Decentralized Optimization Resilient Against Local Data Poisoning Attacks

Yanwen Mao, Deepesh Data, Suhas Diggavi, *Fellow, IEEE*, and Paulo Tabuada, *Fellow, IEEE*

***Abstract*—In this article, we study the problem of decentralized optimization in the presence of adversarial attacks. 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 compute 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 should agree on the said minimizer despite an adversary that can arbitrarily change the local functions of a fraction of the nodes. We present, the resilient averaging gradient descent (RAGD) algorithm, a decentralized, consensus+outlier filtering algorithm that is resilient to such 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. We verify the performance of the RAGD algorithm via numerical examples.**

***Index Terms*—Adversarial machine learning, consensus control, fault tolerant computer networks.**

I. Introduction

THIS article concerns a decentralized optimization problem that has seen several applications in the past decade [1], [2], including federated learning [3], [4]. In this problem, we have a set of nodes connected via a communication network, each equipped with a local function, which collectively compute 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 attack-free networks [5], [6] invalid. Therefore, it is of significant importance to develop learning algorithms that are robust to attacks.

A. Existing Work

Several papers have addressed the robust learning problem (or the robust optimization problem) in the distributed case, see [7], [8], [9], [10], [11], [12], [13], [14], and [15] 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 article, we do not assume the existence of such a central server, i.e., we consider a decentralized setting. In our problem formulation, each node should 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 datasets (or the local functions) across nodes are different.

The problem of global optimization in peer-to-peer networks without centralized coordination (i.e., decentralized setting) has been well-studied [16], [17], [18]. 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 minimizer x^* of the aggregated local functions (which is typically assumed to uniquely exist) and the set S_i of minimizers of each 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 dataset, or when the data samples at all nodes are drawn i.i.d. from the same statistical distribution. In this case, even without communication, nonadversarial 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 [19] and [20]. In [20], each node is asked to perform coordinatewise gradient descent based on a resiliently aggregated version of its received gradients. The solution in [19] is similar to [20] but nodes are asked to perform vanilla gradient descent instead. Both [19] and [20] 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 [21]: In this

Manuscript received 24 January 2023; revised 18 November 2023 and 3 June 2024; accepted 22 June 2024. Date of publication 8 July 2024; date of current version 31 December 2024. This work was supported in part by the Army Research Laboratory through Cooperative Agreement under Grant W911NF-17-2-0196 and in part by the NSF under Grant 2139304, Grant 2146838, and Grant 2007714. Recommended by Associate Editor G. Notarstefano. *(Corresponding author: Yanwen Mao.)*

The authors are with the Department of Electrical and Computer Engineering, University of California, Los Angeles, CA 90095 USA (e-mail: yanwen.mao@ucla.edu; deepesh.data@ucla.edu; suhasdiggavi@ucla.edu; tabuada@ucla.edu).

Digital Object Identifier 10.1109/TAC.2024.3424693

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 [21] and [22] fall into this class. The algorithm in [21] only applies to the special case when all nodes are connected via a complete network. In [22], it is shown that nodes are able to agree on the global optimal point given that any local cost function can be decomposed into a nontrivial weighted sum of univariate strictly convex function, which comes from a common size-limited set of basis functions. Their later work [23] extended this result to the multivariate case and dropped the weighted sum decomposition assumption made in [22].

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 $x^* \notin S_i$. The robust optimization problem studied in this class is known to be subject to some fundamental limitations [24], [25]. Su and Vaidya [26] were the first who studied this setting under the assumption that all local functions are univariate, 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 nonuniformly weighted sum of the local functions. Similar techniques have been adopted by Sundaram and Gharesifard [27] in their 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 [28] extended the result in [26] and [27] to the multivariate case. However, the fundamental limitation of SBG (or LF)-type algorithms is still present in [27], as the consensus point is only guaranteed to lie in the smallest hyper-rectangle that contains all the local minimum points. Recent work of He et al. [29] gave convergence results under the heterogeneous setting using a novel *ClippedGossip* approach, with a slightly different assumption on the power of the adversary. In addition, differently from the aforementioned works which attempted to find the minimizer of the aggregated local functions in the presence of adversaries, Mhamdi et al. [30] studied this problem from a different perspective by relating it to the averaging agreement problem. More recently, Kuwaranancharoen and Sundaram [31] provided a general algorithmic framework for the robust learning problem.

There is also a vast literature devoted to the learning problem when a fraction of data samples is under attack [32], [33], [34]. However, these works differ from our 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.

B. 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 possess 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 article, 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. Moreover, to limit the impact of the adversary, we assume the existence of some similarities among local functions (see Assumption 3), which has been shown necessary in [25] for this class of problem settings. To the best of the 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 article are as follows.

1) We propose 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-weight 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 nonconvex functions [i.e., satisfies the Polyak–Lojasiewicz (PL) inequality]. 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 1. Moreover, the proposed algorithm tolerates an attack up to half of the nodes.
3) We verify the theoretical results with a numerical example, where 20 nodes in a communication network, each equipped with a nonoverlapping portion of the MNIST dataset, collectively train a binary classification model, despite an adversary, which is able to alter the datasets at three nodes.

To the best of the authors' knowledge, our algorithm outperforms any existing solution, under the same or less stringent problem setting (e.g., a distributed setting). In Table I, we list six representative works on robust centralized/distributed optimization and compare such results with this article's contribution. Compared with the works mentioned in Table I, in this article we study the resilient optimization problem in a more sophisticated decentralized setting and provide an algorithm which is robust against an attack to, at most, half the agents. Estimates of the global optimizer at all agents are guaranteed to lie in a ball centered at the optimizer whose radius is proportional to $\sqrt{n}$

TABLE I
COMPARISON BETWEEN SOLUTIONS TO THE RDGO PROBLEM IN [9], [11], [7], [8], [10], AND [32], AND THIS ARTICLE

	[9], [11]	[7], [8]	[10]	[32]	Our algorithm
Network Topology	Distributed	Distributed	Distributed	Centralized	Decentralized
Maximum fraction of attacked agents	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
Dependence of error on dimension n	$O(n)$	$O(\sqrt{n})$	$O(n)$	$O(\sqrt{n})$	$O(\sqrt{n})$
Dependence of error on fraction of attacked nodes ϵ	$O(\epsilon)$	$O(\sqrt{\epsilon})$	$O(\epsilon)$	$O(\epsilon)$	$O(\epsilon)$

and ϵ, where n is the dimension and ϵ is the portion of attacked agents.

We also note that, any result in this article can be conveniently specialized to the distributed case, even when the attack is Byzantine. More details regarding this observation are provided in Remark 6.

C. Article Organization

The rest of this article is organized as follows. Section II formulates the RDGO problem. In Section III, we introduce resilient averaging gradient descent (RAGD) algorithm, which solves the RDGO problem. This is followed by Section IV 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 V. In Section VI, we validate our theoretical results via a numerical example. Finally, Section VII concludes this article.

II. PRELIMINARIES

A. 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 $A \in \mathbb{R}^{n\times p}$ are 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 $\mathcal{B}(x, r) = \{y \in \mathbb{R}^n | \|y - x\|_2 \le r\}$ 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 triplet 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 $\mathcal{N}_i^{\text{in}} = \{j \in \mathcal{V} | (j, i) \in \mathcal{E}\}$, 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 $\mathcal{N}_i^{\text{out}} = \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$. 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 entrywise. The entry in the ith row and jth column, a_{ij}, satisfies $0 < a_{ij} < 1$ if $(i, j) \in \mathcal{E}$ and otherwise $a_{ij} = 0$.

B. Problem Formulation

We consider a set S of p nodes connected via a communication network, modeled as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$. The set $\mathcal{V}$ 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 $\mathbf{A} \in \mathbb{R}^{p\times p}$.

We now formally define the RDGO problem.

Definition 1 (RDGO Problem): Consider a set S of p nodes connected via a communication network. Each node $i \in S$ 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 minimizer x^* of the aggregation of the local functions as follows:

$$f(x) = \sum_{i\in S} f_i(x)$$

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 $1/N \sum_{i=1}^{p} \sum_{j=1}^{N} l(\mathbf{w}, z_{ij})$ with respect to $\mathbf{w}$ and where l is some loss function. 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 article.

C. Attack Model

The solution to the global optimization problem is well-known in the absence of attacks [16], [17], [18], [35]. However, the problem becomes more challenging when some nodes are subject to attacks. In this article, we assume that a subset $S_b \subset S$ of nodes are subject to a data poisoning attack, which is able to replace the original function f_j of an attacked node $j \in S_b$ 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 article, we assume the attack is perpetrated before the nodes start executing the algorithm that solves the RDGO 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 [36], [37], [38], and [39]) where data poisoning attacks are studied.

Remark 1: It was argued in [26] that it is impossible to exactly recover the optimizer x^* when some local functions are attacked by an adversary and when there are no special relationships between the local functions. Therefore, instead of *exactly* recovering the optimizer x^*, we study in this article how well each node can *approximate* x^* using its possibly attacked local function and messages exchanged with its neighbors.

D. Assumptions

We study the RDGO problem under the following assumptions, some of which have already been discussed.

Assumption 1: 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 $i \in S$.

Assumption 2: There exists an $0 < \epsilon < 1/2$, 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.,

$$\sum_{i \in \mathcal{N}_j^{in} \cap S_b} a_{ij} \leq \epsilon \quad \forall j \in S. \tag{1}$$

Assumption 3: 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.,

$$\|\nabla f_i(x) - \nabla f_j(x)\|_2 \leq \kappa \quad \forall i, j \in S. \tag{2}$$

Assumption 1 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 [40], [41]. Note that the DAC problem is a special case of the RDGO problem,[2] which justifies our Assumption 1. Assumption 2 effectively restricts the power of the adversary. For example, if node j has k neighbors and the weight on each link to node j is $1/k$, Assumption 2 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 [27], with the slight difference that in [27] it is assumed that the number of attacked nodes in a neighborhood is upper-bounded. Lastly, Assumption 3 is shown to be necessary in [25].

Apart from Assumptions 1–3, in this article we also need one of the following two assumptions to solve the RDGO problem.

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

[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 $x^* = x_1 + x_2 + \ldots / + x_p p$.

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

```
1  Input: $\{a_{ij} | i \in \mathcal{N}_j^{in}\}$, $\tilde{f}_j$, $\epsilon$, $\eta$, $\tau \in \mathbb{N}$;
2  Initialization: $x_j^0[0] := 0$;
3  for $t = 0, 1, 2, \ldots$ do
4      for $k = 0, 1, 2, \ldots, \tau - 1$ do
5          Broadcast $x_j^k[t]$ ;
6          Receive $x_i^k[t]$ from $i \in \mathcal{N}_j^{in}$;
7          $x_j^{k+1}[t] := \sum_{i \in \mathcal{N}_j^{in}} a_{ij} x_i^k[t]$;
8      end
9      Compute and broadcast the gradient
         $X_j[t] := \nabla \tilde{f}_j(x_j^\tau[t])$ of its local function;
10     Receive $X_i[t]$ from $i \in \mathcal{N}_j^{in}$;
11     $\hat{\mu}_j[t] := \text{RWSE}(\{(a_{ij}, X_i[t]), i \in \mathcal{N}_j^{in}\}, \epsilon)$, (the
         RWSE algorithm is introduced in Section IV);
12     $x_j^0[t+1] := x_j^\tau[t] - \eta \hat{\mu}_j[t]$;
13 end
14 Output: $x_j^\tau[t]$;
```

Assumption 4: The global function f is L-smooth and ν-strongly convex, each local function f_i is L_1-smooth.

Assumption 5 is based on the following definition [42].

Definition 2: A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ satisfies the PL inequality with parameter $\mu \in \mathbb{R}^+$ if the following inequality holds:

$$\frac{1}{2}\|\nabla f(x)\|_2^2 \geq \mu(f(x) - f(x^*)) \tag{3}$$

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

Assumption 5: 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 $\mu < L$.

III. RAGD Algorithm

In this section, we introduce an algorithm called the RAGD, which enables all nodes to approximate the global minimum x^* and, thus, solves the RDGO problem. Algorithm 1 is a pseudocode description of the RAGD algorithm.

The RAGD algorithm has two loops, an inner loop (lines 4–8) and an outer loop (lines 3–12). 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.[3] The input parameter τ controls the number of iterations executed in the inner loop. The estimate at node j in tth iteration of the outer

[3]To avoid ambiguity, we emphasize that all nodes follow their prescribed rules (i.e., lines 4–8) in the inner loop. This setting is different from some other works (e.g., [43] and [44]) where malfunctioning nodes may deviate from the rules and send arbitrary messages to their neighbors.

loop and kth 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 1: Consider a set S of nodes, each starts with an initial value $x_i^0[t]$, and executes lines 4–8 of the RAGD algorithm in parallel. Define $\bar{x}[t] = \frac{1}{p}\sum_{i\in S} x_i^0[t]$ and $d^k[t] = \max_{i,j\in S}\|x_i^k[t] - x_j^k[t]\|_2$. 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) $\frac{1}{p}\sum_{i\in S} x_i^\tau[t] = \bar{x}[t] \quad \forall \tau \in \mathbb{N}$.
2) there exists an $a \in \mathbb{R}^+$ and a $\rho \in (0,1)$ such that for any $\tau \in \mathbb{N}$, $d^\tau[t] \le a\rho^\tau d^0[t]$.

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 (line 9). 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 robust weighted sum estimation (RWSE) algorithm], which allows each node to resiliently approximate the weighted average of the gradients it receives (lines 10 and 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).

Intuitively, and in contrast with the traditional Byzantine setting [29], [30], the fact that attacked agents only corrupt the local functions (and then behave correctly throughout) plays a key role in the algorithm. This weaker Byzantine model simplifies the problem of consensus of their local estimates.

Remark 2: 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 V.

IV. RWSE Algorithm

In this section, we study the problem of how each node can resiliently compute the weighted sum of its neighbors' gradients under Assumptions 1–3, despite a portion of the gradients having been attacked. To solve this problem, we propose a novel algorithm termed the 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 optimization problem,[4] which will be argued in Remark 6. Moreover, we note that RWSE problem is a generalization of the well-known robust mean estimation (RME) problem,[5] hence, RWSE solves the RME problem assuming Assumptions 1–3 hold.

[4]See [16] for a formal definition of the distributed Byzantine-resilient optimization problem.

[5]See [45] for a formal definition of the robust mean estimation problem.

Algorithm 2: Robust Weighted Sum Estimation (RWSE).

```
1  Input: $\{(a_i, X_i)|i \in S\}, \epsilon$; Initialization: $g := 1$,
     $temp := 0$, $w_0 := 0$, $V := \{X_1, X_2, \ldots, X_p\}$,
     $V_e := \{\}$, $g_e := 0$;
2  while $g > 1 - \epsilon$ do
3      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;
4      $s_i := \sum_{z\in S}(a_z\|X_i - X_z\|_2)$;
5      $s_h := \sum_{z\in S}(a_z\|X_h - X_z\|_2)$;
6      if $s_i > s_h$ then
7          $u := i$;
8      else
9          $u := h$;
10     end
11     $V := V\backslash\{X_u\}$, $g := g - a_u$, $temp := u$;
12     if $a_{temp} > \epsilon$ then
13         $V_e =: V \cup \{X_{temp}\}$, $g_e := g + a_{temp}$;
14     else
15         $V_e := V$, $g_e := g$;
16     end
17 end
18 Output: $\hat{\mu} := \frac{1}{g_e}\sum_{X_i\in V_e} a_i X_i$;
```

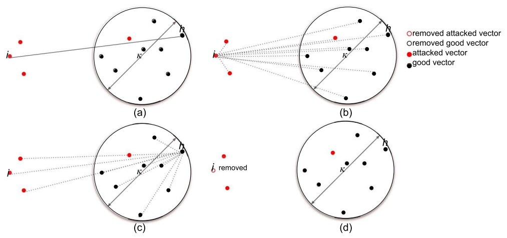


Fig. 1. Visualization of Algorithm 2.

A. Algorithm Description

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

$$\begin{cases} X_i = & \nabla f_i, \quad i \in S_g \\ X_i \neq & \nabla f_i, \quad i \in S_b \end{cases} \tag{4}$$

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 2 and 3. Algorithm 2 is a pseudocode description of the RWSE algorithm.

To explain the RWSE algorithm we describe its execution on the example in Fig. 1. In Fig. 1, a red dot denotes an attacked

TABLE II
ILLUSTRATION OF THE CLASSIFICATION RULE

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

An execution of the RWSE algorithm falls into a certain case if for sufficient number of iterations (as shown in the first column), an indicated pair of vectors (as shown in the second column) is chosen since their distance is larger than any other pair. The third column indicates if a good or attacked vector is removed. 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 2.

vector, whereas a black dot denotes an attack-free vector. All attack-free vectors (black dots) lie in a ball with diameter κ whereas there are no restrictions on the position of attacked vectors. In each iteration, the execution node first finds a pair of vectors (i, h) with the maximum Euclidean distance [see subgraph (a), also lines 4 and 5 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 [see subgraph (b) and (c), also lines 7–10 in Algorithm 2). In the problem instance represented by Fig. 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 line 11 in Algorithm 2.

In addition, we use a scalar variable $temp$ to store the identity of the latest removed vector. By Assumption 2 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 13 or line 15 in Algorithm 2.

B. 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 II, where o stands for a good vector and a stands for an attacked vector. For example, the third row in Table I tells that if in some iteration the maximum distance lies between two good vectors and, hence, a good vector is removed in this iteration, 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 2: If Case 1 or Case 2 holds during the execution of the RWSE algorithm, then the output $\hat{\mu}$ of the RWSE algorithm satisfies the following:

$$\|\hat{\mu} - \mu_g\|_2 \leq \left(2 + \frac{2}{1 - 2\epsilon}\right)\epsilon\kappa. \tag{5}$$

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 3: 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 4: If Case 3 holds during the execution of the RWSE algorithm, then the output $\hat{\mu}$ of the RWSE algorithm satisfies the following:

$$\|\hat{\mu} - \mu_g\|_2 \leq \left(2 + \frac{3 - 4\epsilon}{(1 - 2\epsilon)^2}\right)\epsilon\kappa. \tag{6}$$

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

Lemma 5: Consider the RWSE algorithm with inputs $\{(a_i, X_i) | i \in S\}$ satisfying 1) $a_1 + a_2 + \cdots + a_p = 1$, 2) $a_i > 0$~$\forall i \in S$, 3) $\|\nabla f_i - \nabla f_h\|_2 \leq \kappa$~$\forall i, h \in S$, 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 the following:

$$\|\hat{\mu} - \mu_g\|_2 \leq \left(2 + \frac{3 - 4\epsilon}{(1 - 2\epsilon)^2}\right)\epsilon\kappa. \tag{7}$$

Remark 3: We see from its 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 5, the error of the RWSE algorithm scales with ϵ, which outperforms many RME algorithms whose error scales with $\sqrt{\epsilon}$ [46]. The authors believe this is a consequence of Assumption 3 requiring the dissimilarity among two good gradients to be upper bounded, whereas in RME algorithms usually deal with random data. It is also noteworthy that the error of the RWSE algorithm scales with κ, which implicitly grows with $\sqrt{n}$.

Remark 4: Differently from the trimmed mean algorithm [47], we note that the RWSE algorithm is sensitive to the portion ϵ of attacked nodes in the sense that the guarantee becomes very loose when ϵ approaches $1/2$. The computational complexity of the RWSE algorithm is proportional to p^2.

V. 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 1. Executing lines 9–12 in Algorithm 1 brings the following two consequences:

1) The average of local estimates moves toward 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 toward 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 $d^k[t] = \max_{i,j\in S} \|x_i^k[t] - x_j^k[t]\|_2$ from Lemma 1 and proceed with our first result in this section.

Lemma 6: Consider a set S of nodes in a communication network satisfying Assumption 1. Each node has a local function which satisfies Assumption 3 whereas some local functions are altered by a data poisoning attack which satisfies Assumption 2. Let all nodes run the RAGD algorithm in parallel. Moreover, let either Assumption 4 or 5 hold. For any pair of nodes $i, j \in S$, 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 the following:

$$\|\nabla f_i(x_i^\tau[t]) - \nabla f_j(x_j^\tau[t])\|_2 \le \kappa + 2L_1\, d^\tau[t]. \tag{8}$$

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

$$\|\hat{\mu}_j[t] - \mu_j[t]\|_2 \le c_\epsilon(\kappa + 2L_1\, d^\tau[t]) \tag{9}$$

for any $j \in S$ 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 $c_\epsilon = 3\epsilon - 4\epsilon^2/(1-2\epsilon)^2$.

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

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

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., $t = 0$), the inequality $d^\tau[t] \le r$ holds for any $t \in \mathbb{N}$ provided $\tau \ge \tau_0$ holds. In brief, the choice of τ_0 is dependent on the smoothness of the global function, the convergence rate, and the dissimilarity among local functions. A detailed expression can be found in (42).

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

$$\|\hat{\mu}_j[t] - \mu_j[t]\|_2 \le c_\epsilon(\kappa + 2L_1 r). \tag{10}$$

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

Proposition 2: Under the assumptions of Lemma 6, for any $t \in \mathbb{N}$, $\eta > 0$, $r > 0$, assume $d^\tau[t] \le r$ holds for any $\tau \in \mathbb{N}$, then the following equation holds:

$$\bar{x}[t+1] - \bar{x}[t] = -\frac{\eta}{p}\nabla f(\bar{x}[t]) + \frac{\eta}{p} l[t] \tag{11}$$

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

For simplicity, we define $\xi = pc_\epsilon(\kappa + 2L_1 r) + pL_1 r$. Inequality (11) shows that, if we compare the average at 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 η/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 article.

Theorem 1: Consider a set of nodes in a communication network satisfying Assumption 1, each equipped with a local function satisfying Assumption 3. Moreover, assume a subset of nodes is subject to a data poisoning attack satisfying Assumption 2. Suppose all nodes in the network run the RAGD algorithm with $\eta = \frac{p}{L}$ and parameter $\tau \ge \tau_0$ with τ_0 defined in (42) with respect to any given $r > 0$, then the output of every node $j \in S$ satisfies one of the following two possibilities for any $t \in \mathbb{N}$:

1) Let $\beta = \sqrt{1 - \frac{\nu}{L}}$. Assumption 4 implies the following inequality:

$$\|x_j^\tau[t] - x^*\|_2 \le \beta^t \|x_j^0[0] - x^*\|_2 + \frac{\xi}{(1-\beta)L} + r. \tag{12}$$

2) Let $\beta' = \sqrt{1 - \frac{\mu}{L}}$ and S^* be the set of minimizers of f. Assumption 5 implies the following inequality:

$$\mathcal{D}(x_j^\tau[t], S^*) \le \sqrt{\frac{L}{\mu}}\beta'^t \mathcal{D}\left(x_j^0[0], S^*\right) + \frac{\xi}{\mu} + r. \tag{13}$$

Remark 5: It can also be seen from Theorem 1 that the output $x_j^\tau[t]$ at node j will converge to the ball $\mathcal{B}(x^*, \xi/(1-\beta)L + r)$, if Assumption 4 holds. Moreover, if $x_j^0[0]$ is not in the ball mentioned above, then the estimate $x_j^\tau[t]$ will move toward the ball. From this discussion we also learn that for any $t \in \mathbb{N}$, $x_j^\tau[t]$ always lies in the ball $\mathcal{B}(x^*, \max(\|x^*\|_2, \xi/(1-\beta)L + r))$ since all $x_j^0[0]$ are assumed to be 0. The result in Theorem 1 implicitly assumes that the working domain contains the ball mentioned above. A similar analysis also applies to the case when Assumption 5 holds.

Remark 6: 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 from some existing works [27], [28] in which the computed minimizer is only guaranteed to lie in the smallest hyper-rectangle that contains all local minimizers.[6] Moreover, our error term scales linearly with $\sqrt{n}$ and ϵ when

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

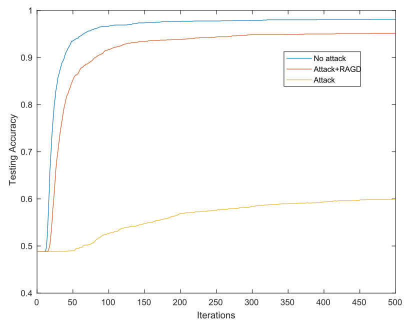


Fig. 2. Testing accuracy over iterations of the RAGD algorithm for decentralized training a logistic regression function on MNIST dataset in the presence of attacks changing gradients.

at most half of nodes are under attack, which matches, or even outperforms its counterpart in the distributed case [7], [9], [48]. We also note that in the absence of attacks, i.e., when $\epsilon = 0$, the RAGD algorithm specializes to the well-known decentralized gradient descent algorithm. Lastly, it is observed that, since the RWSE algorithm can always find a reasonably good estimate of the average of attack-free gradients regardless of the values of the attacked gradients, the RAGD algorithm can be applied to solve the distributed resilient federated learning (respectively, optimization) problem, even when the adversary is Byzantine.

VI. 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 dataset. 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. We note that the logistic regression function satisfies Assumption 4.

In this test, we use 12 000 samples from the MNIST dataset, 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 10 000 and a testing set of size 2000. Moreover, the 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 to be 10. The step size is set to be 1×10^{-6}.

Fig. 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 the attacks. As we can see in Fig. 2, there is a first phase of about 50 iterations where the testing accuracy grows exponentially. This is explained by the bound (12) where the first term on the right-hand side dominates the other two terms. After about 400 iterations, the testing accuracy stabilizes around 95%. At this point the effect of the first term has approximately vanished and we observe the effect of the last two terms that determine the worst case gap between the estimated minimizer and the true minimizer.

VII. Conclusion

In this article, we proposed a RAGD 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.

Appendix

A. Proof of Lemma 1 in Section III

We first introduce the following two results which will be used in the proof of Lemma 1.

Lemma 7: ([49]) Consider a schur stable[7] 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:

$$\|F^n\|_2 \leq m\rho^n. \tag{14}$$

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 follows:

$$J_i^n = \begin{bmatrix} \lambda_i^n & \binom{n}{1}\lambda_i^{n-1} & \binom{n}{2}\lambda_i^{n-2} & \ldots & \binom{n}{s_i-1}\lambda_i^{n-s_i+1} \\ 0 & \lambda_i^n & \binom{n}{1}\lambda_i^{n-1} & \ldots & \binom{n}{s_i-2}\lambda_i^{n-s_i+2} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_i^n & \binom{n}{1}\lambda_i^{n-1} \\ 0 & 0 & \ldots & 0 & \lambda_i^n \end{bmatrix} \tag{15}$$

from which we have the following:

$$\|J_i^n\|_2 \leq \sqrt{n}\|J_i^n\|_1 = \sum_{j=0}^{s_i-1}\binom{n}{j}|\lambda_i|^{n-j}. \tag{16}$$

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

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

enough, since

$$\begin{aligned}\frac{U_i(n+1)}{U_i(n)} &= \frac{\sum_{j=0}^{s_i-1}\binom{n+1}{j}|\lambda_i|^{n+1-j}}{\sum_{j=0}^{s_i-1}\binom{n}{j}|\lambda_i|^{n-j}}\\ &= \frac{\sum_{j=0}^{s_i-1}\frac{(n+1)!}{j!(n+1-j)!}|\lambda_i|^{n+1-j}}{\sum_{j=0}^{s_i-1}\frac{n!}{j!(n-j)!}|\lambda_i|^{n-j}}\\ &\le |\lambda_i| \max_{j\in\{0,1,\dots,s_i-1\}} \frac{\frac{(n+1)!}{j!(n+1-j)!}}{\frac{n!}{j!(n-j)!}}\\ &= |\lambda_i| \max_{j\in\{0,1,\dots,s_i-1\}} \frac{n+1}{n+1-j}\\ &= |\lambda_i|\cdot\frac{n+1}{n+2-s_i}. \end{aligned} \tag{17}$$

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 $n \ge N_0$, $U_i(n+1)/U_i(n) < 1$. By picking $m_i = \max\{1, U_i(N_0)\}$ and $\rho_i = |\lambda_i| \cdot N_0 + 1/N_0 + 2 - s_i$ we finish the proof. ■

Lemma 8: 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 8 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 ith 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 Lemmas 7 and 8 we provide the proof of Lemma 1 in the main file.

Proof of Lemma 1:([50]) To start with, we define $x^k[t] = \left[(x_1^k[t])^T, (x_2^k[t])^T, \dots, (x_p^k[t])^T\right]^T$ for any $k = 0, 1, \dots, \tau$ and $t \in \mathbb{N}$. This definition allows us to write the mathematical representation of the linear iterative algorithm as the following:

$$x^{k+1}[t] = (\mathbf{A}^T \otimes I_n)x^k[t]. \tag{18}$$

To proceed, we find an orthogonal matrix $[R \quad S] \in \mathbb{R}^{p\times p}$ where $R = 1/\sqrt{p}\mathbf{1}_p$ 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 $\{x_i^k[t],\ i \in S\}$, i.e., there exists a set of weights $\{w_1, w_2, \dots, w_p\}$ such that each block in $z_2^k[t]$ can be written as $\sum_{i\in S} w_i x_i^k[t]$. Moreover, by construction of the matrix S, we have $\sum_{i\in S} w_i = 0$. 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 $\{x_i^k[t] - x_j^k[t],\ i, j \in S\}$ 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:

1) Fact 1: $\sum_{i\in S, w_i>0} w_i \le \sqrt{\frac{p}{2}}$, which is obtained by combining facts $\sum_{i\in S} w_i = 0$ and $\sum_{i\in S} w_i^2 = 1$ with the Cauchy–Schwardz inequality;
2) Fact 2: $\sum_{i,j\in S} w'_{ij} = \sum_{i\in S, w_i>0} w_i \le \sqrt{\frac{p}{2}}$.

With the understanding of the change of coordinates, we obtain the following set of equalities starting from (18):

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

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 $[R \quad S]$ is an orthogonal matrix. Moreover, step (c) is true since by Assumption 1 the adjacency matrix of the communication graph $\mathbf{A}$ is doubly-stochastic, and so is its transpose $\mathbf{A}^T$, which implies $\mathbf{A}^TR = R$ as well as $R^T\mathbf{A}^T = R^T$. Lastly, in step (d) we again invoke the orthogonality of the matrix $[R \quad S]$ and the definition of the vector R. From these set of equalities we observe that (18) can be decoupled as follows:

$$\begin{cases} z_1^{k+1}[t] &= z_1^k[t]\\ z_2^{k+1}[t] &= (\mathbf{A}'\otimes I_n)z_2^k[t] \end{cases} \tag{19}$$

where $\mathbf{A}'$ is defined as $\mathbf{A}' = S^T\mathbf{A}^T S$. We note that (19) can be easily generated as follows:

$$\begin{cases} z_1^\tau[t] & = z_1^0[t] \\ z_2^\tau[t] & = (\mathbf{A}' \otimes I_n) z_2^0[t] \end{cases}. \tag{20}$$

Recall the definition $z_1^k[t] = (R^T \otimes I_n)x^k[t] = 1/\sqrt{p}\sum_{i\in S} x_i^0[t] = \sqrt{p}\bar{x}[t]$, 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 ith entry being 1 and its jth entry being -1. This definition allows the following chain:

$$\begin{aligned} & x_i^k[t] - x_j^k[t] \\ & = (B_{ij} \otimes I_n)x^k[t] \\ & = (B_{ij} \otimes I_n)(S \otimes I_n)(S^T \otimes I_n)x^k[t] \\ & = ((B_{ij}S) \otimes I_n)z_2^k[t] \end{aligned} \tag{21}$$

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:

$$\begin{aligned} d^\tau[t] &= \max_{i,j\in S} \|x_i^\tau[t] - x_j^\tau[t]\|_2 \\ &\overset{(a)}{=} \max_{i,j\in S} \|((B_{ij}S)\otimes I_n)z_2^\tau[t]\|_2 \\ &\overset{(b)}{\leq} \max_{i,j\in S} \|((B_{ij}S)\otimes I_n)\|_2 \cdot \|(\mathbf{A}'\otimes I_n)^\tau z_2^0[t]\|_2 \\ &\leq \sqrt{2}\sigma_M(S)\|\mathbf{A}'\otimes I_n\|_2^\tau \cdot \|z_2^0[t]\|_2 \\ &\overset{(c)}{\leq} \sqrt{2}\sigma_M(S) m\rho^\tau \|z_2^0[t]\|_2 \\ &\overset{(d)}{\leq} \sqrt{2}\sigma_M(S) m\rho^\tau \sqrt{p-1}\sqrt{\frac{p}{2}} \max_{i,j\in S}\|x_i^0[t] - x_j^0[t]\|_2 \\ &< \underbrace{p\sigma_M(S)}_{a} m\rho^\tau d^0[t] \end{aligned} \tag{22}$$

where in step (a) and step (b) we plug in (21) and (20), respectively. In step (c) we invoke Lemma 7 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 $\{x_i^0[t] - x_j^0[t],\, i,j \in S\}$, with the non-negative weight values $\{w'_{ij},\, i,j\in S\}$ sum up to at most $\sqrt{p/2}$. This observation leads to the following argument:

$$\begin{aligned} & \left\| \sum_{i,j\in S} w'_{ij}\left(x_i^0[t] - x_j^0[t]\right)\right\|_2 \\ & \leq \sum_{i,j\in S} w'_{ij}\|x_i^0[t] - x_j^0[t]\|_2 \\ & \leq \sum_{i,j\in S} w'_{ij} \max_{i,j\in S}\|x_i^0[t] - x_j^0[t]\|_2 \\ & \leq \sqrt{\frac{p}{2}} \max_{i,j\in S}\|x_i^0[t] - x_j^0[t]\|_2 \end{aligned} \tag{23}$$

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

$$\|z_2^0[t]\|_2 \leq \sqrt{p-1}\sqrt{\frac{p}{2}} \max_{i,j\in S}\|x_i^0[t] - x_j^0[t]\|_2. \tag{24}$$

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

B. Proof of Lemma 2 in Section IV

Proof: We prove the lemma by taking two steps. We first compare $\hat{\mu} = 1/g_e\sum_{i\in V_e} a_i X_i$ with $\hat{\mu}_g = 1/g_e\sum_{i\in V_e} a_i\nabla f_i$. 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 the following:

$$\|\hat{\mu} - \hat{\mu}_g\|_2 \leq \frac{\sum_{i\in V_e\cap S_b}(a_i\|X_i - \nabla f_i\|_2)}{g_e} \leq \frac{2\epsilon}{1-2\epsilon}\kappa. \tag{25}$$

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, whereas μ_g is the average of all good vectors, and 2) by Assumption 3 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:

$$\|\nabla f_i - \hat{\mu}_g\|_2 \leq \frac{\sum_{j\in V_e}(a_j\|\nabla f_i - \nabla f_j\|_2)}{g_e} \leq \kappa. \tag{26}$$

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

$$\begin{aligned} & \|\mu_g - \hat{\mu}_g\|_2 \\ & = \left\|\sum_{i\in S} a_i\nabla f_i - \hat{\mu}_g\right\|_2 \\ & = \left\|\sum_{i\in V_e} a_i\nabla f_i + \sum_{i\in S\setminus V_e} a_i\nabla f_i - (g_e\hat{\mu}_g + (1-g_e)\hat{\mu}_g)\right\|_2 \end{aligned}$$

$$\overset{(a)}{=} \Bigg\| \sum_{i\in V_e} a_i \nabla f_i + \sum_{i\in S\setminus V_e} a_i \nabla f_i - \Big(\sum_{i\in V_e} a_i \nabla f_i + \sum_{i\in S\setminus V_e} a_i \hat{\mu}_g\Big)\Bigg\|_2$$
$$\leq \sum_{i\in S\setminus V_e} (a_i \|\nabla f_i - \hat{\mu}_g\|_2)$$
$$\overset{(b)}{\leq} 2\epsilon\kappa$$

where in step (a) we use the equalities $\hat{\mu}_g = 1/g_e \sum_{i\in V_e} a_i \nabla f_i$ and $\sum_{i\in S\setminus V_e} a_i = 1 - g_e$, and in step (b) we use (26). Summing up (25) and (27) as follows:

$$\|\hat{\mu} - \mu_g\|_2 \leq \|\hat{\mu} - \hat{\mu}_g\|_2 + \|\hat{\mu}_g - \mu_g\|_2 \leq 2\epsilon\kappa + \frac{2\epsilon\kappa}{1-2\epsilon}$$

we obtain the proof of the lemma. ■

C. Proof of Lemma 3

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 as follows:

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

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

Before proving the claim, we first see what it implies. If $\|X_i - X_l\|_2 \leq 1/1 - 2\epsilon\kappa$ holds, and by Assumption 3 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 the following:

$$\|X_i - X_h\|_2 \leq \|X_i - X_l\|_2 + \|X_h - X_l\|_2 \leq \left(1 + \frac{1}{1-2\epsilon}\right)\kappa. \tag{27}$$

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

$$\|X_q - X_h\|_2 \leq \|X_i - X_h\|_2 \leq \left(1 + \frac{1}{1-2\epsilon}\right)\kappa \tag{28}$$

since otherwise Algorithm 2 would not have picked the pair X_i and X_h. Again, we invoke Assumption 3 and obtain that $\|X_h - \nabla f_q\|_2 = \|\nabla f_h - \nabla f_q\|_2 \leq \kappa$. Summing these inequalities up we have the following:

$$\|X_q - \nabla f_q\|_2 \leq \|X_q - X_h\|_2 + \|X_h - \nabla f_q\|_2 \leq \left(2 + \frac{1}{1-2\epsilon}\right)\kappa \tag{29}$$

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 $d \geq 1/1 - 2\epsilon\kappa$. We consider the following set of inequalities:

$$\begin{aligned}
&\sum_{z\in V} a_z(\|X_i - X_z\|_2 - \|X_h - X_z\|_2) \\
&= \sum_{z\in V\cap S_g} a_z(\|X_i - X_z\|_2 - \|X_h - X_z\|_2) + \sum_{z\in V\cap S_b} a_z(\|X_i - X_z\|_2 - \|X_h - X_z\|_2) \\
&\geq \sum_{z\in V\cap S_g} a_z(\|X_i - X_z\|_2 - \|X_h - X_z\|_2) + \left(\sum_{z\in V\cap S_b} a_z\right)(-\|X_i - X_h\|_2) \\
&\overset{(a)}{\geq} \sum_{z\in V\cap S_g} a_z(\|X_i - X_f\|_2 - \|X_h - X_z\|_2) + \left(\sum_{z\in V\cap S_b} a_z\right)(-\|X_i - X_f\|_2 - \|X_f - X_h\|_2) \\
&= \left(\sum_{z\in V\cap S_g} a_z - \sum_{z\in V\cap S_b} a_z\right)\|X_i - X_f\|_2 - \sum_{z\in V\cap S_g} a_z\|X_h - X_z\|_2 - \sum_{z\in V\cap S_b} a_z\|X_f - X_h\|_2 \\
&\overset{(b)}{>} \left(\sum_{z\in V\cap S_g} a_z - \sum_{z\in V\cap S_b} a_z\right) d - \sum_{z\in V\cap S_g} a_z\kappa - \sum_{z\in V\cap S_b} a_z\kappa \\
&\geq (1-2\epsilon)d - \kappa \\
&\overset{(c)}{\geq} 0
\end{aligned} \tag{30}$$

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 $d \geq 1/1 - 2\epsilon\kappa$. From this set of inequalities we reach a contradiction since by assumption the good vector X_i was removed whereas X_h should have been removed. ■

D. Proof of Lemma 4 in Section IV

Proof: Given Lemma 3, the proof of Lemma 4 is similar to the proof of 2. In particular, we have the following:

$$\begin{aligned}
\|\hat{\mu} - \hat{\mu}_g\|_2 &\leq \frac{\sum_{i\in V_e\cap S_b}(a_i\|X_i - \nabla f_i\|_2)}{g_e} \\
&\leq \frac{\epsilon(2 + \frac{1}{1-2\epsilon})\cdot\kappa}{1-2\epsilon} \\
&= \frac{3-4\epsilon}{(1-2\epsilon)^2}\epsilon\kappa
\end{aligned} \tag{31}$$

where in the second step we invoke Lemma 3. Moreover, the following inequality also holds:

$$\|\hat{\mu}_g - \mu_g\|_2 \le 2\epsilon\kappa. \tag{32}$$

Summing up these two sets of inequalities we have the following:

$$\|\hat{\mu} - \mu_g\|_2 \le \left(2 + \frac{3-4\epsilon}{(1-2\epsilon)^2}\right)\epsilon\kappa. \tag{33}$$

■

E. Proof of Lemma 6 in Section V

Proof: We only prove the first claim. The second claim is a natural consequence of the first claim and Lemma 5.

To prove the first claim, we note the following:

$$\begin{aligned}\|x_i^\tau[t] - \bar{x}[t]\|_2 &\le \left\| x_i^\tau[t] - \frac{1}{p}\sum_{j\in S} x_j^\tau[t]\right\|_2 \\ &\le \frac{1}{p}\sum_{j\in S}\|x_i^\tau[t] - x_j^\tau[t]\|_2 \\ &\le d^\tau[t].\end{aligned} \tag{34}$$

The rest of the proof comes from the following direct computation:

$$\begin{aligned}&\|\nabla f_i(x_i^\tau[t]) - \nabla f_j(x_j^\tau[t])\|_2 \\ &\le \|\nabla f_i(x_i^\tau[t]) - \nabla f_i(\bar{x}[t])\|_2 \\ &\quad + \|\nabla f_i(\bar{x}[t]) - \nabla f_j(\bar{x}[t])\|_2 \\ &\quad + \|\nabla f_j(x_j^\tau[t]) - \nabla f_j(\bar{x}[t])\|_2 \\ &\overset{(a)}{\le} L_1\|x_i^\tau[t] - \bar{x}[t]\|_2 + \kappa + L_1\|x_j^\tau[t] - \bar{x}[t]\|_2 \\ &\overset{(b)}{\le} 2L_1\, d^\tau[t] + \kappa\end{aligned} \tag{35}$$

where in step (a) we use Assumption 4 (or Assumption 5) and in step (b) we plug in (34). ■

F. Proof of Proposition 1 in Section V

Proof: We consider an arbitrary pair of nodes $i, j \in S$. By Lemma 6 the following two bounds hold:

$$\|\hat{\mu}_i[t] - \mu_i[t]\|_2 \le (\kappa + 2L_1\, d^\tau[t])c_\epsilon \le (\kappa + 2L_1 r)c_\epsilon \tag{36}$$

$$\|\hat{\mu}_j[t] - \mu_j[t]\|_2 \le (\kappa + 2L_1\, d^\tau[t])c_\epsilon \le (\kappa + 2L_1 r)c_\epsilon. \tag{37}$$

In the proof of Lemma 6 we obtain that $\|\nabla f_i(x_i^\tau[t]) - \nabla f_j(x_j^\tau[t])\|_2 \le \kappa + 2L_1 r$ holds for any pair of nodes $i, j \in S$. This implies the following:

$$\|\mu_i[t] - \mu_j[t]\|_2 \le \kappa + 2L_1 r \tag{38}$$

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:

$$\begin{aligned}&\|x_i^0[t+1] - x_j^0[t+1]\|_2 \\ &= \|(x_i^\tau[t] - \eta\hat{\mu}_i[t]) - (x_j^\tau[t] - \eta\hat{\mu}_j[t])\|_2 \\ &\le \|x_i^\tau[t] - x_j^\tau[t]\|_2 + \eta\|\hat{\mu}_i[t] - \hat{\mu}_j[t]\|_2 \\ &\le \eta\|(\hat{\mu}_i[t] - \mu_i[t]) - (\hat{\mu}_j[t] - \mu_j[t]) + (\mu_i[t] - \mu_j[t])\|_2 \\ &\quad + d^\tau[t] \\ &\le \eta\|\hat{\mu}_i[t] - \mu_i[t]\|_2 + \eta\|\hat{\mu}_j[t] - \mu_j[t]\|_2 \\ &\quad + \eta\|\mu_j[t] - \mu_j[t]\|_2 + r \\ &\le \eta(\kappa + 2L_1 r)(1 + 2c_\epsilon) + r.\end{aligned} \tag{39}$$

Since nodes i and j are picked arbitrarily, inequality (39) equivalently implies the following:

$$d^0[t+1] \le r + \eta(\kappa + 2L_1 r)(1 + 2c_\epsilon). \tag{40}$$

Equation (40) 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 $d^\tau[t+1] \le r$, 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 $d^\tau[t] \le a\rho^\tau d^0[t]$. This provides a lower bound of τ_0 via the following analysis:

$$d^\tau[t+1] \le a\rho^\tau d^0[t+1] \le a\rho^\tau(r + \eta(\kappa + 2L_1 r)(1 + 2c_\epsilon)). \tag{41}$$

In order for $d^\tau[t+1] \le r$, it suffices to pick τ_0 to satisfy the following:

$$\tau_0 \ge \log_{\frac{1}{\rho}} \frac{a(r + \eta(\kappa + 2L_1 r)(1 + 2c_\epsilon))}{r}. \tag{42}$$

■

G. Proof of Proposition 2 in Section V

Proof: By the RAGD algorithm, each node j updates its local parameter according to the following:

$$x_j^0[t+1] = x_j^\tau[t] - \eta\hat{\mu}_j[t] \tag{43}$$

and then executes the linear iterative algorithm to reach consensus in the next iteration. Note that $\bar{x}[t] = \frac{1}{p}\sum_{j\in S} x_j^\tau[t]$ which was argued in Lemma 1. We have the following equalities:

$$\bar{x}[t+1] - \bar{x}[t] = \frac{1}{p}\sum_{j\in S}(x_j^0[t+1] - x_j^\tau[t]) = -\frac{\eta}{p}\sum_{j\in S}\hat{\mu}_j[t]. \tag{44}$$

Recall the definition of $\mu_j[t]$ in Lemma 6: $\mu_j[t] = \sum_{i\in\mathcal{N}_j^{\text{in}}} a_{ij}\nabla f_i(x^\tau[t])$, 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 $j \in S$ in the network, we obtain the following:

$$\begin{aligned}&\sum_{j\in S}\hat{\mu}_j[t] - \sum_{j\in S}\mu_j[t] \\ &= \sum_{j\in S}\hat{\mu}_j[t] - \sum_{j\in S}\sum_{i\in\mathcal{N}_j^{\text{in}}} a_{ij}\nabla f_i(x_i^\tau[t]) \\ &= \sum_{j\in S}\hat{\mu}_j[t] - \sum_{i\in S}\nabla f_i(x_i^\tau[t])\end{aligned}$$

$$
\begin{aligned}
&= \sum_{j\in S}\hat{\mu}_j[t] - \nabla f(\bar{x}[t]) - \sum_{i\in S}(\nabla f_i(x_i^\tau[t]) - \nabla f_i(\bar{x}[t]))\\
&= -\frac{p}{\eta}(\bar{x}[t+1]-\bar{x}[t]) - \nabla f(\bar{x}[t])\\
&\quad - \sum_{i\in S}(\nabla f_i(x_i^\tau[t]) - \nabla f_i(\bar{x}[t])) \qquad (45)
\end{aligned}
$$

where in the last step we used (44). On the other hand, we have the following:

$$
\begin{aligned}
\left\|\sum_{j\in S}\hat{\mu}_j[t] - \sum_{j\in S}\mu_j[t]\right\|_2 &\le \sum_{j\in S}\|\hat{\mu}_j[t]-\mu_j[t]\|_2\\
&\le pc_\epsilon(\kappa + 2L_1 r) \qquad (46)
\end{aligned}
$$

from the triangular inequality as well as (10). Meanwhile, it also holds that for any $j \in S$

$$
\begin{aligned}
\|x_j^\tau[t]-\bar{x}[t]\|_2 &= \|x_j^\tau[t] - \frac{1}{p}\sum_{i\in S}x_i^\tau[t]\|_2\\
&\le \frac{1}{p}\sum_{i\in S}\|x_j^\tau[t]-x_i^\tau[t]\|_2\\
&\le r \qquad (47)
\end{aligned}
$$

which, using Assumption 4 (or Assumption 5), yields the following:

$$
\left\|\sum_{i\in S}(\nabla f_i(x_i^\tau[t]) - \nabla f_i(\bar{x}[t]))\right\|_2 \le pL_1 r. \qquad (48)
$$

The inequalities (45), (46), and (48) together imply the following:

$$
\left\|\bar{x}[t+1]-\bar{x}[t] + \frac{\eta}{p}\nabla f(\bar{x}[t])\right\|_2 \le \eta c_\epsilon(\kappa+2L_1 r) + \eta L_1 r \qquad (49)
$$

which directly implies the claim in the proposition. ■

H. Proof of Theorem 1 in Section V

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 1.

Lemma 9: Suppose function $f : \mathbb{R}^n \to \mathbb{R}$ satisfies Assumption 4. For any $x[t] \in \mathbb{R}^n$ updated according to the following:

$$
x[t+1] = x[t] - \frac{1}{L}(\nabla f(x[t]) - l[t]) \qquad (50)
$$

where $\|l[t]\|_2 \le \xi$ for any $t\in\mathbb{N}$, the following inequality holds for any $t \in \mathbb{N}$ and $\beta = \sqrt{1-\nu/L}$:

$$
\|x[t]-x^*\|_2 \le \beta^t\|x[0]-x^*\|_2 + \frac{\xi}{(1-\beta)L}. \qquad (51)
$$

Proof: By Assumption 4, 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:

$$
f(y) - f(x) \le \nabla f^T(x)(y-x) + \frac{L}{2}\|y-x\|_2^2 \qquad (52)
$$

$$
f(x) - f(y) \ge \nabla f^T(y)(x-y) + \frac{\nu}{2}\|y-x\|_2^2. \qquad (53)
$$

A simple reorganization of (53) yields as follows:

$$
f(y) - f(x) \le \nabla f^T(y)(y-x) - \frac{\nu}{2}\|y-x\|_2^2. \qquad (54)
$$

We consider the following set of equalities and inequalities for any $x, y \in \mathbb{R}^n$ and $x^+ = x - 1/L\nabla f(x)$, which will be used later.

$$
\begin{aligned}
&f(x^+) - f(y)\\
&= f(x^+) - f(x) + f(x) - f(y)\\
&\le \nabla f^T(x)(x^+-x) + \frac{L}{2}\|x^+-x\|_2^2\\
&\quad + \nabla f^T(x)(x-y) - \frac{\nu}{2}\|x-y\|_2^2\\
&= \nabla f^T(x)(x^+-y) + \frac{1}{2L}\|\nabla f(x)\|_2^2 - \frac{\nu}{2}\|x-y\|_2^2\\
&= \nabla f^T(x)(x - \frac{1}{L}\nabla f(x) - y)\\
&\quad + \frac{1}{2L}\|\nabla f(x)\|_2^2 - \frac{\nu}{2}\|x-y\|_2^2\\
&= \nabla f(x)^T(x-y) - \frac{1}{2L}\|\nabla f(x)\|_2^2 - \frac{\nu}{2}\|x-y\|_2^2.
\end{aligned}
$$

In particular, when $y = x^*$, we have the following:

$$
\begin{aligned}
&0 \le f(x^+) - f(x^*)\\
&\le \nabla f^T(x)(x-x^*) - \frac{1}{2L}\|\nabla f(x)\|_2^2 - \frac{\nu}{2}\|x-x^*\|_2^2.
\end{aligned}
$$

In the following we prove the result stated in the lemma:

$$
\begin{aligned}
&\|x[t+1]-x^*\|_2\\
&= \|x[t] - x^* - \frac{1}{L}(\nabla f(x[t]) - l[t])\|_2\\
&\le \|x[t]-x^* - \frac{1}{L}\nabla f(x[t])\|_2 + \frac{1}{L}\|l[t]\|_2\\
&\le \|x[t]-x^* - \frac{1}{L}\nabla f(x[t])\|_2 + \frac{\xi}{L}\\
&\overset{(a)}{\le} \sqrt{\left(1-\frac{\nu}{L}\right)\|x[t]-x^*\|_2^2 +} + \frac{\xi}{L}\\
&\overset{(b)}{=} \beta\|x[t]-x^*\|_2 + \frac{\xi}{L} \qquad (55)
\end{aligned}
$$

where in step (a) we plug in $x = x[t]$ and in step (b) we use the definition $\beta = \sqrt{1-\nu/L}$. We note that $\beta \in (0,1)$. Solving (55) recursively gives the following:

$$
\|x[t]-x^*\|_2 \le \beta^t\|x[0]-x^*\|_2 + \frac{\xi}{(1-\beta)L}. \qquad (56)
$$

■

Lemma 9 will be used in the proof of the first claim in Theorem 1. 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 10: Suppose a function $f:\mathbb{R}^n \to \mathbb{R}$ satisfies Assumption 5. For any $x[t] \in \mathbb{R}$ updated according to the following:

$$x[t+1] = x[t] - \frac{1}{L}\nabla f(x[t]) + \frac{1}{L}l[t] \tag{57}$$

where $\|l[t]\|_2 \leq \xi$ for any $t \in \mathbb{N}$. Let $\beta' = 1-\mu/L \in (0,1)$, the following inequality holds for any $t \in \mathbb{N}$:

$$f(x[t]) - f(x^*) \leq \beta'^t (f(x[0]) - f(x^*)) + \frac{\xi^2}{2L(1-\beta')} \tag{58}$$

where $f(x^*)$ is the minimum of the function f.

Proof: By L-smoothness of function f, we have the following inequality:

$$\begin{aligned} f(x[t+1]) &\leq f(x[t]) + \nabla f^T(x[t])(x[t+1]-x[t]) \\ &\quad + \frac{1}{2}\|x[t+1]-x[t]\|_2^2. \end{aligned} \tag{59}$$

Combining with the update rule (57) yields as follows:

$$\begin{aligned} &f(x[t+1]) - f(x[t]) \\ &\leq \frac{1}{L}\nabla f^T(x[t])(-\nabla f(x[t]) + l[t]) \\ &\quad + \frac{L}{2}\left\| -\frac{1}{L}\nabla f(x[t]) + \frac{1}{L}l[t] \right\|_2^2 \\ &\leq -\frac{1}{2L}\nabla f^T(x[t])\nabla f(x[t]) + \frac{1}{2L}l^T[t]l[t] \\ &\leq -\frac{1}{2L}\|\nabla f(x[t])\|_2^2 + \frac{1}{2L}\xi^2 \\ &\leq -\frac{\mu}{L}(f(x[t]) - f(x^*)) + \frac{1}{2L}\xi^2. \end{aligned} \tag{60}$$

This can equivalently be written as follows:

$$\begin{aligned} &(f(x[t+1]) - f(x^*)) - (f(x[t]) - f(x^*)) \\ &\leq -\frac{\mu}{L}(f(x[t]) - f(x^*)) + \frac{1}{2L}\xi^2 \\ &\leq (1-\frac{\mu}{L})(f(x[t]) - f(x^*)) + \frac{1}{2L}\xi^2 \\ &\leq \beta'(f(x[t]) - f(x^*)) + \frac{1}{2L}\xi^2. \end{aligned} \tag{61}$$

Similarly, by solving (61) recursively we obtain the following:

$$f(x[t]) - f(x^*) \leq (\beta')^t (f(x[0]) - f(x^*)) + \frac{\xi^2}{2L(1-\beta')}. \tag{62}$$

■

Before giving our proof of Theorem 1, we need to state the following lemma, which will be used in the proof.

Lemma 11 ([42]): 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

$$f(x) - f(x^*) \geq \frac{\mu}{2}\|x - x^*\|_2^2. \tag{63}$$

Proof: The proof can be found in [42]. ■

The most important implication of Lemma 11 is that if a function $f:\mathbb{R}^n \to \mathbb{R}$ satisfies the PL inequality with parameter μ, then the following bound:

$$\frac{\mu}{2}\mathcal{D}^2(x, S^*) \leq f(x) - f(x^*) \tag{64}$$

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 1: First, let Assumption 4 hold. From Lemma 9, (47), and (11) in the main file, we observe the following:

$$\begin{aligned} &\|x_j^\tau[t] - x^*\|_2 \\ &\leq \|x_j^\tau[t] - \bar{x}[t]\|_2 + \|\bar{x}[t] - x^*\|_2 \\ &\leq \beta^t\|\bar{x}[0] - x^*\|_2 + \frac{\xi}{(1-\beta)L} + r \\ &= \beta^t\|x_j^\tau[0] - x^*\|_2 + \frac{\xi}{(1-\beta)L} + r. \end{aligned} \tag{65}$$

Similarly, the following inequality can also be obtained from Lemma 10 and (11) in the main file. Let Assumption 5 holds, for any $x^* \in S^*$, we have the following:

$$f(\bar{x}[t]) - f(x^*) \leq (\beta')^t (f(\bar{x}[0]) - f(x^*)) + \frac{\xi^2}{2L(1-\beta')}. \tag{66}$$

Combining (66) with (64), we obtain the following set of inequalities:

$$\begin{aligned} \mathcal{D}(\bar{x}[t], S^*) &\leq \sqrt{\frac{2}{\mu}(\beta')^t(f(\bar{x}[0]) - f(x^*) + \frac{\xi^2}{\mu L(1-\beta')}} \\ &\leq \sqrt{\frac{2}{\mu}(\beta')^t(f(\bar{x}[0]) - f(x^*)} + \sqrt{\frac{\xi^2}{\mu L(1-\beta')}} \\ &\leq \sqrt{\frac{L}{\mu}(\beta')^t\mathcal{D}^2(\bar{x}[0], S^*)} + \sqrt{\frac{\xi^2}{\mu L(1-\beta')}} \\ &= \sqrt{\frac{L}{\mu}}\left(\sqrt{\beta'}\right)^t\mathcal{D}(\bar{x}[0], S^*) + \sqrt{\frac{\xi^2}{\mu L(1-\beta')}} \end{aligned}$$

where in the third step we used L-smoothness of function f. In the end, we plug in inequality (47) into (67) and obtain the following inequality:

$$\mathcal{D}(\bar{x}_j^\tau[t], S^*) \leq \sqrt{\frac{L}{\mu}}\left(\sqrt{\beta'}\right)^t\mathcal{D}(x_j^0[0], S^*) + \sqrt{\frac{\xi^2}{\mu L(1-\beta')}} + r. \tag{67}$$

Plugging in the definition $\beta' = 1-\mu/L$ into (67) gives us the following:

$$\mathcal{D}(\bar{x}_j^\tau[t], S^*) \leq \sqrt{\frac{L}{\mu}}\left(\sqrt{\beta'}\right)^t\mathcal{D}\left(x_j^0[0], S^*\right) + \frac{\xi}{\mu} + r. \tag{68}$$

■

References

[1] J. B. Predd, S. B. Kulkarni, and H. V. Poor, “Distributed learning in wireless sensor networks,” *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 56–69, Jul. 2006.

[2] S. Boyd, N. Parikh, and E. Chu, *Distributed Optimization and Statistical Learning Via the Alternating Direction Method of Multipliers*. Boston, MA, USA: Now Pub., 2011.

[3] M. Ma, A. N. Nikolakopoulos, and G. B. Giannakis, “Fast decentralized learning via hybrid consensus ADMM,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2018, pp. 3829–3833.

[4] P. Kairouz et al., “Advances and open problems in federated learning,” *Found. Trends Mach. Learn.*, vol. 14, no. 1–2, pp. 1–210, 2021.

[5] M. Castro et al., “Practical Byzantine fault tolerance,” in *Proc. USENIX Conf. Operating Syst. Des. Implementation*, 1999, pp. 173–186.

[6] K. Driscoll, B. Hall, M. Paulitsch, P. Zumsteg, and H. Sivencrona, “The real Byzantine generals,” in *Proc. 23rd Digit. Avionics Syst. Conf.*, 2004, pp. 6.D.4–61-11.

[7] D. Data and S. Diggavi, “Byzantine-resilient high-dimensional SGD with local iterations on heterogeneous data,” in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2021, pp. 2478–2488.

[8] D. Data and S. Diggavi, “Byzantine-resilient SGD in high dimensions on heterogeneous data,” in *Proc. IEEE Int. Symp. Inf. Theory*, 2021, pp. 2310–2315.

[9] A. Ghosh, R. K. Maity, S. Kadhe, A. Mazumdar, and K. Ramchandran, “Communication-efficient and byzantine-robust distributed learning with error feedback,” *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 3, pp. 942–953, Sep. 2021.

[10] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, “RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets,” in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 1544–1551.

[11] A. Ghosh, J. Hong, D. Yin, and K. Ramchandran, “Robust federated learning in a heterogeneous environment,” 2019, *arXiv:1906.06629*.

[12] D. Alistarh, Z. A.-Zhu, and J. Li, “Byzantine stochastic gradient descent,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4618–4628.

[13] Y. Chen, L. Su, and J. Xu, “Distributed statistical machine learning in adversarial settings: Byzantine gradient descent,” *Proc. ACM Meas. Anal. Comput. Syst.*, 2017, pp. 1–25.

[14] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, “Byzantine-robust distributed learning: Towards optimal statistical rates,” in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 5650–5659.

[15] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, “Data poisoning attacks against federated learning systems,” in *Proc. 25th Eur. Symp. Res. Comput, Secur.*, 2020, pp. 480–501.

[16] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[17] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, “A decentralized second-order method with exact linear convergence rate for consensus optimization,” *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 4, pp. 507–522, Dec. 2016.

[18] F. Bullo, *Lectures on Network Systems*. Seattle, WA, USA: Kindle Direct Pub., 2019.

[19] C. Fang, Z. Yang, and W. U. Bajwa, “BRIDGE: Byzantine-resilient decentralized gradient descent,” *IEEE Trans. Signal Inf. Process. Netw.*, vol. 8, pp. 610–626, 2022.

[20] Z. Yang and W. U. Bajwa, “ByRDiE: Byzantine-resilient distributed coordinate descent for decentralized learning,” *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 4, pp. 611–627, Dec. 2019.

[21] N. Gupta and N. H. Vaidya, “Resilience in collaborative optimization: Redundant and independent cost functions,” 2020, *arXiv:2003.09675*.

[22] L. Su and N. H. Vaidya, “Robust multi-agent optimization: Coping with byzantine agents with input redundancy,” in *Proc. Int. Symp. Stabilization, Safety, Secur. Distrib. Syst.*, 2016, pp. 368–382.

[23] N. Gupta, T. T. Doan, and N. H. Vaidya, “Byzantine fault-tolerance in decentralized optimization under minimal redundancy,” 2020, *arXiv:2009.14763*.

[24] S. Liu, N. Gupta, and N. H. Vaidya, “Approximate byzantine fault-tolerance in distributed optimization,” in *Proc. 2021 ACM Symp. Princ. Distrib. Comput.*, 2021, pp. 379–389.

[25] S. P. Karimireddy, L. He, and M. Jaggi, “Byzantine-robust learning on heterogeneous datasets via bucketing,” in *Proc. 10th Int. Conf. Learn. Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=jXKKDEi5vJt

[26] L. Su and N. H. Vaidya, “Fault-tolerant multi-agent optimization: Optimal iterative distributed algorithms,” in *Proc. 2016 ACM Symp. Princ. Distrib. Comput.*, 2016, pp. 425–434.

[27] S. Sundaram and B. Gharesifard, “Distributed optimization under adversarial nodes,” *IEEE Trans. Autom. Control*, vol. 64, no. 3, pp. 1063–1076, Mar. 2019.

[28] K. Kuwaranancharoen, L. Xin, and S. Sundaram, “Byzantine-resilient distributed optimization of multi-dimensional functions,” in *Proc. Amer. Control Conf.*, 2020, pp. 4399–4404.

[29] L. He, S. P. Karimireddy, and M. Jaggi, “Byzantine-robust decentralized learning via clippedgossip,” 2022, *arXiv:2202.01545*.

[30] E. M. E. -Mhamdi, S. Farhadkhani, R. Guerraoui, A. Guirguis, L.-N. Hoang, and S. Rouault, “Collaborative learning in the jungle (decentralized, byzantine, heterogeneous, asynchronous and non-convex learning),” in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 25 044–25 057.

[31] K. Kuwaranancharoen and S. Sundaram, “On the geometric convergence of byzantine-resilient distributed optimization algorithms,” 2023, *arXiv:2305.10810*.

[32] E. Rosenfeld, E. Winston, P. Ravikumar, and Z. Kolter, “Certified robustness to label-flipping attacks via randomized smoothing,” in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2020, pp. 8230–8241.

[33] E. Wong, L. Rice, and J. Z. Kolter, “Fast is better than free: Revisiting adversarial training,” 2020, *arXiv:2001.03994*.

[34] A. Raghunathan, J. Steinhardt, and P. Liang, “Certified defenses against adversarial examples,” 2018, *arXiv:1801.09344*.

[35] A. Mokhtari, Q. Ling, and A. Ribeiro, “Network newton distributed optimization methods,” *IEEE Trans. Signal Process.*, vol. 65, no. 1, pp. 146–161, 2016.

[36] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli, “Is feature selection secure against training data poisoning?,” in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2015, pp. 1689–1698.

[37] N. Baracaldo, B. Chen, H. Ludwig, and J. A. Safavi, “Mitigating poisoning attacks on machine learning models: A data provenance based approach,” in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, 2017, pp. 103–110.

[38] Y. Shoukry, P. Martin, P. Tabuada, and M. Srivastava, “Non-invasive spoofing attacks for anti-lock braking systems,” in *Proc. Int. Conf. Cryptographic Hardware Embedded Syst.*, 2013, pp. 55–72.

[39] Y. Shoukry, P. Martin, Y. Yona, S. Diggavi, and M. Srivastava, “PyCRA: Physical challenge-response authentication for active sensors under spoofing attacks,” in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1004–1015.

[40] L. Xiao, S. Boyd, and S.-J. Kim, “Distributed average consensus with least-mean-square deviation,” *J. Parallel Distrib. Comput.*, vol. 67, no. 1, pp. 33–46, 2007.

[41] K. Cai and H. Ishii, “Average consensus on arbitrary strongly connected digraphs with time-varying topologies,” *IEEE Trans. Autom. Control*, vol. 59, no. 4, pp. 1066–1071, Apr. 2014.

[42] H. Karimi, J. Nutini, and M. Schmidt, “Linear convergence of gradient and proximal-gradient methods under the Polyak-Ojasiewicz condition,” in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2016, pp. 795–811.

[43] S. Farhadkhani, R. Guerraoui, N. Gupta, R. Pinot, and J. Stephan, “Byzantine machine learning made easy by resilient averaging of momentums,” in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2022, pp. 6246–6283.

[44] S. P. Karimireddy, L. He, and M. Jaggi, “Learning from history for byzantine robust optimization,” in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2021, pp. 5311–5319.

[45] S. B. Hopkins and J. Li, “How hard is robust mean estimation?,” in *Proc. Conf. Learn. Theory*, 2019, pp. 1649–1682.

[46] I. Diakonikolas, G. Kamath, D. Kane, J. Li, A. Moitra, and A. Stewart, “Robust estimators in high-dimensions without the computational intractability,” *SIAM J. Comput.*, vol. 48, no. 2, pp. 742–864, 2019.

[47] G. Lugosi and S. Mendelson, “Robust multivariate mean estimation: The optimality of trimmed mean,” *Ann. Statist.*, vol. 49, no. 1, pp. 393–410, 2021.

[48] L. He, S. P. Karimireddy, and M. Jaggi, “Byzantine-robust learning on heterogeneous datasets via resampling,” 2020, *arXiv:2006.09365*.

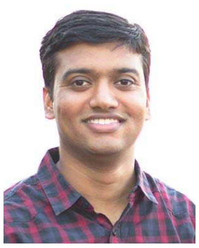
[49] C.-T. Chen, *Linear System Theory and Design*. London, U.K.: Oxford Univ. Press, 1999.

[50] S. S. Kia, B. V. Scoy, J. Cortes, R. A. Freeman, K. M. Lynch, and S. Martinez, “Tutorial on dynamic average consensus: The problem, its applications, and the algorithms,” *IEEE Control Syst. Mag.*, vol. 39, no. 3, pp. 40–72, Jun. 2019.

Yanwen Mao received the B.E. degree in electrical engineering and power automation from Shanghai Jiao Tong University, Shanghai, China, in 2017, the M.S. and Ph.D. degrees in electrical and computer engineering from the University of California, Los Angeles, CA, USA, in 2019 and 2022, respectively.

His current research interests include decentralized optimization and robust networked control systems.

Dr. Mao was the recipient of the Ultra High Voltage Scholarship at Shanghai Jiao Tong University.



Deepesh Data received the B.Tech. degree in computer science and engineering from the International Institute of Information Technology (IIIT-H), Hyderabad, India, in 2011, and the M.Sc. and Ph.D. degrees in computer science from the Tata Institute of Fundamental Research (TIFR), Mumbai, India, in 2017 and 2022, respectively.

From 2018 to 2022, he was a Postdoctoral Scholar with the University of California, Los Angeles (UCLA), and from 2017 to 2018, with the Indian Institute of Technology Bombay (IIT-B). He is currently a Research Scientist with Meta Platforms, Inc., Menlo Park, CA, USA. His research interests are in distributed optimization, machine learning, differential privacy, cryptography, algorithms, and information theory, with a current focus on privacy-preserving machine learning.

Dr. Data was the recipient of the Best Paper Award from the ACM Conference on Computer and Communications Security (CCS) 2021, ACM India Doctoral Dissertation Award for 2019 (Honorable Mention), TIFR-Sasken Best Ph.D. Thesis Award for 2017–18 in Technology and Computer Sciences, and Microsoft Research India Ph.D. Fellowship for 2014–17.

Suhas Diggavi (Fellow, IEEE) received the undergraduate degree from IIT, Delhi, and the Ph.D. degree from Stanford University, Stanford, CA, USA.

He was a Principal Member Research Staff with AT&T Shannon Laboratories and directed the Laboratory for Information and Communication Systems (LICOS), EPFL. He is currently a Professor of electrical and computer engineering with the University of California (UCLA), Los Angeles, CA, USA, where he directs the Information Theory and Systems Laboratory. He has eight issued patents. His research interests include information theory and its applications to several areas, including machine learning, security and privacy, wireless networks, data compression, cyber-physical systems, and bioinformatics and neuroscience.

Dr. Diggavi was the recipient of several recognitions for his research with IEEE and ACM, including the 2013 IEEE Information Theory Society & Communications Society Joint Paper Award, the 2021 ACM Conference on Computer and Communications Security (CCS) Best Paper Award, the 2013 ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc) Best Paper Award, the 2006 IEEE Donald Fink prize paper award among others, and the 2019 Google Faculty Research Award, 2020 Amazon Faculty Research Award, and 2021 Facebook/Meta faculty research award. He was selected as a Guggenheim Fellow in 2021. He was a IEEE Distinguished Lecturer and was also on board of governors for the IEEE Information Theory Society (2016–2021). He has been an Associate Editor for IEEE TRANSACTIONS ON INFORMATION THEORY, ACM/IEEE TRANSACTIONS ON NETWORKING, and other journals and special issues, as well as in the program committees of several IEEE conferences. He has also helped organize IEEE and ACM conferences including serving as the Technical Program CoChair for 2012 IEEE Information Theory Workshop (ITW), the Technical Program CoChair for the 2015 IEEE International Symposium on Information Theory (ISIT), and General CoChair for ACM Mobihoc 2018.

Paulo Tabuada (Fellow, IEEE) was born in Lisbon, Portugal, one year after the Carnation Revolution. He received the “Licenciatura” degree in aerospace engineering from Instituto Superior Tecnico, Lisbon, Portugal, in 1998, and the Ph.D. degree in electrical and computer engineering from the Institute for Systems and Robotics, Instituto Superior Tecnico, in 2002.

From 2002 to 2003 he was a Postdoctoral Researcher with the University of Pennsylvania. After spending three years with the University of Notre Dame, as an Assistant Professor, he joined the Electrical and Computer Engineering Department with the University of California, Los Angeles, CA, USA, where he currently is the Vijay K. Dhir Professor of Engineering.

Dr. Tabuada was the recipient of multiple awards including the NSF CAREER Award in 2005, the Donald P. Eckman Award in 2009, the George S. Axelby Award in 2011, the Antonio Ruberti Prize in 2015 for his contributions to control and cyber-physical systems. He was awarded the grade of Fellow by IFAC in 2019. He has been Program Chair and General Chair for several conferences in the areas of control and of cyber-physical systems, such as NecSys, HSCC, and ICCPS. He is currently the Chair of HSCC’s steering committee and was on the editorial board of the IEEE EMBEDDED SYSTEMS LETTERS and the IEEE TRANSACTIONS ON AUTOMATIC CONTROL.