17], and 2) computationally efficient relaxations. The methods reported in the first class are better suited for small systems as the computational complexity grows combinatorially with the number of sensors. Noteworthy examples of the second class include: convex relaxations [FTD14, YFF16], distributed detection filters [PDB13], specialized observers under sparsity constraints [ST15], satisfiability modulo theory techniques [SNS18], and safety envelopes [TDJ14].

The distributed version of the SSR problem has also attracted a substantial amount of interest given the distributed nature of CPSs. Several authors have studied the problem of estimating a static vector from a set of corrupted measurements, either over a distributed sensor network [CKM18a,SS19], or over a connected-on-average network [CKM18b]. A control-theoretic approach to distributed function calculation was developed in [SH10]. Follow-up works have analyzed the resilient consensus problem, both for discrete [LZK13], and continuous-time [LZS13] systems. The work in [TV15] also evaluates this method in various network topologies. The problem of guaranteeing resilience in the context of distributed state estimation, when the state of the system evolves over time (based on potentially unstable dynamics) has been recently explored in [DUS19], [MS16], and [MS19]. In particular, the authors in [MS19] develop a fully-distributed algorithm that reconstructs the evolving state despite attacks on certain sensors in the network.

Despite the wealth of literature on the security of CPSs, to the best of the authors' knowledge, a detailed characterization of the complexity of the SSR problem is still lacking. On the one hand, the chapters [FTD14,YFF16,PDB13,ST15,SNS18,TDJ14] suggest that the SSR problem is computationally hard since they propose efficient relaxations to the problem. On the other hand, the chapter [MS19] implicitly proposes a polynomial-time solution to the SSR poblem for certain cases. These observations naturally call for a better understanding of the complexity of the SSR problem, which is precisely the goal of this chapter.

As we shall soon see, two alternate notions of observability, namely "sparse observability" introduced in [FTD14, ST15] (see also [SH10] for an equivalent notion in continuous time), and "eigenvalue observability" [Che98], [MS18], will play key roles in our characterization of the SSR problem complexity. Our contributions are the following:

1. We show that the SSR problem is NP-hard.

2. We provide a decomposition that identifies portions of the state that can be reconstructed in polynomial time and portions that are NP-hard to reconstruct.

3. We offer a polynomial-time solution for the SSR problem under an eigenvalue observability assumption.

4. We show that checking sparse observability is coNP-complete.

5. We show that the notions of sparse observability and eigenvalue observability are equivalent when the geometric multiplicity of each eigenvalue of the system matrix $\mathbf{A}$ is 1.

These results can be understood as follows. Although the SSR problem is NP-hard, in general, there may be portions of the state that can be reconstructed in polynomial time. We perform a system decomposition to identify these different portions of the state. In particular, when all the eigenvalues of the system matrix $\mathbf{A}$ have unitary geometric multiplicity, the decomposition results in scalar SSR problems. This establishes the equivalence between sparse observability, a necessary and sufficient condition for the SSR problem to be solvable, and eigenvalue observability, a sufficient condition for the existence of a polynomial time algorithm. Interestingly, even if the unitary geometric multiplicity condition is not satisfied, we may still check eigenvalue observability and, if successful, solve the SSR problem in polynomial time. When the system does not satisfy the eigenvalue observability condition, we conjecture that the SSR problem is intractable since even checking sparse observability is coNP-complete. This chapter improves upon the preliminary results in [MMS19] by in-

troducing a decomposition technique that is key to the aforementioned contributions 1 and 2.

The rest of the chapter is organized as follows. In Section 2.2, we define the notation used throughout the chapter. In Section 2.3, we introduce the system model and give a formal definition of the SSR problem, sparse observability, and eigenvalue observability. We prove that the SSR problem is NP-hard in Section 2.4. This is then followed by a result on breaking the overall SSR problem into several smaller independent SSR problems. As a special case, we show in Section 2.6 that under an eigenvalue observability assumption, the SSR problem can be solved in polynomial time. While checking eigenvalue observability can be done in polynomial time, in Section 2.7 we show that checking sparse observability is coNP-complete. We connect these two notions in Section 2.8 by showing that they are equivalent when the geometric multiplicity of each eigenvalue of the system matrix $\mathbf{A}$ is 1. Finally, we conclude the chapter in Section 2.9.

2.2 Additional Notations

For a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, we use $\ker \mathbf{A}$ to denote the kernel of $\mathbf{A}$, $\mathrm{Im}(\mathbf{A})$ to denote the image of $\mathbf{A}$ and $\mathbf{A}|_V$ to denote the restriction of the linear map defined by $\mathbf{A}$ to the subspace V. We also denote by $\mathbf{A}(V)$ the set $\{y \in \mathbb{R}^n | y = \mathbf{A}x, x \in V\}$.

Let V be a vector space. The collection of vector spaces $\{V^j\}_{j=1,\ldots,r}$, with $V^j \subseteq V$, is said to be an internal direct sum of V, denoted by $V = \bigoplus_{j=1,\ldots,r} V^j$, if any vector $v \in V$ can be uniquely written as $v = v_1 + \ldots + v_r$ with $v_j \in V^j$. The direct sum comes equipped with canonical inclusions $\imath_j : V^j \to V$ taking $v_j \in V^j$ to $\imath_j(v_j) = v_j \in V$, and canonical projections $\pi_j : V \to V^j$ taking $v \in V$ to $\pi_j(v) = v_j \in V^j$.

As an example, consider $V = \mathbb{R}^4$ and let $V^1 = \mathrm{Im}(\mathbf{M}_1)$, $V^2 = \mathrm{Im}(\mathbf{M}_2)$, and $V^3 = \mathrm{Im}(\mathbf{M}_3)$

where $\mathbf{M}_1$, $\mathbf{M}_2$, and $\mathbf{M}_3$ are the following linear transformations:

$$\mathbf{M}_1 = \begin{bmatrix} 2 & 0 \\ -1 & 1 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{M}_2 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix}, \quad \mathbf{M}_3 = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 1 \end{bmatrix}. \tag{2.1}$$

The collection $\{V^1, V^2, V^3\}$ is an internal direct sum of V since all the column vectors are linearly independent. The canonical inclusions $\imath_j$ can be represented by $\mathbf{I}_4|_{V^j}$, the identity matrix $\mathbf{I}_4$ of order 4 restricted to the subspace V^j, since $\imath_j$ maps any vector $v \in V^j$ to $v \in V$. Conversely, the canonical projections π_j are represented by the matrices $\mathbf{P}_j = \mathbf{M}_i\mathbf{U}_j\mathbf{M}^{-1}$, where $\mathbf{U}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$, $\mathbf{U}_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$, $\mathbf{U}_3 = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$, as well as $\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & \mathbf{M}_2 & \mathbf{M}_3 \end{bmatrix}$.

Let $V = \bigoplus_{j=1,\ldots,r} V^j$, $W = \bigoplus_{j=1,\ldots,r} W^j$, and consider a linear map $F : V \to W$ satisfying $F(V^j) \subseteq W^j$. Then, the linear map $F^{(j)} : V^j \to W^j$ defined by $F^{(j)} = \pi_j \circ F \circ \imath_j$ satisfies:

$$F^{(j)} \circ \pi_j = \pi_j \circ F \tag{2.2}$$

$$\imath_j \circ F^{(j)} = F \circ \imath_j, \tag{2.3}$$

where $\circ$ denotes function composition.

Continuing with our example, let $\mathbf{F}$ be represented by the matrix:

$$\mathbf{F} = \frac{1}{2}\begin{bmatrix} 2 & 0 & 0 & -4 \\ 1 & 3 & -1 & 4 \\ -1 & -1 & 3 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix}, \tag{2.4}$$

and note that $\mathbf{F}(V^j) \subseteq V^j$. The maps $\mathbf{F}^{(j)}$ are then given by $\mathbf{F}^{(1)} = \mathbf{P}_1\mathbf{F} \circ \imath_1 = \mathbf{P}_1\mathbf{F}|_{V^1} = \mathbf{I}_4|_{V^1}$, $\mathbf{F}^{(2)} = \mathbf{P}_2\mathbf{F} \circ \imath_2 = \mathbf{P}_2\mathbf{F}|_{V^2} = 2\mathbf{I}_4|_{V^2}$, as well as $\mathbf{F}^{(3)} = \mathbf{P}_3\mathbf{F} \circ \imath_3 = \mathbf{P}_3\mathbf{F}|_{V^3} = 3\mathbf{I}_4|_{V^3}$. Since the vector subspaces V^j are the generalized eigenspaces of $\mathbf{F}$ corresponding to each different

eigenvalue, the matrices $\mathbf{F}^{(j)}$ are simply the identity matrix restricted to V^j multiplied by the corresponding eigenvalue.

2.3 Problem Formulation

2.3.1 System Model

Consider a discrete-time linear time-invariant system under sensor attacks of the following form:

$$\mathbf{x}[t+1] = \mathbf{A}\mathbf{x}[t] \tag{2.5}$$

$$\mathbf{y}_i[t] = \mathbf{C}_i\mathbf{x}[t] + \mathbf{e}_i[t], \tag{2.6}$$

where $\mathbf{x}[t] \in \mathbb{R}^n$ and $\mathbf{y}_i[t] \in \mathbb{R}^{p_i}$ represent the state of the system and the measurement acquired by sensor i respectively. The vector $\mathbf{e}_i[t] \in \mathbb{R}^{p_i}$ models the attack on sensor i. If sensor i is attacked by an adversary, then $\mathbf{e}_i[t]$ can be arbitrary, otherwise, $\mathbf{e}_i[t]$ remains zero for any k. Let $\mathcal{V}$ denote the set of sensors, and let $N = |\mathcal{V}|$. We use $\mathbf{C} = \left[\mathbf{C}_1^T|\mathbf{C}_2^T|\cdots|\mathbf{C}_N^T\right]^T$ to denote the collection of the sensor observation matrices, $\mathbf{y}[t] = \begin{bmatrix}\mathbf{y}_1^T[t] & \cdots & \mathbf{y}_N^T[t]\end{bmatrix}^T$ and $\mathbf{e}[t] = \begin{bmatrix}\mathbf{e}_1^T[t] & \cdots & \mathbf{e}_N^T[t]\end{bmatrix}^T$ to represent the collective measurement vector and the collective attack vector, respectively.

We define $\mathcal{O}_i = \left[\mathbf{C}_i^T|(\mathbf{C}_i\mathbf{A})^T|\ldots|(\mathbf{C}_i\mathbf{A}^{\tau_i-1})^T\right]^T$ to be the observability matrix of sensor i with τ_i being the observability index of the pair $(\mathbf{A}, \mathbf{C}_i)$. We also define two more vectors $\mathbf{Y}_i = \begin{bmatrix}\mathbf{y}_i^T[0] & \ldots & \mathbf{y}_i^T[\tau_i-1]\end{bmatrix}^T$ and $\mathbf{E}_i = \begin{bmatrix}\mathbf{e}_i^T[0] & \ldots & \mathbf{e}_i^T[\tau_i-1]\end{bmatrix}^T$ to be the collection of measurements and attacks of sensor i over the time horizon $[0, \tau_i - 1]$, respectively. An equivalent expression for the measurements is:

$$\mathbf{Y}_i = \mathcal{O}_i\mathbf{x}[0] + \mathbf{E}_i. \tag{2.7}$$

In the remainder of the chapter, we drop the time indices to simplify notation.

2.3.2 The Secure State-Reconstruction Problem

Problem 1 *(Secure state-reconstruction)*

Input: *Matrices* $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{C}_i \in \mathbb{R}^{p_i\times n},\ i = 1,\dots,N$, *and a set of vectors* $\mathbf{Y}_i \in \mathbb{R}^{p_i\tau_i},\ i = 1,\dots,N$.

Question: *Find a vector* $\mathbf{x} \in \mathbb{R}^n$ *and a set* $\mathcal{I}$ *of minimal cardinality such that* $\mathbf{Y}_j = \mathcal{O}_j\mathbf{x}$ *for all* $j \notin \mathcal{I}$.

In other words, the SSR problem requires the reconstruction of a state $\mathbf{x}$ and the simplest attack explanation in the form of the least number of attacked sensors. Note that when the solution $\mathbf{x}$ is unique, we have found the state of the linear system. Although uniqueness of solutions is essential when handling attacks, we can study the complexity of the SSR problem independently of the number of solutions. To make this clear, we will explicitly state the uniqueness requirements when needed.

2.3.3 Sparse Observability and Eigenvalue Observability

The notions of sparse observability and eigenvalue observability are instrumental to the results in this chapter.

Definition 1 (Sparse observability index) *The sparse observability index of the pair* $(\mathbf{A}, \mathbf{C})$ *in system* (2.5)-(2.6) *is the largest integer* k *such that* $\ker\ \mathcal{O}_{\mathcal{V}\setminus\mathcal{K}} = \{0\}$ *for any* $\mathcal{K} \subseteq \mathcal{V},\ |\mathcal{K}| \leq k$. *When the sparse observability index is* r, *we say that system* (2.5)-(2.6) *is* $r-$*sparse observable.*

It is proved in [FTD14,ST15] (see also [CWH15] for a similar notion in continuous time) that the possibility of uniquely reconstructing the state $\mathbf{x}[t]$ is characterized by the sparse observability index.

Theorem 1 ([FTD14, CWH15, ST15]) *Consider the linear system* (2.5)-(2.6) *where at*

most s sensors are subject to attacks. The state $\mathbf{x}[t]$ can be uniquely reconstructed if and only if the sparse observability index of the pair $(\mathbf{A}, \mathbf{C})$ is at least $2s$.

In view of this result, computing the sparse observability index of a system is of great interest since it characterizes the maximum number of arbitrary sensor attacks that can be tolerated without compromising the ability to uniquely reconstruct the state.

In addition to sparse observability, we will require the notion of eigenvalue observability [Che98, MS18].

Definition 2 (Eigenvalue observability index) *We say that an eigenvalue $\lambda \in sp(\mathbf{A})$ is observable w.r.t. sensor i if the linear map defined by $\begin{bmatrix} \mathbf{A} - \lambda \mathbf{I}_n \\ \mathbf{C}_i \end{bmatrix}$ is injective.*

If the above condition is satisfied, we say that "sensor i can observe the states in the generalized eigenspace corresponding to λ", or briefly, we say "sensor i can observe eigenvalue λ". Let the set of all sensors that can observe an eigenvalue λ be denoted $\mathcal{S}_\lambda$. The eigenvalue observability index of system (2.5)-(2.6) is the largest integer k such that each eigenvalue of the matrix $\mathbf{A}$ is observable by at least $k+1$ distinct sensors. When the eigenvalue observability index is k, we say that system (2.5)-(2.6) is k-eigenvalue observable.

We study the SSR problem under the following assumptions.

Assumption 1: For each sensor $i \in \{1, \ldots, N\}$ under attack, the adversary can only manipulate sensor i's measurements through the signal $\mathbf{e}_i[t]$ in (2.6).

Assumption 2: The adversary is omniscient, i.e., we assume the adversary has full knowledge of the system state, measurements, and plant model. Moreover, all the attacked sensors are allowed to work cooperatively.

2.4 SSR is Hard

Fawzi *et al.* established in [FTD14] a connection between the SSR problem and compressed sensing by drawing inspiration from the ideas of Candes and Tao in [CT05]. We take this approach further by also using the ideas in [CT05] to establish that the SSR problem is NP-hard. To do so, we first define the compressed sensing problem.

Problem 2 *(Compressed sensing)*
Input: *A full row rank matrix* $\mathbf{F} \in \mathbb{Q}^{m \times n}$*, a vector* $\mathbf{b} \in \mathbb{Q}^m$*.*
Question: *Find the sparsest solution of* $\mathbf{Fx} = \mathbf{b}$*.*

The compressed sensing problem yields the solution to the minimization problem:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \|\mathbf{x}\|_0 \\ \text{s.t.} \quad & \mathbf{Fx} = \mathbf{b}. \end{aligned} \tag{2.8}$$

Theorem 2 ([FTD14]) *The SSR problem is NP-hard.*

Proof. Given an instance of the compressed sensing problem, we generate an instance of the SSR problem as follows. Let the system matrix be of the form $\mathbf{A} = \mathbf{I}_n$, and the collective observation matrix $\mathbf{C}$ satisfy $\mathrm{Im}\mathbf{C} = \ker \mathbf{F}$. Let the measurements of the sensors be scalar-valued, i.e., let $\mathbf{C}_i$ be the i-th row of $\mathbf{C}$. Note that based on the above $\mathbf{A}$ matrix, the observability index for each sensor $i \in \{1, \ldots, N\}$ is given by $\tau_i = 1$, and thus $\mathcal{O}_i = \mathbf{C}_i$. Finally, let $\mathbf{Y}$ be any solution to the equation $\mathbf{FY} = \mathbf{b}$. Since the linear equation $\mathbf{FY} = \mathbf{b}$ is underdetermined, finding a solution $\mathbf{Y}$ can be done in polynomial time [Lau04]. For each $i \in \{1, \ldots, N\}$, set $\mathbf{Y}_i$ to be the i-th row of $\mathbf{Y}$. Thus, given an instance of the compressed sensing problem, the instance of the SSR problem described above can be constructed in polynomial time.

The SSR problem for the constructed instance degenerates to:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{e}} \quad & \|\mathbf{e}\|_0 \\ \text{s.t.} \quad & \mathbf{C}\mathbf{x} + \mathbf{e} = \mathbf{Y}. \end{aligned} \tag{2.9}$$

We now show these two problems have the same solution. It is simple to see that any solution $(\mathbf{x}, \mathbf{e})$ of $\mathbf{C}\mathbf{x} + \mathbf{e} = \mathbf{Y}$ provides a solution to $\mathbf{F}\mathbf{e} = \mathbf{b}$, since by applying $\mathbf{F}$ we obtain:

$$\begin{aligned} \mathbf{F}(\mathbf{C}\mathbf{x} + \mathbf{e}) &= \mathbf{F}\mathbf{Y} \\ \Leftrightarrow \mathbf{F}\mathbf{e} &= \mathbf{b}. \end{aligned} \tag{2.10}$$

To prove the converse, we show that for every $\mathbf{e}$ such that $\mathbf{F}\mathbf{e} = \mathbf{b}$, there exists some $\mathbf{x}$ satisfying $\mathbf{C}\mathbf{x} + \mathbf{e} = \mathbf{Y}$. Recalling that $\mathbf{F}\mathbf{Y} = \mathbf{b}$, we obtain $\mathbf{F}(\mathbf{Y} - \mathbf{e}) = \mathbf{0}$, i.e., $\mathbf{Y} - \mathbf{e} \in \ker \mathbf{F}$. Since $\ker \mathbf{F} = \text{Im}\mathbf{C}$, there exists an $\mathbf{x}$ such that $\mathbf{C}\mathbf{x} = \mathbf{Y} - \mathbf{e}$, as desired.

Noticing that the equations $\mathbf{F}\mathbf{e} = \mathbf{b}$ and $\mathbf{C}\mathbf{x} + \mathbf{e} = \mathbf{Y}$ have the same solutions for $\mathbf{e}$, we conclude that they also have the same sparsest solution. In other words, if there exists an algorithm $\mathcal{A}$ that solves the SSR problem for the specific instance constructed by us, such an algorithm will also yield a solution to the given instance of the compressed sensing problem. It then follows that since the compressed sensing problem is NP-hard [Nat95], the secure state reconstruction problem is also NP-hard. □

2.5 System Decomposition

In the previous section, we proved that the SSR problem is in general NP-hard. This means there does not exist a polynomial-time solution unless $P = NP$. Despite this fact, we show in this section how to decompose the SSR problem into smaller instances. In the next section, we identify which of these smaller instances are NP-hard, and which ones are solvable in polynomial time.

Lemma 1 *Assume the existence of a collection of vector spaces* $\{X^j\}_{j=1,\dots,r}$ *satisfying:*

1. $\mathbb{C}^n = \bigoplus_{j=1,\ldots,r} X^j$;

2. $\mathbf{A}(X^j) \subseteq X^j$ *for* $j = 1, \ldots, r$;

3. $\mathcal{O}_i(\mathbb{C}^n) = \bigoplus_{j=1,\ldots,r} \mathcal{O}_i^j(X^j) \quad$ *for* $i = 1, \ldots, p$,

then for any $\mathbf{Y}_i$*, a solution* $\mathbf{x}$ *of the equation:*

$$\mathbf{Y}_i = \mathcal{O}_i \mathbf{x}, \tag{2.11}$$

whenever it exists, can be written as $\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2 + \ldots + \mathbf{x}_r$ *with* $\mathbf{x}_j = \pi_j(\mathbf{x}) \in X^j$ *given by the solution of:*

$$\mathbf{Y}_i^j = \mathcal{O}_i^j \mathbf{x}_j, \tag{2.12}$$

for $\mathbf{Y}_i^j = \pi_j(\mathbf{Y}_i) \in \mathcal{O}_i^j(X^j)$ *and* $\mathcal{O}_i^j = \pi_j \circ \mathcal{O}_i \circ \imath_j$.

Proof. Let $\mathbf{x}_j$ be the solution of (2.12) and note that:

$$\mathbf{Y}_i^j = \mathcal{O}_i^j \mathbf{x}_j \Rightarrow \imath_j(\mathbf{Y}_i^j) = \imath_j \circ \mathcal{O}_i^j(\mathbf{x}_j) = \mathcal{O}_i \circ \imath_j(\mathbf{x}_j) = \mathcal{O}_i \mathbf{x}_j, \tag{2.13}$$

where the third equality follows from (2.3). By summing over j we obtain:

$$\mathbf{Y}_i = \sum_{j=1}^{r} \imath_j(\mathbf{Y}_i^j) = \sum_{j=1}^{r} \mathcal{O}_i \mathbf{x}_j = \mathcal{O}_i \sum_{j=1}^{r} \mathbf{x}_j = \mathcal{O}_i \mathbf{x}. \tag{2.14}$$

Hence, the solutions to (2.12) provide a solution to (2.11). Consider now (2.11):

$$\begin{aligned} \mathbf{Y}_i = &\mathcal{O}_i \mathbf{x} \Rightarrow \pi_j(\mathbf{Y}_i) = \pi_j \circ \mathcal{O}_i(\mathbf{x}) \\ \Rightarrow &\mathbf{Y}_i^j = \mathcal{O}_i^j \circ \pi_j(\mathbf{x}) = \mathcal{O}_i^j \mathbf{x}_j. \end{aligned} \tag{2.15}$$

where the third equality follows from (2.2). Hence, if $\mathbf{x}$ is a solution to (2.11), then $\mathbf{x}_i$ is a solution to (2.12). □

Intuitively, we treat the state-space $\mathbb{R}^n$ as the direct sum of multiple subspaces. If the images of these subspaces under the linear map $\mathcal{O}_i$ are pairwise non-overlapping, we are able to project the state vector $\mathbf{x}$ onto these subspaces, project the measurement $\mathbf{Y}_i$ onto

the image under the linear map $\mathcal{O}_i$ of these subspaces, and then establish a one-to-one correspondence between the projected state vector and the projected measurement. This effectively decomposes the original problem into r sub-problems, each of dimension $\dim(X^j)$. As formalized in the next result, the spaces X^j can always be taken to be the generalized eigenspaces of $\mathbf{A}$.

Proposition 1 *The generalized eigenspaces $V^1, \ldots, V^r$ of $\mathbf{A}$ satisfy properties (1)-(3) in Lemma 1.*

Proof. Properties (1) and (2) in Lemma (1) follow directly from the definition of generalized eigenspace. To simplify notation, we will drop the sensor index i in this proof.

It also follows from the definition of generalized eigenspace that $\cup_{j=1,\ldots,r} V^j$ spans $\mathbb{C}^n$. Therefore, the set $\cup_{j=1,\ldots,r} \mathcal{O}(V^j)$ spans $\mathcal{O}(\mathbb{C}^n)$. Given this, to conclude property (3) we only need to show:

$$\mathcal{O}(V^j) \cap \mathcal{O}(V^k) = \{0\}, \qquad \forall j \neq k.$$

Moreover, it suffices to show that for any $\mathbf{x}_j \in V^j$ and $\mathbf{x}_k \in V^k$, with $j \neq k$, the equality $\mathcal{O}(\mathbf{x}_j + \mathbf{x}_k) = 0$ can only be satisfied if $\mathcal{O}\mathbf{x}_j = 0 = \mathcal{O}\mathbf{x}_k$.

We have the following sequence of equalities that is explained thereafter:

$$\begin{aligned} 0 &= \mathcal{O}(\mathbf{x}_j + \mathbf{x}_k) && (2.16)\\ &= \mathcal{O}(\mathbf{A} - \lambda_k \mathbf{I}_n)^{\alpha(\lambda_k)}(\mathbf{x}_j + \mathbf{x}_k) && (2.17)\\ &= \mathcal{O}(\mathbf{A} - \lambda_k \mathbf{I}_n)^{\alpha(\lambda_k)}(\mathbf{x}_j) && (2.18)\\ &= \mathcal{O}\mathbf{x}_j. && (2.19) \end{aligned}$$

The second step follows from $\ker \mathcal{O} \subseteq \ker \mathcal{O}(\mathbf{A} - \lambda_k \mathbf{I}_n)^{\alpha(\lambda_k)}$, the third step follows from

$\mathbf{x}_k \in V^k = \ker(\mathbf{A} - \lambda_k \mathbf{I}_n)^{\alpha(\lambda_k)}$, and the fourth from the following sequence of steps:

$$\begin{aligned}
\dim\ker \mathcal{O}\big|_{V^j} &\leq \dim\ker \mathcal{O}(\mathbf{A} - \lambda_k \mathbf{I}_n)^{\alpha(\lambda_k)}\big|_{V^j} && (2.20)\\
&= \dim\ker(\mathbf{A} - \lambda_k \mathbf{I}_n)^{\alpha(\lambda_k)}\big|_{V^j} && (2.21)\\
&+ \dim\ker \mathcal{O}\big|_{(\mathbf{A}-\lambda_k \mathbf{I}_n)^{\alpha(\lambda_k)}V^j} && (2.22)\\
&= \dim\ker \mathcal{O}\big|_{(\mathbf{A}-\lambda_k \mathbf{I}_n)^{\alpha(\lambda_k)}V^j} && (2.23)\\
&\leq \dim\ker \mathcal{O}\big|_{V^j}. && (2.24)
\end{aligned}$$

The first step comes from $\ker \mathcal{O} \subseteq \ker \mathcal{O}(\mathbf{A} - \lambda_k \mathbf{I}_n)$. To show that the second step is true, we observe that
$\dim\ker \mathbf{MN} = \dim\ker \mathbf{N} + \dim\ker(\mathbf{M}\big|_{\mathbf{N}(\mathbb{C}^n)})$ for any matrices $\mathbf{M}, \mathbf{N} \in \mathbb{C}^{n\times n}$. The third step comes from the map $(\mathbf{A} - \lambda_j \mathbf{I}_n)^{\alpha(\lambda_j)}\big|_{V^j}$ being injective if $j \neq k$, as the generalized eigenspaces V^j and V^k intersect only at the origin, and $\ker(\mathbf{A} - \lambda_j \mathbf{I}_n)^{\alpha(\lambda_j)} = V^j$. The fourth step follows by the $\mathbf{A}-$invariant nature of eigenspace V^j. This shows $\dim\ker \mathcal{O}\big|_{V^j} = \dim\ker \mathcal{O}(\mathbf{A} - \lambda_k \mathbf{I}_n)^{\alpha(\lambda_k)}\big|_{V^j}$ which, combined with $\ker \mathcal{O}\big|_{V^j} \subseteq \ker \mathcal{O}(\mathbf{A} - \lambda_k \mathbf{I}_n)^{\alpha(\lambda_k)}\big|_{V^j}$, can only hold when $\ker \mathcal{O}\big|_{V^j} = \ker \mathcal{O}(\mathbf{A} - \lambda_k \mathbf{I}_n)^{\alpha(\lambda_k)}\big|_{V^j}$. A symmetric argument can be used to show that $\mathcal{O}\mathbf{x}_k = 0$ and the claim is thus proved. □

Combining Lemma (1) and Proposition (1) results in a decomposition of the sensor measurements in (2.7):

$$\mathbf{Y}_i^j = \mathcal{O}_i^j \mathbf{x}_j, \quad j = 1, 2, \ldots, r, \tag{2.25}$$

where $\mathbf{Y}_i^j = \pi_j(Y_i)$ is the projection of measurement $\mathbf{Y}_i$ onto the vector space $\mathcal{O}_i(V^j)$, the linear transformation $\mathcal{O}_i^j$ is defined by $\mathcal{O}_i^j = \pi_j \circ \mathcal{O}_i \circ \imath_j$, $\mathbf{x}_j$ is given by $\mathbf{x}_j = \pi_j(\mathbf{x})$, $\pi_j : \mathbb{R}^n \to V^j$ is the canonical projection and $\imath_j : V^j \to \mathbb{R}^n$ is the canonical inclusion.

Theorem 3 *A solution* $\mathbf{x}$ *of the SSR problem with inputs* $\mathbf{A}, \mathbf{C}_i, \mathbf{Y}_i$ *is given by* $\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_j$ *where* $\mathbf{x}_i$ *is the solution to the SSR problem with inputs* $\mathbf{A}^{(j)} = \pi_j \circ \mathbf{A} \circ \imath_j$, $\mathbf{C}_i^j = \mathbf{C}_i \circ \imath_j$, $\mathbf{Y}_i^j$.

Proof. Follows directly from Lemma 1, Proposition 1, and the properties of generalized eigenspaces. □

Theorem (3) lays the theoretical foundation for decomposing the SSR problem with n states into r sub-problems of the form:

$$\begin{aligned} \mathbf{x}_j[t+1] &= \mathbf{A}^{(j)}\mathbf{x}_j[t], \\ \mathbf{Y}_i^j[t] &= \mathcal{O}_i^j\mathbf{x}_j[t] + \mathbf{E}_i^j[t], \end{aligned} \tag{2.26}$$

each with $\alpha(\lambda_1), \alpha(\lambda_2), \ldots, \alpha(\lambda_r)$ states. The attack vector $\mathbf{E}_i^j$ is identically zero when sensor i is not under attack. The state of the original problem can be reconstructed by summing up the state reconstructions of each sub-problem.

We now illustrate the decomposition of (2.5)-(2.6) into (2.26) through an example. The matrix $\mathbf{A}$ is the same as the matrix $\mathbf{F}$ defined in (2.4) and the matrices $\mathbf{C}_i$ are given by:

$$\begin{aligned} \mathbf{C}_1 &= \begin{bmatrix} 3 & 2 & 0 & 2 \end{bmatrix}, \quad & \mathbf{C}_2 &= \begin{bmatrix} 2 & 3 & 1 & -1 \end{bmatrix}, \\ \mathbf{C}_3 &= \begin{bmatrix} 2 & 2 & 0 & 0 \end{bmatrix}, \quad & \mathbf{C}_4 &= \begin{bmatrix} 2 & 3 & -1 & 0 \end{bmatrix}. \end{aligned}$$

As we discussed below (2.4), the generalized eigenspaces of $\mathbf{A}$ are $V^1 = \mathrm{Im}(\mathbf{M}_1), V^2 = \mathrm{Im}(\mathbf{M}_2)$, and $V^3 = \mathrm{Im}(\mathbf{M}_3)$ corresponding to eigenvalues $1, 2$, and 3 respectively, where $\mathbf{M}_j$ are defined in (2.1) for $j = 1, 2, 3$. Also, recall that the projections π_1, π_2, and π_3 are $\mathbf{P}_j = \mathbf{M}_j(\mathbf{M}_j^T\mathbf{M}_j)^{-1}\mathbf{M}_j^T$ for $j = 1, 2, 3$. By definition, we have $\mathbf{x}_1 = \mathbf{P}_1\mathbf{x}$, $\mathbf{x}_2 = \mathbf{P}_2\mathbf{x}$, $\mathbf{x}_3 = \mathbf{P}_3\mathbf{x}$, and $\mathbf{A}^{(1)} = \mathbf{P}_1\mathbf{A}|_{V^1}$, $\mathbf{A}^{(2)} = \mathbf{P}_2\mathbf{A}|_{V^2}$, $\mathbf{A}^{(3)} = \mathbf{P}_3\mathbf{A}|_{V^3}$. Hence the decomposition of $\mathbf{x}[t+1] = \mathbf{A}\mathbf{x}[t]$ is given by:

$$\mathbf{P}_j\mathbf{x}[t+1] = (\mathbf{P}_j\mathbf{A}|_{V^j})(\mathbf{P}_j\mathbf{x}[t]), \quad j = 1, 2, 3.$$

We now illustrate how to decompose the measurement equation $\mathbf{Y}_1[t] = \mathcal{O}_1\mathbf{x}[t] + \mathbf{E}_1[t]$

for sensor 1. The observability matrix $\mathcal{O}_1$ of sensor 1 is given by:

$$\mathcal{O}_1 = \begin{bmatrix} 3 & 2 & 0 & 2 \\ 4 & 3 & -1 & 4 \\ 6 & 5 & -3 & 10 \\ 10 & 9 & -7 & 28 \end{bmatrix}.$$

We first compute the projections $\widetilde{\pi}_1^1, \widetilde{\pi}_1^2$ and $\widetilde{\pi}_1^3$ that map $\mathcal{O}_1(\mathbb{R}^4)$ to $\mathcal{O}_1(V^1), \mathcal{O}_1(V^2)$, and $\mathcal{O}_1(V^3)$, respectively. To do this, we define the matrices:

$$\widetilde{\mathbf{M}}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \ \widetilde{\mathbf{M}}_2 = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 8 \end{bmatrix}, \text{ and } \widetilde{\mathbf{M}}_3 = \begin{bmatrix} 1 \\ 3 \\ 9 \\ 27 \end{bmatrix},$$

which satisfy $\mathcal{O}_1(V^1) = \mathrm{Im}(\widetilde{\mathbf{M}}_1)$, $\mathcal{O}_1(V^2) = \mathrm{Im}(\widetilde{\mathbf{M}}_2)$, and $\mathcal{O}_1(V^3) = \mathrm{Im}(\widetilde{\mathbf{M}}_3)$. We also remark that the collection $\{\mathcal{O}_1(V^1), \mathcal{O}_1(V^2), \mathcal{O}_1(V^3)\}$ is an internal direct sum of the vector space $\mathcal{O}_1(\mathbb{R}^4)$. Therefore, by defining $\widetilde{\mathbf{M}} = \begin{bmatrix} \widetilde{\mathbf{M}}_1 & \widetilde{\mathbf{M}}_2 & \widetilde{\mathbf{M}}_3 \end{bmatrix}$ and $\widetilde{\mathbf{U}_1} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$, $\widetilde{\mathbf{U}_2} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$, $\widetilde{\mathbf{U}_3} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$, each projection $\widetilde{\pi}_1^i$ can be represented by the projection matrix:

$$\widetilde{\mathbf{P}}_1^i = \widetilde{\mathbf{M}}_i \widetilde{\mathbf{U}}_i (\widetilde{\mathbf{M}}^T \widetilde{\mathbf{M}})^{-1} \widetilde{\mathbf{M}}^T, \ i = 1, 2, 3.$$

By definition, $\mathbf{Y}_1^j = \widetilde{\mathbf{P}}_1^j \mathbf{Y}_1$, $\mathbf{E}_1^j = \widetilde{\mathbf{P}}_1^j \mathbf{E}_1$ and $\mathcal{O}_1^j = \widetilde{\mathbf{P}}_1^j \mathcal{O}_1|_{V^j}$ for $j = 1, 2, 3$. In summary, the decomposition of measurement $\mathbf{Y}_1[t] = \mathcal{O}_1 \mathbf{x}[t] + \mathbf{E}_1[t]$ is given by:

$$\widetilde{\mathbf{P}}_1^j \mathbf{Y}_1[t] = (\widetilde{\mathbf{P}}_1^j \mathcal{O}_1|_{V^j})(\mathbf{P}_1^j \mathbf{x}[t]) + \widetilde{\mathbf{P}}_1^j \mathbf{E}_1[t], \quad j = 1, 2, 3.$$

2.6 Classes of SSR Problems Solvable in Polynomial Time

While in the previous section we established that the SSR problem is NP-hard, in this section we leverage the results in Section 2.5 to answer a simple but important question: when can we solve the SSR problem in polynomial time? Our answer relies heavily on the

system decomposition technique introduced in Section 2.5. The first result establishes that the decomposition can be done in polynomial time.

Proposition 2 *The computational complexity of decomposing the system* (2.5)-(2.6) *into sub-systems* (2.26) *is within* $O(pn^3)$.

Proof. To prove this result, we list all the steps involved in the decomposition from (2.5)-(2.6) to (2.26) and list the computational complexity of each step.

Offline preparation 1: compute the observability matrix of each sensor $\mathcal{O}_i$. The computational complexity of this step is $O(pn^2)$.

Offline preparation 2: find the eigenvalues of the matrix $\mathbf{A}$ as well as its generalized eigenspaces V^j. This can be done by finding the Jordan form of $\mathbf{A}$. The computational complexity of this step is $O(n^3)$.

Offline preparation 3: determine the image of each generalized eigenspace V^j under the observability matrix $\mathcal{O}_i$, i.e., $\mathcal{O}_i(V^j)$. In this step, we perform p times two $n \times n$ matrix multiplications and thus the complexity of this step is $O(pn^3)$.

Offline preparation 4: find the projection matrix for each generalized eigenspace and each sensor. The computational complexity of this step is $O(pn^3)$.

Online task: at each time instance, project the measurements $\mathbf{Y}_i[t]$ of each sensor i onto each generalized eigenspace. In this step, for each sensor we multiply a $n \times n$ matrix by a $n \times 1$ vector r times. This requires $O(pn^2 r)$ time.

We thus conclude that we can decompose the system (2.5)-(2.6) into sub-systems (2.26) within $O(pn^3)$ and finish the proof. □

Before giving an answer to the question we stated at the beginning of this section, we relate the sparse observability index defined for the system (2.5)-(2.6) and the sparse observability index for each subsystem (2.26) with j ranging from 1 to r in the following two results. Note that, since the state space of (2.26) is V^j, sparse observability is characterized

by the injectivity of $\mathcal{O}_i^j|_{V^j}$ whereas eigenvalue observability is characterized by injectivity of the linear map $\begin{bmatrix} \mathbf{A}^{(j)} - \lambda_j \mathbf{I}_n^{(j)} \\ \mathbf{C}_i^j \end{bmatrix}$, where we define $\mathbf{I}_n^{(j)} = \pi_j \circ \mathbf{I}_n \circ \imath_j$. We now have the following results.

Theorem 4 *The system* (2.5)-(2.6) *is k-sparse observable if and only if for each* $j \in \{1, 2, \ldots, r\}$*, the system* (2.26) *is k-sparse observable.*

Proof. This result can be easily established by observing that $\ker \mathcal{O}_i = \oplus_{j=1}^r \ker \mathcal{O}_i^j$ holds for any sensor i. We omit the proof here in the interest of space. □

Similarly, to relate the eigenvalue observability index defined for the overall system and the eigenvalue observability index for each subsystem, we have the following result.

Theorem 5 *The system* (2.5)-(2.6) *is k-eigenvalue observable if and only if for each* $j \in \{1, 2, \ldots, r\}$*, the system* (2.26) *is k-eigenvalue observable.*

Proof. By the definition of eigenvalue observability, it suffices to show the matrix $\begin{bmatrix} \mathbf{A} - \lambda_j \mathbf{I}_n \\ \mathbf{C}_i \end{bmatrix}$ has full column rank if and only if each matrix $\begin{bmatrix} \mathbf{A}^{(j)} - \lambda_j \mathbf{I}_n^{(j)} \\ \mathbf{C}_i^j \end{bmatrix}$ defines an injective map with domain V^j, for j ranging from 1 to r.

Consider the map $F : V \to V \times \mathbb{R}^{p_i}$ defined by the matrix $\begin{bmatrix} \mathbf{A} - \lambda_j \mathbf{I}_n \\ \mathbf{C}_i \end{bmatrix}$ and note that F being injective is equivalent to $\ker F = \{0\}$. Note also that the result immediately follows if we establish that $\ker F \subseteq V^j$. This can be seen by noting that $F\mathbf{x} = 0$ for $\mathbf{x} \in \mathbb{R}^n$ degenerates to $F\mathbf{x} = 0$ for $\mathbf{x} \in V^j$ and (given $\mathbf{x} = \imath_j \mathbf{x}$) can be written as $F\imath_j \mathbf{x} = 0$:

$$\begin{bmatrix} \mathbf{A} \circ \imath_j - \lambda_j \imath_j \\ \mathbf{C}_i \circ \imath_j \end{bmatrix} \mathbf{x} = 0. \tag{2.27}$$

Moreover, since $(\mathbf{A}-\lambda_j\mathbf{I}_n)(V^j) \subseteq V^j$ we have the equality $\pi_j(\mathbf{A}-\lambda_j\mathbf{I}_n)\imath_j\mathbf{x} = (\mathbf{A}-\lambda_j\mathbf{I}_n)\imath_j\mathbf{x}$. Therefore, (2.27) degenerates into:

$$\begin{bmatrix} \pi_j \circ \mathbf{A} \circ \imath_j - \lambda_j \pi_j \circ \imath_j \\ \mathbf{C}_i \circ \imath_j \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{A}^{(j)} - \lambda_j \mathbf{I}_n^{(j)} \\ \mathbf{C}_i^j \end{bmatrix} \mathbf{x} = 0. \tag{2.28}$$

Therefore, we proceed by showing that $\ker F \subseteq V^j$. The equality $F\mathbf{x} = 0$ implies $(\mathbf{A} - \lambda_j\mathbf{I}_n)\mathbf{x} = 0$. If we write $\mathbf{x}$ as $\mathbf{x}_j + \mathbf{x}_{\bar{j}}$ with $\mathbf{x}_j = \pi_j(\mathbf{x})$ and $\mathbf{x}_{\bar{j}} = \sum_{k=1,k\neq j}^{r} \pi_k(\mathbf{x})$ we have $(\mathbf{A} - \lambda_j\mathbf{I}_n)(\mathbf{x}_j + \mathbf{x}_{\bar{j}}) = 0$. We now make two observations. The first is that $(\mathbf{A} - \lambda_j\mathbf{I}_n)\mathbf{x}_{\bar{j}} = 0$ implies $\mathbf{x}_{\bar{j}} = 0$ since $\mathbf{x}_{\bar{j}} \neq 0$ would imply that $\mathbf{x}_{\bar{j}} \in V^j$, by definition of V^j. The second observation is that $(\mathbf{A}-\lambda_j\mathbf{I}_n)(V^\ell) \subseteq V^\ell$, for $\ell \in \{1,\ldots,r\}$, implies that $(\mathbf{A}-\lambda_j\mathbf{I}_n)(\mathbf{x}_j+\mathbf{x}_{\bar{j}}) = 0$ iff $(\mathbf{A} - \lambda_j\mathbf{I}_n)\mathbf{x}_j = 0$ and $(\mathbf{A} - \lambda_j\mathbf{I}_n)\mathbf{x}_{\bar{j}} = 0$. Together with the first observation we have $\mathbf{x}_{\bar{j}} = 0$ which implies that $\mathbf{x} \in V^j$ and concludes the proof. □

Based on the above decomposition and the assumption that at most s sensors are attacked, we partition the set of eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_r\}$ as follows:

- We define $\mathcal{J}_1 \subseteq \{\lambda_1, \lambda_2, \ldots, \lambda_r\}$ to be the set of eigenvalues whose corresponding subsystems (2.26) are not $2s$-sparse observable.
- We define $\mathcal{J}_2 \subseteq \{\lambda_1, \lambda_2, \ldots, \lambda_r\} \setminus \mathcal{J}_1$ to be the set of eigenvalues whose corresponding subsystems (2.26) are $2s$-eigenvalue observable.
- We define $\mathcal{J}_3 = \{\lambda_1, \lambda_2, \ldots, \lambda_r\} \setminus \{\mathcal{J}_1 \cup \mathcal{J}_2\}$ to be the set of eigenvalues whose corresponding subsystems (2.26) are $2s$-sparse observable but not $2s-$eigenvalue observable.

2.6.1 Impossibility of Reconstructing Substates Corresponding to Eigenvalues in the Set $\mathcal{J}_1$

It is established in Section (2.3) that the SSR problem does not admit a unique solution if it is not $2s-$sparse observable. Therefore, it is impossible to reconstruct the substates corresponding to eigenvalues in $\mathcal{J}_1$. Furthermore, by Theorem (4) if $\mathcal{J}_1$ is not empty, the

overall system defined in (2.5)-(2.6) is not $2s-$sparse observable, which in turn means the solution is not unique.

2.6.2 Reconstructing the Substates Corresponding to Eigenvalues in the Set $\mathcal{J}_2$

We learned from Theorem (5) that if λ_j is observable w.r.t. sensor i, then after decomposing the system, λ_j is also observable w.r.t. to sensor i in the j-th sub-system corresponding to this sensor. By the Popov-Belevitch-Hautus (PBH) test, the j-th sub-system $(\mathbf{A}^{(j)}, \mathbf{C}_i^j)$ is observable, which shows that $\mathbf{x}_j$ can be reconstructed using only measurements from sensor i.

We now explain how to reconstruct the substates corresponding to eigenvalues in $\mathcal{J}_2$ based on majority voting. Consider any eigenvalue $\lambda_j \in \mathcal{J}_2$. Let S_{λ_j} represent the set of sensors w.r.t. which λ_j is observable. The result of the PBH test implies that $\mathbf{x}_j$ can be recovered using the measurements of each of the sensors in the set $\mathcal{S}_{\lambda_j}$. We denote by $x_j^{(l)}$ the lth component of $\mathbf{x}_j$. Based on the definition of the set $\mathcal{J}_2$, we have $|\mathcal{S}_{\lambda_j}| \geq (2s+1)$. Consequently, since at most s sensors have been compromised, we are guaranteed at least $s+1$ consistent copies of the state $x_j^{(l)}$. Thus, each component of the vector $x_j^{(l)}$ can be recovered via majority voting and therefore all the substates corresponding to eigenvalues in $\mathcal{J}_2$ can be reconstructed in polynomial time.

2.6.3 Computational Complexity of Reconstructing Substates Corresponding to Eigenvalues in the Set $\mathcal{J}_3$

The NP-hardness of solving the SSR problem has been established in Section 2.4. In this subsection, we argue that with the prescribed decomposition technique, the computational complexity of solving the SSR problem for substates corresponding to eigenvalues in $\mathcal{J}_3$ could be reduced whenever we only need to reconstruct substates whose dimension is smaller than n. Assuming s is the upper bound of the number of attacked sensors, we have the following

theorem.

Theorem 6 *By applying the decomposition* (2.26)*, the SSR problem can be solved in time* $\sum_{\lambda_j \in \mathcal{J}_3} \mathcal{C}(p, n_j) + O(pn^3)$ *if the system* (2.5)-(2.6) *is* $2s-$*sparse observable, where* $\mathcal{C}(p, n)$ *is the time complexity of solving an instance of the SSR problem with* n *states and* p *sensors whose corresponding system is* $2s-$*sparse observable.*

Before providing a proof we first discuss how this result may reduce the computational complexity of solving the SSR problem. For a large-scale CPS, it's not uncommon for the number of sensors to greatly exceed the number of states, i.e., $p \gg n$. We note that the computational complexity of brute force search grows exponentially with p. Also, the computational complexity of some brute force search algorithms (such as [CWH15]) to determine whether a set of sensors is attacked is at least $O(n^2)$. In other words, for such algorithms $\mathcal{C}(p, n) \geq O(p^2 n^2)$. By assuming $p \gg n$ we make the following observations:

1. $O(p^2 n^2) \geq \sum_{j=1}^{r} O(p^2 n_j^2)$, and equality holds only when $r = 1$.
2. $O(pn^3) \ll \sum_{j=1}^{r} O(p^2 n_j^2)$.

The first observation shows that the computation required to solve all the sub-problems is smaller than what is required to solve the original problem. The second observation shows that, compared with the computational complexity of solving the SSR problem, the computation required for decomposition of the original system is negligible. These two facts indicate that by decomposing the SSR problem into simpler instances, we reduce the computational complexity of solving the SSR problem.

Proof of Theorem 6: We already established that reconstructing the state of each decomposed system is also an SSR problem and the solution $\mathbf{x}$ of the original problem is obtained by summing over all the projections, i.e., $\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_r$. Therefore any algorithm that solves the SSR problem can be applied to solve each subproblem, i.e., we may solve

each subproblem corresponding to $\lambda_j \in \mathcal{J}_3$ within time complexity $\mathcal{C}(p, n_j)$ since there are p sensors and n_j states. By the assumption that the system (2.5)-(2.6) is $2s-$sparse observable as well as Theorem (4), all sub-systems are $2s-$sparse observable and hence $\mathcal{J}_1 = \{\phi\}$, and for each subproblem corresponding to $\lambda_j \in \mathcal{J}_2$ the time complexity of the majority voting algorithm is within $O(pn^2)$. In summary, the total computational complexity is:

$$\sum_{\lambda_j \in \mathcal{J}_2} O(pn_j^2) + \sum_{\lambda_j \in \mathcal{J}_3} \mathcal{C}(p, n_j) + O(pn^3) \tag{2.29}$$

$$= \sum_{\lambda_j \in \mathcal{J}_3} \mathcal{C}(p, n_j) + O(pn^3), \tag{2.30}$$

which finishes the proof.

□

Remark 1 *The actual complexity might be even smaller than* $\sum_{\lambda_j \in \mathcal{J}_3} \mathcal{C}(p, n_j) + O(pn^3)$. *This can be seen by noting that we solve each smaller SSR problem sequentially, and thus we can remove measurements from sensors that have been identified as being attacked when solving subsequent problems.*

To conclude, we have the following result which answers the question at the beginning of this section by pointing out when the SSR problem can be solved in polynomial time, which actually is a corollary of Theorem (6).

Corollary 1 *Consider the system* (2.5)-(2.6)*, and suppose at most* s *sensors are attacked. Let the eigenvalue observability index of system* (2.5)-(2.6) *be at least* $2s$*. Then, the SSR problem can be solved in polynomial time.*

Remark 2 *Another understanding of this classification of eigenvalues into* $\mathcal{J}_1$*,* $\mathcal{J}_2$*, and* $\mathcal{J}_3$ *is provided by the vulnerability of the corresponding substates. Substates in* $\mathcal{J}_1$ *are the most vulnerable to attack since the defender may not even be able to identify the attacked set of sensors. Substates in* $\mathcal{J}_2$ *are robust against attacks since attacked sensors can be easily*

determined. For substates $\mathcal{J}_3$, the defender is able to identify the attacked sensors, but this task requires a substantially higher computational effort.

In other words, in the view of the adversary, a wise attacking strategy is to attack the substates corresponding to eigenvalues in $\mathcal{J}_1$, and it should avoid attacking states in $\mathcal{J}_2$ since majority voting will allow the defender to easily identify the compromised sensors.

2.6.4 Example - Continued

In this subsection we continue the example in Section 2.2 and Section 2.5 and show how to classify each subsystem under the assumption that the adversary can attack at most $s = 1$ sensor. We recall that V^1, V^2, V^3 are the eigenspaces corresponding to eigenvalues 1, 2, and 3, respectively. Also, after decomposition, we have $\mathbf{A}^{(j)} = \mathbf{P}_j \mathbf{A}|_{V^j}$ as well as $\mathcal{O}_i^j = \widetilde{\mathbf{P}}_i^j \mathcal{O}_i|_{V^j}$ for $i = 1, 2, 3, 4$ and $j = 1, 2, 3$.

We first claim that $\lambda_3 = 3$ belongs to $\mathcal{J}_1$. To see why this is true, we remove $2s = 2$ sensors, sensor 1 and sensor 4, and explicitly compute $\mathcal{O}_2^3$ and $\mathcal{O}_3^3$. We have:

$$\mathcal{O}_2 = \begin{bmatrix} 2 & 3 & 1 & -1 \\ 3 & 4 & 0 & -1 \\ 5 & 6 & -2 & -1 \\ 9 & 10 & -6 & -1 \end{bmatrix}, \mathcal{O}_3 = \begin{bmatrix} 2 & 2 & 0 & 0 \\ 3 & 3 & -1 & 0 \\ 5 & 5 & -3 & -0 \\ 9 & 9 & -7 & 0 \end{bmatrix},$$

and $\mathcal{O}_2(V^3) = \mathcal{O}_3(V^3) = \{0\}$ which yields $(\widetilde{\mathbf{P}}_2^3 \mathcal{O}_2)\mathbf{x}_3' = 0$ and $(\widetilde{\mathbf{P}}_3^3 \mathcal{O}_3)\mathbf{x}_3'' = 0$ for any $\mathbf{x}_3'$ and $\mathbf{x}_3''$ in V^3. Therefore, we have $\mathcal{O}_2^3 = \mathcal{O}_3^3 = 0$. By the definition of sparse observability, we have $\ker \mathcal{O}_{\{2,3\}}^3 = V^3$ and hence the subsystems corresponding to eigenvalue 3 are not $2s-$sparse observable. Also, a similar analysis reveals that subsystems corresponding to eigenvalues λ_1 and λ_2 are both $2s-$sparse observable, hence $1 \notin \mathcal{J}_1$ and $2 \notin \mathcal{J}_1$.

Next we argue that $\lambda_2 = 2$ belongs to $\mathcal{J}_2$. To see why this is true, we first recall that

$\mathbf{A}^{(2)} = \mathbf{P}_2\mathbf{A}|_{V^2}$, $\mathbf{I}_4^{(2)} = \mathbf{I}_4|_{V^2}$, $\mathbf{C}_i^2 = \mathbf{C}_i|_{V^2}$, and then check that for sensor 1, the matrix:

$$\begin{bmatrix} \mathbf{A}^{(2)} - 2\mathbf{I}_4^{(2)} \\ \mathbf{C}_1^2 \end{bmatrix} = \begin{bmatrix} -2 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 \\ -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & -2 \\ 3 & 2 & 0 & 2 \end{bmatrix}\Bigg|_{V^2},$$

defines an injective map. We also run the same check on sensor $2, 3$, and 4 to conclude that eigenvalue λ_2 is observable by all 4 sensors. Hence the subsystems corresponding to λ_2 are $2s-$eigenvalue observable. Proceeding in the same fashion we conclude that subsystems corresponding to eigenvalue λ_1 are not $2s-$eigenvalue observable. Therefore, the eigenvalue $\lambda_1 = 1$ belongs to $\mathcal{J}_3$.

In summary, the substates in V^3 cannot be securely reconstructed, the substates in V^1 can be securely reconstructed in the presence of at most 1 attacked sensor, and the substates in V^2 can be securely reconstructed and the reconstruction can be done efficiently.

2.7 Complexity of Checking Sparse Observability

In the previous two sections, we studied the complexity of the SSR problem, and in particular, identified instances of the problem that can be solved in polynomial time. Recall that under at most s sensor attacks on the system (2.5)-(2.6), $2s$-sparse observability is necessary and sufficient for the SSR problem to yield a unique solution, namely the true initial state vector $\mathbf{x}[0]$. Given this result, we now take a step back and ask: what is the complexity of deciding whether a given system is $2s$-sparse observable? This question is highly relevant since it aims to identify the maximum number of sensor attacks that can be tolerated by a given system of the form (2.5)-(2.6). In what follows, we show that determining the sparse-observability index (see Definition 1) of a system is computationally hard; we will focus on the case of scalar-valued sensors throughout, as it suffices to establish the computational complexity of

the problem.

Problem 3 *(r-sparse observability)*
Input: *A matrix* $\mathbf{A} \in \mathbb{Q}^{n \times n}$*, a matrix* $\mathbf{C} \in \mathbb{Q}^{p \times n}$ *and a positive integer* r*.*
Question: *Is the pair* $(\mathbf{A}, \mathbf{C})$ *r-sparse observable?*

Note that if the answer to an instance of the r-sparse observability problem is "no", then there is a simple proof: one can provide a set of r rows of $\mathbf{C}$ that, if removed, result in a system that is no longer observable. However, it is not clear whether there is a similarly simple proof for "yes" instances. Thus, the r-sparse observability problem is in the class coNP.[1]

The complement of a decision problem is the problem obtained by switching the "yes" and "no" answers to all instances of that problem. If a problem is in the class coNP, then its complement is in the class NP, and vice versa.

We will show that the r-sparse observability problem is coNP-hard by showing that its complement is NP-hard. Specifically, we define the following complement problem to r-sparse observability.

Problem 4 *(r-sparse unobservability)*
Input: *A matrix* $\mathbf{A} \in \mathbb{Q}^{n \times n}$*, a matrix* $\mathbf{C} \in \mathbb{Q}^{p \times n}$ *and a positive integer* r*.*
Question: *Is there a set of* r *rows that can be removed from* $\mathbf{C}$ *in order to yield a matrix* $\bar{\mathbf{C}}$ *such that* $(\mathbf{A}, \bar{\mathbf{C}})$ *is unobservable?*

Note that the answer to an instance of r-sparse unobservability is "yes" if and only if the answer to the corresponding instance of r-sparse observability is "no" and vice versa. Further note that r-sparse unobservability is in the class NP.

[1] See, e.g., [CLR09] for additional details on the complexity classes NP and coNP.

We show that r-sparse unobservability is NP-complete by providing a reduction from the following *Linear Degeneracy* problem. This problem was shown to be NP-complete in [Kha95].

Problem 5 *(Linear Degeneracy [Kha95])*
Input: *A full column rank matrix* $\mathbf{F} \in \mathbb{Q}^{p \times n}$.
Question: *Does* $\mathbf{F}$ *contain a degenerate (i.e., noninvertible)* $n \times n$ *submatrix?*

In other words, the linear degeneracy problem asks whether it is possible to remove $p-n$ rows from matrix $\mathbf{F}$ so that the resulting (square) matrix is not full rank. We are now ready to prove the following result.

Theorem 7 ([MMS19]) *The r-sparse unobservability problem is NP-complete. Thus, the r-sparse observability problem is coNP-complete.*

Proof. Given an instance of the linear degeneracy problem (with matrix $\mathbf{F} \in \mathbb{Q}^{p \times n}$), we construct an instance of the r-sparse unobservability problem as follows: set $\mathbf{A} = \mathbf{I}_n$, $\mathbf{C} = \mathbf{F}$, and $r = p - n$.

We now show that the answer to the constructed instance of r-sparse unobservability is "yes" if and only if the answer to the given instance of linear degeneracy is "yes".

First, suppose that the answer to the constructed instance of r-sparse unobservability is "yes." Then there exists a set of r rows of $\mathbf{C}$ that can be removed such that the remaining rows are not sufficient to yield observability. However, since $\mathbf{A} = \mathbf{I}_n$, the above implies that there is a set of r rows of $\mathbf{C}$ that can be removed such that the remaining rows are not full column rank. Since $\mathbf{C} = \mathbf{F}$ and $r = p - n$, this means that there is an $n \times n$ submatrix of $\mathbf{F}$ that loses rank, and thus the answer to the linear degeneracy problem is "yes."

Next, we show that if the answer to the given instance of linear degeneracy is "yes," then the answer to the constructed instance of r-sparse unobservability is "yes." We will

do this by showing the contrapositive: if the answer to the constructed instance of r-sparse unobservability is "no", then the answer to the given instance of linear degeneracy is "no." Suppose the answer to the constructed instance of r-sparse unobservability is "no." Then, by definition, the pair $(\mathbf{A}, \mathbf{C})$ is observable even after removing any arbitrary r rows from $\mathbf{C}$. However, since $\mathbf{A} = \mathbf{I}_n$, in order for the system to remain observable after removing r rows from $\mathbf{C}$, it must be the case that the remaining rows of $\mathbf{C}$ have full column rank. Thus, if the answer to the constructed instance of r-sparse unobserability is "no", then $\mathbf{C}$ has full column rank after removing any arbitrary $r = p - n$ rows. This means that every $n \times n$ submatrix of $\mathbf{C}$ is invertible. Since $\mathbf{C} = \mathbf{F}$, the answer to the given instance of linear degeneracy is "no" (i.e., there is no $n \times n$ submatrix of $\mathbf{F}$ that is degenerate).

Thus, we have shown that the answer to the constructed instance of r-sparse unobservability is "yes" if and only if the answer to the given instance of linear degeneracy is "yes". Since linear degeneracy is NP-complete, so is r-sparse unobservability.

Finally, since r-sparse observability is the complement of r-sparse unobservability, we have that r-sparse observability is coNP-complete. □

Remark 3 *In [MS19], certain necessary conditions were presented for estimating the state of a plant despite attacks in a distributed setting, i.e., where measurements of the plant are dispersed over a network of sensors. Specifically, these conditions impose certain requirements on the observation model (in addition to requirements on the communication structure), the complexity of checking which was left open. Interestingly, Theorem 7 resolves this question, and establishes that checking the necessary conditions in [MS19] is computationally hard; since the focus of our chapter is on centralized systems, we do not present details of this result here.*

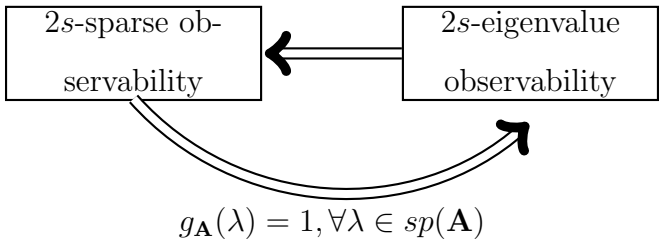

Figure 2.1: Figure illustrating the hierarchy of relationships between different notions of observability.

2.8 Connections Between Sparse Observability and Eigenvalue Observability

In Sections 2.4 and 2.7, we showed that the SSR problem and the problem of determining the sparse observability index of a system are each computationally hard. At the same time, Section 2.6 gave us the positive result that certain instances of the SSR problem can be efficiently solved. In line with this finding, we are now motivated to ask: Can the sparse observability index of a system be computed in polynomial time for certain specific instances? In this section, we show that this is indeed the case by identifying instances of the problem where the notions of sparse observability and eigenvalue observability coincide. Given that the eigenvalue observability index of a system can always be computed in polynomial time based on simple rank tests, an equivalence between the two notions of observability immediately yields instances of the problem where the sparse observability index of the system can also be computed in polynomial time. With this in mind, in this section we will prove each of the implications indicated in Figure 2.1. We begin with the following simple result.

Proposition 3 ([MMS19]) *Consider the linear system* (2.5)-(2.6)*, and suppose its eigenvalue observability index is* $2s$*. Then, the pair* $(\mathbf{A}, \mathbf{C})$ *is at least* $2s$*-sparse observable.*

Proof. Consider any subset of sensors $\mathcal{F} \subset \mathcal{V}$, such that $|\mathcal{F}| \leq 2s$. To establish that the pair $(\mathbf{A}, \mathbf{C})$ is at least $2s$-sparse observable, we need to show that the pair $(\mathbf{A}, \mathbf{C}_{\mathcal{V}\setminus\mathcal{F}})$ is observable.

Based on the PBH test, this amounts to checking that each eigenvalue $\lambda \in sp(\mathbf{A})$ is observable w.r.t. the observation matrix $\mathbf{C}_{\mathcal{V}\setminus\mathcal{F}}$. Let S_λ represent the set of sensors w.r.t. which λ is observable. A sufficient condition for this to happen is $|(\mathcal{V} \setminus \mathcal{F}) \cap \mathcal{S}_\lambda| \geq 1$, which is indeed true given that an eigenvalue observability index of $2s$ implies $|\mathcal{S}_\lambda| \geq (2f+1), \forall \lambda \in sp(\mathbf{A})$, and the fact that $|\mathcal{F}| \leq 2s$. □

To see that the reverse implication does not hold in general, consider the following example.

Example 1 *Consider an LTI system of the form* (2.5)-(2.6) *monitored by 6 sensors, with parameters as follows:*

$$\mathbf{A} = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}, \mathbf{C}_i = \begin{cases} \begin{bmatrix} 1 & 0 \end{bmatrix}, & \text{if } i \in \{1,2,3\}, \\ \\ \begin{bmatrix} 0 & 1 \end{bmatrix}, & \text{if } i \in \{4,5,6\}. \end{cases} \tag{2.31}$$

Here $\lambda \in \mathbb{R}, |\lambda| \geq 1$. Suppose $s = 1$. Then, the removal of at most 2 sensors will ensure that at least one sensor from each of the sets $\{1,2,3\}$ and $\{4,5,6\}$ remains unattacked; given the measurement model in (2.31)*, this is sufficient to preserve observability w.r.t. the remaining sensors. In other words, the system is 2-sparse observable. However, it is easy to verify that the eigenvalue λ is not observable w.r.t. any sensor.*

In view of Proposition 3 and Example 1, we conclude that $2s$-sparse observability of a system is in general less restrictive than the condition that the eigenvalue observability index of the system is $2s$. In what follows, we establish that the two aforementioned notions coincide when additional structure is imposed on the spectrum of $\mathbf{A}$.

Proposition 4 ([MMS19]) *Consider the linear system model given by* (2.5)-(2.6)*, and suppose $\lambda \in sp(\mathbf{A})$ has geometric multiplicity* 1*. Consider any non-empty subset of sensors*

$\mathcal{S} = \{i_1, i_2, \ldots, i_{|\mathcal{S}|}\} \subseteq \mathcal{V}$. *Then, the eigenvalue* λ *is observable w.r.t. the pair* $(\mathbf{A}, \mathbf{C}_{\mathcal{S}})$ *if and only if there exists a sensor* $i_p \in \mathcal{S}$ *such that* λ *is observable w.r.t. sensor* i_p*, i.e.,* λ *is observable w.r.t. the pair* $(\mathbf{A}, \mathbf{C}_{i_p})$.

Proof. Consider a similarity transformation that maps $\mathbf{A}$ to its Jordan canonical form $\mathbf{J}$. Let this transformation map $\mathbf{C}_{\mathcal{S}}$ to $\bar{\mathbf{C}}_{\mathcal{S}}$, and $\mathbf{C}_{i_j}$ to $\bar{\mathbf{C}}_{i_j}$, for each $i_j \in \mathcal{S}$. Since λ has geometric multiplicity 1, there exists a single Jordan block corresponding to λ in $\mathbf{J}$. Let this Jordan block be denoted $\mathbf{J}_\lambda$. Without loss of generality, suppose $\mathbf{J}$ is of the following form:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_\lambda & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{J}} \end{bmatrix}, \tag{2.32}$$

where $\bar{\mathbf{J}}$ is the collection of the Jordan blocks corresponding to eigenvalues in $sp(\mathbf{A})\backslash\{\lambda\}$. Based on the PBH test, λ is observable w.r.t. the pair $(\mathbf{J}, \bar{\mathbf{C}}_{\mathcal{S}})$ if and only if the following condition holds:

$$\operatorname{rank} \begin{bmatrix} \mathbf{J} - \lambda \mathbf{I}_n \\ \bar{\mathbf{C}}_{\mathcal{S}} \end{bmatrix} = n. \tag{2.33}$$

Given the structure of $\mathbf{J}$ in (2.32), and the fact that λ has geometric multiplicity 1, it is easy to see that (2.33) holds if and only if there is at least one non-zero entry in the first column of $\bar{\mathbf{C}}_{\mathcal{S}}$. However, the preceding condition holds if and only if there exists some sensor $i_p \in \mathcal{S}$ with at least one non-zero entry in the first column of $\bar{\mathbf{C}}_{i_p}$; the latter is precisely the condition for observability of λ w.r.t. the sensor i_p, given that $g_{\mathbf{A}}(\lambda) = 1$. To complete the proof, it suffices to notice that a similarity transformation preserves the observability of an eigenvalue. □

We now make use of the previous result to establish an equivalence between sparse observability and eigenvalue observability.

Proposition 5 *Consider the linear system model* (2.5)-(2.6)*, and suppose every eigenvalue of* $\mathbf{A}$ *has geometric multiplicity* 1*. Then, the pair* $(\mathbf{A}, \mathbf{C})$ *is* $2s$*-sparse observable if and only if the eigenvalue observability of the system is* $2s$*.*

Proof. For necessity, we proceed via contradiction. Suppose the pair $(\mathbf{A}, \mathbf{C})$ is $2s$-sparse observable, but there exists some $\lambda \in sp(\mathbf{A})$ that is observable w.r.t. at most $2s$ distinct sensors. Recall that the set of sensors w.r.t. which λ is observable is denoted $\mathcal{S}_\lambda$. Based on our hypothesis, $|\mathcal{S}_\lambda| \leq 2s$. Suppose $|\mathcal{S}_\lambda| = 2s$ (since an identical argument can be sketched when $|\mathcal{S}_\lambda| < 2s$). Since $(\mathbf{A}, \mathbf{C})$ is $2s$-sparse observable, the pair $(\mathbf{A}, \mathbf{C}_{\mathcal{V}\setminus\mathcal{S}_\lambda})$ is observable. However, based on Proposition 4, this requires λ to be observable w.r.t. at least one sensor in $\mathcal{V} \setminus \mathcal{S}_\lambda$, leading to the desired contradiction. This completes the proof of necessity. For sufficiency, note from Proposition 3 that the pair $(\mathbf{A}, \mathbf{C})$ is at least $2s$-sparse observable whenever its eigenvalue observability index is $2s$; the fact that the observability index is no more than $2s$ follows from the additional assumption on the geometric multiplicity of eigenvalues, and arguments similar to those used for establishing necessity. □

It directly follows from the definition of eigenvalue observability that the eigenvalue observability index of a system can be computed in polynomial time. Hence, we have the following corollaries of Proposition 5.

Corollary 2 *When all the eigenvalues of the matrix* $\mathbf{A}$ *have geometric multiplicity 1, the sparse observability index of the system can be computed in polynomial time.*

Corollary 3 *For a 2s-sparse observable system* (2.5)-(2.6)*, when all the eigenvalues of the matrix* $\mathbf{A}$ *have geometric multiplicity 1, the SSR problem can be solved in polynomial time.*

Proof. It is shown in Proposition 5 that under the unitary geometric multiplicity assumption, a $2s$-sparse observable system is also $2s$-eigenvalue observable. Thus, such a system satisfies the hypotheses in the statement of Theorem 1, and we immediately obtain the existence of a polynomial-time solution for the SSR problem. □

2.9 Conclusion

In this chapter, we showed that when the eigenvalues of the system matrix $\mathbf{A}$ have unitary geometric multiplicity, the SSR problem is tractable since both checking the sparse observability (see Corollary 2) as well as solving the SSR problem (see Theorem 1) can be performed in polynomial time. When at least one of the eigenvalues has geometric multiplicity greater than one, we can still compute the eigenvalue observability index and, if it is at least $2s$, solve the SSR problem in polynomial time if at most s sensors are attacked. However, in this case, eigenvalue observability is no longer necessary for the SSR problem to be solvable. Since even checking sparse observability is coNP-complete, we conjecture that the SSR problem may be intractable in this case. The authors are currently investigating this conjecture. However, even in this case, the computational complexity of solving the SSR problem can be reduced, when the system matrix $\mathbf{A}$ has at least 2 distinct eigenvalues.

CHAPTER 3

Secure State-Reconstruction Over Networks Subject to Attacks

3.1 Introduction

This chapter is concerned with the problem of reconstructing the state of a linear time-invariant system from measurements that have been corrupted by an adversary. Several examples of such attacks have been reported in the literature [DDZ15, DP17]. Moreover, the increasingly distributed and interconnected nature of Cyber-Physical Systems, including IoT devices, creates new opportunities for such attacks [CAS08, WOM19]. Hence the security of CPSs is a problem of vital importance [GUC18].

The problem studied in this chapter, state reconstruction despite attacks on the information to be processed, is termed the Secure State-Reconstruction problem [MMS19], which we will discuss in more detail in Section 3.2.5. Most literature on the SSR problem has focused on determining necessary and sufficient conditions on the number of attacked sensors for solvability [PDB13, BPG17], on efficient algorithms [FTD14, YFF16, ST15, SNS18, TDJ14, AY18, MG16, AY17], and more recently on the algorithmic complexity of this problem [MMS19].

Although none of the above chapters considers networks, it is known that communication channels in CPS are vulnerable to certain types of attacks [WOM19]. A related line of work, [MS19], [MRB19], considers the problem of distributed estimation with sensor measurements subject to attacks and, in particular, [MS19] presents necessary conditions for this problem to be solvable. We note that the SSR problem becomes much more intricate when the adversary

attacks not only the sensors but also the network that transports the sensor measurements to the location where they are processed. In this chapter we go beyond the results proposed in [MS19] by providing necessary and sufficient conditions for the SSR problem to be solved in the more general setting where both sensors and the communication network are under attack. A different line of work addresses the resiliency of communication networks while being agnostic to what the transmitted data will be used for, the most prominent line of work is secure network coding [YC06] where the amount of secure information rate can be expressed in term of min-cut values. For example, [CFP15] develops achievability protocols and outer bounds for the secure network coding problem where the edges are subject to packet erasures, and [LDJ18] explores the capacity region of a quadratically constrained channel corrupted by a causal adversary. Another example comes from the use of error-correction coding techniques, see [KK08, EV08], which can be used to detect attacked links and even recover information from partially corrupted messages.

In contrast with the previously cited literature that mostly focused on attacks only on sensors or attacks only on network links and nodes, in this chapter we consider the scenario where sensor measurements are subject to attacks and transported to a receiver location by a network whose nodes and links are also subject to attacks. In this context we ask: how many sensors, network links, and network nodes can be attacked while ensuring solvability of the SSR problem? We give a complete solution to this question by providing necessary and sufficient conditions for solvability of the SSR problem.

The rest of the chapter is organized as follows. In Section 3.2, we introduce the system model, the network model and give a formal definition of the SSR problem. As our main result, we introduce necessary and sufficient conditions for the SSR problem to be solvable in Section 3.3. The sufficiency of these conditions is proved in Section 3.4, by designing a coding algorithm, whereas necessity is proved in Section 3.5 by proposing an attacking strategy for the adversary/attacker.

3.2 Problem Setting

3.2.1 System Model

Consider a linear time-invariant dynamical system:

$$x[k+1] = Ax[k], \tag{3.1}$$

where $k \in \mathbb{N}$ is the time index, $x[k] \in \mathbb{R}^n$ is the state vector and $A \in \mathbb{R}^{n \times n}$ is the system matrix. The system is monitored by a set P of p sensors $\{w_1, w_2, \ldots, w_p\}$, and the ith sensor w_i measures the state according to:

$$y_i[k] = C_i x[k], \quad i = 1, 2, \ldots, p, \tag{3.2}$$

where $y_i[k] \in \mathbb{R}^{\nu_i}$ and $C_i \in \mathbb{R}^{\nu_i \times n}$. In the context of this chapter, the specific values of the observability indices do not play a role, hence we assume n to be the observability index for all (A, C_i) without loss of generality. By collecting n measurements, we can write the observations from the ith sensor as follows:

$$Y_i[k] = \mathcal{O}_i x[k], \tag{3.3}$$

where $Y_i[k] = \left[y_i^T[k]|y_i^T[k+1]|\ldots|y_i^T[k+n-1]\right]^T$ and $\mathcal{O}_i$ is the observability matrix of sensor i, which is defined by $\mathcal{O}_i = \left[C_i^T|(C_iA)^T|\ldots|(C_iA^{n-1})^T)\right]^T$.

3.2.2 Network model

As shown in Fig. (1), sensing nodes, i.e., network nodes equipped with sensors, transmit their measurements via a synchronous relay network to the receiver. We model the relay network, together with the sensing nodes, by a directed graph $G = (P \cup V, E)$, where P and V represent the set of sensing nodes and relay nodes (nodes in the relay network) respectively, and $(i, j) \in E$ represents the direct communication link from $i \in P \cup V$ to $j \in P \cup V$. Note that a node can both be a sensing node and a relay node, i.e., it is possible that $P \cap V \neq \varnothing$.

We assume the following properties about the network.

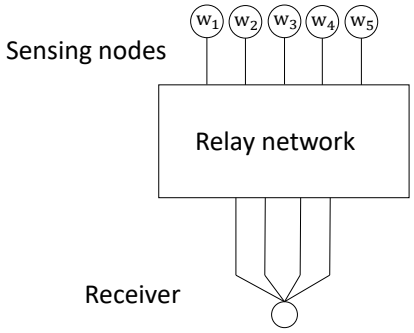

Figure 3.1: Measurements are transmitted to the receiver via a relay network.

1. The pair (A, C_P) is observable[1].

2. Each node has a unique identifier, and knows the identity of the in-neighbor from which it receives a message.

3. Each communication link has infinite capacity, i.e., we assume that real numbers can be transmitted through a link. Moreover, an attack-free link reliably transmits messages, i.e., messages are not lost, delayed, neither corrupted.

4. The set of attacked links and nodes does not change over time.

5. The network operates synchronously. In other words, all the nodes receive, compute, and transmit messages in a synchronized manner and this process takes exactly one time step.

Among all these assumptions, the system observability (assumption (1)) and stability of the attack (assumption (4)) are standard in SSR problem formulations, see [CWH15, ST15, PDB13]. The unique identifier and local knowledge of the network (assumption (2)) are also widely used, see, e.g, [MS19, SV16]. Moreover, the synchronous communication network model (assumption (5)) is widely accepted, see [FS07]. Although we make assumption (3)

[1]Recall that C_P is the matrix obtained from $C = \left[C_1^T | C_2^T | \dots | C_p^T\right]^T$ by removing all the rows whose indices are not in P.

to avoid working over finite fields, we note that our impossibility results for networks with infinite capacity directly carry over to networks with finite capacity. Also note that the network can be either wired, in which case nodes can send different messages to different out-neighbors at the same time instant; or wireless, where all out-neighbors of a node receive the same message from this node at the same time. More detailed network models, including asynchrony and links with capacities, make the problem more challenging and are left to future work.

3.2.3 Adversary Model

We consider that a subset $M_1 \subseteq E$ of links and a subset $M_2 \subseteq (P \cup V)$ of nodes is subject to active attacks. We assume the adversary is omniscient, i.e., it is aware of the system state, the measurements of all sensing nodes, the network topology, etc. In terms of capability, an attacked link is allowed to arbitrarily alter messages passing through it, including erasing, dropping or delaying the messages, and an attacked node may arbitrarily deviate from any prescribed rules and thus send any message to its out-neighbors. Attacked nodes are also allowed to send different messages to different out-neighbors at the same time instant under the wired network assumption. Moreover, attacked links and nodes are allowed to work cooperatively. The only assumption we make is that the power of the adversary is limited, i.e., the number of attacked links and nodes is upper bounded by f_1 and f_2. In other words, the following inequalities hold $|M_1| \leq f_1$ and $|M_2| \leq f_2$. Note that f_1 and f_2 are known to the receiver.

3.2.4 Definitions

Intuitively, in order for the receiver to reconstruct the state despite the existence of attacked nodes and links, the network must possess 2 kinds of redundancies: measurement redundancy and transmission redundancies, which can be formulated through the notions of critical set

(first introduced in [MS19]) and mix cut with respect to (w.r.t.) a critical set.

Definition 3 (Critical set) *A set $S \subseteq P$ is said to be a critical set if the pair $(A, C_{P\setminus S})$ is not observable.*

Definition 4 (Mix cut with respect to a critical set) *Consider a critical set S. A set $H_S \subseteq (P \cup V \cup E)$ is called a mix cut w.r.t. S if removal of $H_S \cap (V \cup E)$ from G disconnects the receiver and $H_S \cap P$. We denote by $L(H_S) = |H_S \cap E|$ the number of links in the mix cut and $N(H_S) = |H_S \cap (P \cup V)|$ the number of nodes in the mix cut.*

3.2.5 The Secure State-Reconstruction Problem

We now formally define the secure state-reconstruction problem studied in this chapter.

Problem 6 *(Secure state-reconstruction problem)*
Input: *a linear system, defined by* (3.1) *and* (3.2)*, and satisfying assumption 1, a network satisfying assumptions 2, 3, and 4, the numbers f_1 of attacked links and f_2 of attacked nodes.*
Question: *Is it possible for the receiver to reconstruct the initial value $x[0]$ of the state of the linear system, provided as input, from received messages, knowing that at most f_1 links and f_2 nodes in the network are subject to the actions of an attacker respecting assumption 5?*

Note that in our problem setting, the receiver has no knowledge about the network topology. Although the formal statement of the SSR problem only requires the initial value $x[0]$ of the state to be reconstructed, all the results in this chapter can be suitably extended to the reconstruction of the value $x[t]$ of the state at any time $t \in \mathbb{R}_0^+$.

3.3 Main Result

We present the main contribution of this chapter in the next result. Note that it can be conveniently extended to the case of multiple receivers (i.e., MIMO) who want to reconstruct

the same initial value $x[0]$ of the state.

Theorem 8 *The SSR problem is solvable if and only if for every critical set S and every mix cut H_S w.r.t S, the following bounds are satisfied:*

$$L(H_S) > 2f_1 \text{ or } N(H_S) > 2f_2.$$

Theorem (8) allows us to recover several results in the literature. To do so, however, we need to introduce the notion of minimum cut.

Definition 5 (Minimum cut) *A cut between a critical set S and the receiver is a set of links by removing which disconnects S from the receiver. A minimum cut L_S between S and the receiver is a cut with the smallest cardinality (if there are multiple, any of them is a minimum cut).*

If the attacked units are restricted to: (1) communication links, or (2) sensing nodes and relay nodes, we obtain the following corollaries.

Corollary 4 *If there are no attacked nodes, i.e., $f_2 = 0$, the SSR problem is solvable if and only if for every critical set S, $f_1 < |L_S|/2$.*

This corollary states that if the adversary is only able to attack communication links, then the SSR problem is solvable if and only if strictly less than half of the links in the minimum cut between any critical set S and the receiver are attacked. Similar results are known in the network coding literature [YC06], although the setting is slightly different due to finite capacity constraints.

Corollary 5 *If there are only attacked nodes, i.e., $f_1 = 0$, the SSR problem is solvable if and only if removal of any subset of $2f_2$ nodes does not disconnect any critical set S and the receiver.*

Note that the "only if" part of this corollary has been proved in Theorem 1 in [MS19]. Similarly to corollary (4), if there are only attacked nodes, then the SSR problem is solvable if and only if strictly less than half of the nodes in any cut between any critical set S and the receiver are attacked.

A preliminary version of this result when each node has one attack-free communication link to the receiver (i.e., $f_1 = 0$) was first proposed in [PDB13]. This simpler version of the SSR problem is proved to be solvable if and only if the pair (A, C) remaining observable even after the removal of any $2f_2$ sensing nodes. This is also known as $2f_2-$sparse observability, which was first proposed implicitly in [CWH15] and then explicitly in [ST15, SNS18].

3.4 Sufficiency: Coding and Decoding Algorithms

To show that the SSR problem is solvable when the sufficient condition is met, we design a coding algorithm and prove that the receiver is able to reconstruct the initial value $x[0]$ of the state.

3.4.1 Encoding Algorithm

We ask each node in the relay network to send the same message to all its out-neighbors. Each message is composed of several bulletins. Each bulletin has 2 sections: the information section and the routing section.

To start a transmission, each sensing node creates a bulletin with its measurement in the information section, and leaves the routing section blank. Whenever a message arrives from an in-neighbor, a relay node attaches the identifier of the in-neighbor at the end of the routing section of each bulletin of the received message, while keeping the information section untouched.

After receiving and processing messages from all its in-neighbors, the relay node creates

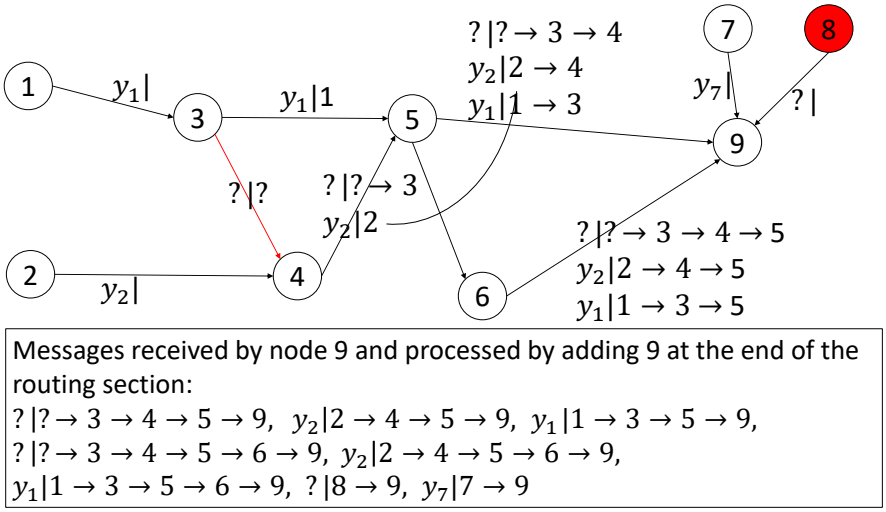

Figure 3.2: A simple illustration of the encoding algorithm.

a new message whose measurement section consists of the concatenation of the bulletins in the measurements sections of the processed packages. The sequence by which these bulletins are concatenated will be of no consequence for the algorithm's correctness. The relay node then sends this message to all its out-neighbors.

At last, the receiver attaches the identifier of the in-neighbor followed by its own identifier at the end of the routing section of each bulletin whenever it receives a message. For a bulletin that does not pass through any attacked link or node, the sequence of node identifiers in its routing section shows the route it has passed from the sensing node to the receiver. Therefore we regard two consecutive node identifiers in the routing section as a link.

Fig. 3.2 provides a simple illustration of our encoding algorithm. Since attacked links and nodes, which are colored in red, can arbitrarily change messages passing through, we denote by "?" the messages transmitted by them. All messages are represented on the figure. Note that the adversary should still keep the format of the message (i.e., bulletins, an information section and a routing section), otherwise the receiver or a relay node can easily detect it has been corrupted by an adversary and erase it. In this example, nodes 1, 2, 7, and 8 are sensing nodes, 3, 4, 5, and 6 are relay nodes, and node 9 is the receiver. Node 8 and the link

from node 3 to node 4 are attacked and denoted in red. As an illustration, node 6 sends to node 9 a message composed of 3 bulletins. Each bulletin has an information section and a routing section, divided by a vertical line. For example, in the third bulletin, $y_1|1 \to 3 \to 5$, y_1 is the value of the information section and $1 \to 3 \to 5$ is the value of the routing section.

The following Lemma is a direct consequence of the above described protocol.

Lemma 2 *If a message sent by a sensing node does not pass through any attacked link or attacked node, the receiver will receive a bulletin with the sensing node's measurement in the information section and the routing in the routing section.*

We will use this lemma to show that if a bulletin is relayed by attacked links or nodes, then at least one of them is recorded in its routing section.

Lemma 3 *For each bulletin in each received message by the receiver, if the first identifier i in the routing section does not belong to a sensing node, or the information section value $\tilde{y}_i[0]$ does not satisfy $\tilde{y}_i[0] = C_i x[0]$, then there is at least 1 attacked link or node in the routing section.*

Proof. By Lemma (2) we know that if a bulletin from node w_i does not pass through attacked links and nodes, the value in the information section of the corresponding bulletin satisfies $\tilde{y}_i[0] = C_i x[0]$. In other words, if $\tilde{y}_i[0] \neq C_i x[0]$ then this bulletin must have been altered and relayed by some attacked links or nodes. □

Now we show that the last attacked link or node this bulletin passes must be recorded in the routing section of the bulletin. Consider the first non-attacked node (note that it can also be the receiver) this bulletin passes through after being altered by the last attacked link or node in its route. By assumption, the non-attacked node knows the identifier of its in-neighbors, this node will attach the in-neighbor's identifier in the routing section, and thereby automatically adds the link from the in-neighbor to itself, with at least one of them

being attacked. And since all upcoming nodes and links are non-attacked, the identifier of the attacked link or node will be transmitted to the receiver in the information section of the bulletin.

3.4.2 Decoding Algorithm

We first notice that for every bulletin with correct information, the receiver can determine the time at which the message was sent by counting the number of links the message transversed.

The receiver is able to reconstruct the initial value $x[0]$ of the state after it receives all bulletins generated by non-attacked sensing nodes at time instances $0, 1, \ldots, n-1$. To do this, the receiver analyzes each bulletin in each message and stores those sent between 0 and $n-1$ (both included) until it is convinced that all bulletins generated during this period and relayed by only non-attacked links and nodes are received[2].

The naive decoding algorithm is as follows:

Step 1. The receiver categorizes all stored bulletins into different classes according to the information in their routing sections: each different path corresponds to a different class.

Step 2. For each class j, the receiver computes the time instance when each bulletin in it was generated and then does the following:

If there is exactly one bulletin generated at each time instance $0, \ldots, n-1$, then the receiver stacks the values of the information section of each bulletin into the vector:

$$\tilde{Y}_i^j[0] = \left[\tilde{y}_i^T[0]|\tilde{y}_i^T[1]|\ldots|\tilde{y}_i^T[n-1]\right]^T$$

where $i \in P$ is the first identifier in the routing section of class j. Otherwise, the receiver removes all bulletins in class j from its storage.

Step 3. The receiver picks a set $L \subseteq (P \cup V \cup E)$ of f_1 links and f_2 nodes[3] and removes

[2]To do this the receiver should have a reasonable estimate of the size of the network.

[3]Note that the receiver does not need prior knowledge of the network since it can observe the identifiers

from storage any bulletin in a class whose routing section contains at least 1 element in L.

Step 4. The receiver then checks whether the remaining bulletins are consistent (the consistency of bulletins will be defined afterwards). If the remaining bulletins are consistent, the state value that is consistent with all the remaining bulletins is the correct initial value of the state. Otherwise, it restores those removed bulletins in step (3), then goes to step (3) and picks another L.

The previous algorithm used the notion of consistency between bulletins. A set of bulletins generating $\tilde{Y}_i^j, i \in P$ in step (2), is said to be consistent if there exists a state $\tilde{x}[0]$ such that[4]:

$$\tilde{Y}_i^j[0] = \mathcal{O}_i \tilde{x}[0] \tag{3.4}$$

for each $\tilde{Y}_i^j[0]$ in step (2). In this case we also say the state $\tilde{x}[0]$ is consistent with these bulletins[5].

Before proving correctness of the decoding algorithm we return to the example in Subsection 3.4.1 (see Fig. (2)) to illustrate how the decoding algorithm works.

We consider a scalar system with trivial dynamics (i.e., $A = 1$ and $C_i = 1$) for all sensing nodes which yields $y_i = x$. We use the same network as in Fig. (2) in Subsection 3.4.1. All messages are represented on the figure. Recall that a vertical line divides each bulletin into an information section and a routing section. In this example, we assume $x = 1$ hence $y_1 = y_2 = y_7 = 1$, and the attacked link changes the value of the information section from 1 to 2 whenever a bulletin passes through. Also, when node 8 generates a new bulletin, it places value 2 in the information section.

The receiver node 9 receives 9 bulletins in total, among which the first, fourth and eighth are attacked. Since the receiver knows that at most 1 link and 1 node are attacked, it

---

of nodes and links, by analyzing the messages (including those attacked ones) it received.

[4]Again we note that k is the time index when the message is sent instead of received.

[5]The presence of bounded noise would not affect the results substantially since instead of checking equality (3.4) we would check instead $|\tilde{Y}_i^j[0] - \mathcal{O}_i \tilde{x}[0]| < \epsilon$ where ϵ is chosen based on the noise magnitude.

selects a set composed of 1 link and 1 node and removes the bulletins containing them. For example, if the receiver picks link $3 \to 5$ and node 4, and then removes all bulletins whose routing section contains them, the only remaining bulletins are $2|8 \to 9$ and $1|7 \to 9$, which are inconsistent. Then by step 4 in the decoding algorithm, the receiver restores all removed messages and chooses another set. Since the combinations of 1 link and 1 node are finite, the receiver will eventually choose link $3 \to 4$ and node 8. Then, by removing all the bulletins whose routing section contains $3 \to 4$ and 8, the receiver concludes that the remaining bulletins are consistent and thus obtains the correct initial value $x[0]$ of the state.

3.4.3 Correctness of Decoding Algorithm

Lemma 4 *If L, the set the receiver picks in step (3) of the decoding algorithm, contains all attacked links and attacked nodes, the remaining bulletins in step (3) of the algorithm are consistent and the initial value $x[0]$ of the state is the unique vector satisfying $\tilde{Y}_i^j[0] = \mathcal{O}_i x[0]$ with j ranging through the bulletins of the messages whose routing section contains no elements in L.*

Proof. By Lemma (3), if all the attacked links and nodes are in L, all messages that might be incorrect are removed by the receiver. Hence by Lemma (2) the value of the information section $\tilde{y}_i$ of each bulletin whose routing section contains no elements of L satisfies $\tilde{y}_i[0] = C_i x[0]$, or equivalently, $\tilde{Y}_i^j[0] = \mathcal{O}_i x[0]$ since they are in the same class. In other words, the estimated $x[0]$ is consistent with all these bulletins.

To show that the solution $x[0]$ is unique, we assume, for the purpose of contradiction, the existence of a state $\tilde{x}[0]$, different from $x[0]$, and consistent with the bulletins whose routing section contains no elements in L, i.e., $\tilde{Y}_i^j[0] = \mathcal{O}_i \tilde{x}[0]$. We focus on the subset S of transmitting nodes defined by $S = \{w_i | \mathcal{O}_i(\tilde{x}[0] - x[0]) \neq 0\}$. By Assumption (1) in Section (3.2), S is non-empty. Note that S is a critical set since the vector space spanned by $\tilde{x}[0] - x[0]$ is in the unobservable space of the set $P \backslash S$, i.e., for any sensing node $w_i \in P \backslash S$

we have $\mathcal{O}_i x[0] = 0$. But we also notice that, by assumption, for this critical set S, every mix cut H_S satisfies $L(H_S) > 2f_1$ or $N(H_S) > 2f_2$. This implies that by removing L from the network there should be at least 1 node $w_i \in S$ whose measurements $Y_i^j[0] = \mathcal{O}_i x[0]$ appears in the message sections of a certain bulletin during this time at the receiver without being altered. This establishes the contradiction since $Y_i^j[0] - \mathcal{O}_i \tilde{x}[0] = \mathcal{O}_i(x[0] - \tilde{x}[0]) \neq 0$.

□

Lemma 5 *If there are bulletins with attacked measurements remaining in step 2 of the decoding algorithm, the remaining bulletins are inconsistent.*

Proof. We follow an argument similar to the one used in the proof of Lemma (4). For the purpose of contradiction, we assume the remaining measurements are consistent, i.e., there exists a state $\tilde{x}[0]$ satisfying $Y_i^j[0] = \mathcal{O}_i \tilde{x}[0]$ for all remaining bulletins. Since there is at least 1 bulletin with altered measurements at the receiver, $\tilde{x}[0] \neq x[0]$. Still we focus on the non-empty set of sensing nodes $S = \{w_i | \mathcal{O}_i(\tilde{x}[0] - x[0]) \neq 0\}$, which is also a critical set as discussed above. By the assumption that for all remaining bulletins $Y_i^j[0] = \mathcal{O}_i \tilde{x}[0]$, no measurement from $w_i \in S$ arrives at the receiver without being altered. In other words, L, together with the actual set of attacked links and nodes M, forms a cut between S and the receiver. Since both L and M have at most f_1 links and f_2 nodes, we reach a contradiction with the fact that any cut has at least $2f_1$ links or $2f_2$ nodes. □

The correctness of the algorithm is a natural consequence of Lemma (4) and Lemma (5). Moreover, since the proposed flooding protocol does not require relay nodes to transmit different messages to different out-neighbors, our algorithm applies both to wired and wireless networks.

3.5 Necessity: Strategy of the Adversary

To show that the SSR problem is unsolvable if there exists a cut containing $2f_1$ links and $2f_2$ nodes or less, we prove that there always exists an attacking strategy for the adversary that prevents the adversary from correctly reconstructing the state.

The adversary's strategy is as follows:

Step 1. Find a critical set S and a mix cut H_S such that $L(H_S) \leq 2f_1$ and $N(H_S) \leq 2f_2$.

Step 2. Fabricate another state $\hat{x}[0]$ such that $x[0] - \hat{x}[0]$ is in the unobservable space of $P \backslash S$.

Step 3. Attack f_1 links and f_2 nodes in H_S. We denote by H_S^{ho} the set of non-attacked links and nodes, and H_S^{mal} the set of attacked links and nodes. For each link and node in H_S^{ho}, the attack assigns a link or a node in H_S^{mal} without repetition to simulate its behavior (which will be illustrated next).

Therefore, each message passing through an attacked link (resp. node) is replaced with the message the honest link (resp. node) being simulated would have sent if the state was $\hat{x}$ instead of x.

To prove that such a strategy works for the adversary, we first notice that since S is a critical cut and the pair $(A, P \backslash S)$ is not observable, the state $\hat{x}$ described in step 2 is guaranteed to exist. Also the assignment in step 3 is feasible since there are more links and nodes in H_S^{mal} than in H_S^{ho}.

The receiver node is unable to distinguish the following 2 cases. **Case 1**: links and nodes in H_S^{mal} are attacked and the state is $x[0]$; and **Case 2**: links and nodes in H_S^{ho} are attacked and the state is $\hat{x}[0]$.

To prove this claim, we first notice that the measurements $Y_i^j[0]$ ($w_i \in P \backslash S$) do not help distinguish between these two cases, since $\mathcal{O}_i(x[0] - \hat{x}[0]) = 0$ implies that $Y_i^j[0] = \mathcal{O}_i x[0] = \mathcal{O}_i \hat{x}[0] = \hat{Y}_i^j[0]$ for all $w_i \in P \backslash S$. Therefore, the removal of sensing nodes $P \backslash S$ and the

corresponding links will not interfere with distinguishing these two cases.

Now we only focus on the cut H_S and the network from H_S to the receiver. We call each link and node in H_S a virtual sensing link and node respectively. The rest of the network is equivalent to the following: a set of virtual sensing nodes and links with at least half being attacked, and an error-free network connecting these virtual sources to the receiver. The correct information carried by virtual sources comes in pairs with an incorrect one (recall that the adversary assigns without repetition an attacked link and node to simulate the behavior of honest ones respectively), which is exactly the same except for the state x or $\hat{x}$ they encode.

In this case, if the receiver, somehow, decodes the correct message x, then in the scenario where everything is the same except that the adversary attacks H_S^{ho} using the same strategy, the receiver will conclude the state to be $\hat{x}[0]$. Therefore, the receiver cannot distinguish between these two cases.

Combining the results in Section 3.4 and 3.5 we obtain Theorem 8.

3.6 Conclusion

In this chapter we investigated the SSR problem under an attack model that allows for attacks on sensors and also on communication. We provided necessary and sufficient conditions for the SSR problem to be solvable that generalize the existing results for the case of sensor only attacks and of communication only attacks on synchronous networks.

CHAPTER 4

Decentralized Secure State-Tracking in Multi-Agent Systems

4.1 Introduction

Over the past decade the topic of decentralized state-tracking has received considerable attention, due to the increasingly decentralized nature of complex systems such as traffic networks and power grids. In this problem, a group of nodes is required to collectively track the state of a linear dynamical system using measurements from their own sensors and messages exchanged with neighboring nodes via a communication network.

In the absence of attacks, the decentralized state-tracking problem has been well studied [HTW18, KSC16, WM17, MS19]. However, reports on CPSs attacks [FR11, Win] remind us of how vulnerable these systems can be. Motivated by this consideration, in this chapter we tackle a more challenging version of the decentralized state-tracking problem where some nodes are subject to sensor attacks spoofing its measurements. We refer to this problem as the Decentralized Secure State-Tracking (DSST) problem.

4.1.1 Related Work

Closely related to the DSST problem is the Secure State-Reconstruction problem whose formal definition can be found in [MMS22]. Roughly speaking, in the SSR problem a central server, which has access to all measurements, is asked to reconstruct the state of a linear

system despite an attack on some sensors. A preliminary version of the SSR problem was first considered in [STJ10], and rigorously defined in [FTD11]. Moreover, the solvability of the SSR problem was settled in [FTD11,ST15,CWH15]. In particular, in [FTD11] and [CWH15] it is pointed out that in order to reconstruct the state in the presence of s attacked sensors, the linear system must remain observable after a removal of any subset of $2s$ sensors. This property of a linear system is referred to as $2s-$sparse observability in [ST15].

Although it has been known for long that the SSR problem is NP-hard [FTD14], much progress was reported on reducing the computational complexity of solving the SSR problem. Many works, such as [FTD14, ST15, MS19, MMS22], carve out subsets of the SSR problem instances which allow for a polynomial-time solution. More results on reducing the computational complexity of the SSR problem can be found in [YFF16] and [HMX19]. In particular, the solution of the SSR problem proposed in [SCW18], which is built upon satisfiability modulo theory, is of the utmost practical interest due to its good performance in simulations. We show in this chapter that, with our algorithm, any DSST problem can be reduced to an SSR problem thereby enabling the use of any of the aforementioned algorithms to solve the DSST problem.

Compared with the SSR problem, the DSST problem is much more complicated since there is no longer a central server thus implying that each node only has partial information obtained via its own sensors and messages exchanged with neighboring nodes. Although of significant importance, the current understanding of this more challenging DSST problem is limited. To the best of the authors' knowledge, only three chapters addressed the DSST problem. Moreover, they only solve a subset of the DSST problem since they make assumptions either on system dynamics, network, or both. For example, both [LKS20] and [MS19] make assumptions on system dynamics. In particular, they require the existence of a basis such that the unobservable space of all the (or enough) sensors is the span of a subset of this basis. We will refer this property as Scalar Decomposability, or SD in brief. Intrinsically, SD enables one to decompose a DSST problem into multiple sub-problems each associated

with a scalar system. Interestingly, [LKS20] and [MS19] exploit SD in different ways which leads to different types of solutions of the DSST problem. In [LKS20], with the help of SD, the DSST problem is formulated as a distributed convex optimization problem with time-varying loss function and a high-gain observer is proposed to reconstruct the state with the help of "blended dynamics approach" introduced in [LS20]. In contrast, [MS19] proposes a local filter which forces the estimate of an attack-free sensor to always lie in the convex hull of the estimates of its attack-free neighbors. Although this strategy allows extensions to defend against more powerful classes of attacks, it also places an additional assumption on the attacker capabilities by requiring that it does not attack too many neighbors of each network node. Moreover, the tracking algorithm proposed in [LKS20] has steady-state error. Compared with [MS19] and [LKS20], the state observer proposed in [HRS20] does not require SD, but it is still based on an assumption involving both system dynamics and network topology so it still only solves a subset of DSST problem instances. Moreover, it requires a communication frequency much higher than the sampling rate, which is not typical in applications.

4.1.2 Our Approach and Contributions

In this chapter, we study the DSST problem from a new perspective by relating it to the consensus problem. According to this perspective, the objective of each node is to reach consensus on the state of the system. Since the state evolves over time, the relevant type of consensus is the dynamic average consensus. A thorough literature review of the dynamic average consensus problem is provided in the kia2019tutorial chapter [KVC19] to which we refer all the interested readers for a discussion of the relevant literature, including [SOM05], [KCM15], and [BFL10].

Our solution of the DSST problem is based on the simple observation that instead of processing all measurements from all nodes in the network, a suitably compressed version of the measurements suffices for each node to reconstruct the state. In particular, in Section 4.5

we will show that reconstructing the state from compressed measurements can be formulated as a special case of the SSR problem and, hence, any algorithm that solves the most general case of the SSR problem can be used to solve the DSST problem. Therefore, a solution of the DSST problem is obtained provided each node can track the compressed version of the measurements. To achieve this goal, we draw inspiration from [BFL10] and design an observer that provides each node with error-free tracking of the compressed measurements.

We make the following contributions in this chapter:

1. We propose a necessary and sufficient condition for the DSST problem to be solvable.
2. We provide a solution to the most general case of the DSST problem.

Compared with [LKS20], our solution of the DSST problem does not require SD, and the tracking of the state is error-free. Compared with [HRS20], we do no rely on unnecessary assumptions regarding system dynamics or network topology, and we adopt the widely-accepted setting where the communication rate equals the sampling rate. The major disadvantage of our algorithm is its computational complexity. Our algorithm is combinatorial in the most general case. Although our algorithm runs in polynomial time when SD holds, it still requires more computations than the algorithms proposed in [MS19] and [LKS20]. Moreover, our solution of the DSST problem requires sufficiently fast sampling rates. We summarize the comparison[1] between our solution of the DSST problem and the solutions in [MS19], [LKS20], and [HRS20], in Table 1.

4.1.3 Chapter Organization

The organization of the chapter is as follows. The decentralized secure state-tracking problem is formulated in Section 4.2, including all the assumptions we made. In Section 4.3 we discuss

[1]Note that the computational complexity is measured by the number of additions and multiplications needed by the algorithm during one round of state update. Although the results in [LKS20] are based on a continuous-time representation of the system, computational complexity is assessed for its discrete-time version.

	[MS19]	[LKS20]	[HRS20]	Our algorithm
Constraints	SD and topology constraints	SD	Constraints on dynamics and topology	No constraints
Computational complexity	\	\	Polynomial but high	NP-hard
Computational complexity if SD holds	Polynomial	Polynomial	Not specified	Polynomial but higher than [LKS20]
Convergence	Asymptotic	\	Asymptotic	Exponential
Existence of steady-state error?	No	Yes	No	No
Working criterion	Continuous and Discrete-time	Continuous-time	High communication frequency	Requirement on sampling rate
Against Byzantine attacks?	Yes	No	No	No

Table 4.1: A comparison between solutions to the DSST problem in [MS19], [LKS20], [HRS20], and this chapter.

how to compress the measurements and state a result regarding the solvability of the DSST problem. Then in Section 4.4 we relate the DSST problem to the dynamic average consensus problem. We provide our solution to the DSST problem in Section 4.5. The chapter concludes with Section 4.7.

4.2 Problem Formulation and Key Idea

In this section we introduce the decentralized secure state-reconstruction problem.

4.2.1 System Model

We consider a linear time-invariant system monitored by a network of p nodes whose sensors are subject to attacks:

$$\begin{aligned} x[t+1] &= Ax[t], \\ y_i[t] &= C_i x[t] + e_i[t], \end{aligned} \tag{4.1}$$

where $x[t] \in \mathbb{R}^n$ is the system state at time $t \in \mathbb{N}$, $y_i[t] \in \mathbb{R}$ is the measurement of node i where $i \in P \triangleq \{1, 2, \ldots, p\}$, which is assumed to be a scalar, and the matrices A, and C_i

have appropriate dimensions.

The vector $e_i[t] \in \mathbb{R}$ models the attack on the sensor at node i (which we will refer to as sensor i for brevity). If sensor i is attacked by an adversary, then $e_i[t]$ can be arbitrary, otherwise, $e_i[t]$ remains zero for any t, and $y_i[t] = C_i x[t]$ holds which means node i receives correct measurements from its sensor. We also assume that the adversary is omniscient, i.e., the adversary has knowledge about the system model, the algorithm being executed at each node and, for any time slot t, the adversary knows the system state $x[t]$ and the measurements $y_i[t]$ from all nodes. The only assumption we make on the adversary is that it can only attack a fixed set of at most $s \in \mathbb{N}$ sensors. Note that this set of attacked sensors is unknown to any node in the network.

Collecting n consecutive measurements (the reason for which will be discussed in sections 4.4 and 4.5) over time, the output of sensor i can be written in a more compact form:

$$Y_i[t] = \mathcal{O}_i x[t] + E_i[t], \quad i = 1, \ldots, p, \tag{4.2}$$

where $Y_i[t]$ and $E_i[t]$ are obtained by stacking vertically over time the measurements of sensor i and the attack vector, respectively, and the matrix $\mathcal{O}_i$ is the observability matrix of sensor i. These three matrices are defined by:

$$Y_i[t] = \begin{bmatrix} y_i[t] \\ y_i[t+1] \\ \vdots \\ y_i[t+n-1] \end{bmatrix} \in \mathbb{R}^n, \; E_i[t] = \begin{bmatrix} e_i[t] \\ e_i[t+1] \\ \vdots \\ e_i[t+n-1] \end{bmatrix} \in \mathbb{R}^n,$$

$$\mathcal{O}_i = \begin{bmatrix} C_i \\ C_i A \\ \vdots \\ C_i A^{n-1} \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

In a similar way, we stack over nodes the measurements, the observability matrices, and

the attack vectors, from which a more concise representation of the linear system is obtained:

$$\begin{aligned} x[t+1] &= Ax[t], \\ Y[t] &= \mathcal{O}x[t] + E[t], \end{aligned} \tag{4.3}$$

where $Y[t]$, $\mathcal{O}$, and $E[t]$ are obtained by stacking vertically each $Y_i[t]$, $\mathcal{O}_i$, and $E_i[t]$, respectively, for $i \in \{1, 2, \ldots, p\}$, i.e.:

$$Y[t] = \begin{bmatrix} Y_1[t] \\ Y_2[t] \\ \vdots \\ Y_p[t] \end{bmatrix} \in \mathbb{R}^{pn}, \ \mathcal{O} = \begin{bmatrix} \mathcal{O}_1 \\ \mathcal{O}_2 \\ \vdots \\ \mathcal{O}_p \end{bmatrix} \in \mathbb{R}^{pn \times n},$$

$$E[t] = \begin{bmatrix} E_1[t] \\ E_2[t] \\ \vdots \\ E_p[t] \end{bmatrix} \in \mathbb{R}^{pn}.$$

We also note that since the adversary can only attack at most s nodes, and since for an attack-free sensor i we have $E_i[t] = 0$ for any t, the vector $E[t]$ is sparse at any t. In the end, we define the matrices $y[t] = \begin{bmatrix} y_1[t] & y_2[t] & \ldots & y_p[t] \end{bmatrix}^T \in \mathbb{R}^p$, $C = \begin{bmatrix} C_1^T & C_2^T & \ldots & C_p^T \end{bmatrix}^T \in \mathbb{R}^{p \times n}$ and $e[t] = \begin{bmatrix} e_1[t] & e_2[t] & \ldots & e_p[t] \end{bmatrix}^T \in \mathbb{R}^p$ for future use.

We assume that the communication between nodes in the network can be modeled by an undirected graph. We recall the definitions made in the introduction related to graph. Each node is modeled by a vertex $i \in \mathcal{V}$, and a communication link from node i to node j is modeled by an edge $(i, j) \in \mathcal{E}$ from vertex i to j. Since we assume the graph is undirected, $(i, j) \in \mathcal{E}$ implies $(j, i) \in \mathcal{E}$ which shows that node j can also send messages to node i.

Remark 4 *Although the linear system* (4.3) *is modelled without inputs, we note that, all results in this chapter can be conveniently extended to the case when the input is known to every node in the network.*

4.2.2 Assumptions

Here we list all the assumptions we use in this chapter. Some of them have already been discussed when introducing the adversary model.

Assumption 1 *The network can be modeled by a communication graph which is time-invariant, undirected and connected.*

Assumption 2 *The adversary is only able to attack at most s nodes. The set of attacked nodes remains constant over time.*

Assumption 3 *The system dynamics are known to all nodes in the network.*

Assumption 4 *The adversary is only able to change the measurements of the attacked nodes. Each attacked node still executes its algorithm correctly.*

The Assumptions 1 2, 3, and 4 are in line with the assumptions in [LKS20] and [HRS20] except that we also assume each node knows the C_i matrices of all other nodes throughout the network[2]. We also require the following assumptions:

Assumption 5 *All the measurements y_i are scalars.*

Assumption 6 *For any eigenvalue $m + n\mathbf{i}$ of A with magnitude greater than or equal to 1 and for any non-zero eigenvalue λ of the communication graph laplacian $\mathcal{L}$, the inequality $\left(m - \frac{\lambda^2}{\lambda^2_{\max}(\mathcal{L})}\right)^2 + n^2 < 1$ holds.*

Assumption 5 is not necessary but we adopt it for simplicity. It can be easily seen that a node whose sensor produces a measurement in $\mathbb{R}^p$ can be modeled as p virtual nodes, each having a scalar sensor.

[2]This assumption can easily be enforced by endowing each node with the relevant information during initialization.

Whenever the discrete-time linear system (4.1) is the time discretization of an underlying continuous-time linear system:

$$\dot{\tilde{x}} = \tilde{A}\tilde{x}, \tag{4.4}$$

Assumption 6 can be interpreted as a requirement on the sampling time τ used to obtain (4.1) from (4.4). If τ is small enough, then $A = e^{\tilde{A}\tau}$ can be made arbitrarily close to the identity matrix. In other words, by increasing the sampling rate, m can be made arbitrary close to 1 and n close to 0, and such a pair of m and n satisfies Assumption 6.

We also note that, Assumption 6 implies for any eigenvalue $m + n\mathbf{i}$ of A, the inequality $(m-1)^2 + n^2 < 1$ holds.

4.2.3 The Decentralized Secure State-Tracking Problem

In this section we provide the definition of the decentralized secure state-tracking problem.

In plain words, to solve the DSST problem, each node i must maintain a state estimate $\hat{x}_i[t]$ which converges asymptotically to the true state $x[t]$. We also refer to this property by saying that $\hat{x}_i[t]$ tracks $x[t]$. The rigorous definition of the DSST problem is as follows:

Definition 6 (Decentralized Secure State-Tracking Problem) *Consider a linear system subject to attacks* (4.1) *satisfying Assumptions 2-6 and a communication network satisfying Assumption 1. The decentralized secure state-tracking problem asks for an algorithm running at each node $i \in P$ with measurements $y_i \in \mathbb{R}$ and messages from neighboring nodes as its input, and such that its output $\hat{x}_i[t]$ satisfies:*

$$\lim_{t\to\infty} \|\hat{x}_i[t] - x[t]\| = 0.$$

Remark 5 *Differently from centralized SSR problem (see for example [MMS22] for its definition), the DSST problem does not require each node to explicitly know which subset of nodes in the network is attacked. For the SSR problem, it has been argued in [FTD14] that*

knowing the true state $x[t]$ and knowing the set of attacked sensors is equivalent, while this is not the case in the DSST problem setting.

Remark 6 *In the DSST problem setting, it is possible to require **all** nodes, including attacked nodes and attack-free nodes, to maintain a state estimate which asymptotically tracks the system state $x[t]$. This follows from Assumption 1 which restricts the adversary to only alter sensor measurements. In particular, nodes with spoofed measurements are still able to correctly execute their algorithms. Therefore, the attacked nodes are still able reconstruct the state $x[t]$ and may even determine that their own measurements have been altered.*

4.2.4 Key Idea

The key idea for solving the DSST problem is based on the simple observation that instead of having access to measurements $Y = \begin{bmatrix} Y_1^T & Y_2^T & \dots & Y_p^T \end{bmatrix}^T$ of all the sensors, a compressed version $(D \otimes I_n)Y$ of the measurements may suffice to reconstruct the state, where the measurement compression matrix $D \in \mathbb{R}^{v \times p}$ reduces the measurements from $\mathbb{R}^p$ to $\mathbb{R}^v$ with $v \leq p$. Compression is possible, in most cases, and thus v will be strictly smaller than p. We will elaborate on the feasible choices for a measurement compression matrix D in Section 4.3.

Equipped with the observation that the compressed version of measurements $(D \otimes I_n)Y$ suffices to reconstruct state, it is natural to ask: how can each node have access to the compressed measurements? We show how to reformulate this problem as a dynamic average consensus problem in Section 4.4 and an algorithm for each node to track $(D \otimes I_n)Y$ is provided in Section 4.5. As we shall see, the connectivity of the network affects the convergence speed of the tracking algorithm, which is the only role network topology plays in our solution to the DSST problem.

Lastly, in Section 4.5 we show how to reconstruct the state x from the compressed measurements $(D \otimes I_n)Y$ at each node. Since each node now has enough information to reconstruct the state, communication is no longer required in this step. We will prove that

by suitably introducing a slack variable, this problem can be reduced to an SSR problem. This observation implies that we may employ any algorithm for the SSR problem that does not require additional assumptions, even though the compression slightly changes the attack model. It then follows that, as each sensor's estimate of the compressed measurements converges, so does the state reconstructed from the estimated compressed measurements. These three steps provide a solution to the DSST problem.

4.3 Design of the Compression Matrix and Solvability of DSST

There are two considerations involved in the choice of the compression matrix D. On the one hand, in order to reduce communications and storage, we want the D matrix to have the least possible number of rows. On the other hand, the compressed measurements $(D \otimes I_n)Y$ must provide enough information for each node to correctly reconstruct the state. We start with the definition of sparse detectability with respect to a matrix, which is a generalization of sparse observability [ST15, CWH15] as well as sparse detectability [NM15] but stronger, as we will very soon see.

Definition 7 (Detectability [AM07]) *A pair (A, C) is detectable if all the unobservable eigenvalues of A are stable.*

In other words, if (A, C) is detectable, then for any two trajectories $x_1[t] = A^t x_1[0]$ and $x_2[t] = A^t x_2[0]$, equality $C(x_1[t] - x_2[t]) = 0$ holding for all $t \in \mathbb{N}$ implies $\lim_{t\to\infty}(x_1[t] - x_2[t]) = 0$. This property will be used in the proof of Theorem 7.

Definition 8 (Sparse detectability [NM15]) *The sparse detectability index of the system* (4.1) *is the largest integer k such that for any $\mathcal{K} \subseteq P$ satisfying $|\mathcal{K}| \geq p - k$ the pair $(A, C_{\mathcal{K}})$ is detectable. When the sparse detectability index is k, we say that system* (4.1) *is k-sparse detectable.*

In plain words, if we remove any subset of at most s sensors and the remaining system represented by the pair $(A, C_{\mathcal{K}})$ is still detectable, then we say that the original system is s-sparse detectable. We now present another perspective on sparse observability, paving the way for the definition of sparse detectability with respect to a matrix, which plays a critical role in our study. We first define the set:

$$\mathbf{Q}_s = \{L \in \mathbb{R}^{*\times p} | \ker(L) = \text{span } V, V \subseteq \mathbf{E}_p, |V| \leq s\},$$

that we use in the following equivalent definition of sparse detectability:

Definition 9 (Sparse detectability) *The sparse detectability index of the system* (4.1) *is the largest integer k such that the pair (A, LC) is detectable for any $L \in \mathbf{Q}_k$. When the sparse detectability index is k, we say that system* (4.1) *is k-sparse detectable.*

Intuitively, by left multiplying C by L we remove the measurements of a subset of s sensors without losing information from any remaining sensors. The equivalence between these two definitions of sparse detectability is trivial and, due to space limitations, we do not provide a proof. We proceed by introducing the new notion of sparse detectability with respect to a matrix:

Definition 10 (Sparse detectability with respect to a matrix) *Consider the system* (4.1)*, a matrix $D \in \mathbb{R}^{v\times p}$, and define the following set:*

$$\mathbf{P}_s = \{L \in \mathbb{R}^{*\times v} | \ker(L) = D(\textit{span } V), V \subseteq \mathbf{E}_p, |V| \leq s\}.$$

The sparse detectability index of the system (4.1) *with respect to D is the largest integer k such that the pair (A, LDC) is detectable for any $L \in \mathbf{P}_k$. When the sparse detectability index with respect to D is k, we say that system* (4.1) *is k-sparse detectable with respect to D.*

Remark 7 *We note that, sparse detectability with respect to I coincides with the definition of sparse detectability.*

For later use, we formalize the contrapositive of Remark 7 in the next lemma.

Lemma 6 *Any pair (A, C) that fails to be s-sparse detectable is not $s-$sparse detectable with respect to any matrix of compatible dimensions.*

Intuitively, the stacked measurement Y provides redundant information about the state x. Some information redundancy is necessary for the system to be resilient against s attacked sensors. The compression matrix D aims to compress down the redundant information while keeping enough redundancy to provide resilience against the attacks. In the following lemma we we characterize such compression matrices.

Lemma 7 *If there exists an algorithm producing a state estimate $\hat{x}$ satisfying $\lim_{t\to\infty} \|x[t] - \hat{x}[t]\| = 0$ for trajectory $x[t]$ of system (4.1) satisfying Assumptions 2-4 and starting at any initial condition in $\mathbb{R}^n$ only using $(D \otimes I_n)Y[t]$ as input and for any attack signal $e[t] \in \mathbb{R}^p$, then (A, C) is $2s-$sparse detectable with respect to D.*

Proof. For the sake of contradiction, we assume that (A, C) is not $2s-$sparse detectable with respect to D. By the definition of sparse detectability with respect to a matrix, there exist two state trajectories $\tilde{x}_1[t] = A^t\tilde{x}_1[0]$ and $\tilde{x}_2[t] = A^t\tilde{x}_2[0]$ such that $\lim_{t\to\infty}(\tilde{x}_1[t] - \tilde{x}_2[t]) \neq 0$ and $LDC(\tilde{x}_1[t] - \tilde{x}_2[t]) = 0$ for some $L \in \mathbf{P}_{2s}$ and any $t \in \mathbb{N}$. This also implies the existence of two $s-$sparse signals $e_1[t] \in \mathbb{R}^p$ and $e_2[t] \in \mathbb{R}^p$ such that the equation $D(C\tilde{x}_1[t] - C\tilde{x}_2[t] + e_1[t] - e_2[t]) = 0$ holds for any $t \in \mathbb{N}$, which is equivalent to:

$$u[t] \triangleq D(C\tilde{x}_1[t] + e_1[t]) = D(C\tilde{x}_2[t] + e_2[t]). \tag{4.5}$$

Consider the following 2 scenarios: (1) the state trajectory is $\tilde{x}_1[t]$ and the attack signal is $e_1[t]$; (2) the state trajectory is $\tilde{x}_2[t]$ and the attack signal is $e_2[t]$. By (4.5) we know that if a node only has access to $u[t]$ then it cannot distinguish between these 2 scenarios, and since $\lim_{t\to\infty}(\tilde{x}_1[t] - \tilde{x}_2[t]) \neq 0$, it is thus unable to track the state. □

Lemma 7 shows that it is necessary to pick the compression matrix D such that the pair (A, C) is $2s$-sparse detectable with respect to D, otherwise it is impossible to track the state from compressed measurements $(D \otimes I_n)Y$. Later in Lemma 8 we will see that this condition also suffices to solve the DSST problem. Anticipating the results in Section 4.5 we state here the main contribution of this chapter, $2s$-sparse detectability with respect to D is both necessary and sufficient to solve the DSST problem:

Theorem 9 *The DSST problem associated with the linear system subject to attacks defined in* (4.1) *satisfying Assumptions 2-6 and a communication network satisfying Assumption 1 is solvable only if the pair* (A, C) *is 2s-sparse detectable.*

Proof. This result comes easily from Lemma 7 by picking $D = I_p$. □

4.4 Reduction to Dynamic Average Consensus

One key step of our solution to the DSST problem is to have each node tracking the compressed measurements $(D \otimes I_n)Y$. To do this, we ask each node i to maintain an estimate vector $W_i = \begin{bmatrix}(W_i^1)^T & (W_i^2)^T & \dots & (W_i^v)^T\end{bmatrix}^T \in \mathbb{R}^{vn}$, whose j-th block, W_i^j, tracks the j-th linear combination of measurements $\sum_i d_{ji}Y_i$, where d_{ji} is the entry at the j-th row and i-th column of D.

In this section, and in the one that follows, we focus on the problem of tracking $(D_1 \otimes I_n)Y$, where D_1 is the first row of D, which can also be written as $\sum_i d_{1i}Y_i$, and the tracked value is stored in the block W_i^1. For technical reasons we will use $\frac{1}{p}\sum_i d_{1i}Y_i$, which serves the same purpose since by Assumption 3 the value of p is known to all nodes. Note that any algorithm that can track $\frac{1}{p}\sum_i d_{1i}Y_i$ can be extended to track $\frac{1}{p}(D \otimes I_n)Y$ but running v concurrent copies, each with a different set of weights $\{d_{ji}\}$. We observe that this problem can be seen as an instance of the dynamic average consensus problem [KVC19]. In brief, suppose that each agent in the network has a time-varying local reference signal $\phi_i(t) : [0, \infty) \to \mathbb{R}^n$. The

dynamic average consensus problem asks for an algorithm that allows individual agents to track the time-varying average of the reference signals, given by:

$$u^{\text{avg}}[t] = \frac{1}{p}\sum_{i=1}^{p}\phi_i[t]. \tag{4.6}$$

In our problem setting, we pick the external input corresponding to $(D_1 \otimes I_n)Y$ to be $\phi_i^1[t] = d_{1i}Y_i[t]$ and what we want to track is $\frac{1}{p}\sum_{i=1}^{p}\phi_i^1[t]$. In other words, we may adopt any algorithm that solves the dynamic average consensus algorithm thereby enabling all nodes to track $(D \otimes I_n)Y$.

However, in the setting of dynamic average consensus problem, no knowledge about reference signals ϕ_i is assumed, whereas in our problem, for an attack-free node i, we have the following:

$$Y_i[t+1] = \underbrace{\begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\alpha_0 & -\alpha_1 & -\alpha_2 & \dots & -\alpha_{n-1} \end{bmatrix}}_{\hat{A}} Y_i[t]. \tag{4.7}$$

This equality comes from the construction $Y_i[t] = \begin{bmatrix} y_i[t] & y_i[t+1] & \dots & y_i[t+n-1] \end{bmatrix}^T$ containing n consecutive measurements where $\alpha_1, \alpha_2, \dots, \alpha_{n-1}$ are the coefficients of the characteristic polynomial of A. This can be seen by noticing that $y_i[t+n] = C_iA^nx[t] = C_i(\alpha_{n-1}A^{n-1} + \dots + \alpha_1A + \alpha_0)x[t] = \alpha_{n-1}y_i[t+n-1] + \dots + \alpha_1y_i[t+1] + \alpha_0y_i[t]$ whenever node i is attack free. Moreover, we note that $\hat{A}$ is the controller form of A and has the same eigenvalues as A.

Remark 8 *Since the state portion corresponding to the eigenvalues with magnitudes strictly less than 1 will decay to zero with the elapse of time, in the remainder of the chapter we will only be focusing on the state portion corresponding to eigenvalues with magnitudes greater than or equal to 1.*

Based on Equation (4.7), we define the following local sanity check:

Local sanity check: given measurement $y_i[t]$ of node i, we say sensor i passes the local sanity check if $\|Y_i[t+1] - \hat{A}Y_i[t]\| \leq \epsilon$ for any $t > 0$, where ϵ is a chosen error tolerance.

In practice, we ask all nodes in the network to constantly run the local sanity check on their sensors. It is trivially seen that all attack-free sensors would pass the check at any time. On the other hand, if a sensor fails the local sanity check, then we immediately reach the conclusion that this sensor is under attack. Therefore, we assume that all the attack vectors e_i corresponding to attacked sensors are constructed so that the resulting measurements $Y_i[t]$ pass the local sanity check, i.e., $\|Y_i[t+1] - \hat{A}Y_i[t]\| \leq \epsilon$.

The purpose of the local sanity check is to force the dynamics of Y_i to be governed by (4.7), including those from attacked nodes. We will exploit this additional knowledge of $d_{1i}Y_i$ (or ϕ_i^1) to achieve better tracking results.

Remark 9 *We emphasize that we are not asking attacked sensors to maintain integrity (i.e., follow Equation* (4.7)*), but instead are assuming the worst case when attacked sensors can pass the sanity check. Consider the case when an attacked sensor deviates from the rule* (4.7) *and does not pass the check, then the associated node will be aware of it and inform all other nodes in the network to exclude this sensor from the network.*

4.5 Solving the DSST Problem

In this section we solve the DSST problem by showing that all nodes are able to asymptotically track the compressed measurements (in subsection A), and can reconstruct the state of the system from compressed measurement with a suitable choice of the compression matrix (in subsection B).

4.5.1 Tracking the Compressed Measurements

We argued in Section 4.4 that the tracking of the compressed measurements $(D \otimes I_n)Y$ is intrinsically identical to a dynamic average consensus problem. We extend the results in [BFL10] from the scalar-input case to the vector case and present such extension in the discrete-time domain.

We ask each node to update its estimate of $\frac{1}{p}\sum_i d_{1i}Y_i[t]$ according to:

$$\begin{cases} W_i^1[t+1] = & (\hat{A}-I)W_i^1[t] + 2k_I \sum_{j\in\mathcal{N}_i}(\eta_j[t]-\eta_i[t]) \\ & \quad + \phi_i^1[t], \\ b_i[t+1] = & \hat{A}b_i[t] - k_I \sum_{j\in\mathcal{N}_i}(W_j^1[t]-W_i^1[t]), \\ \eta_i[t] = & k_P b_i[t] - k_I \sum_{j\in\mathcal{N}_i}(W_j^1[t]-W_i^1[t]), \end{cases} \tag{4.8}$$

where $W_i^1[t]$ is the estimate at node i of the average of the input signal $\frac{1}{p}\sum_i \phi_i^1[t]$, $k_I, k_P \in \mathbb{R}$ are design parameters, $b_i[t]$ and $\eta_i[t]$ are internal states of node i. The following theorem states that under suitable choices of k_I and k_P the estimate of each node $W_i^1[t]$ approaches the true state $\frac{1}{p}\sum_i \phi_i^1[t]$.

Lemma 8 *Consider the average tracking algorithm in* (4.8) *where the input signal satisfies* $\phi_i^1[t+1] = \hat{A}\phi_i^1[t]$ *for all* $i \in \{1, 2, \ldots, p\}$. *There exist constants* $k_I \in \mathbb{R}$ *and* $k_P \in \mathbb{R}$ *such that the state estimate* $W_i^1[t]$ *at each node tracks the average of the input signals exponentially fast, i.e.:*

$$\left\| W_i^1[t] - \frac{1}{p}\sum_i \phi_i^1[t] \right\| < \beta\alpha^t,$$

for some $0 < \alpha < 1$ *and* $\beta > 0$.

Proof. Proof of Lemma 8 can be found in the appendix. □

4.5.2 Reconstructing the State with Compressed Measurements

In this subsection we argue that if $W_i[t]$ is close enough to $(D \otimes I_n)Y[t]$ then reconstructing the state $x[t]$ is nothing more than solving a standard SSR problem. Once again, for the sake of simplicity we are not going to provide the rigorous definition of the SSR problem in this chapter, which can be found in [MMS22].

In the reconstruction step, each node seeks a state $\hat{x} \in \mathbb{R}^n$ and a vector $\hat{E} \in \mathbb{R}^{pn}$ containing no more than s non-zero blocks that best explain the tracked compressed measurements W_i. This problem can be solved in different ways. In this chapter we adopt an optimization formulation by requiring each node to minimize $\|W_i - (D \otimes I_n)(\mathcal{O}\hat{x} + \hat{E})\|_2$. Note that we dropped the time index t here to simplify notation. In other words, each node solves the following optimization problem:

$$\begin{aligned} (\hat{x}, \hat{E}) = \quad & \text{argmin}_{(\tilde{x}, \tilde{E})} \quad \|W_i - (D \otimes I_n)(\mathcal{O}\tilde{x} + \tilde{E})\|_2, \\ & s.t. \qquad\quad \|\tilde{E}\|_{l_0/l_r} \leq s, \end{aligned} \tag{4.9}$$

where $\|\tilde{E}\|_{l_0/l_r}$ denotes the number of non-zero blocks of the vector $\tilde{E}$. The following lemma states that the solution $\hat{x}$ of (4.9) is a good estimate of the system state when the tracking of the compressed measurements is accurate enough.

Lemma 9 *Let $D \in \mathbb{R}^{v \times p}$ and assume (A, C) is $2s$-sparse detectable with respect to D. There exists a constant $\beta \in (0, +\infty)$ such that for any $\alpha \in (0, +\infty)$, $x \in \mathbb{R}^n$, $W_i \in \mathbb{R}^{vn}$, and $Y \in \mathbb{R}^{pn}$ satisfying $Y = \mathcal{O}x$ and $\|W_i - (D \otimes I_n)Y\|_2 \leq \alpha$, the solution $\hat{x}$ to the optimization problem (4.9) satisfies $\|\hat{x} - x\|_2 \leq \beta\alpha$.*

Proof. Proof of Lemma 9 can be found in the appendix. □

As a special case, when $W_i = (D \otimes I_n)Y$, the optimization problem in (4.9) degenerates to the following equality:

$$(D \otimes I_n)Y = (D \otimes I_n)\mathcal{O}\hat{x}_i + (D \otimes I_n)\hat{E}_i.$$

Solving this equality is equivalent to finding the state $\hat{x}_i$, a vector $T \in \ker(D \otimes I_n)$ and a vector $\hat{E}_i$ containing no more than s non-zero blocks such that:

$$Y = \mathcal{O}\hat{x}_i + \hat{E}_i + T,$$

holds. Now we define a matrix N which has np rows and the least possible number of columns such that $\mathcal{R}(N) = \ker(D \otimes I_n)$, and the optimization problem (4.9) is further reduced into finding the state $\hat{x}_i$, a vector $\hat{r}_i$ of appropriate dimension and a vector $\hat{E}_i$ containing no more than s non-zero blocks satisfying:

$$Y = \mathcal{O}\hat{x}_i + \hat{E}_i + N\hat{r}_i = \begin{bmatrix} \mathcal{O} & N \end{bmatrix} \begin{bmatrix} \hat{x}_i \\ \hat{r}_i \end{bmatrix} + \hat{E}_i.$$

Hereby we conclude that reconstructing the state x from compressed measurements $(D \otimes I_n)Y$ is a special case of the SSR problem since Y contains n consecutive measurements from all nodes in the network, and any algorithmic solution to the SSR problem, for example [SCW18] and [MSK16], can be applied to solve the DSST problem if they do not require additional assumptions.

The preceding discussion directly leads to the sufficiency result:

Theorem 10 *Consider the linear system subject to attacks defined in* (4.1) *satisfying Assumptions 2-6 and a communication network satisfying Assumption 1. The tracking algorithm* (4.8) *together with a state-reconstruction algorithm enables each node to asymptotically track the state of the system* (4.1) *and consequently solves the DSST problem, if the pair* (A, C) *is 2s-sparse detectable.*

Remark 10 *We note that, the tracking of compressed measurements has a delay of* $n-1$ *time steps, and consequently, our solution of the DSST problem (i.e., tracking of the state* $x[t]$*) also has a delay of* $n-1$ *time steps. However, in the special cases when the observability index of the system* (4.1) *is smaller than* n*, and the evolution of stacked measurements (i.e., Equation* (4.7)*) is governed by a matrix of smaller size, the delay will be shorter than* $n-1$ *time steps.*

Combining Theorem 9 (necessary part) and Theorem 10 (sufficient part) we obtain the main result of the chapter.

Theorem 11 *The DSST problem associated with a communication network satisfying Assumption 1 and a linear system subject to attacks defined in* (4.1) *satisfying Assumptions 2-6 is solvable if and only if the pair* (A, C) *is 2s-sparse detectable.*

4.6 Numerical Example

In this section we borrow an example from [HRS20] with slight modifications to illustrate how we solve the DSST problem by asking all nodes to track the compressed version of the measurements. With this example we also compare the performance of our algorithm and the one proposed in [LKS20].

We consider a scalar system with $A = 1.0006$, $C_i = 1,\ i = 1, \ldots, 30$. The initial state satisfies $x[0] = 1$ and consequently $x[t] = 1.0006^t$. All thirty nodes are connected via a communication network, the topology of which is shown in Figure 1. This communication graph is identical to figure 1 in [HRS20]. In this graph, every circle represents a node and a line connecting two circles represents a communication link between the connected nodes. Attack-free nodes are colored in gray, while attacked nodes are colored in red, i.e., nodes 2, 16, and 29 are attacked.

This scalar system has 30 nodes and we can easily conclude that it is 29-sparse detectable. In other words, this system can tolerate up to 14 attacked sensors. In this example, we assume that only 3 sensors (sensors 2, 16, and 29) are attacked by an adversary. This gap makes possible for designing a compression matrix D which compresses down the redundant part of the measurements.

The matrix D should satisfy the necessary conditions, i.e., the system should be 7-sparse detectable with respect to D. In this case, any matrix of proper dimensions whose rank is

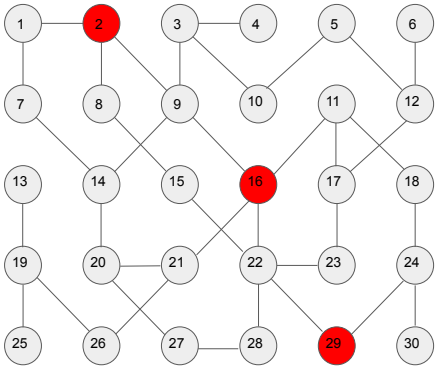

Figure 4.1: The communication graph.

greater than or equal to 7 is a good candidate for D. For simplicity, we pick the compression matrix $D \in \mathbb{R}^{7\times 30}$ as follows:

$$D_{ij} = \begin{cases} 0 & \text{if } 4 \leq (j-i-3) \bmod 30 \leq 26, \\ 1 & \text{otherwise,} \end{cases} \tag{4.10}$$

where D_{ij} is the (i,j)th entry in the matrix D. We now use this simple example to illustrate why tracking compressed measurements DY suffices to reconstruct the state. We first note that, the most naive way of solving the DSST problem associated with this system is by asking all nodes to collect measurements from all sensors throughout the network, which requires all nodes to store 30 scalar variables. However, we notice that only three out of thirty nodes are under attack, therefore seven suitably chosen linear combinations of measurements (with different weight sets) suffice to reconstruct the state despite the attacks. In this case, all nodes only need to store 7 scalars, which reduces communication cost and storage.

Practically, we ask all nodes to run concurrently seven versions of the tracking algorithm (4.8), each associated with a different weight set. For example, to track[3] $D_1 Y$, each node i

[3]We recall that D_1 is the first row of D.

executes algorithm (4.8) with reference signal $\phi_i^1[t] = d_{1i}y_i[t]$, or explicitly:

$$\phi_i^1[t] = \begin{cases} 1.0006^t & i = 1, 3, 4, 5, 6, 7, \\ k \times 1.0006^t & i = 2, \\ 0 & \text{otherwise}, \end{cases} \tag{4.11}$$

for some $k \in \mathbb{R}$ arbitrary chosen by the attacker[4]. In the experiment we pick $k = 3$ and consequently $D_1Y = 0.3 \times 1.0006^t$. In Figure 4.2 we show the tracking error of Algorithm (4.8) and observe that all the nodes are able to track the compressed measurements D_1Y, which verifies our theoretical results.

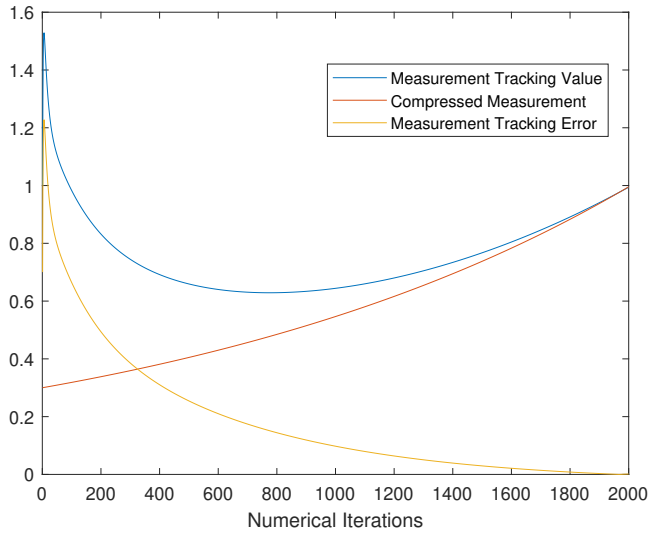

Figure 4.2: The measurement tracking error $\|W^1 - D_1Y\|$ versus iterations.

The reconstruction step is simple since the state can be reconstructed via majority voting for a scalar system. In Figure 4.3 we show that the reconstructed state tracks the true state asymptotically. We also compare our algorithm with the distributed median solver in [LKS20]. As we can see from Figure 4.3, solution in [LKS20] reaches steady state quickly at around 3.1ms after the algorithm started executing. However, it has a steady-state error

[4]We note that the measurement of the attacked sensor 2 must be in this form in order to pass the local sanity check.

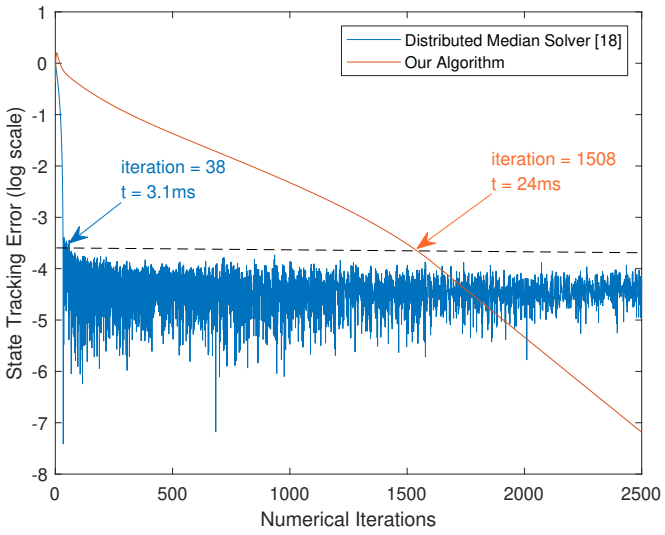

Figure 4.3: The state tracking error $\|\hat{x} - x\|$ versus iterations.

of around 0.03. Compared with [LKS20], our algorithm takes longer to converge, 24ms for the tracking error to decrease below 0.03. However, our algorithm asymptotically converges to the correct solution whereas the algorithm in [LKS20] does not.

4.7 Concluding Remarks

In this chapter, we studied the problem of tracking the state of a linear system against sensor attacks in a network. Motivated by recent works in the dynamic average consensus literature and the simple observation that a compressed version of measurements suffices to reconstruct the state, we proposed a novel decentralized observer that enables each node in the network to track the state of the linear system without making any non-necessary assumptions except one regarding the sampling rate.

Future directions include developing a decentralized state-tracking algorithm robust against other types of adversarial attacks, for example, a byzantine attack which is not only able to alter measurements but also forces attacked nodes to deviate from their prescribed algo-

rithm. One additional problem of interest is the extension of the proposed results to the case where inputs are present but unknown to the nodes.

CHAPTER 5

Decentralized Robust Optimization

5.1 Introduction

This chapter concerns decentralized optimization problem that has seen several applications in the past decade [PKP06, BPC11], including federated learning [MNG18, KMA19]. In this problem, we have a set of nodes connected via a communication network, each equipped with a local function, which are collectively searching for the minimizer of the aggregation of their local functions. However, in some scenarios, nodes may suffer from malicious attacks, which render most solutions developed for faultless networks [CL99, DHP04] invalid. Therefore, it is of significant importance to develop learning algorithms that are robust to attacks.

5.1.1 Existing Works

Some works have addressed the robust learning problem (or the robust optimization problem) in the distributed case [DD21, DD20, GMK21, LXC19, GHY19] and references therein. In these works, the existence of a central server is assumed, which is connected to all nodes in the network and is responsible for learning the model or computing the minimizer. In this chapter, we do not assume the existence of such a central server, i.e., we consider a decentralized setting. We ask each node to learn a model (or obtain the minimizer), using its own local information and messages exchanged with its neighbors. Moreover, we adopt the more general heterogeneous problem setting, where the data set (or the local function) at each node is different.

The problem of global optimization when nodes are spread in a network (i.e., decentralized setting) has been well-studied [NO09, MSL16, Bul19]. However, these solutions are vulnerable to attacks: they completely break down if some local functions are altered by an adversary. This consideration motivated some other works which focus on developing robust decentralized optimization algorithms. Depending on the relationship between the global minimum point x^* (which is typically assumed to uniquely exist) and the set of minimizers, Si, of the local function f_i, these works can be roughly divided into the following three classes:

Class one: Each local function has only one local minimum point which coincides with the global optimizer, i.e., $S_i = \{x^*\},\ i = 1, 2, \ldots, p$. This scenario typically takes place in a machine learning problem setting where all nodes are collaboratively learning a model by performing stochastic gradient descent using the same data set, or when the data samples at all nodes are drawn i.i.d. from the same statistical distribution. In this case, even without communication, non-adversarial nodes can reach consensus on the global optimizer, hence most efforts have been devoted to accelerating convergence speed by suitably exchanging messages between neighboring nodes. Representative works in this class include [YB19a, YB19b]. In [YB19b], each node is asked to perform coordinate-wise gradient descent based on a resiliently aggregated version of its received gradients. The solution in [YB19a] is similar to [YB19b] but nodes are asked to perform vanilla gradient descent instead. Both [YB19a] and [YB19b] showed that collaboration among nodes in a network increases the speed of model-training notwithstanding a small fraction of nodes being attacked by a Byzantine adversary.

Class two: The global optimizer belongs to the set of local minimum points of any node, i.e., $x^* \in S_i,\ i = 1, 2, \ldots, p$, and at least one local function has two or more minimizers. In this case, nodes must rely on messages exchanged with their neighbors in order to obtain the global optimal point. The following interesting observation was made in [GV20]: in this setting, it is possible for all nodes to retrieve the exact global minimum point whereas in the

most general case, when there is no direct relationship between the global minimum point and the sets of local minimum points, x^* is not retrievable. Papers [SV16] and [GV20] fall into this class. In particular, [SV16] showed that nodes are able to agree on the global optimal point given that any local cost function can be decomposed into a non-trivial weighted sum of uni-variate strictly convex functions which comes from a common size-limited set of basis functions. Their later work [GDV20] extended this result to the multivariate case and dropped the weighted sum decomposition assumption made in [SV16]. However, the algorithm in [GV20] only applies to the special case when all nodes are connected via a complete network.

Class three: The global optimizer x^* is not the local minimum point for some functions, i.e., there exists $i \in \{1, 2, \ldots, p\}$ so that $x^* \notin S_i$. Paper [SV16] was the first work that studied this setting under the assumption that all local functions are uni-variate, in which the authors novelly proposed the Synchronous Byzantine Gradient (SBG) method which forces all nodes to reach consensus on the minimum point of a non-uniformly weighted sum of the local functions. Similar techniques have been adopted by [SG18] in its Local Filtering (LF) algorithm, which significantly outperforms the SBG algorithm in terms of communication load at the price of requiring an unnecessary assumption on the communication graph topology. A later work [KXS20] extended the result in [SV16] and [SG18] to the multivariate case. However, the fundamental limitation of SBG (or LF)-type algorithms is still present in [SG18], as the consensus point is only guaranteed to lie in the smallest hyper-rectangle that contains all the local minimum points.

There is also a vast literature devoted to the learning problem when a fraction of data samples is under attack [RWR20, WRK20, RSL18]. However, these works differ from our RDGO problem setting in the sense that they assume a fraction of data samples to be attacked, whereas we assume that all the data samples in a fraction of nodes are attacked. As a consequence, we are able to filter the information from a set of nodes whereas such approaches are not applicable in most adversarial learning problems.

5.1.2 Our Contributions

As we can see from the previous discussion, all existing works on decentralized optimization in the presence of attacks have its limitations: they either consider a simpler case where some special relationships exist between the global optimal point and the sets of local optimal points, or are only able to provide a loose bound on the distance between the consensus point and the global minimum point x^*. However, it is noteworthy that all these works consider the attacker to be byzantine and to possesses full knowledge of the system: including, but not limited to, the graph topology, all local functions, and the algorithms running at each node. Moreover, an attacked node may arbitrarily deviate from its prescribed rules if it is attacked by a byzantine adversary. In this chapter, we consider a milder type of attacks known as data poisoning attacks, where the adversary still has full knowledge of the system, but is only able to change the local functions of the attacked nodes. The main difference between byzantine attacks and data poisoning attacks is that an attacked node is still able to execute its program if it is subject to data poisoning attacks. To the best of authors' knowledge, the decentralized optimization problem against data poisoning attacks, which we refer to as the Resilient Decentralized Global Optimization (RDGO) problem, has not yet been studied.

The main contributions of this chapter are:

1. We propose a a novel filtering algorithm which robustly estimates the weighted sum of a set of vectors in $\mathbb{R}^n$ in the presence of data poisoning attacks. The algorithm is given in Algorithm 2. Moreover, the distance between the computed weighted sum and the true value scales well with the dimension n ($\propto \sqrt{n}$), and the fraction ϵ of attacked vectors ($\propto \epsilon$). The algorithm is also light-weighted since its computational complexity scales linearly with n.

2. We propose an algorithm that solves the RDGO problem when the aggregated function is either convex or belongs to a special class of non-convex functions. The algorithm is given

in Algorithm 2. The algorithm guarantees the Euclidean distance between the obtained minimizer and the true one (in the absence of attacks) to be proportional to $\sqrt{n}$ and ϵ, which is proved in Theorem 12. Moreover, the proposed algorithm tolerates an attack up to half of the nodes.

3. We verify the theoretical results with a numerical example, where we ask 20 nodes in a communication network, each equipped with a non-overlapping portion of the MNIST data set, to collectively train a binary classification model, despite an adversary which is able to alter the data sets at three nodes.

To the best of authors' knowledge, our algorithm outperforms any existing solution, under the same or less stringent problem setting (e.g., a distributed setting). In the following table we list 3 representative works on robust centralized/distributed optimization and compare our results with theirs.

	[GHY19, GMK21]	[DD20, DD21]	[LXC19]	[RWR20]	Our algorithm
Network Topology	distributed	distributed	distributed	centralized	decentralized
Error Tolerance	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
Error vs Dimension	$O(n)$	$O(\sqrt{n})$	$O(n)$	$O(\sqrt{n})$	$O(\sqrt{n})$
Error vs Fraction of Attacked Nodes	$O(\epsilon)$	$O(\sqrt{\epsilon})$	$O(\epsilon)$	$O(\epsilon)$	$O(\epsilon)$

Table 5.1: A comparison between solutions to the RDGO problem in [GHY19, GMK21], [DD20, DD21], [LXC19], [RWR20], and this chapter.

We also note that, any result in this chapter can be conveniently applied to the distributed case, even when the attack is Byzantine. More details regarding this claim can be obtained in Remark 15.

5.1.3 Chapter Organization

The remainder of the chapter is organized as follows. Section 5.2 formulates the RDGO problem. In Section 5.3 we introduce Resilient Averaging Gradient Descent (RAGD) algorithm which solves the RDGO problem. This is followed by Section 5.4 in which how to robustly estimate the weighted sum of a set of vectors is investigated. The performance of the RAGD algorithm is studied in Section 5.5. Lastly, in Section 5.6 we verify our theoretical results via a numerical example, and in Section 5.7 we conclude this chapter.

5.2 Preliminaries

5.2.1 Notation

Let $\mathbb{R}$, $\mathbb{R}^+$, and $\mathbb{N}$ denote the set of real, positive real, and natural numbers, respectively. Given a vector $v \in \mathbb{R}^n$ where n is a positive natural number, we use $\|v\|_2$ to denote the l_2 norm of v. Also, we define the all-ones vector of length n by $\mathbf{1}_n = (1, 1, \ldots, 1)^T$ and I_n to be the identity matrix of order n. The largest and smallest singular values of a matrix $A \in \mathbb{R}^{n\times p}$ is denoted by $\sigma_M(A)$ and $\sigma_m(A)$, respectively, where n, p are positive natural numbers. Moreover, we use $\nabla f(x)$ to denote the gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ evaluated at $x \in \mathbb{R}^n$. Further let $r \in \mathbb{R}^+$. We denote by $\mathcal{B}(x, r) = \{y \in \mathbb{R}^n | \|y - x\|_2 \leq r\}$ the ball centered at x with radius r. Moreover, the distance $\mathcal{D}(x, S)$ between a point $x \in \mathbb{R}^n$ and a set $S \subseteq \mathbb{R}^n$ is defined by $\mathcal{D}(x, S) = \inf_{y\in S} \|x - y\|_2$.

A weighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ is a triple consisting of a set of vertices $\mathcal{V} = \{v_1, v_2, \ldots, v_p\}$ with cardinality p, a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, and a weighted adjacency matrix $\mathbf{A} \in \mathbb{R}^{p\times p}$ which will be defined very soon. The set of in-neighbors of a vertex $i \in \mathcal{V}$, denoted by $\mathcal{N}_i^{in} = \{j \in \mathcal{V} | (j, i) \in \mathcal{E}\}$, is the set of vertices connected to i by an edge. Similarly, the set of out-neighbors of a vertex $i \in \mathcal{V}$ is defined by $\mathcal{N}_i^{out} = \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$. We assume each vertex is both an in-neighbor and an out-neighbor of itself. The weighted adjacency

matrix $\mathbf{A}$ of the graph $\mathcal{G}$ is defined entry-wise. The entry in the i-th row and j-th column, a_{ij}, satisfies $0 < a_{ij} < 1$ if $(i,j) \in \mathcal{E}$ and otherwise $a_{ij} = 0$.

5.2.2 Problem Formulation

We consider a set P of p nodes connected via a communication network, modelled as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$. The set V in $\mathcal{G}$ represents the set of nodes and the set $\mathcal{E}$ represents the set of communication links between all pairs of nodes. In particular, an edge $(i,j) \in \mathcal{E}$ exists if and only if node j can receive information from node i. Moreover, each communication link (i,j) is associated with a positive scalar value $a_{ij} > 0$, which, we recall, is the (i,j)-th entry of $\mathbf{A}$.

We now formally define the RDGO problem.

Definition 11 *Consider a set P of p nodes connected via a communication network. Each node $i \in P$ is equipped with a local function $f_i : \mathbb{R}^n \to \mathbb{R}$ where $x \in \mathbb{R}^n$ is the optimization variable. The RDGO problem asks each node to find the optimizer x^* of the aggregation of the local functions:*

$$f = \sum_{i \in P} f_i,$$

using its local function f_i and messages exchanged with its neighboring nodes, notwithstanding some local functions have been altered by a data poisoning attack.

In the decentralized federated learning problem, each node i has a local data set $\mathcal{Z}_i = \{z_{i1}, z_{i2}, \ldots, z_{iN}\}$ of cardinality N. The federated learning problem asks all nodes to collectively minimize the following risk function $\frac{1}{N}\sum_{i=1}^{p}\sum_{j=1}^{N} l(\mathbf{w}, z_{ij})$ with respect to $\mathbf{w}$. In this case, each local function $f_i(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N} l(\mathbf{w}, z_{ij})$ is implicitly defined by the local data set $\mathcal{Z}_i$ at node i. The aggregation f of local functions is also named as the global function in this chapter.

5.2.3 Attack Model

The solution to the global optimization problem is trivial in the absence of attacks [NO09, MSL16, MLR16, Bul19]. However, the problem gets more interesting and also complicated when some nodes are subject to attacks. In this note, we assume that a subset $P_b \subset P$ of nodes are subject to a data poisoning attack, which is able to replace the original function f_j of an attacked node $j \in P_b$ with $\tilde{f}_j \neq f_j$. Moreover, we define $S_g = S \backslash S_b$ to be the set of attack-free nodes. For simplicity, we also use $\tilde{f}_i$ to denote the local function of an attack-free node $i \in S_g$ after the data poisoning attack. It is trivially seen that, for an attack-free node i, $\tilde{f}_i = f_i$.

The adversary that launches the data poisoning attack is assumed to be omniscient, i.e., it has full knowledge of the communication graph, the local functions of all nodes, the algorithm each node executes, etc. Moreover, it is able to arbitrarily alter the local functions of the attacked nodes. However, differently from Byzantine attacks, we assume that all nodes, even those subject to data poisoning attacks, are able to execute their protocols correctly. Moreover, in the context of this chapter, we assume the attack is perpetrated before the nodes start executing the algorithm that solves the RAGD problem, which we will soon discuss in the next section. The attacked local functions will not change once the algorithm starts running. This definition of the data poisoning attack is in line with other works (for example [XBB15], [BCL17], [SMT13], and [SMY15]) where data poisoning attacks are studied.

Remark 11 *It was argued in [SV16] that it is impossible to exactly recover the optimizer x^* when some local functions are attacked by an adversary and when there is no special relationships between the local functions. Therefore, instead of exactly recovering the optimizer x^*, we study in this chapter how well each node can approximate x^* using its possibly attacked local function and messages exchanged with its neighbors.*

5.2.4 Assumptions

We study the RGDO problem under the following assumptions, some of which have already been discussed:

Assumption 7 *The communication graph is fixed, connected, and doubly-stochastic (i.e., the adjacency matrix* $\mathbf{A}$ *of the graph is a doubly-stochastic matrix). Moreover, the weight associated with each link is known to the corresponding receiver node, for example, node* j *is aware of* a_{ij} *for any* $i \in P$.

Assumption 8 *There exists an* $0 < \epsilon < \frac{1}{2}$*, known to all nodes in the network, such that for any node* j*, the sum of link weights corresponding to its attacked in-neighbors is upper bounded by* ϵ*, i.e.:*

$$\sum_{i \in \mathcal{N}_j^{in} \cap S_b} a_{ij} \leq \epsilon, \ \forall j \in P. \tag{5.1}$$

Assumption 9 *Each local function is differentiable, and the Euclidean distance between the gradients of any two local functions evaluated at any point* x *in the working domain*[1] *is upper bounded by some constant* $\kappa > 0$*, i.e.:*

$$\|\nabla f_i(x) - \nabla f_j(x)\|_2 \leq \kappa, \ \forall i, j \in P. \tag{5.2}$$

Assumption 7 is a constraint on the communication network topology. This assumption is the simplest one that enables a solution to the Decentralized Average Consensus (DAC) problem, where a network of nodes, each having a local initial value, seeks to agree on the average of their initial values [XBK07, CI14]. Note that the DAC problem is a special case of the RDGO problem[2], which justifies our Assumption 7. Assumption 8 restricts the power of the adversary. For example, if node j has k neighbors and the weight on each link to node j is

[1] *A brief discussion on the working domain will be provided at the end of the chapter.*

[2] Consider a special class of the RDGO problem where the local function at node i is chosen to be $f_i(x) = (x - x_i)^T(x - x_i)$. We note that the global optimizer of this RDGO problem is $x^* = \frac{x_1 + x_2 + \cdots + x_p}{p}$.

$1/k$, Assumption 8 requires that less than half of the links can be attacked since ϵ is required to be smaller than $1/2$. Similar assumptions were made in [SG18], with the slight difference that in [SG18] it is assumed that the number of attacked nodes in a neighborhood is upper-bounded. Lastly, Assumption 9 is widely accepted in decentralized (distributed) machine learning literature, for example, [YB19b] and [DD21] adopted very similar assumptions to enable filtering information from attacked nodes.

Apart from Assumptions 7-9, in this chapter we also need one of the following two assumptions to solve the RDGO problem.

Assumption 10 *The global function f is L-smooth and ν-strongly convex, each local function f_i is L_1-smooth.*

Assumption 11 is based on the following definition [KNS16].

Definition 12 *A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ satisfies the Polyak-Lojasiewicz (PL) inequality with parameter $\mu \in \mathbb{R}^+$ if the following inequality holds:*

$$\frac{1}{2}\|\nabla f(x)\|_2^2 \geq \mu(f(x) - f(x^*)), \tag{5.3}$$

for any $x \in \mathbb{R}^n$ and x^* being a minimizer of the function f.

Assumption 11 *The global function f is L-smooth and satisfies the PL inequality with parameter μ, each local function f_i is L_1-smooth.*

We note that a ν-strongly convex function must satisfy the PL inequality. This result can be observed by choosing $\mu = \nu$. However, the opposite does not hold. For example, a function that satisfies the PL inequality may have multiple minimizers. We also note that if a function f is both L-smooth and satisfies the PL inequality with parameter μ, then $\mu < L$.

5.3 The Resilient Averaging Gradient Descent Algorithm

In this section, we introduce an algorithm called the Resilient Averaging Gradient Descent (RAGD), which enables all nodes to approximate the global minimum x^* and thus solves the RDGO problem. Algorithm 1 is a pseudo-code description of the RAGD algorithm.

Algorithm 1: Resilient Averaging Gradient Descent (RAGD) Algorithm for Node j

Input: $\{a_{ij}|i \in \mathcal{N}_j^{in}\}$, $\tilde{f}_j$, ϵ, η, $\tau \in \mathbb{N}$;

Initialization: $x_j^0[0] := 0$;

for $t = 0, 1, 2, \ldots$ **do**

  for $k = 0, 1, 2, \ldots, \tau - 1$ **do**

    Broadcast $x_j^k[t]$;

    Receive $x_i^k[t]$ from $i \in \mathcal{N}_j^{in}$;

    $x_j^{k+1}[t] := \sum_{i \in \mathcal{N}_j^{in}} a_{ij} x_i^k[t]$;

  end

  Compute and broadcast the gradient $X_j[t] := \nabla \tilde{f}_j(x_j^\tau[t])$ of its local function;

  Receive $X_i[t]$ from $i \in \mathcal{N}_j^{in}$;

  $\hat{\mu}_j[t] :=$ RWSE($\{(a_{ij}, X_i[t]), i \in \mathcal{N}_j^{in}\}, \epsilon$) where the RWSE algorithm will be introduced in Section 5.4;

  $x_j^0[t+1] := x_j^\tau[t] - \eta \hat{\mu}_j[t]$;

end

Output: $x_j^\tau[t]$;

The RAGD algorithm has two loops, an inner loop (lines 4-8) and an outer loop (lines 3-13). In the inner loop, all the nodes are asked to run a linear iterative algorithm aiming at reaching consensus on the average of their local estimates. The input parameter τ controls the number of iterations executed in the inner loop. The estimate at node j in t-th iteration

of the outer loop and k-th iteration of the inner loop is denoted by $x_j^k[t]$. To proceed, we directly provide the following result on the convergence property of the inner loop.

Lemma 10 *Consider a set P of nodes, each starts with an initial value $x_i^0[t]$, executes lines 4-8 of the RAGD algorithm in parallel. Define $\bar{x}[t] = \frac{1}{p}\sum_{i\in P} x_i^0[t]$ and $d^k[t] = \max_{i,j\in P} \|x_i^k[t] - x_j^k[t]\|_2$. The following two properties regarding $x_1^\tau[t], x_2^\tau[t], \ldots, x_p^\tau[t]$ hold for any $t \in \mathbb{N}$:*

1. *$\frac{1}{p}\sum_{i\in P} x_i^\tau[t] = \bar{x}[t]$, $\forall \tau \in \mathbb{N}$.*
2. *there exists an $a \in \mathbb{R}^+$ and a $\rho \in (0,1)$ such that for any $\tau \in \mathbb{N}$, $d^\tau[t] \leq a\rho^\tau d^0[t]$.*

In the outer loop, all nodes are first asked to reach consensus on the average of their local estimates by executing the inner loop. Then each node is asked to compute and broadcast the gradient of its (possibly attacked) local function (lines 9-10). We note that some gradients are not reliable since some local functions have been altered by the data poisoning attack. Upon receiving gradients from all its neighbors, each node runs a screening algorithm (the RWSE algorithm) which allows each node to resiliently approximate the weighted average of the gradients it receives (line 11), and in the end updates its local parameter by performing a gradient descent step based on the output of the RWSE algorithm (line 12).

Remark 12 *In line 9 of Algorithm 1, we ask each node j to compute the gradient of its (possibly attacked) local function evaluated at its current local estimate $x_j^\tau[t]$. If node j is free from attack, then $X_j[t] = \nabla f_j(x_j^\tau[t])$, i.e., the computed gradient equals the gradient of its original local function evaluated at the same point. However, if node j is attacked, we make no assumptions on the relationship between $X_j[t]$ and $\nabla f_j(x_j^\tau[t])$ except that $X_j[t]$ exists.*

Detailed discussion on the RAGD algorithm will be presented in Section 5.5.

5.4 The Robust Weighted Sum Estimation Algorithm

In this section, we study the problem of how each node can resiliently compute the weighted sum of its neighbors' gradients under assumptions 7-9, despite a portion of the gradients having been attacked. To solve this problem, we propose a novel algorithm termed the Robust Weighted Sum Estimation (RWSE) algorithm.

The RWSE algorithm is not only the key for solving the RDGO problem, but also has other applications, for example, it can be conveniently applied to solve the distributed Byzantine-resilient federated learning problem[3]. Moreover, we note that the robust weighted sum estimation problem is a generalization of the well-known RME problem, hence RWSE solves the robust mean estimation problem assuming Assumptions 7-9 hold.

5.4.1 Algorithm Description

Since the execution node j is fixed, in this section we drop this index and represent the weights $\{a_{ij} : i \in P\}$ by $\{a_i : i \in P\}$. Note that $\sum_{i\in P} a_i = 1$ by Assumption 7. Let $S = \mathcal{N}_j^{in}$, the message X_i that node j receives from node i satisfies:

$$\begin{cases} X_i = & \nabla f_i, \quad i \in S_g, \\ X_i \neq & \nabla f_i, \quad i \in S_b, \end{cases} \tag{5.4}$$

where we also dropped time indices t and τ and used ∇f_i in lieu of $\nabla f_i(x_i^\tau[t])$ for simplicity. The goal is to approximate the weighted average $\mu_g = \sum_{i\in S} a_i \nabla f_i$ using the data $\{(a_1, X_1), (a_2, X_2), \ldots, (a_p, X_p)\}$ under Assumptions 8 and 9. Algorithm 2 is a pseudo code description of the RWSE algorithm.

To explain the RWSE algorithm we describe its execution on the example in Figure 1. In Figure 1, a red dot denotes an attacked vector while a black dot denotes an attack-free vector. All attack-free vectors (black dots) lie in a ball with diameter κ whereas there are

[3]See [DD21] for a formal definition of the distributed Byzantine-resilient federated learning problem.

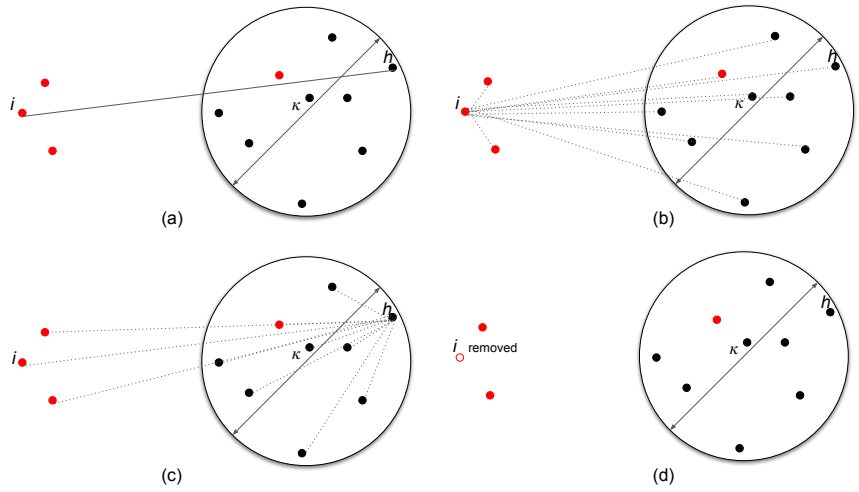

Figure 5.1: Visualization of Algorithm 2.

no restrictions on the position of attacked vectors. In each iteration, the execution node first finds the pair of vectors (i, h) with the maximum Euclidean distance (subgraph (a), also line 4 in Algorithm 2), then computes and compares the weighted sum of the distance between vector i and all other vectors and the weighted sum of the distance between vector h and all other vectors (subgraph (b) and (c), also lines 5-6 in Algorithm 2). In the problem instance represented by Figure 1, vector h is closer to the rest of vectors compared with the attacked vector i, hence in the last subgraph (d) vector i is removed according to lines 6-12 in Algorithm 2.

In addition, we use a scalar variable *temp* to store the identity of the latest removed vector. By Assumption 8 if the weight a_i associated to a vector X_i satisfies $a_i > \epsilon$, then this vector cannot be attacked. Therefore, if a vector X_i with weight $a_i > \epsilon$ is removed at some iteration and then the algorithm terminates, there is no harm restoring this vector X_i since it must be a good vector. By doing so we have the following guarantee of the weight sum of the remaining vectors: $g_e \geq 1 - 2\epsilon$, where g_e is defined in line 14 and line 16 in Algorithm 2.

5.4.2 Performance

It is trivial that Algorithm 2 will terminate (once enough nodes are removed the guard in line 3 will be violated). Therefore we only need to check how close the output $\hat{\mu}$ is to the true average μ_g. To do this, we divide all possibilities regarding the execution of Algorithm 2 into three cases. The classification rule is shown in Table 1, where r stands for a good vector and a stands for an attacked vector. An execution of the RWSE algorithm falls into a certain case if for sufficient number of iterations as shown in the first column of this row, an indicated pair of vectors in the second column is chosen since their distance is larger than any other pair, and in the end a certain (good or attack) vector is removed as indicated by the third column in this row. For example, if for at least one iteration, two good vectors are chosen and one of them is removed, then the execution falls into Case two.

Now we study the execution of the RWSE algorithm case by case.

Case 1: During each iteration one attacked vector is removed.

Observation 1 *If Case 1 holds during the execution of the RWSE algorithm, then for any two vectors X_i, X_h such that $i, h \in V_e$ we have $\|X_i - X_h\|_2 \leq \kappa$.*

We proceed by examining case 2. As we will soon see, Case 1 and Case 2 are very similar.

Case 2: In some iterations the distance between a pair of good vectors is larger than any other pair. Due to this reason a good vector is removed in this iteration.

Observation 2 *If Case 2 holds during the execution of the RWSE algorithm, then for any two vectors X_i, X_h such that $i, h \in V_e$ we have $\|X_i - X_h\|_2 \leq \kappa$.*

Now we study the performance of the RWSE algorithm when either Case 1 or Case 2 holds. The following lemma guarantees that the output $\hat{\mu}$ of the RWSE algorithm is a good estimate of μ_g if either Case 1 or Case 2 holds.

Lemma 11 *If Case 1 or Case 2 holds during the execution of the RWSE algorithm, then the output $\hat{\mu}$ of the RWSE algorithm satisfies:*

$$\|\hat{\mu} - \mu_g\|_2 \leq \left(2 + \frac{2}{1-2\epsilon}\right)\epsilon\kappa. \tag{5.5}$$

Case 3: In some iterations the maximum distance lies between a good vector and a bad vector, leading to a good vector being removed.

We first make a claim on the distance between any remaining attacked vector X_q and the vector ∇f_q which was replaced. This result will be used in the analysis of the RWSE algorithm if Case 3 holds.

Lemma 12 *If Case 3 holds, then for any $q \in V_e \cap S_b$, we have $\|X_q - \nabla f_q\|_2 \leq (2 + \frac{1}{1-2\epsilon})\kappa$.*

Lemma 13 *If Case 3 holds during the execution of the RWSE algorithm, then the output $\hat{\mu}$ of the RWSE algorithm satisfies:*

$$\|\hat{\mu} - \mu_g\|_2 \leq \left(2 + \frac{3-4\epsilon}{(1-2\epsilon)^2}\right)\epsilon\kappa. \tag{5.6}$$

By taking the worst case over the bounds for each of the three cases we obtain the following result:

Lemma 14 *Consider the RWSE algorithm with inputs $\{(a_1, X_1), (a_2, X_2), \ldots, (a_p, X_p)\}$ satisfying (1) $a_1 + a_2 + \cdots + a_p = 1$, (2) $a_i > 0,\ \forall i \in P$, (3) $\|\nabla f_i - \nabla f_h\|_2 \leq \kappa,\ \forall i, h \in P$, and ϵ satisfying (4) $\epsilon < \frac{1}{2}$. Define $\mu_g = \sum_{i=1}^{p} a_i \nabla f_i$. The output $\hat{\mu}$ of the RWSE algorithm satisfies:*

$$\|\hat{\mu} - \mu_g\|_2 \leq \left(2 + \frac{3-4\epsilon}{(1-2\epsilon)^2}\right)\epsilon\kappa. \tag{5.7}$$

Remark 13 *We see from the description that the RWSE algorithm scales well with the dimension n, since the computational complexity of the RWSE algorithm grows linearly with the increase of n. Moreover, according to Lemma 14, the error of the RWSE algorithm scales*

with ϵ, which outperforms many RME algorithms whose error scales with $\sqrt{\epsilon}$ [DKK19]. The authors believe this is a consequence of Assumption 9 where we assume the dissimilarity among two good gradients is upper bounded, whereas in RME algorithms usually deal with randomized data. It is also noteworthy that the error of the RWSE algorithm scales with κ, which implicitly grows with $\sqrt{n}$.

5.5 Performance of the RAGD Algorithm

In this section we prove the correctness of the RAGD algorithm. We start with an intuitive explanation of the RAGD algorithm: the inner loop can be considered as an initialization step, in which each node initializes their estimate to be the average of estimates of all nodes throughout the network, up to some error which decreases exponentially with τ by Lemma 10. Executing lines 9-12 in Algorithm 1 brings the following two consequences:

1. The average of local estimates moves towards the minimum point (or a minimum point if there are multiple), up to some constant error.

2. The distance between two local estimates may increase.

The proof idea is simple. We will prove the following two facts: (1) the local estimates of all nodes are clustered in a ball, and (2) the centroid of the ball moves towards the minimizer up to a constant error. It is a natural consequence of these two facts that the estimate at any node is close to the minimizer. We recall the definition $d^k[t] = \max_{i,j \in P} \|x_i^k[t] - x_j^k[t]\|_2$ from and proceed with our first result in this section:

Lemma 15 *Consider a set P of nodes in a communication network satisfying Assumption 7. Each node has a local function which satisfies Assumption 9 whereas some local functions are altered by a data poisoning attack which satisfies Assumption 8. Let all nodes run the RAGD algorithm in parallel. Moreover, let either Assumption 10 or 11 hold. For any pair of nodes $i, j \in P$, any $t \in \mathbb{N}$ and any $\tau \in \mathbb{N}$, the output of these nodes $x_i^\tau[t]$, $x_j^\tau[t]$ in iteration*

t and the maximum distance $d^k[t]$ satisfy:

$$\|\nabla f_i(x_i^\tau[t]) - \nabla f_j(x_j^\tau[t])\|_2 \leq \kappa + 2L_1 d^\tau[t]. \tag{5.8}$$

Moreover, in this case, the variable $\hat{\mu}_j[t]$ in line 11 of the RWSE algorithm at any node $j \in P$ and iteration $t \in \mathbb{N}$ satisfies:

$$\|\hat{\mu}_j[t] - \mu_j[t]\|_2 \leq c_\epsilon(\kappa + 2L_1 d^\tau[t]), \tag{5.9}$$

for any $j \in P$ and any $t \in \mathbb{N}$, where $\mu_j[t] = \sum_{i \in \mathcal{N}_j^{in}} a_{ij} \nabla f_i(x_i^\tau[t])$ and $c_\epsilon = \frac{3\epsilon - 4\epsilon^2}{(1-2\epsilon)^2}$.

Moreover, in the following proposition we show that $d^\tau[t]$ can be uniformly upper bounded over time t if τ is large enough:

Proposition 6 *Under the assumptions of Lemma 15, for any step size $\eta > 0$ and any given input $r > 0$, there always exist a $\tau_0 \in \mathbb{N}$ such that $d^\tau[t] \leq r$ implies $d^\tau[t+1] \leq r$ for any $t \in \mathbb{N}$, if $\tau \geq \tau_0$.*

This proposition tells that, for any given r, as long as all nodes start from the same initial value in the first iteration (i.e., $t = 0$), then $d^\tau[t] \leq r$ holds for any $t \in \mathbb{N}$ if $\tau \geq \tau_0$ holds.

Combining Lemma 15 with Proposition 6, we notice that for any iteration t and any node $j \in P$, the variable (defined in line 11 of Algorithm 1) $\hat{\mu}_j[t]$ satisfies:

$$\|\hat{\mu}_j[t] - \mu_j[t]\|_2 \leq c_\epsilon(\kappa + 2L_1 r). \tag{5.10}$$

Now we are ready to analyze the performance of the RAGD algorithm. Recall the definition $\bar{x}[t] = \frac{1}{p}\sum_{j \in P} x_j^0[t]$. The following result shows that in the RAGD algorithm the average of local estimates is approximately updated with a gradient descent step:

Proposition 7 *Under the assumptions of Lemma 15, for any $t \in \mathbb{N}$, $\eta > 0$, $r > 0$, assume $d^\tau[t] \leq r$ holds for any $\tau \in \mathbb{N}$, then the following equation holds:*

$$\bar{x}[t+1] - \bar{x}[t] = -\frac{\eta}{p}\nabla f(\bar{x}[t]) + \frac{\eta}{p} l[t], \tag{5.11}$$

for some $l[t]$ satisfying $\|l[t]\|_2 \leq p c_\epsilon(\kappa + 2L_1 r) + pL_1 r$.

For simplicity, we define $\xi = pc_\epsilon(\kappa + 2L_1 r) + pL_1 r$. Inequality (5.11) shows that, if we compare the average at the iteration $t+1$ and the average at iteration t, we determine that the average $\bar{x}[t]$ is updated with a "polluted" gradient, which differs from the true gradient $\nabla f(\bar{x}[t])$ by a vector $l[t]$ whose Euclidean norm is upper bounded by ξ, using step size $\frac{\eta}{p}$. As we will soon see, performing gradient descent on a strongly convex function (or a function that satisfies the PL condition) using an approximate gradient makes the average estimate $\bar{x}[t]$ converge to the optimizer x^*, up to some constant error.

We proceed by describing the performance guarantee of the RAGD algorithm, which is also the main result of this chapter.

Theorem 12 *Consider a set of nodes in a communication network satisfying Assumption 7, each equipped with a local function satisfying Assumption 9. Moreover, assume a subset of nodes are subject to a data poisoning attack satisfying Assumption 8. Suppose all nodes in the network run the RAGD algorithm with $\eta = \frac{p}{L}$ and parameter $\tau \geq \tau_0$ in Equation[4] (7.43), then the output of every node $j \in P$ satisfies one of the two possibilities:*

1. *Let $\beta = \sqrt{1 - \frac{\nu}{L}}$. Assumption 10 implies the following inequality:*

$$\|x_j^\tau[t] - x^*\|_2 \leq \beta^k \|x_j^0[0] - x^*\|_2 + \frac{\xi}{(1-\beta)L} + r, \tag{5.12}$$

2. *Let $\beta' = 1 - \frac{\mu}{L}$ and S^* be the set of minimizers of f. Assumption 11 implies the following inequality:*

$$\mathcal{D}(x_j^\tau[t], S^*) \leq \sqrt{\frac{L}{\mu}} \beta'^{\frac{t}{2}} \mathcal{D}(x_j^0[0], S^*) + \frac{L\xi}{\mu} + r. \tag{5.13}$$

Remark 14 *It can also be seen from Theorem 12 that the output $x_j^\tau[t]$ at node j will converge to the ball $\mathcal{B}(x^*, \frac{\xi}{(1-\beta)L} + r)$, if Assumption 10 holds. Moreover, if $x_j^0[0]$ is not in the ball mentioned above, then the estimate $x_j^\tau[t]$ will move towards the ball. From this discussion we*

[4] *Equation (7.43) can be found in the supplementary file.*

also learn that for any $t \in \mathbb{N}$, $x_j^\tau[t]$ always lies in the ball $\mathcal{B}(x^*, \max(\|x^*\|_2, \frac{\xi}{(1-\beta)L} + r))$ since all $x_j^0[0]$ are assumed to be 0. The result in Theorem 12 implicitly assumes that the working domain contains the ball mentioned above. A similar analysis also applies to the case when Assumption 11 holds.

Remark 15 *In either case, our RAGD algorithm guarantees that the distance between the computed minimizer and the true global minimizer (in the absence of attacks) is bounded by a constant error term. This differs some existing works [SG18, KXS20] in which the computed minimizer is only guaranteed to lie in the smallest hyper-rectangle that contains all local minimizers*[5]*. Moreover, our error term scales linearly with $\sqrt{n}$ and ϵ when at most half of nodes are under attack, which matches, or even outperforms its counter part in the distributed case [DD21, GMK21, HKJ20]. We also note that in the absence of attacks, i.e., when $\epsilon = 0$, the RAGD algorithm degenerates to the well-known decentralized gradient descent algorithm. Lastly, it is observed that the RAGD algorithm can be applied to solve the distributed resilient federated learning problem, even when the adversary is Byzantine.*

5.6 Numerical Results

In this section, we use a numerical example to illustrate our theoretical results. We consider a decentralized binary classification problem using the MNIST handwritten digits data set. In this task, a total of 20 nodes in a randomly generated communication network are asked to classify digits in two classes corresponding to the digit 0 and the digit 1, using a logistic regression function collectively trained by themselves.

In this test, we use 12000 samples from the MNIST data set, 6000 samples are pictures of handwritten zero digits and the rest are pictures of handwritten one digits. The set of samples is split into a training set of size 10000 and a testing set of size 2000. Moreover, the

[5] *Some of these works assume the attack to be Byzantine.*

training set is equally split into 20 subsets each of size 500, and each node in the network has access to only one subset of samples.

We first generate a random doubly-stochastic square matrix of size 20 to represent the communication graph. In this experiment, we perform an attack which changes all the gradients from two nodes to a random vector generated using Gaussian distribution $\mathcal{N}(0, I_n)$). In the simulation we set $r = 0.05$ and choose the number of iterations of the inner loop τ_0 to be 10. The step size is set to be 1×10^{-6}.

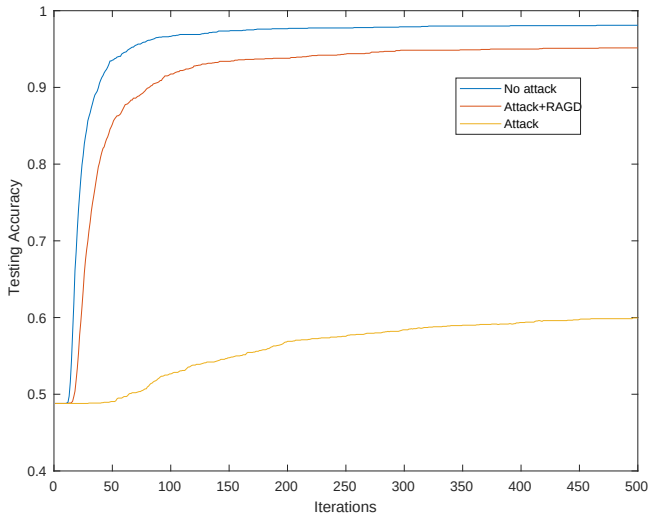

Figure 5.2: Testing accuracy over iterations of the RAGD algorithm for decentralized training a logistic regression function on MNIST data set in the presence of attacks changing gradients.

Figure 2 shows a typical execution of the proposed algorithm in comparison with the absence of attacks and the attack-only case. The yellow curve corresponds to the testing accuracy over iterations when two nodes are subject to the attack mentioned above. In this case, the testing accuracy is around 60%, whereas in the absence of attacks the classification accuracy is above 98%. The red curve corresponds to the case when we implement our RAGD algorithm to combat against the attacks. As we can see in Figure 2, the RAGD algorithm significantly enhances the testing accuracy to more than 95%.

5.7 Conclusion

In this chapter, we proposed a resilient averaging gradient descent algorithm, which solves the decentralized global optimization problem in the presence of data poisoning attacks. The proposed algorithm enables all nodes to approximate the global optimizer, with an error that scales linearly with $\sqrt{n}$ where n is the dimension and the fraction ϵ of attacked nodes.

Algorithm 2: Robust Weighted Sum Estimation (RWSE)

Input: $\{(a_1, X_1), (a_2, X_2), \ldots, (a_p, X_p)\}, \epsilon$;

Initialization: $g := 1$, $temp := 0$, $w_0 := 0$, $V := \{X_1, X_2, \ldots, X_p\}$, $V_e := \{\}$, $g_e := 0$;

while $g > 1 - \epsilon$ **do**

compute the Euclidean distance between every pair of nodes and find out a pair with the maximum distance. If there are multiple pairs, pick one of them arbitrarily. Without loss of generality we assume that vectors X_i and X_h are picked;

$s_i := \sum_{z \in V} (a_z \| X_i - X_z \|_2)$;

$s_h := \sum_{z \in V} (a_z \| X_h - X_z \|_2)$;

if $s_i > s_h$ **then**

$u := i$;

else

$u := h$;

end

$V := V \backslash \{X_u\}$, $g := g - a_u$, $temp := u$;

end

if $a_{temp} > \epsilon$ **then**

$V_e =: V \cup \{X_{temp}\}$, $g_e := g + a_{temp}$;

else

$V_e := V$, $g_e := g$;

end

Output: $\hat{\mu} := \frac{1}{g_e} \sum_{i \in V_e} a_i X_i$;

# of iterations	$\{i, h\}$	u	Case
All iterations	$\{r, a\} \vee \{a, a\}$	a	1
At least 1 iteration	$\{r, r\}$	r	2
At least 1 iteration	$\{r, a\}$	r	3

Table 5.2: Illustration of the classification rule.

CHAPTER 6

Future Directions

In this chapter we conclude the dissertation by outlining some problems that represent my ongoing research.

In this dissertation, we made an attempt to identify the attacked nodes in a CPS by studying the messages sent by each nodes and having them compared against one another. However, the algorithm we design is not able to remove attacked nodes. In plain words, they simply enable attack-free nodes to filter out erroneous messages, or "neglect" to them. The attacked nodes and messaged still lie in the network and they are simply neglected. However, it is not always the case that people want to live along with attacks in a system. In many cases, people want to prevent attacks from coming into being, and, if there is really attacked nodes, people want to remove them and replace them. For example, people in industry tend to build firewalls and come up with data encryption/decryption algorithms to keep the network safe. This motivates my current research, which is based on the following beliefs:

1. No CPS is absolutely safe.

2. We can make a CPS safer with hybrid security measures.

As we see from the previous discussion, they way people in industry deal with attacks is quite different from my approach presented in this dissertation. However, these two kinds of safety measures are not mutually exclusive - we can build a firewall, or implement two-factor authentication measure, and simultaneously double check each piece of information

we receive from neighboring nodes to make sure they make sense. How the combination of these two safety measures restricts the strategy of the attacker and how to robustify a system is part of my ongoing research.

To the best of my knowledge, not too many works have addressed this problem. A most recent work [KP22] considers the state reconstruction problem in a centralized system when data transmission can be partially checked using an authentication rule, and an earlier work [KLS16] showed that by encrypting controller signals the power of the adversary will be limited and thereby increase the robustness of networked systems.

This research area is rich and many questions can be asked. In the following context I'll list some questions that worth exploring:

1. How can a safety measure (like a 2-factor authentication) limit the capability of the adversary? Instead of arbitrarily changing the measurements from attacked sensors, which is a common assumption in the related literature, attackers must carefully design how an attack could be launched without being spotted/stopped by safety measures.

2. Is stealthy attack still possible in a given CPS system with an alarm? If so, how much damage can the adversary do before the alarm rings? Some systems are protected by an alarm system, which rings when the value of a certain internal state exceeds a given threshold. However, it seems different attack strategies will lead to different outcomes regarding this internal state, which might be very sensitive to one kind of attack, but may not even ring if another attack is launched.

3. How much redundancy can we save with shared information? As said, a vast literature is devoted to identifying attacked nodes by comparing messages from that node and other messages in the network. To do this, one needs to deploy more nodes than necessary in order to do the job, where the information redundancy is used for checking if a certain set of messages is reliable. However, with even a little bit of shared information, some nodes will be able to know if a given node, paired with him, is attacked or not.

The tradeoff between the amount of shared information and the redundancy needed is another interesting topic.

There are also other interesting ideas that worth a try:

4. How do we collaborate a network with attacked nodes and also with trusted nodes?

5. How to define the robustness of a graph subject to different kinds of cyber-attacks, and, if we have the freedom of adding more nodes in the graph, how should we design these nodes and where should we place them?

6. It is easily seen that any algorithm robust against attacks will be outperformed by its counterpart (which only works in the attack-free case). A simple reason is the attack-free case can be viewed as an attack-existing case when the number of attacked nodes is zero. However, there lacks an understanding of the tradeoff between the performance of the algorithm and the robustness of the algorithm. How to balance these two aspects might be another interesting research topic.

CHAPTER 7

Appendix

The missing proofs in Chapters 4 and 5 are provided in this chapter.

7.1 Proofs for Chapter 4

7.1.1 Proof of Lemma 8

We first present the following three lemmas which will be used in the proof of Lemma 8.

Lemma 16 *Consider a positive semi-definite matrix $B = B^T \in \mathbb{R}^{n\times n}$ and a matrix $S \in \mathbb{R}^{n\times m}$ such that $\mathcal{R}(S) \cap \ker(B) = \{0\}$. The matrix S^TBS is diagonalizable and positive definite.*

Proof. The matrix S^TBS being symmetric implies it is diagonalizable and positive semi-definite since B is also positive semi-definite. Assume, for the sake of contradiction, that it is not positive definite. Then, there exists $x \neq 0 \in \mathbb{R}^n$ such that $x^TS^TBSx = 0$. Since $x^TS^TBSx = \|\sqrt{B}Sx\|_2^2$ we conclude that $\sqrt{B}Sx = 0$, i.e., $Sx \in \ker(B)$ since any matrix B and its square root $\sqrt{B}$ have the same kernel, a contradiction with $\mathcal{R}(S)\cap\ker(B) = \{0\}$. □

Lemma 17 *Consider a block square matrix $H = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$, where $A, B, C, D \in \mathbb{R}^{r\times r}$ and $r \in \mathbb{N}$ satisfy the following properties: (1) blocks A and D are upper-triangular and have $a_1, a_2, \ldots, a_r$ and $d_1, d_2, \ldots, d_r$ as their diagonal entries respectively, and (2) blocks B and*

C are diagonal and have $b_1, b_2, \ldots, b_r$ and $c_1, c_2, \ldots, c_r$ as their diagonal entries respectively. The matrix H is stable if every 2×2 matrix $\begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix}$ is stable for i ranging from 1 to r.

Proof. We will only prove the special case when $r = 2$ since the rest of the proof follows by an induction argument, which we omit for the sake of brevity.

We explicitly write out the matrix H:

$$H = \left[\begin{array}{cc|cc} a_1 & a_0 & b_1 & \\ & a_2 & & b_2 \\ \hline c_1 & & d_1 & d_0 \\ & c_2 & & d_2 \end{array}\right],$$

and compare with another matrix $\hat{H}$:

$$\hat{H} = \left[\begin{array}{cc|cc} a_1 & b_1 & a_0 & \\ c_1 & d_1 & & d_0 \\ \hline & & a_2 & b_2 \\ & & c_2 & d_2 \end{array}\right].$$

We observe that $\hat{H}$ can be obtained from H by a similarity transformation: $\hat{H} = T^{-1}HT$, where

$$T = T^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

In other words, matrices H and $\hat{H}$ share the same set of eigenvalues. Moreover, we obtain that $\hat{H}$ is a block diagonal matrix with both of its diagonal blocks $\begin{bmatrix} a_1 & b_1 \\ c_1 & d_1 \end{bmatrix}$ and $\begin{bmatrix} a_2 & b_2 \\ c_2 & d_2 \end{bmatrix}$ being stable. This shows $\hat{H}$ is a stable matrix, which finishes our proof.

Comment 1 *Moreover, to simplify we write the dynamics $x[t+1] = Hx[t]$ as $x^+ = Hx$ since t does not play a role in our analysis.*

We consider the following linear system of order 4:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}^+ = \underbrace{\left[\begin{array}{cc|cc} a_1 & a_0 & b_1 & \\ & a_2 & & b_2 \\ \hline c_1 & & d_1 & d_0 \\ & c_2 & & d_2 \end{array}\right]}_{H} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}. \tag{7.1}$$

To prove H is stable, it is sufficient to prove the system (7.1) is stable, under the assumption that both matrices $\begin{bmatrix} a_1 & b_1 \\ c_1 & d_1 \end{bmatrix}$ *and* $\begin{bmatrix} a_2 & b_2 \\ c_2 & d_2 \end{bmatrix}$ *are stable. We note that system (7.1) can be decomposed into the following 2 sub-systems each of order 2:*

$$\begin{cases} \begin{bmatrix} x_2 \\ x_4 \end{bmatrix}^+ = \begin{bmatrix} a_2 & b_2 \\ c_2 & d_2 \end{bmatrix} \begin{bmatrix} x_2 \\ x_4 \end{bmatrix}, \\ \begin{bmatrix} x_1 \\ x_3 \end{bmatrix}^+ = \begin{bmatrix} a_1 & b_1 \\ c_1 & d_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} + \begin{bmatrix} a_0 x_2 \\ d_2 x_4 \end{bmatrix}. \end{cases} \tag{7.2}$$

The first sub-system has x_2 and x_4 as its states. By the assumption that the matrix $\begin{bmatrix} a_2 & b_2 \\ c_2 & d_2 \end{bmatrix}$ *is stable, this system is a stable system. Furthermore, the second sub-system with states x_1 and x_3 is input-to-state stable if we treat the last term* $\begin{bmatrix} a_0 x_2 \\ d_2 x_4 \end{bmatrix}$ *as the input, since by assumption* $\begin{bmatrix} a_1 & b_1 \\ c_1 & d_1 \end{bmatrix}$ *is also stable. Since the interconnection of an ISS system with an asymptotically stable system results in an asymptotically stable system (see, e.g., [Son08]), we conclude that H is a stable matrix.*

□

Lemma 18 *Consider a block matrix B of the form* $B = \begin{bmatrix} A - I - 2k_I^2\lambda^2 I_n & -2k_I k_P \lambda I_n \\ k_I \lambda I_n & A \end{bmatrix}$

where $A \in \mathbb{R}^{n\times n}$ and λ is a non-zero eigenvalue of $\mathcal{L}$. There always exist $k_I, k_P \in \mathbb{R}$ such that B is stable[1] *if A satisfies Assumption 6 in Section 4.2.*

Proof. We first find the Jordan decomposition of A. Let $J = T^{-1}AT$ be the Jordan form of A, J is an upper-triangular matrix with entries $a_1, a_2, \ldots, a_n$ on its diagonal. Instead of matrix B, we focus on the following matrix which has the same eigenvalues as B:

$$\begin{bmatrix} T^{-1} & \\ & T^{-1} \end{bmatrix} \begin{bmatrix} A - I - 2k_I^2\lambda^2 I_n & -2k_P k_I \lambda I_n \\ k_I \lambda I_n & A \end{bmatrix} \begin{bmatrix} T & \\ & T \end{bmatrix}$$
$$= \begin{bmatrix} J - I - 2k_P^2\lambda^2 I_n & -2k_P k_I \lambda I_n \\ k_P \lambda I_n & J \end{bmatrix}.$$

Now we invoke Lemma (17) and obtain that B is stable if the matrix

$$R(a,\lambda) = \begin{bmatrix} a - 1 - 2k_I^2\lambda^2 & -2k_P k_I \lambda \\ k_I \lambda & a \end{bmatrix}$$

is stable for any $a \in \{a_1, a_2, \ldots, a_n\}$ being an eigenvalue of A and any λ being a non-zero eigenvalue of $\mathcal{L}$. In the next step we compute the eigenvalues k_1, k_2 of matrix $R(a,\lambda)$:

$$\begin{aligned} & (a - 1 - 2k_I^2\lambda^2 - k)(a - k) + 2k_P k_I^2 \lambda^2 = 0 \\ \Rightarrow \quad & k^2 + (2k_I^2\lambda^2 - 2a + 1)k + (2k_p k_I^2 \lambda^2 + a^2 - a - 2ak_I^2\lambda^2) \\ & = 0. \end{aligned}$$

The eigenvalues of the matrix $R(a,\lambda)$ are given by:

$$\begin{aligned} 2k_1 &= -2k_I^2\lambda^2 + 2a - 1 + \sqrt{(2k_I^2\lambda^2 + 1)^2 - 8k_P k_I^2 \lambda^2}, \\ 2k_2 &= -2k_I^2\lambda^2 + 2a - 1 - \sqrt{(2k_I^2\lambda^2 + 1)^2 - 8k_P k_I^2 \lambda^2}. \end{aligned}$$

Lastly, we provide a possible choice of λ and c such that for any aforementioned a and λ, both eigenvalues k_1 and k_2 are stable. We pick $k_P = 1$ and $k_I = \frac{1}{\sqrt{2}\lambda_{\max}(\mathcal{L})}$, where we recall that $\lambda_{\max}(\mathcal{L})$ is the largest eigenvalue of $\mathcal{L}$. This choice leads to:

$$k_1 = a - 1, \; k_2 = a - \frac{\lambda^2}{\lambda_{\max}^2(\mathcal{L})}.$$

[1] *We recall that a matrix is stable if and only if the magnitude of all its eigenvalues is strictly less than 1.*

By Assumption 6 both k_1 and k_2 are stable for any feasible choice of a and λ, which finishes our proof. □

Now we are ready to prove Lemma 8.

Proof of Lemma 8: To begin with, we reproduce Algorithm (4.8) for the sake of convenience:

$$\begin{cases} W_i^1[t+1] = & (\hat{A}-I)W_i^1[t] + 2k_I \sum_{j\in\mathcal{N}_i}(\eta_j[t]-\eta_i[t]) \\ & \quad + \phi_i^1[t], \\ b_i[t+1] = & \hat{A}b_i[t] - k_I \sum_{j\in\mathcal{N}_i}(W_j^1[t]-W_i^1[t]), \\ \eta_i[t] = & k_P b_i[t] - k_I \sum_{j\in\mathcal{N}_i}(W_j^1[t]-W_i^1[t]), \end{cases} \tag{7.3}$$

and rewrite it in the following compact form:

$$\begin{cases} W^1[t+1] = & (I_p\otimes\hat{A}-I_{np})W^1[t] + \phi^1[t] \\ & \quad - 2k_I(\mathcal{L}\otimes I_n)\eta[t], \\ b[t+1] = & (I_p\otimes\hat{A})b[t] + k_I(\mathcal{L}\otimes I_n)W^1[t], \\ \eta[t] = & k_P b[t] + k_I(\mathcal{L}\otimes I_n)W^1[t], \end{cases} \tag{7.4}$$

where $W^1 = \begin{bmatrix}(W_1^1)^T & (W_2^1)^T & \dots & (W_p^1)^T\end{bmatrix}^T$ and $\phi^1 = \begin{bmatrix}(\phi_1^1)^T & (\phi_2^1)^T & \dots & (\phi_p^1)^T\end{bmatrix}^T$ are obtained by concatenating estimates and weighted measurements from all nodes in the network, $b = \begin{bmatrix}b_1^T & b_2^T & \dots & b_p^T\end{bmatrix}^T$ and $\eta = \begin{bmatrix}\eta_1^T & \eta_2^T & \dots & \eta_p^T\end{bmatrix}^T$ are similarly defined.

We will take three steps to prove Lemma 8. In step 1, we decompose the stacked version of the tracking algorithm (7.4) and show that, the sum of local estimates $\sum_{i=1}^p W_i^1$ and the dissimilarity between local estimates at different nodes $W_i^1 - W_j^1 (i \neq j)$ can be studied independently. This motivates our step 2 in which we show that the sum of local estimates $\sum_{i=1}^p W_i^1$ converges to the sum of local reference signals $\sum_{i=1}^p \phi_i^1$. Lastly, in step 3, we argue that local estimates at all nodes are identical if t is sufficiently large. Combining these three steps we finish our proof.

Step 1: Decomposition. We first introduce the matrix $R = \frac{1}{\sqrt{p}}\mathbf{1}_p$ and pick $S \in \mathbb{R}^{n\times(n-1)}$ such that the matrix $\begin{bmatrix} R & S \end{bmatrix} \in \mathbb{R}^{n\times n}$ is orthogonal. Then we left multiply $\begin{bmatrix} R^T \otimes I_n \\ S^T \otimes I_n \end{bmatrix}$ on both sides of all three equations in the set (7.4). The following set of equalities is obtained:

$$\begin{cases} \begin{bmatrix} R^T \otimes I_n \\ S^T \otimes I_n \end{bmatrix} W^1[t+1] = & \begin{bmatrix} R^T \otimes I_n \\ S^T \otimes I_n \end{bmatrix} (I_p \otimes \hat{A} - I_{np}) \begin{bmatrix} R \otimes I_n & S \otimes I_n \end{bmatrix} \begin{bmatrix} R^T \otimes I_n \\ S^T \otimes I_n \end{bmatrix} W^1[t] \\ & - 2k_I \begin{bmatrix} R^T \otimes I_n \\ S^T \otimes I_n \end{bmatrix} (\mathcal{L} \otimes I_n) \begin{bmatrix} R \otimes I_n & S \otimes I_n \end{bmatrix} \begin{bmatrix} R^T \otimes I_n \\ S^T \otimes I_n \end{bmatrix} \eta[t] \\ & + \begin{bmatrix} R^T \otimes I_n \\ S^T \otimes I_n \end{bmatrix} \phi^1[t], \\ \begin{bmatrix} R^T \otimes I_n \\ S^T \otimes I_n \end{bmatrix} b[t+1] = & \begin{bmatrix} R^T \otimes I_n \\ S^T \otimes I_n \end{bmatrix} (I_p \otimes \hat{A}) \begin{bmatrix} R \otimes I_n & S \otimes I_n \end{bmatrix} \begin{bmatrix} R^T \otimes I_n \\ S^T \otimes I_n \end{bmatrix} b[t] \\ & + k_I \begin{bmatrix} R^T \otimes I_n \\ S^T \otimes I_n \end{bmatrix} (\mathcal{L} \otimes I_n) \begin{bmatrix} R \otimes I_n & S \otimes I_n \end{bmatrix} \begin{bmatrix} R^T \otimes I_n \\ S^T \otimes I_n \end{bmatrix} W^1[t], \\ \begin{bmatrix} R^T \otimes I_n \\ S^T \otimes I_n \end{bmatrix} \eta[t] = & k_P \begin{bmatrix} R^T \otimes I_n \\ S^T \otimes I_n \end{bmatrix} \begin{bmatrix} R \otimes I_n & S \otimes I_n \end{bmatrix} \begin{bmatrix} R^T \otimes I_n \\ S^T \otimes I_n \end{bmatrix} b[t] \\ & + k_I \begin{bmatrix} R^T \otimes I_n \\ S^T \otimes I_n \end{bmatrix} (\mathcal{L} \otimes I_n) \begin{bmatrix} R \otimes I_n & S \otimes I_n \end{bmatrix} \begin{bmatrix} R^T \otimes I_n \\ S^T \otimes I_n \end{bmatrix} W^1[t]. \end{cases} \tag{7.5}$$

Exploiting the facts that $\mathcal{L}\mathbf{1}_p = 0$ and $\mathcal{L}$ is a symmetric matrix, Equation (7.5) is simplified to:

$$z_1[t+1] = (\hat{A} - I_n) z_1[t] + \varphi_1[t], \tag{7.6}$$

and:

$$\begin{cases} z_2[t+1] = & (I_{p-1} \otimes \hat{A} - I_{n(p-1)}) z_1[t] + \varphi_2[t] \\ & \quad - 2k_I(\mathcal{L}' \otimes I_n)\eta_2[t], \\ b_2[t+1] = & (I_{p-1} \otimes \hat{A}) b_2[t] + k_I(\mathcal{L}' \otimes I_n)\eta_2[t], \\ \eta_2[t] = & k_P b_2[t] - k_I(\mathcal{L}' \otimes I_n) z_2[t], \end{cases} \tag{7.7}$$

where $z_1[t] = (R^T \otimes I_n)W^1[t]$, $z_2[t] = (S^T \otimes I_n)W^1[t]$, $\varphi_1[t] = (R^T \otimes I_n)\phi^1[t]$, $\varphi_2[t] = (S^T \otimes I_n)\phi^1[t]$, $b_2[t] = (S^T \otimes I_n)b[t]$, $\eta_2[t] = (S^T \otimes I_n)\eta_t$, and $\mathcal{L}' = S^T \mathcal{L} S$. To proceed, we invoke Lemma 16 here and observe that $\mathcal{L}'$ is diagonalizable and positive definite. Moreover, by construction of matrix S, $\mathcal{L}$ and $\mathcal{L}'$ share the same set of non-zero eigenvalues.

Step 2: Track the Sum of Reference Signals. In this part we show that z_1 (which resembles the sum of estimates at all nodes) converges to φ_1 (which resembles the sum of reference signals).

We have the following set of equalities:

$$\begin{aligned} & z_1[t+1] - \varphi_1[t+1] \\ \overset{(a)}{=} \; & z_1[t+1] - (R^T \otimes I_n)\phi^1[t+1] \\ \overset{(b)}{=} \; & (\hat{A} - I_n) z_1[t] + \varphi_1[t] - (R^T \otimes I_n)\phi^1[t+1] \\ = \; & (\hat{A} - I_n) z_1[t] + \varphi_1[t] - (R^T \otimes I_n)(I_p \otimes \hat{A})\phi^1[t] \\ = \; & (\hat{A} - I_n) z_1[t] + \varphi_1[t] - (R^T \otimes \hat{A})\phi^1[t] \\ = \; & (\hat{A} - I_n) z_1[t] + \varphi_1[t] - (1 \otimes \hat{A})(R^T \otimes I_n)\phi^1[t] \\ \overset{(c)}{=} \; & (\hat{A} - I_n) z_1[t] + \varphi_1[t] - \hat{A}\varphi_1[t] \\ = \; & (\hat{A} - I_n)(z_1[t] - \varphi_1[t]), \end{aligned} \tag{7.8}$$

where in steps (a) and (c) we use the definition $\varphi_1 = (R^T \otimes I_n)\phi^1[t]$, and in step (b) we plug in Equation (7.6).

To see why $\hat{A} - I_n$ is a stable matrix, we consider any eigenvalue $m + n\mathbf{i}$ of $\hat{A} - I_n$, we observe that $m + n\mathbf{i} + 1$ must be an eigenvalue of $\hat{A}$, which is also an eigenvalue of A since $\hat{A}$

and A share the same spectrum. By Assumption 6 we have $m^2 + n^2 < 1$, which shows that $\hat{A} - I_n$ is a stable matrix.

From Equation (7.8) and the fact that $(\hat{A} - I_n)$ is a stable matrix we draw the following conclusion:

Conclusion 1 *For any solution $z_1[t]$ of (7.6), there exist $0 < \alpha < 1$ and $\beta > 0$ such that:*

$$\|z_1[t] - \varphi_1[t]\| < \beta\alpha^t.$$

Step 3: Reach Consensus. In this part we show that the local estimates W_i^1 at all nodes reach consensus. Practically, we prove that $z_2 = (S^T \otimes I_n)W^1$ converges to 0 which serves the same purpose.

Substituting η_2 with $k_P b_2 - k_I(\mathcal{L}' \otimes I_n)z_2$ (i.e., the third equation in (7.7)), we obtain the following dynamics:

$$\begin{bmatrix} z_2[t+1] \\ b_2[t+1] \end{bmatrix} = B \cdot \begin{bmatrix} z_2[t] \\ b_2[t] \end{bmatrix} + \begin{bmatrix} \varphi_2[t] \\ 0 \end{bmatrix}, \tag{7.9}$$

where $B =$

$$\begin{bmatrix} I_{p-1} \otimes \hat{A} - I_{n(p-1)} - 2k_I^2(\mathcal{L}'^2 \otimes I_n) & -2k_I k_P(\mathcal{L}' \otimes I_n) \\ k_I(\mathcal{L}' \otimes I_n) & I_{p-1} \otimes \hat{A} \end{bmatrix}.$$

It is simple to see that, if B is stable, for any two solutions w and w' of (7.9), we have $\lim_{t\to\infty}(w[t] - w'[t]) = 0$, since $z = w - w'$ satisfies $z[t+1] = Bz[t]$. In other words, if we can prove: *1.* there exists a function $\hat{b}_2[t] : \mathbb{R} \to \mathbb{R}^{n(p-1)}$ such that $(0, \hat{b}_2)$ is a solution of (7.9), and *2.* the matrix B is Hurwitz, then the proposition is proved.

First we prove that $(0, \hat{b}_2[t])$ is the solution of (7.9) for some $\hat{b}_2[t]$. By substituting $z_2 = 0$ into (7.9) we obtain:

$$\begin{cases} \varphi_2[t] = & 2k_P k_I(\mathcal{L}' \otimes I_n)\hat{b}_2[t], \\ \hat{b}_2[t+1] = & (I_{p-1} \otimes \hat{A})\hat{b}_2[t]. \end{cases} \tag{7.10}$$

We make a guess that $(\hat{z}_2 = 0, \hat{b}_2[t] = \frac{1}{2k_P k_I}(\mathcal{L}'^{-1} \otimes I_n)\varphi_2[t])$ is a feasible solution of Equation (7.9) since the first equation in (7.10) is naturally satisfied. To prove this is a solution of (7.9), we show that if the second equality in (7.10) holds at time t, then it will also hold at time $t+1$. Then a simple induction can be conducted and the proof could thus be obtained. We check that:

$$\begin{aligned}
\hat{b}_2[t+1] &= \frac{1}{2k_P k_I}(\mathcal{L}'^{-1} \otimes I_n)\varphi_2[t+1] \\
&= \frac{1}{2k_P k_I}(\mathcal{L}'^{-1} \otimes I_n)(S^T \otimes I_n)\phi^1[t+1] \\
&= \frac{1}{2k_P k_I}(\mathcal{L}'^{-1} \otimes I_n)(S^T \otimes I_n)(I_p \otimes \hat{A})\phi^1[t] \\
&= \frac{1}{2k_P k_I}(\mathcal{L}'^{-1} \otimes I_n)(I_{p-1} \otimes \hat{A})(S^T \otimes I_n)\phi^1[t] \\
&= \frac{1}{2k_P k_I}(\mathcal{L}'^{-1} \otimes I_n)(I_{p-1} \otimes \hat{A})\varphi_2[t] \\
&= \frac{1}{2k_P k_I}(I_{p-1} \otimes \hat{A})(\mathcal{L}'^{-1} \otimes I_n)\varphi_2[t] \\
&= (I_{p-1} \otimes \hat{A})\hat{b}_2[t].
\end{aligned}$$

This concludes our proof that $(\hat{z}_2, \hat{b}_2)$ is a solution to (7.9).

To prove that matrix B can be made stable by properly choosing k_P and k_I, we recall that $\mathcal{L}'$ is positive definite, which implies $\mathcal{L}'$ is diagonalizable. Assume $P^{-1}\mathcal{L}'P$ is a diagonal matrix, where $P \in \mathbb{R}^{p-1}$, we consider the following similarity transformation of B:

$$\tilde{B} = \begin{bmatrix} P^{-1} \otimes I_n & 0 \\ 0 & P^{-1} \otimes I_n \end{bmatrix} B \begin{bmatrix} P \otimes I_n & 0 \\ 0 & P \otimes I_n \end{bmatrix}.$$

For the sake of brevity we do not write down matrix $\tilde{B}$ explicitly. However, we note that $\tilde{B}$ and B have the same set of eigenvalues. We can also conclude, from the structure of the matrix $\tilde{B}$, that $\tilde{B}$ is stable if all the following blocks:

$$\begin{bmatrix} A - I - 2k_I^2\lambda^2 I_n & -2k_I k_P \lambda I_n \\ k_I \lambda I_n & A \end{bmatrix},$$

are stable, where λ is an eigenvalue of $\mathcal{L}'$. We invoke Lemma 18 and see that there exist a choice of k_I and k_P such that for any λ being an eigenvalue of $\mathcal{L}'$, the aforementioned matrix is stable, which finishes our proof that B can be made Hurwitz.

The conclusion of step 3 is as follows:

Conclusion 2 *For any solution* $(z_2[t], b_2[t])$ *of* (7.7) *we have:*

$$\|z_2[t]\| < \beta\alpha^t,$$

for some $0 < \alpha < 1$ *and* $\beta > 0$.

Combining Conclusions 1 and 2 we obtain the proof of Lemma 8. □

7.1.2 Proof of Lemma 9

Proof. The solution $(\hat{x}, \hat{E})$ to the optimization problem (4.9) satisfies the following set of inequalities:

$$\begin{aligned} & \|W_i - (D \otimes I_n)(\mathcal{O}\hat{x} + \hat{E})\|_2 \\ \leq\ & \|W_i - (D \otimes I_n)(\mathcal{O}x + E)\|_2 \\ \leq\ & \alpha, \end{aligned}$$

which implies:

$$\|(D \otimes I_n)(\mathcal{O}(\hat{x} - x) + (\hat{E} - E))\|_2 \leq 2\alpha. \tag{7.11}$$

Moreover, let $\mathcal{K}_1$ be the index set of non-zero blocks in E, $\mathcal{K}_2$ be the index set of non-zero blocks in $\hat{E}$, and $\mathcal{K} = \mathcal{K}_1 \cup \mathcal{K}_2$. Note that $|\mathcal{K}| \leq 2s$. We pick $L \in \mathbf{P}_{2s}$ such that[2] $\ker(L) = \text{span}\{\mathbf{e}_i, i \in \mathcal{K}\}$ and $\|L\|_2 = 1$. We observe that the following inequality holds:

$$\|((L \otimes I_n)(D \otimes I_n)(\mathcal{O}(\hat{x} - x))\|_2 \leq 2\alpha. \tag{7.12}$$

[2]The definition of $\mathbf{P}_{2s}$ can be found in the definition of sparse detectability with respect to a matrix in Section 4.3, and the norm condition can be satisfied by a proper normalization.

By assumption (A, C) is $2s$-sparse detectable with respect to D, it follows that the matrix $(L \otimes I_n)(D \otimes I_n)\mathcal{O}$ has full column rank. In other words, we can pick $\beta^{-1} = \frac{1}{2}\sigma_{\min}((L \otimes I_n)(D \otimes I_n)\mathcal{O}) > 0$ and then Equation (7.12) directly implies $\|\hat{x} - x\|_2 \leq \beta\alpha$, which finishes our proof. □

7.2 Proofs for Chapter 5

7.2.1 Proof of Lemma 10

We first introduce the following two results which will be used in the proof of Lemma 10.

Lemma 19 *([CLS04]) Consider a schur stable[3] matrix $F \in \mathbb{R}^{r\times r}$. There always exist $m \geq 1$ and $0 < \rho < 1$ such that for any $n \in \mathbb{N}$, the following bound holds:*

$$\|F^n\|_2 \leq m\rho^n. \tag{7.13}$$

Proof. We first perform a Jordan decomposition of the matrix F: $F = T^{-1}JT$ where $T \in \mathbb{R}^{n\times n}$ is an invertible matrix and J is in block diagonal form, i.e., $J = \text{diag}\{J_1, J_2, \ldots, J_l\}$, where each J_i is a Jordan block. To prove Lemma 4.1, it suffices to prove the existence of $m_i \geq 1$ and $0 < \rho_i < 1$ for each block J_i such that $\|J_i^n\|_2 \leq m_i\rho_i^n$ holds.

Let the eigenvalue corresponding to block J_i be λ_i, and furthermore assume J_i is of size $s_i \times s_i$. We explicitly write out J_i^n as:

$$J_i^n = \begin{bmatrix} \lambda_i^n & \binom{n}{1}\lambda_i^{n-1} & \binom{n}{2}\lambda_i^{n-2} & \ldots & \binom{n}{s_i-1}\lambda_i^{n-s_i+1} \\ 0 & \lambda_i^n & \binom{n}{1}\lambda_i^{n-1} & \ldots & \binom{n}{s_i-2}\lambda_i^{n-s_i+2} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_i^n & \binom{n}{1}\lambda_i^{n-1} \\ 0 & 0 & \ldots & 0 & \lambda_i^n \end{bmatrix}, \tag{7.14}$$

[3] *A matrix is called schur stable if all of its eigenvalues lie strictly in the unit circle.*

from which we have:

$$\|J_i^n\|_2 \leq \sqrt{n}\|J_i^n\|_1 = \sum_{j=0}^{s_i-1} \binom{n}{j} |\lambda_i|^{n-j}. \tag{7.15}$$

For simplicity, we define $U_i(n) = \sum_{j=0}^{s_i-1} \binom{n}{j} |\lambda_i|^{n-j}$. We note that $U_i(n)$ is a decaying sequence of n when n is large enough, since:

$$\begin{aligned}
\frac{U_i(n+1)}{U_i(n)} &= \frac{\sum_{j=0}^{s_i-1} \binom{n+1}{j} |\lambda_i|^{n+1-j}}{\sum_{j=0}^{s_i-1} \binom{n}{j} |\lambda_i|^{n-j}} && (7.16)\\
&= \frac{\sum_{j=0}^{s_i-1} \frac{(n+1)!}{j!(n+1-j)!} |\lambda_i|^{n+1-j}}{\sum_{j=0}^{s_i-1} \frac{n!}{j!(n-j)!} |\lambda_i|^{n-j}} && (7.17)\\
&\leq |\lambda_i| \max_{j \in \{0,1,\ldots,s_i-1\}} \frac{\frac{(n+1)!}{j!(n+1-j)!}}{\frac{n!}{j!(n-j)!}} && (7.18)\\
&= |\lambda_i| \max_{j \in \{0,1,\ldots,s_i-1\}} \frac{n+1}{n+1-j} && (7.19)\\
&= |\lambda_i| \cdot \frac{n+1}{n+2-s_i}. && (7.20)
\end{aligned}$$

By assumption F is a schur stable matrix, its eigenvalue λ_i satisfies $|\lambda_i| < 1$, which shows there exists $N_0 \in \mathbb{N}$ such that for any $n \geq N_0$, $\frac{U_i(n+1)}{U_i(n)} < 1$. By picking $m_i = \max\{1, U_i(N_0)\}$ and $\rho_i = |\lambda_i| \cdot \frac{N_0+1}{N_0+2-s_i}$ we finish the proof. □

Lemma 20 *The doubly-stochastic adjacency matrix $\mathbf{A} \in \mathbb{R}^{p\times p}$ associated with a connected graph has exactly one eigenvalue with value 1 and corresponding eigenvector $\mathbf{1}_p$. Any other eigenvalue of $\mathbf{A}$ lies strictly inside the unit circle.*

Proof. Lemma 20 is a standard result. For the sake of completeness we provide a sketch of its proof.

The proof is based on the definition and known facts about the degree matrix $\mathbf{D}$ and the Laplacian matrix $\mathcal{L}$ corresponding to a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{A}\}$. The degree matrix $\mathbf{D} \in \mathbb{R}^{p\times p}$ of the graph $\mathcal{G}$ is a diagonal matrix with its i-th diagonal element defined by $d_{ii} = \sum_{j=1}^{p} a_{ij}$. The Laplacian matrix $\mathcal{L}$ of the graph $\mathcal{G}$ is defined by $\mathcal{L} = \mathbf{D} - \mathbf{A}$. It is well-known in

the literature that if the graph is connected, by spectral theory, 0 is an eigenvalue of $\mathcal{L}$ of algebraic multiplicity 1 and its corresponding eigenvalue is $\mathbf{1}_p$.

It is also well-known that the magnitude of any eigenvalue of a doubly-stochastic matrix $\mathbf{A}$ is less or equal to 1. Moreover, since $\mathcal{L} = \mathbf{D} - \mathbf{A}$ has an eigenvalue 0 with algebraic multiplicity 1, and $\mathbf{D}$ is the identity matrix since $\mathbf{A}$ is doubly-stochastic, it is trivially seen that the algebraic multiplicity of eigenvalue 1 of $\mathbf{A}$ is 1. □

With Lemma 19 and Lemma 20 we provide the proof of Lemma 10 in the main file.

Proof of Lemma 10.([KVC19]) We define $x^k[t] = \left[(x_1^k[t])^T, (x_2^k[t])^T, \ldots, (x_p^k[t])^T\right]^T$ for any $k = 0, 1, \ldots, \tau$ and $t \in \mathbb{N}$. This definition allows us to write the mathematical representation of the linear iterative algorithm as the following:

$$x^{k+1}[t] = (\mathbf{A}^T \otimes I_n)x^k[t]. \tag{7.21}$$

To proceed, we find an orthogonal matrix $\begin{bmatrix} R & S \end{bmatrix} \in \mathbb{R}^{p\times p}$ where $R = \frac{1}{\sqrt{p}}\mathbf{1}_p$ and define the following change of coordinates: $z_1^k[t] = (R^T \otimes I_n)x^k[t]$ which is the sum of local values, and $z_2^k[t] = (S^T \otimes I_n)x^k[t]$. To understand the implication of the vector $z_2^k[t]$, we note that $z_2^k[t]$ can be equally divided into $p-1$ blocks each of size n and each of which can be explained as a linear combination of vectors in the set $\{x_i^k[t],\ i \in P\}$, i.e., there exists a set of weights $\{w_1, w_2, \ldots, w_p\}$ such that each block in $z_2^k[t]$ can be written as $\sum_{i\in P} w_i x_i^k[t]$. Moreover, by construction of the matrix S, we have $\sum_{i\in P} w_i = 0$. This observation shows that by properly combining terms with positive and negative coefficients, each block in $z_2^k[t]$ can be alternatively expressed by a weighted sum of vectors in the set $\{x_i^k[t] - x_j^k[t],\ i, j \in P\}$ where any weight w'_{ij} is non-negative. This argument shows that $z_2^k[t]$ is closely related to the difference among local values from different nodes. We also point out the following two properties regarding the weight values w_i and w'_{ij}, which will very soon be used in this proof:

- Fact 1: $\sum_{i\in P, w_i>0} w_i \leq \sqrt{\frac{p}{2}}$, which is obtained by combining facts $\sum_{i\in P} w_i = 0$ and $\sum_{i\in P} w_i^2 = 1$ with the Cauchy-schwardz inequality,

- Fact 2: $\sum_{i,j\in P} w'_{ij} = \sum_{i\in P, w_i>0} w_i \leq \sqrt{\frac{p}{2}}$.

With the understanding of the change of coordinates, we obtain the following set of equalities stating from (7.21):

$$
\begin{aligned}
& x^{k+1}[t] = (\mathbf{A}^T \otimes I_n) x^k[t] \\
\overset{(a)}{\Rightarrow}\ & \left(\begin{bmatrix} R^T \\ S^T \end{bmatrix} \otimes I_n\right) x^{k+1}[t] = \left(\begin{bmatrix} R^T \\ S^T \end{bmatrix} \otimes I_n\right)(\mathbf{A}^T \otimes I_n) x^k[t] \\
\overset{(b)}{\Rightarrow}\ & \left(\begin{bmatrix} R^T \\ S^T \end{bmatrix} \otimes I_n\right) x^{k+1}[t] = \left(\begin{bmatrix} R^T \\ S^T \end{bmatrix} \otimes I_n\right)(\mathbf{A}^T \otimes I_n)\left(\begin{bmatrix} R & S \end{bmatrix} \otimes I_n\right)\left(\begin{bmatrix} R^T \\ S^T \end{bmatrix} \otimes I_n\right) x^k[t] \\
\Rightarrow\ & \left(\begin{bmatrix} R^T \\ S^T \end{bmatrix} \otimes I_n\right) x^{k+1}[t] = \left(\begin{bmatrix} R^T\mathbf{A}^T R & R^T\mathbf{A}^T S \\ S^T\mathbf{A}^T R & S^T\mathbf{A}^T S \end{bmatrix} \otimes I_n\right)\left(\begin{bmatrix} R^T \\ S^T \end{bmatrix} \otimes I_n\right) x^k[t] \\
\overset{(c)}{\Rightarrow}\ & \left(\begin{bmatrix} R^T \\ S^T \end{bmatrix} \otimes I_n\right) x^{k+1}[t] = \left(\begin{bmatrix} R^T R & R^T S \\ S^T R & S^T\mathbf{A}^T S \end{bmatrix} \otimes I_n\right)\left(\begin{bmatrix} R^T \\ S^T \end{bmatrix} \otimes I_n\right) x^k[t] \\
\overset{(d)}{\Rightarrow}\ & \left(\begin{bmatrix} R^T \\ S^T \end{bmatrix} \otimes I_n\right) x^{k+1}[t] = \left(\begin{bmatrix} 1 & 0 \\ 0 & S^T\mathbf{A}^T S \end{bmatrix} \otimes I_n\right)\left(\begin{bmatrix} R^T \\ S^T \end{bmatrix} \otimes I_n\right) x^k[t],
\end{aligned}
$$

where in step (a) we left multiply a matrix $\left(\begin{bmatrix} R^T \\ S^T \end{bmatrix} \otimes I_n\right)$ on both sides of the equation and in step (b) we use the fact that $\begin{bmatrix} R & S \end{bmatrix}$ is an orthogonal matrix. Moreover, step (c) is true since by Assumption 7 the adjacency matrix of the communication graph $\mathbf{A}$ is doubly-stochastic, and so is its transpose $\mathbf{A}^T$, which implies $\mathbf{A}^T R = R$ as well as $R^T\mathbf{A}^T = R^T$. Lastly, in step (d) we again invoke the orthogonality of the matrix $\begin{bmatrix} R & S \end{bmatrix}$ and the definition of the vector R. From these set of equalities we observe that Equation (7.21) can be decoupled into:

$$
\begin{cases}
z_1^{k+1}[t] = & z_1^k[t], \\
z_2^{k+1}[t] = & (\mathbf{A}' \otimes I_n) z_2^k[t],
\end{cases}
\tag{7.22}
$$

where $\mathbf{A}'$ is defined as $\mathbf{A}' = S^T\mathbf{A}^T S$. We note that Equation (7.22) can be easily generated

to:

$$\begin{cases} z_1^\tau[t] = & z_1^0[t], \\ z_2^\tau[t] = & (\mathbf{A}' \otimes I_n) z_2^0[t], \end{cases} \tag{7.23}$$

Recall the definition $z_1^k[t] = (R^T \otimes I_n)x^k[t] = \frac{1}{\sqrt{p}} \sum_{i \in P} x_i^0[t] = \sqrt{p}\bar{x}[t]$, the first property is obtained.

To prove the second property, we note that $x_i^k[t] - x_j^k[t] = (B_{ij} \otimes I_n)x^k[t]$ where $B_{ij} \in \mathbb{R}^{1 \times p}$ is a sparse row vector with zeros almost everywhere except its i-th entry being 1 and its j-th entry being -1. This definition allows the following chain of equations:

$$x_i^k[t] - x_j^k[t] = (B_{ij} \otimes I_n)x^k[t] = (B_{ij} \otimes I_n)(S \otimes I_n)(S^T \otimes I_n)x^k[t] = ((B_{ij}S) \otimes I_n)z_2^k[t], \tag{7.24}$$

where the second step holds since B_{ij} is perpendicular to the left kernel of S. Moreover, we note that the following matrix:

$$\begin{bmatrix} R^T \\ S^T \end{bmatrix} \mathbf{A} \begin{bmatrix} R & S \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & S^T \mathbf{A} S \end{bmatrix}$$

has exactly the same set of eigenvalues as matrix $\mathbf{A}$ by construction of $\begin{bmatrix} R & S \end{bmatrix}$. Since $\mathbf{A}$ has only one eigenvalue 1 and the rest of its eigenvalues lie strictly in the unit circle, we conclude that all eigenvalues of $(\mathbf{A}' \otimes I_n)$ strictly lie in the unit circle. Combining these two observations we obtain the following set of inequalities:

$$\begin{aligned} d^\tau[t] &= \max_{i,j \in P} \|x_i^\tau[t] - x_j^\tau[t]\|_2 \\ &\overset{(a)}{=} \max_{i,j \in P} \|((B_{ij}S) \otimes I_n)z_2^k[t]\|_2 \\ &\overset{(b)}{\leq} \max_{i,j \in P} \|((B_{ij}S) \otimes I_n)\|_2 \cdot \|(\mathbf{A}' \otimes I_n)^\tau z_2^0[t]\|_2 \\ &\leq \sqrt{2}\sigma_M(S)\|\mathbf{A}' \otimes I_n\|_2^m \cdot \|z_2^0[t]\|_2 \\ &\overset{(c)}{\leq} \sqrt{2}\sigma_M(S)m\rho^\tau\|z_2^0[t]\|_2 \\ &\overset{(d)}{\leq} \sqrt{2}\sigma_M(S)\rho^\tau\sqrt{p-1}\sqrt{\frac{p}{2}} \max_{i,j \in P} \|x_i^0[t] - x_j^0[t]\|_2 \\ &< \underbrace{p\sigma_M(S)}_{a} \rho^\tau d^0[t], \end{aligned} \tag{7.25}$$

where in step (a) and step (b) we plug in Equations (7.24) and (7.23), respectively. In step (c) we invoke Lemma 19 since $\mathbf{A}'$ is proved to be a schur stable matrix.

To see why step (d) holds, we recall that the vector $z_2^0[t]$ can be equally divided into $p-1$ blocks each of size n. Moreover, echoing our discussion beneath the definition of $z_2^k[t]$, each block in $z_2^0[t]$ can be expressed by a non-negative weighted sum of vectors in the set $\{x_i^0[t] - x_j^0[t],\ i,j \in P\}$, with the non-negative weight values $\{w'_{ij},\ i,j \in P\}$ sum up to at most $\sqrt{\frac{p}{2}}$. This observation leads to the following argument:

$$\begin{aligned} & \left\| \sum_{i,j \in P} w'_{ij}(x_i^0[t] - x_j^0[t]) \right\|_2 \\ \leq & \sum_{i,j \in P} w'_{ij} \|x_i^0[t] - x_j^0[t]\|_2 \\ \leq & \sum_{i,j \in P} w'_{ij} \max_{i,j \in P} \|x_i^0[t] - x_j^0[t]\|_2 \\ \leq & \sqrt{\frac{p}{2}} \max_{i,j \in P} \|x_i^0[t] - x_j^0[t]\|_2, \end{aligned} \tag{7.26}$$

which justifies our step (d) in which we claimed:

$$\left\| z_2^0[t] \right\|_2 \leq \sqrt{p-1}\sqrt{\frac{p}{2}} \max_{i,j \in P} \|x_i^0[t] - x_j^0[t]\|_2. \tag{7.27}$$

By choosing a to be $a = p\sigma_M(S)$ we finish the proof. □

7.2.2 Proof of Lemma 11

Proof. We prove the lemma by taking two steps. We first compare $\hat{\mu} = \frac{1}{g_e} \sum_{i \in V_e} a_i X_i$ with $\hat{\mu}_g = \frac{1}{g_e} \sum_{i \in V_e} a_i \nabla f_i$. This is understood as follows: there might be some attacked vectors left in the set of vectors $\{X_i \in \mathbb{R}^n | i \in V_e\}$, this is because these attacked vectors are so close to good vectors that the filter is unable to filter them out. For the same reason they do not pollute the estimate significantly. We first show if we restore the value of these vectors (i.e., replace X_i with ∇f_i for $i \in V_e \cap S_b$), the mean estimate does not change too much.

We have $\|X_i - \nabla f_i\|_2 \leq 2\kappa$ for any $i \in V_e \cap S_b$. To see why this holds, we arbitrarily

choose an $h \in V_e \cap S_g$, and we have $\|X_i - X_h\|_2 \leq \kappa$ as well as $\|\nabla f_i - \nabla f_h\|_2 \leq \kappa$. Since $X_h = \nabla f_h$, we get the result by adding them up.

Therefore, we have:

$$\|\hat{\mu} - \hat{\mu}_g\|_2 \leq \frac{\sum_{i \in V_e \cap S_b} (a_i \|X_i - \nabla f_i\|_2)}{g_e} \leq \frac{2\epsilon}{1 - 2\epsilon}\kappa. \tag{7.28}$$

Moreover, we observe that $\hat{\mu}_g$ is close to the desired value μ_g due to the following two reasons. (1) $\hat{\mu}_g$ is the average of the majority of good vectors while μ_g is the average of all good vectors, and (2) by Assumption 9 any two good vectors do not differ too much. We note first that if either Case 1 or Case 2 holds, then for any $i \in S$ we have the following set of inequalities:

$$\|\nabla f_i - \hat{\mu}_g\|_2 \leq \frac{\sum_{j \in V_e} (a_j \|\nabla f_i - \nabla f_j\|_2)}{g_e} \leq \kappa. \tag{7.29}$$

With these inequalities we can upper bound the distance between $\hat{\mu}_g$ and μ_g, in the following way:

$$\begin{aligned}
\|\mu_g - \hat{\mu}_g\|_2 &= \left\| \sum_{i \in S} a_i \nabla f_i - \hat{\mu}_g \right\|_2 \\
&= \left\| \sum_{i \in V_e} a_i \nabla f_i + \sum_{i \in S \setminus V_e} a_i \nabla f_i - (g_e \hat{\mu}_g + (1 - g_e)\hat{\mu}_g) \right\|_2 \\
&\overset{(a)}{=} \left\| \sum_{i \in V_e} a_i \nabla f_i + \sum_{i \in S \setminus V_e} a_i \nabla f_i - (\sum_{i \in V_e} a_i \nabla f_i + \sum_{i \in S \setminus V_e} a_i \hat{\mu}_g) \right\|_2 \\
&= \left\| \sum_{i \in S \setminus V_e} a_i \nabla f_i - \sum_{i \in S \setminus V_e} a_i \hat{\mu}_g \right\|_2 \\
&\leq \sum_{i \in S \setminus V_e} (a_i \|\nabla f_i - \hat{\mu}_g\|_2) \\
&\overset{(b)}{\leq} 2\epsilon\kappa,
\end{aligned} \tag{7.30}$$

where in step (a) we use the equalities $\hat{\mu}_g = \frac{1}{g_e} \sum_{i \in V_e} a_i \nabla f_i$ and $\sum_{i \in S \setminus V_e} a_i = 1 - g_e$, and in step (b) we use Equation (7.29). Summing up Equations (7.28) and (7.30):

$$\|\hat{\mu} - \mu_g\|_2 \leq \|\hat{\mu} - \hat{\mu}_g\|_2 + \|\hat{\mu}_g - \mu_g\|_2 \leq 2\epsilon\kappa + \frac{2\epsilon\kappa}{1 - 2\epsilon} = \left(2 + \frac{2}{1 - 2\epsilon}\right)\epsilon\kappa, \tag{7.31}$$

we obtain the proof of the lemma. □

7.2.3 Proof of Lemma 12

Proof. We consider the first time when an attacked vector X_i and a good vector X_h are picked since they have the maximum distance, and X_h is going to be removed at the end of this iteration. Note that at this point (before X_h is removed), the following inequality holds since all the removed vectors are attacked:

$$\sum_{z \in V \cap S_g} a_z \geq 1 - \epsilon.$$

We prove that, if it is the case, then there always exist an $l \in V \cap S_g$ such that $\|X_i - X_l\|_2 \leq \frac{1}{1-2\epsilon}\kappa$.

Before proving the claim, we first see what it implies. If $\|X_i - X_l\|_2 \leq \frac{1}{1-2\epsilon}\kappa$ holds, and by Assumption 9 the following holds: $\|\nabla f_h - \nabla f_l\|_2 = \|X_h - X_l\|_2 \leq \kappa$ since we assumed that $h, l \in S_g$, from which we conclude that:

$$\|X_i - X_h\|_2 \leq \|X_i - X_l\|_2 + \|X_h - X_l\|_2 \leq \left(1 + \frac{1}{1-2\epsilon}\right)\kappa. \tag{7.32}$$

On the other hand, we note that for any $q \in V_e \cap S_b$, we have:

$$\|X_q - X_h\|_2 \leq \|X_i - X_h\|_2 \leq \left(1 + \frac{1}{1-2\epsilon}\right)\kappa, \tag{7.33}$$

since otherwise Algorithm 2 would not have picked the pair X_i and X_h. Again, we invoke Assumption 9 and obtain that $\|X_h - \nabla f_q\|_2 = \|\nabla f_h - \nabla f_q\|_2 \leq \kappa$. Summing these inequalities up we have:

$$\|X_q - \nabla f_q\|_2 \leq \|X_q - X_h\|_2 + \|X_h - \nabla f_q\|_2 \leq \left(2 + \frac{1}{1-2\epsilon}\right)\kappa, \tag{7.34}$$

which is exactly the claim in the lemma.

Now we prove the claim by its contrapositive. Let X_f be the closest vector in $V \cap S_g$ to X_i in the Euclidean sense. For the sake of contradiction, assume the distance between X_i and X_f is strictly larger than d, where $d \geq \frac{1}{1-2\epsilon}\kappa$. We consider the following set of inequalities:

$$
\begin{aligned}
& \sum_{z\in V} a_z(\|X_i - X_z\|_2 - \|X_h - X_z\|_2) \\
= & \sum_{z\in V\cap S_g} a_z(\|X_i - X_z\|_2 - \|X_h - X_z\|_2) + \sum_{z\in V\cap S_b} a_z(\|X_i - X_z\|_2 - \|X_h - X_z\|_2) \\
\geq & \sum_{z\in V\cap S_g} a_z(\|X_i - X_z\|_2 - \|X_h - X_z\|_2) + \left(\sum_{z\in V\cap S_b} a_z\right)(-\|X_i - X_h\|_2) \\
\overset{(a)}{\geq} & \sum_{z\in V\cap S_g} a_z(\|X_i - X_f\|_2 - \|X_h - X_z\|_2) + \left(\sum_{z\in V\cap S_b} a_z\right)(-\|X_i - X_f\|_2 - \|X_f - X_j\|_2) \\
= & \left(\sum_{z\in V\cap S_g} a_z - \sum_{z\in V\cap S_b} a_z\right)\|X_i - X_f\|_2 - \sum_{z\in V\cap S_g} a_z\|X_h - X_z\|_2 - \sum_{z\in V\cap S_b} a_z\|X_f - X_j\|_2 \\
\overset{(b)}{>} & \left(\sum_{z\in V\cap S_g} a_z - \sum_{z\in V\cap S_b} a_z\right)d - \sum_{z\in V\cap S_g} a_z\kappa - \sum_{z\in V\cap S_b} a_z\kappa \\
\geq & (1-2\epsilon)d - \kappa \\
\overset{(c)}{\geq} & 0, \qquad (7.35)
\end{aligned}
$$

where in step (a) we use the fact that X_f is the closest attack-free vector in V to X_i, and in step (b) we directly replace with $\|X_i - X_f\|_2$ with d, and the inequality sign holds because $\sum_{z\in V\cap S_g} a_z - \sum_{z\in V\cap S_b} a_z > 0$, i.e., the weight sum of attack-free vectors is higher than the weight sum of attacked vectors. The last step (c) comes from the assumption that $d \geq \frac{1}{1-2\epsilon}\kappa$. From this set of inequalities we reach a contradiction since by assumption the good vector X_j was removed whereas X_h should have been removed. □

7.2.4 Proof of Lemma 13

Proof. Given Lemma 12, the proof of Lemma 13 is similar to the proof of 11. In particular, we have:

$$\|\hat{\mu} - \hat{\mu}_g\|_2 \leq \frac{\sum_{i\in V_e\cap S_b}(a_i\|X_i - \nabla f_i\|_2)}{g_e} \leq \frac{\epsilon(2 + \frac{1}{1-2\epsilon})\cdot\kappa}{1-2\epsilon} = \frac{3-4\epsilon}{(1-2\epsilon)^2}\epsilon\kappa,$$

where in the second step we invoke Lemma 5.4. Moreover, the following inequality also holds:

$$\|\hat{\mu}_g - \mu_g\|_2 \leq 2\epsilon\kappa.$$

Summing up these two sets of inequalities we have:

$$\|\hat{\mu} - \mu_g\|_2 \leq \left(2 + \frac{3-4\epsilon}{(1-2\epsilon)^2}\right)\epsilon\kappa.$$

□

7.2.5 Proof of Lemma 15

Proof. We only prove the first claim. The second claim is a natural consequence the first claim and Lemma 5.6.

To prove the first claim, we note that:

$$\|x_i^\tau[t] - \bar{x}[t]\|_2 \leq \left\| x_i^\tau[t] - \frac{1}{p}\sum_{j\in P} x_j^\tau[t] \right\|_2 \leq \frac{1}{p}\sum_{j\in P}\|x_i^\tau[t] - x_j^\tau[t]\|_2 \leq d^\tau[t]. \tag{7.36}$$

The rest of the proof comes from the following direct computation:

$$\begin{aligned}
& \|\nabla f_i(x_i^\tau[t]) - \nabla f_j(x_j^\tau[t])\|_2 \\
\leq\ & \|\nabla f_i(x_i^\tau[t]) - \nabla f_i(\bar{x}[t])\|_2 + \|\nabla f_i(\bar{x}[t]) - \nabla f_j(\bar{x}[t])\|_2 + \|\nabla f_j(x_j^\tau[t]) - \nabla f_j(\bar{x}[t])\|_2 \\
\overset{(a)}{\leq}\ & L_1\|x_i^\tau[t] - \bar{x}[t]\|_2 + \kappa + L_1\|x_j^\tau[t] - \bar{x}[t]\|_2 \\
\overset{(b)}{\leq}\ & 2L_1 d^\tau[t] + \kappa,
\end{aligned}$$

where in step (a) we use Assumption 10 (or Assumption 11) and in step (b) we plug in Equation (7.36). □

7.2.6 Proof of Proposition 6

Proof. We consider an arbitrary pair of nodes $i, j \in P$. By Lemma 15 the following two bounds hold:

$$\|\hat{\mu}_i[t] - \mu_i[t]\|_2 \leq (\kappa + 2L_1 d^\tau[t])c_\epsilon \leq (\kappa + 2L_1 r)c_\epsilon, \tag{7.37}$$

$$\|\hat{\mu}_j[t] - \mu_j[t]\|_2 \leq (\kappa + 2L_1 d^\tau[t])c_\epsilon \leq (\kappa + 2L_1 r)c_\epsilon. \tag{7.38}$$

In the proof of Lemma 15 we obtain that $\|\nabla f_i(x_i^\tau[t]) - \nabla f_j(x_j\tau[t])\|_2 \leq \kappa + 2L_1 r$ holds for any pair of nodes $i, j \in P$. This implies:

$$\|\mu_i[t] - \mu_j[t]\|_2 \leq \kappa + 2L_1 r, \tag{7.39}$$

since both $\mu_i[t]$ and $\mu_j[t]$ are weighted sums of local gradients. Combining all these facts, we have the following set of inequalities:

$$\begin{aligned}
& \|x_i^0[t+1] - x_j^0[t+1]\|_2 \\
= \; & \|(x_i^\tau[t] - \eta\hat{\mu}_i[t]) - (x_j^\tau[t] - \eta\hat{\mu}_j[t])\|_2 \\
\leq \; & \|x_i^\tau[t] - x_j^\tau[t]\|_2 + \eta\|\hat{\mu}_i[t] - \hat{\mu}_j[t]\|_2 \\
\leq \; & d^\tau[t] + \eta\|(\hat{\mu}_i[t] - \mu_i[t]) - (\hat{\mu}_j[t] - \mu_j[t]) + (\mu_i[t] - \mu_j[t])\|_2 \\
\leq \; & r + \eta\|\hat{\mu}_i[t] - \mu_i[t]\|_2 + \eta\|\hat{\mu}_j[t] - \mu_j[t]\|_2 + \eta\|\mu_j[t] - \mu_j[t]\|_2 \\
\leq \; & r + \eta(\kappa + 2L_1 r)(1 + 2c_\epsilon).
\end{aligned} \tag{7.40}$$

Since nodes i and j are picked arbitrarily, inequality (7.40) equivalently implies:

$$d^0[t+1] \leq r + \eta(\kappa + 2L_1 r)(1 + 2c_\epsilon). \tag{7.41}$$

Equation (7.41) suggests that after the execution of lines 9-12 in each iteration of the RAGD algorithm, the distance among the local parameters of a pair of nodes may increase, but will not increase dramatically, i.e., the distance is upper bounded by $(\kappa+2L_1 r)(1+2c_\epsilon)$. In order to make $d^\tau[t+1] \leq r$, we should mitigate the increase of distance by executing the inner loop for sufficiently many iterations. This is made possible by Lemma 4.1 which shows the existence of an $a > 0$ and a $\rho \in (0, 1)$ such that $d^\tau[t] \leq a\rho^\tau d^0[t]$. This provides a lower bound of τ_0 via the following analysis:

$$d^\tau[t+1] \leq a\rho^\tau d^0[t+1] \leq a\rho^\tau(r + \eta(\kappa + 2L_1 r)(1 + 2c_\epsilon)). \tag{7.42}$$

In order for $d^\tau[t+1] \leq r$, it suffices to pick τ_0 to satisfy:

$$\tau_0 \geq \log_{\frac{1}{\rho}} \frac{a(r + \eta(\kappa + 2L_1 r)(1 + 2c_\epsilon))}{r}. \tag{7.43}$$

□

7.2.7 Proof of Proposition 7

Proof. By the RAGD algorithm, each node j updates its local parameter according to:

$$x_j^0[t+1] = x_j^\tau[t] - \eta\hat{\mu}_j[t], \tag{7.44}$$

and then executes the linear iterative algorithm to reach consensus in the next iteration. Note that $\bar{x}[t] = \frac{1}{p}\sum_{j\in P} x_j^\tau[t]$ which was argued in Lemma 10. We have the following equalities:

$$\bar{x}[t+1] - \bar{x}[t] = \frac{1}{p}\sum_{j\in P}(x_j^0[t+1] - x_j^\tau[t]) = -\frac{\eta}{p}\sum_{j\in P}\hat{\mu}_j[t]. \tag{7.45}$$

Recall the definition of $\mu_j[t]$ in Lemma 15: $\mu_j[t] = \sum_{i\in\mathcal{N}_j^{in}} a_{ij}\nabla f_i(x^\tau[t])$, which is the weighted sum of gradients node j should receive in the absence of attacks. Summing the difference between μ_j and $\hat{\mu}_j$ over all nodes $j \in P$ in the network, we obtain:

$$\begin{aligned}
&\sum_{j\in P}\hat{\mu}_j[t] - \sum_{j\in P}\mu_j[t] \\
=\ &\sum_{j\in P}\hat{\mu}_j[t] - \sum_{j\in P}\sum_{i\in\mathcal{N}_j^{in}} a_{ij}\nabla f_i(x_i^\tau[t]) \\
=\ &\sum_{j\in P}\hat{\mu}_j[t] - \sum_{i\in P}\nabla f_i(x_i^\tau[t]) \\
=\ &\sum_{j\in P}\hat{\mu}_j[t] - \nabla f(\bar{x}[t]) - \sum_{i\in P}(\nabla f_i(x_i^\tau[t]) - \nabla f_i(\bar{x}[t])) \\
=\ &-\frac{p}{\eta}(\bar{x}[t+1] - \bar{x}[t]) - \nabla f(\bar{x}[t]) - \sum_{i\in P}(\nabla f_i(x_i^\tau[t]) - \nabla f_i(\bar{x}[t])),
\end{aligned} \tag{7.46}$$

where in the last step we used Equation (7.45). On the other hand, we have:

$$\left\|\sum_{j\in P}\hat{\mu}_j[t] - \sum_{j\in P}\mu_j[t]\right\|_2 \le \sum_{j\in P}\|\hat{\mu}_j[t] - \mu_j[t]\|_2 \le pc_\epsilon(\kappa + 2L_1 r), \tag{7.47}$$

from the triangular inequality as well as Equation (5.10). Meanwhile, it also holds that for any $j \in P$:

$$\|x_j^\tau[t] - \bar{x}[t]\|_2 = \|x_j^\tau[t] - \frac{1}{p}\sum_{i\in P} x_i^\tau[t]\|_2 \le \frac{1}{p}\sum_{i\in P}\|x_j^\tau[t] - x_i^\tau[t]\|_2 \le r, \tag{7.48}$$

which, using Assumption 10 (or Assumption 11), yields:

$$\left\| \sum_{i \in P} (\nabla f_i(x_i^\tau[t]) - \nabla f_i(\bar{x}[t])) \right\|_2 \leq pL_1 r. \tag{7.49}$$

The inequalities (7.46), (7.47), and (7.49) together imply:

$$\left\| \bar{x}[t+1] - \bar{x}[t] + \frac{\eta}{p} \nabla f(\bar{x}[t]) \right\|_2 \leq \eta c_\epsilon (\kappa + 2L_1 r) + \eta L_1 r, \tag{7.50}$$

which directly implies the claim in the proposition. □

7.2.8 Proof of Theorem 12

We prove the following lemmas which state that instead of performing accurate gradient descent, if we only have access to a gradient which is distance-bounded from the true one by at most a constant (which we call a roughly correct gradient), the minimal point of a strongly-convex function can be obtained up to some error. These results directly lead to Theorem 12.

Lemma 21 *Suppose function $f : \mathbb{R}^n \to \mathbb{R}$ satisfies Assumption 10. For any $x[t] \in \mathbb{R}^n$ updated according to:*

$$x[t+1] = x[t] - \frac{1}{L}(\nabla f(x[t]) - l[t]), \tag{7.51}$$

where $\|l[t]\|_2 \leq \xi$ for any $t \in \mathbb{N}$, the following inequality holds for any $t \in \mathbb{N}$ and $\beta = \sqrt{1 - \frac{\nu}{L}}$:

$$\|x[t] - x^*\|_2 \leq \beta^t \|x[0] - x^*\|_2 + \frac{\xi}{(1-\beta)L}. \tag{7.52}$$

Proof. By Assumption 10, the function f is both ν-strongly convex and L-smooth. This implies for any pair $x, y \in \mathbb{R}^n$, the following two inequalities hold:

$$f(y) - f(x) \leq \nabla f^T(x)(y - x) + \frac{L}{2}\|y - x\|_2^2, \tag{7.53}$$

$$f(x) - f(y) \geq \nabla f^T(y)(x - y) + \frac{\nu}{2}\|y - x\|_2^2. \tag{7.54}$$

A simple reorganization of Equation (7.54) yields:

$$f(y) - f(x) \leq \nabla f^T(y)(y-x) - \frac{\nu}{2}\|y-x\|_2^2. \tag{7.55}$$

We consider the following set of equalities and inequalities for any $x, y \in \mathbb{R}^n$ and $x^+ = x - \frac{1}{L}\nabla f(x)$, which will be used later.

$$\begin{aligned}
f(x^+) - f(y) &= f(x^+) - f(x) + f(x) - f(y) \\
&\leq \nabla f^T(x)(x^+ - x) + \frac{L}{2}\|x^+ - x\|_2^2 + \nabla f^T(x)(x-y) - \frac{\nu}{2}\|x-y\|_2^2 \\
&= \nabla f^T(x)(x^+ - y) + \frac{1}{2L}\|\nabla f(x)\|_2^2 - \frac{\nu}{2}\|x-y\|_2^2 \\
&= \nabla f^T(x)(x - \frac{1}{L}\nabla f(x) - y) + \frac{1}{2L}\|\nabla f(x)\|_2^2 - \frac{\nu}{2}\|x-y\|_2^2 \\
&= \nabla f(x)^T(x-y) - \frac{1}{2L}\|\nabla f(x)\|_2^2 - \frac{\nu}{2}\|x-y\|_2^2.
\end{aligned}$$

In particular, when $y = x^*$, we have:

$$0 \leq f(x^+) - f(x^*) \leq \nabla f^T(x)(x-x^*) - \frac{1}{2L}\|\nabla f(x)\|_2^2 - \frac{\nu}{2}\|x-x^*\|_2^2. \tag{7.56}$$

In the following we prove the result stated in the lemma.

$$\begin{aligned}
&\|x[t+1] - x^*\|_2 \\
= \;& \|x[t] - x^* - \frac{1}{L}(\nabla f(x[t]) - l[t])\|_2 \\
\leq \;& \|x[t] - x^* - \frac{1}{L}\nabla f(x[t])\|_2 + \frac{1}{L}\|l[t]\|_2 \\
\leq \;& \|x[t] - x^* - \frac{1}{L}\nabla f(x[t])\|_2 + \frac{\xi}{L} \\
= \;& \sqrt{\|x[t]-x^*\|_2^2 + \frac{1}{L^2}\|\nabla f(x[t])\|_2^2 - \frac{2}{L}\nabla f^T(x[t])(x[t]-x^*)} + \frac{\xi}{L} \\
\overset{(a)}{\leq} \;& \sqrt{(1-\frac{\nu}{L})\|x[t]-x^*\|_2^2 +} + \frac{\xi}{L} \\
\overset{(b)}{=} \;& \beta\|x[t]-x^*\|_2 + \frac{\xi}{L},
\end{aligned} \tag{7.57}$$

where in step (a) we plug in Equation (7.56) with $x = x[t]$ and in step (b) we use the definition $\beta = \sqrt{1-\frac{\nu}{L}}$. We note that $\beta \in (0,1)$. Solving Equation (7.57) recursively gives:

$$\|x[t]-x^*\|_2 \leq \beta^t\|x[0]-x^*\|_2 + \frac{\xi}{(1-\beta)L}. \tag{7.58}$$

□

Lemma 21 will be used in the proof of the first claim in Theorem 12. A similar result when the global function f satisfies the PL inequality instead of the convexity condition will also be provided in the following lemma.

Lemma 22 *Suppose a function $f : \mathbb{R}^n \to \mathbb{R}$ satisfies Assumption 11. For any $x[t] \in \mathbb{R}$ updated according to:*

$$x[t+1] = x[t] - \frac{1}{L}\nabla f(x[t]) + l[t], \tag{7.59}$$

where $\|l[t]\|_2 \leq \xi$ for any $t \in \mathbb{N}$. Let $\beta' = 1 - \frac{\mu}{L} \in (0,1)$, the following inequality holds for any $t \in \mathbb{N}$ and:

$$f(x[t]) - f(x^*) \leq \beta'^t(f(x[0]) - f(x^*)) + \frac{L\xi^2}{2(1-\beta')}, \tag{7.60}$$

where $f(x^*)$ is the minimum of the function f.

Proof. By L-smoothness of function f, we have the following inequality:

$$f(x[t+1]) \leq f(x[t]) + \nabla f^T(x[t])(x[t+1] - x[t]) + \frac{L}{2}\|x[t+1] - x[t]\|_2^2. \tag{7.61}$$

Combining with the update rule (7.59) yields:

$$\begin{aligned}
& f(x[t+1]) - f(x[t]) \leq \nabla f^T(x[t])(-\frac{1}{L}\nabla f(x[t]) + l[t]) + \frac{L}{2}\left\|-\frac{1}{L}\nabla f(x[t]) + l[t]\right\|_2^2 \\
\Rightarrow\ & f(x[t+1]) - f(x[t]) \leq -\frac{1}{2L}\nabla f^T(x[t])\nabla f(x[t]) + \frac{L}{2}l^T[t]l[t] \\
\Rightarrow\ & f(x[t+1]) - f(x[t]) \leq -\frac{1}{2L}\|\nabla f(x[t])\|_2^2 + \frac{L}{2}\xi^2 \\
\overset{(a)}{\Rightarrow}\ & f(x[t+1]) - f(x[t]) \leq -\frac{\mu}{L}(f(x[t]) - f(x^*)) + \frac{L}{2}\xi^2 \\
\Rightarrow\ & (f(x[t+1]) - f(x^*)) - (f(x[t]) - f(x^*)) \leq -\frac{\mu}{L}(f(x[t]) - f(x^*)) + \frac{L}{2}\xi^2 \\
\Rightarrow\ & (f(x[t+1]) - f(x^*)) \leq (1 - \frac{\mu}{L})(f(x[t]) - f(x^*)) + \frac{L}{2}\xi^2 \\
\Rightarrow\ & (f(x[t+1]) - f(x^*)) \leq \beta(f(x[t]) - f(x^*)) + \frac{L}{2}\xi^2
\end{aligned} \tag{7.62}$$

Similarly, by solving Equation (7.62) recursively we obtain:

$$f(x[t]) - f(x^*) \leq (\beta')^t (f(x[0]) - f(x^*)) + \frac{L\xi^2}{2(1-\beta')}. \tag{7.63}$$

□

Before giving our proof of Theorem 12, we need to state the following lemma, which will be used in the proof.

Lemma 23 ([KNS16]) *Let a function $f : \mathbb{R}^n \to \mathbb{R}$ satisfy the PL inequality with parameter μ. For any $x \in \mathbb{R}^n$, there always exist a minimizer x^* of f such that:*

$$f(x) - f(x^*) \geq \frac{\mu}{2} \|x - x^*\|_2^2. \tag{7.64}$$

Proof. The proof can be found in [KNS16]. □

The most important implication of Lemma 23 is that if a function $f : \mathbb{R}^n \to \mathbb{R}$ satisfies the PL inequality with parameter μ, then the following bound:

$$\frac{\mu}{2} \mathcal{D}^2(x, S^*) \leq f(x) - f(x^*), \tag{7.65}$$

holds, where S^* is the set of minimizers of f. This bound will be used in the proof of the main theorem.

Proof of Theorem 12. First let Assumption 10 hold. From Lemma 21, Equation (7.48), and Equation (5.11) in the main file, we observe that:

$$\begin{aligned} & \|x_j^\tau[t] - x^*\|_2 \\ \leq\ & \|x_j^\tau[t] - \bar{x}[t]\|_2 + \|\bar{x}[t] - x^*\|_2 \\ \leq\ & \beta^k \|\bar{x}[0] - x^*\|_2 + \frac{\xi}{(1-\beta)L} + r \\ =\ & \beta^k \|x_j^\tau[0] - x^*\|_2 + \frac{\xi}{(1-\beta)L} + r. \end{aligned} \tag{7.66}$$

Similarly, the following inequality can also be obtained from Lemma 22 and Equation (5.11) in the main file. Let Assumption 11 holds, for any $x^* \in S^*$, we have:

$$f(\bar{x}[t]) - f(x^*) \leq (\beta')^t (f(\bar{x}[0]) - f(x^*)) + \frac{L\xi^2}{2(1-\beta')}. \tag{7.67}$$

Combining Equation (7.67) with Equation (7.65), we obtain the following set of inequalities:

$$\begin{aligned}
\mathcal{D}(\bar{x}[t], S^*) &\leq \sqrt{\frac{2}{\mu}(\beta')^t (f(\bar{x}[0]) - f(x^*)) + \frac{L\xi^2}{\mu(1-\beta')}} \\
&\leq \sqrt{\frac{2}{\mu}(\beta')^t (f(\bar{x}[0]) - f(x^*))} + \sqrt{\frac{L\xi^2}{\mu(1-\beta')}} \\
&\leq \sqrt{\frac{L}{\mu}(\beta')^t \mathcal{D}^2(\bar{x}[0], S^*)} + \sqrt{\frac{L\xi^2}{\mu(1-\beta')}} \\
&= \sqrt{\frac{L}{\mu}} \left(\sqrt{\beta'}\right)^t \mathcal{D}(\bar{x}[0], S^*) + \sqrt{\frac{L\xi^2}{\mu(1-\beta')}}.
\end{aligned} \tag{7.68}$$

where in the third step we used L-smoothness of function f. In the end, we plug in inequality (7.48) into (7.68) and obtain the following inequality:

$$\mathcal{D}(\bar{x}_j^\tau[t], S^*) \leq \sqrt{\frac{L}{\mu}} \left(\sqrt{\beta'}\right)^t \mathcal{D}(x_j^0[0], S^*) + \sqrt{\frac{L\xi^2}{\mu(1-\beta')}} + r, \tag{7.69}$$

Plugging in the definition $\beta' = 1 - \frac{\mu}{L}$ into Equation (7.69) gives us:

$$\mathcal{D}(\bar{x}_j^\tau[t], S^*) \leq \sqrt{\frac{L}{\mu}} \left(\sqrt{\beta'}\right)^t \mathcal{D}(x_j^0[0], S^*) + \frac{L\xi}{\mu} + r. \tag{7.70}$$

□

REFERENCES

[ALS10] Saurabh Amin, Xavier Litrico, S Shankar Sastry, and Alexandre M Bayen. "Stealthy deception attacks on water SCADA systems." In *Proc. of the 13th ACM Int. Conference on Hybrid Systems: Computation and Control*, pp. 161–170, 2010.

[AM07] Panos J Antsaklis and Anthony N Michel. *Linear systems*. Springer Science & Business Media, 2007.

[AY17] Liwei An and Guang-Hong Yang. "Secure state estimation against sparse sensor attacks with adaptive switching mechanism." *IEEE Transactions on Automatic Control*, **63**(8):2596–2603, 2017.

[AY18] Liwei An and Guang-Hong Yang. "State estimation under sparse sensor attacks: A constrained set partitioning approach." *IEEE Transactions on Automatic Control*, **64**(9):3861–3868, 2018.

[BCL17] Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, and Jaehoon Amir Safavi. "Mitigating poisoning attacks on machine learning models: A data provenance based approach." In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 103–110, 2017.

[BFL10] He Bai, Randy A Freeman, and Kevin M Lynch. "Robust dynamic average consensus of time-varying inputs." In *49th IEEE Conference on Decision and Control (CDC)*, pp. 3104–3109. IEEE, 2010.

[BPC11] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[BPG17] Cheng-Zong Bai, Fabio Pasqualetti, and Vijay Gupta. "Data-injection attacks in stochastic control systems: Detectability and performance tradeoffs." *Automatica*, **82**:251 – 260, 2017.

[Bul19] Francesco Bullo. *Lectures on network systems*. Kindle Direct Publishing, 2019.

[CAS08] Alvaro A. Cárdenas, Saurabh Amin, and Shankar Sastry. "Research Challenges for the Security of Control Systems." In *Proceedings of the 3rd Conference on Hot Topics in Security*, HOTSEC'08, 2008.

[CFP15] László Czap, Christina Fragouli, Vinod M Prabhakaran, and Suhas Diggavi. "Secure network coding with erasures and feedback." *IEEE Transactions on Information Theory*, **61**(4):1667–1686, 2015.

[Che98] Chi-Tsong Chen. *Linear system theory and design*. Oxford University Press, Inc., 1998.

[CI14] Kai Cai and Hideaki Ishii. "Average consensus on arbitrary strongly connected digraphs with time-varying topologies." *IEEE Transactions on Automatic Control*, **59**(4):1066–1071, 2014.

[CKM18a] Yuan Chen, Soummya Kar, and Jose MF Moura. "Resilient distributed estimation through adversary detection." *IEEE Transactions on Signal Processing*, **66**(9):2455–2469, 2018.

[CKM18b] Yuan Chen, Soummya Kar, and José MF Moura. "Topology free resilient distributed estimation." *arXiv: 1812.08902*, 2018.

[CL99] Miguel Castro, Barbara Liskov, et al. "Practical byzantine fault tolerance." In *OSDI*, volume 99, pp. 173–186, 1999.

[CLR09] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[CLS04] Ben M Chen, Zongli Lin, and Yacov Shamash. *Linear Systems Theory*. 2004.

[CT05] Emmanuel J Candes and Terence Tao. "Decoding by linear programming." *IEEE Transactions on Information Theory*, **51**(12):4203–4215, 2005.

[CWH15] Michelle S Chong, Masashi Wakaiki, and Joao P Hespanha. "Observability of linear systems under adversarial attacks." In *2015 American Control Conference (ACC)*, pp. 2439–2444. IEEE, 2015.

[DD20] Deepesh Data and Suhas Diggavi. "Byzantine-resilient SGD in high dimensions on heterogeneous data." *arXiv preprint arXiv:2005.07866*, 2020.

[DD21] Deepesh Data and Suhas Diggavi. "Byzantine-resilient high-dimensional SGD with local iterations on heterogeneous data." In *International Conference on Machine Learning*, pp. 2478–2488. PMLR, 2021.

[DDZ15] Katherine R Davis, Charles M Davis, Saman A Zonouz, Rakesh B Bobba, Robin Berthier, Luis Garcia, and Peter W Sauer. "A cyber-physical modeling and assessment framework for power grid infrastructures." *IEEE Transactions on smart grid*, **6**(5):2464–2475, 2015.

[DHP04] Kevin Driscoll, Brendan Hall, Michael Paulitsch, Phil Zumsteg, and Hakan Sivencrona. "The real byzantine generals." In *The 23rd Digital Avionics Systems Conference (IEEE Cat. No. 04CH37576)*, volume 2, pp. 6–D. IEEE, 2004.

[DKK19] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. "Robust estimators in high-dimensions without the computational intractability." *SIAM Journal on Computing*, **48**(2):742–864, 2019.

[DP17] D. D'Auria and F. Persia. "A Collaborative Robotic Cyber Physical System for Surgery Applications." In *2017 IEEE International Conference on Information Reuse and Integration (IRI)*, pp. 79–83, Aug 2017.

[DUS19] Mohammad Deghat, Valery Ugrinovskii, Iman Shames, and Cedric Langbort. "Detection and mitigation of biasing attacks on distributed estimation networks." *Automatica*, **99**:369–381, 2019.

[EV08] Tuvi Etzion and Alexander Vardy. "Error-correcting codes in projective space." In *2008 IEEE International Symposium on Information Theory*, pp. 871–875. IEEE, 2008.

[FR11] James P Farwell and Rafal Rohozinski. "Stuxnet and the future of cyber war." *Survival*, **53**(1):23–40, 2011.

[FS07] Christina Fragouli and Emina Soljanin. "Network Coding Fundamentals." *Foundations and Trends® in Networking*, **2**(1):1–133, 2007.

[FTD11] Hamza Fawzi, Paulo Tabuada, and Suhas Diggavi. "Secure state-estimation for dynamical systems under active adversaries." In *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 337–344. IEEE, 2011.

[FTD14] H. Fawzi, P. Tabuada, and S. Diggavi. "Secure Estimation and Control for Cyber-Physical Systems Under Adversarial Attacks." *IEEE Transactions on Automatic Control*, **59**(6):1454–1467, 2014.

[GDV20] Nirupam Gupta, Thinh T Doan, and Nitin H Vaidya. "Byzantine Fault-Tolerance in Decentralized Optimization under Minimal Redundancy." *arXiv preprint arXiv:2009.14763*, 2020.

[GHY19] Avishek Ghosh, Justin Hong, Dong Yin, and Kannan Ramchandran. "Robust federated learning in a heterogeneous environment." *arXiv preprint arXiv:1906.06629*, 2019.

[GLB10] Abhishek Gupta, Cédric Langbort, and Tamer Başar. "Optimal control in the presence of an intelligent jammer with limited actions." In *Proc. of the 49th IEEE Conference on Decision and Control*, pp. 1096–1101, 2010.

[GMK21] Avishek Ghosh, Raj Kumar Maity, Swanand Kadhe, Arya Mazumdar, and Kannan Ramchandran. "Communication-Efficient and Byzantine-Robust Distributed Learning with Error Feedback." *IEEE Journal on Selected Areas in Information Theory*, 2021.

[GUC18] Jairo Giraldo, David Urbina, Alvaro Cardenas, Junia Valente, Mustafa Faisal, Justin Ruths, Nils Ole Tippenhauer, Henrik Sandberg, and Richard Candell. "A survey of physics-based attack detection in cyber-physical systems." *ACM Computing Surveys (CSUR)*, **51**(4):1–36, 2018.

[GV20] Nirupam Gupta and Nitin H Vaidya. "Resilience in collaborative optimization: redundant and independent cost functions." *arXiv preprint arXiv:2003.09675*, 2020.

[HKJ20] Lie He, Sai Praneeth Karimireddy, and Martin Jaggi. "Byzantine-robust learning on heterogeneous datasets via resampling." *arXiv preprint arXiv:2006.09365*, 2020.

[HMX19] Duo Han, Yilin Mo, and Lihua Xie. "Convex optimization based state estimation against sparse integrity attacks." *IEEE Transactions on Automatic Control*, **64**(6):2383–2395, 2019.

[HRS20] Xingkang He, X. Ren, H. Sandberg, and K. Johansson. "How to Secure Distributed Filters Under Sensor Attacks?" *ArXiv*, **abs/2004.05409**, 2020.

[HTW18] Weixin Han, Harry L Trentelman, Zhenhua Wang, and Yi Shen. "A simple approach to distributed observer design for linear systems." *IEEE Transactions on Automatic Control*, **64**(1):329–336, 2018.

[KCM15] Solmaz S Kia, Jorge Cortés, and Sonia Martinez. "Dynamic average consensus under limited control authority and privacy requirements." *International Journal of Robust and Nonlinear Control*, **25**(13):1941–1966, 2015.

[Kha95] Leonid Khachiyan. "On the complexity of approximating extremal determinants in matrices." *Journal of Complexity*, **11**(1):138–153, 1995.

[KK08] R. Koetter and F. R. Kschischang. "Coding for Errors and Erasures in Random Network Coding." *IEEE Transactions on Information Theory*, **54**(8):3579–3591, Aug 2008.

[KLS16] Junsoo Kim, Chanhwa Lee, Hyungbo Shim, Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. "Encrypting controller using fully homomorphic encryption for security of cyber-physical systems." *IFAC-PapersOnLine*, **49**(22):175–180, 2016.

[KMA19] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. "Advances and open problems in federated learning." *arXiv preprint arXiv:1912.04977*, 2019.

[KNS16] Hamed Karimi, Julie Nutini, and Mark Schmidt. "Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition." In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 795–811. Springer, 2016.

[KP22] Amir Khazraei and Miroslav Pajic. "Attack-resilient state estimation with intermittent data authentication." *Automatica*, **138**:110035, 2022.

[KSC16] Taekyoo Kim, Hyungbo Shim, and Dongil Dan Cho. "Distributed Luenberger observer design." In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 6928–6933. IEEE, 2016.

[KVC19] Solmaz S Kia, Bryan Van Scoy, Jorge Cortes, Randy A Freeman, Kevin M Lynch, and Sonia Martinez. "Tutorial on dynamic average consensus: The problem, its applications, and the algorithms." *IEEE Control Systems Magazine*, **39**(3):40–72, 2019.

[KXS20] Kananart Kuwaranancharoen, Lei Xin, and Shreyas Sundaram. "Byzantine-resilient distributed optimization of multi-dimensional functions." In *2020 American Control Conference (ACC)*, pp. 4399–4404. IEEE, 2020.

[Lan11] Ralph Langner. "Stuxnet: Dissecting a cyberwarfare weapon." *IEEE Security & Privacy*, **9**(3):49–51, 2011.

[Lau04] Alan J. Laub. *Matrix Analysis for Scientists and Engineers*. SIAM, 2004.

[LDJ18] T. Li, B. K. Dey, S. Jaggi, M. Langberg, and A. D. Sarwate. "Quadratically Constrained Channels with Causal Adversaries." In *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018.

[LKS20] J. G. Lee, J. Kim, and H. Shim. "Fully Distributed Resilient State Estimation Based on Distributed Median Solver." *IEEE Transactions on Automatic Control*, **65**(9):3935–3942, 2020.

[LS20] Jin Gyu Lee and Hyungbo Shim. "A tool for analysis and synthesis of heterogeneous multi-agent systems under rank-deficient coupling." *Automatica*, **117**:108952, 2020.

[LXC19] Liping Li, Wei Xu, Tianyi Chen, Georgios B Giannakis, and Qing Ling. "RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1544–1551, 2019.

[LY17] AnYang Lu and GuangHong Yang. "Secure state estimation for cyber-physical systems under sparse sensor attacks via a switched Luenberger observer." *Information sciences*, **417**:454–464, 2017.

[LZK13] Heath J LeBlanc, Haotian Zhang, Xenofon Koutsoukos, and Shreyas Sundaram. "Resilient asymptotic consensus in robust networks." *IEEE Journal on Selected Areas in Communications*, **31**(4):766–781, 2013.

[LZS13] Heath J LeBlanc, Haotian Zhang, Shreyas Sundaram, and Xenofon Koutsoukos. "Resilient continuous-time consensus in fractional robust networks." In *Proc. of the American Control Conference*, pp. 1237–1242. IEEE, 2013.

[MG16] Yilin Mo and Emanuele Garone. "Secure dynamic state estimation via local estimators." In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 5073–5078. IEEE, 2016.

[MLR16] Aryan Mokhtari, Qing Ling, and Alejandro Ribeiro. "Network Newton distributed optimization methods." *IEEE Transactions on Signal Processing*, **65**(1):146–161, 2016.

[MMS19] Yanwen Mao, Aritra Mitra, Shreyas Sundaram, and Paulo Tabuada. "When is the Secure State-Reconstruction Problem Hard?" In *Proc. of the 58th IEEE Conference on Decision and Control*, pp. 5368–5373. IEEE, 2019.

[MMS22] Yanwen Mao, Aritra Mitra, Shreyas Sundaram, and Paulo Tabuada. "On the computational complexity of the secure state-reconstruction problem." *Automatica*, **136**:110083, 2022.

[MNG18] Meng Ma, Athanasios N Nikolakopoulos, and Georgios B Giannakis. "Fast decentralized learning via hybrid consensus ADMM." In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3829–3833. IEEE, 2018.

[MRB19] Aritra Mitra, John A. Richards, Saurabh Bagchi, and Shreyas Sundaram. "Resilient distributed state estimation with mobile agents: overcoming Byzantine adversaries, communication losses, and intermittent measurements." *Autonomous Robots*, **43**(3):743–768, Mar 2019.

[MS16] Aritra Mitra and Shreyas Sundaram. "Secure distributed observers for a class of linear time invariant systems in the presence of Byzantine adversaries." In *Proc. of the 55th IEEE Conference on Decision and Control*, pp. 2709–2714, 2016.

[MS18] Aritra Mitra and Shreyas Sundaram. "Distributed observers for LTI systems." *IEEE Transactions on Automatic Control*, **63**(11):3689–3704, 2018.

[MS19] Aritra Mitra and Shreyas Sundaram. "Byzantine-resilient distributed observers for LTI systems." *Automatica*, **108**:108487, 2019.

[MSK16] Shaunak Mishra, Yasser Shoukry, Nikhil Karamchandani, Suhas N Diggavi, and Paulo Tabuada. "Secure state estimation against sensor attacks in the presence of noise." *IEEE Transactions on Control of Network Systems*, **4**(1):49–59, 2016.

[MSL16] Aryan Mokhtari, Wei Shi, Qing Ling, and Alejandro Ribeiro. "A decentralized second-order method with exact linear convergence rate for consensus optimization." *IEEE Transactions on Signal and Information Processing over Networks*, **2**(4):507–522, 2016.

[Nat95] Balas K. Natarajan. "Sparse approximate solutions to linear systems." *SIAM journal on computing*, **24**(2):227–234, 1995.

[NM15] Yorie Nakahira and Yilin Mo. "Dynamic state estimation in the presence of compromised sensory data." In *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 5808–5813. IEEE, 2015.

[NO09] Angelia Nedic and Asuman Ozdaglar. "Distributed subgradient methods for multi-agent optimization." *IEEE Transactions on Automatic Control*, **54**(1):48–61, 2009.

[PDB12] Fabio Pasqualetti, Florian Dörfler, and Francesco Bullo. "Cyber-physical security via geometric control: Distributed monitoring and malicious attacks." *Proc. of the 51st IEEE Conference on Decision and Control*, pp. 3418–3425, 2012.

[PDB13] Fabio Pasqualetti, Florian Dörfler, and Francesco Bullo. "Attack detection and identification in cyber-physical systems." *IEEE Transactions on Automatic Control*, **58**(11):2715–2729, 2013.

[PKP06] Joel B Predd, Sanjeev B Kulkarni, and H Vincent Poor. "Distributed learning in wireless sensor networks." *IEEE Signal Processing Magazine*, **23**(4):56–69, 2006.

[RSL18] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. "Certified defenses against adversarial examples." *arXiv preprint arXiv:1801.09344*, 2018.

[RWR20] Elan Rosenfeld, Ezra Winston, Pradeep Ravikumar, and Zico Kolter. "Certified robustness to label-flipping attacks via randomized smoothing." In *International Conference on Machine Learning*, pp. 8230–8241. PMLR, 2020.

[SCW18] Yasser Shoukry, Michelle Chong, Masashi Wakaiki, Pierluigi Nuzzo, Alberto Sangiovanni-Vincentelli, Sanjit A Seshia, Joao P Hespanha, and Paulo Tabuada. "SMT-based observer design for cyber-physical systems under sensor attacks." *ACM Transactions on Cyber-Physical Systems*, **2**(1):1–27, 2018.

[SG18] Shreyas Sundaram and Bahman Gharesifard. "Distributed optimization under adversarial nodes." *IEEE Transactions on Automatic Control*, **64**(3):1063–1076, 2018.

[SH10] Shreyas Sundaram and Christoforos N Hadjicostis. "Distributed function calculation via linear iterative strategies in the presence of malicious agents." *IEEE Transactions on Automatic Control*, **56**(7):1495–1508, 2010.

[Smi11] Roy S Smith. "A decoupled feedback structure for covertly appropriating networked control systems." *IFAC Proceedings Volumes*, **44**(1):90–95, 2011.

[Smi15] Roy S Smith. "Covert misappropriation of networked control systems: Presenting a feedback structure." *IEEE Control Systems Magazine*, **35**(1):82–92, 2015.

[SMT13] Yasser Shoukry, Paul Martin, Paulo Tabuada, and Mani Srivastava. "Non-invasive spoofing attacks for anti-lock braking systems." In *International Conference on Cryptographic Hardware and Embedded Systems*, pp. 55–72. Springer, 2013.

[SMY15] Yasser Shoukry, Paul Martin, Yair Yona, Suhas Diggavi, and Mani Srivastava. "Pycra: Physical challenge-response authentication for active sensors under spoofing attacks." In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1004–1015, 2015.

[SNS18] Yasser Shoukry, Pierluigi Nuzzo, Alberto L Sangiovanni-Vincentelli, Sanjit A Seshia, George J Pappas, and Paulo Tabuada. "Smc: Satisfiability Modulo Convex Programming." *Proc. of the IEEE*, **106**(9):1655–1679, 2018.

[SOM05] Demetri P Spanos, Reza Olfati-Saber, and Richard M Murray. "Dynamic consensus on mobile networks." In *IFAC world congress*, pp. 1–6. Citeseer, 2005.

[Son08] Eduardo D Sontag. "Input to state stability: Basic concepts and results." In *Nonlinear and optimal control theory*, pp. 163–220. Springer, 2008.

[SS19] Lili Su and Shahin Shahrampour. "Finite-time guarantees for byzantine-resilient distributed state estimation with noisy measurements." *IEEE Transactions on Automatic Control*, 2019.

[ST15] Yasser Shoukry and Paulo Tabuada. "Event-triggered state observers for sparse sensor noise/attacks." *IEEE Transactions on Automatic Control*, **61**(8):2079–2091, 2015.

[STJ10] Henrik Sandberg, André Teixeira, and Karl H Johansson. "On security indices for state estimators in power networks." In *First Workshop on Secure Control Systems (SCS), Stockholm, 2010*, 2010.

[SV16] Lili Su and Nitin H Vaidya. "Fault-tolerant multi-agent optimization: optimal iterative distributed algorithms." In *Proceedings of the 2016 ACM symposium on principles of distributed computing*, pp. 425–434, 2016.

[TDJ14] Ashish Tiwari, Bruno Dutertre, Dejan Jovanović, Thomas de Candia, Patrick D Lincoln, John Rushby, Dorsa Sadigh, and Sanjit Seshia. "Safety envelope for security." In *Proc. of the 3rd International Conference on High Confidence Networked Systems*, pp. 85–94. ACM, 2014.

[TV15] Lewis Tseng and Nitin H Vaidya. "Fault-tolerant consensus in directed graphs." In *Proc. of the 2015 ACM Symposium on Principles of Distributed Computing*, pp. 451–460, 2015.

[Win] "State Farm Falls Victim to Credential-Stuffing Attack." *https://threatpost.com/state-farm-credential-stuffing-attack/147139/*. Accessed: 2020-12-23.

[WM17] Lili Wang and A Stephen Morse. "A distributed observer for a time-invariant linear system." *IEEE Transactions on Automatic Control*, **63**(7):2123–2130, 2017.

[WOM19] Sean Weerakkody, Omur Ozel, Yilin Mo, and Bruno Sinopoli. "Resilient Control in Cyber-Physical Systems: Countering Uncertainty, Constraints, and Adversarial Behavior." *Foundations and Trends® in Systems and Control*, **7**(1-2):1–252, 2019.

[WRK20] Eric Wong, Leslie Rice, and J Zico Kolter. "Fast is better than free: Revisiting adversarial training." *arXiv preprint arXiv:2001.03994*, 2020.

[XBB15] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. "Is feature selection secure against training data poisoning?" In *international conference on machine learning*, pp. 1689–1698. PMLR, 2015.

[XBK07] Lin Xiao, Stephen Boyd, and Seung-Jean Kim. "Distributed average consensus with least-mean-square deviation." *Journal of parallel and distributed computing*, **67**(1):33–46, 2007.

[YB19a] Zhixiong Yang and Waheed U Bajwa. "BRIDGE: Byzantine-resilient decentralized gradient descent." *arXiv preprint arXiv:1908.08098*, 2019.

[YB19b] Zhixiong Yang and Waheed U Bajwa. "ByRDiE: Byzantine-resilient distributed coordinate descent for decentralized learning." *IEEE Transactions on Signal and Information Processing over Networks*, **5**(4):611–627, 2019.

[YC06] Raymond Yeung and Ning Cai. "Network Error Correction, I: Basic Concepts and Upper Bounds." *Communications in Information and Systems*, **6**, Jan 2006.

[YFF16] Sze Zheng Yong, Ming Qing Foo, and Emilio Frazzoli. "Robust and resilient estimation for cyber-physical systems under adversarial attacks." In *2016 American Control Conference (ACC)*, pp. 308–315. IEEE, 2016.