

A survey on secure decentralized optimization and learning

Changxin Liu^a, Nicola Bastianello^a, Wei Huo^b, Yang Shi^c, Karl H. Johansson^a

^a*Division of Decision and Control Systems, KTH Royal Institute of Technology, and Digital Futures, Stockholm, Sweden*

^b*Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong SAR, China*

^c*Department of Mechanical Engineering, University of Victoria, Victoria, BC, Canada*

Abstract

Decentralized optimization has become a standard paradigm for solving large-scale decision-making problems and training large machine learning models without centralizing data. However, this paradigm introduces new privacy and security risks, with malicious agents potentially able to infer private data or impair the model accuracy. Over the past decade, significant advancements have been made in developing secure decentralized optimization and learning frameworks and algorithms. This survey provides a comprehensive tutorial on these advancements. We begin with the fundamentals of decentralized optimization and learning, highlighting centralized aggregation and distributed consensus as key modules exposed to security risks in federated and distributed optimization, respectively. Next, we focus on privacy-preserving algorithms, detailing three cryptographic tools and their integration into decentralized optimization and learning systems. Additionally, we examine resilient algorithms, exploring the design and analysis of resilient aggregation and consensus protocols that support these systems. We conclude the survey by discussing current trends and potential future directions.

Keywords: Decentralized optimization, federated learning, cyber-physical system security, multi-agent systems

Contents

1	Introduction	1	3.4 Secret sharing-based algorithms	17	
1.1	Organization of the paper	2	3.5 Discussions	17	
1.2	Related survey papers	3			
2	Decentralized optimization and learning	3	4	Resilient decentralized optimization	19
2.1	Optimization problem and system architectures	4	4.1	Objective in the presence of attackers .	20
2.2	Federated optimization	5	4.2	Resilient aggregation and consensus .	20
2.3	Distributed optimization	7	4.3	Metrics of resilience	23
2.4	Constraint-coupled problems	9	4.4	Resilient federated optimization	25
3	Privacy-preserving decentralized optimization	9	4.5	Resilient distributed consensus and optimization	25
3.1	Cryptographic tools	9	4.6	Discussions	28
3.2	Differential privacy-based algorithms . .	13	5	Conclusion and Outlook	29
3.3	Homomorphic encryption-based algorithms	15	5.1	Conclusion	29
			5.2	Topics not covered in this survey	29
			5.3	Future research directions	29

1. Introduction

The confluence of several technological trends in recent years has revolutionized the training of high-performance machine learning models. Indeed, connected devices are increasingly widespread on the customer and industrial markets, allowing to collect mas-

Email addresses: changxin@kth.se (Changxin Liu), nicolba@kth.se (Nicola Bastianello), whuoaa@connect.ust.hk (Wei Huo), yshi@uvic.ca (Yang Shi), kallej@kth.se (Karl H. Johansson)

sive amounts of data. These devices are also equipped with growing computational power, enabling efficient model training. However, data privacy concerns and regulatory restrictions, such as the General Data Protection Regulation (GDPR) [1] and the EU AI Act [2] in Europe, make the traditional approach of consolidating data into a single repository undesirable, and often also impractical.

The goal of training efficient models without compromising privacy has thus spurred the adoption of decentralized algorithms. In this paradigm, spatially distributed computing units, utilizing wireless communication networks and cloud-based computing platforms, collaboratively train machine learning models without disclosing local private data. Decentralized learning algorithms are employed in an increasingly wide range of applications. Examples include the training of predictive keyboards for smartphones by Google [3]; training diagnostic models for healthcare leveraging multi-institution collaborations [4]; intelligent transportation [5, 6]; and many more, see [7] for a survey.

Besides regulatory constraints to safeguard privacy, we remark that adopting the decentralized paradigm also yields a practical advantage. Indeed, tasks like training foundational models with billions of parameters could take years of computation if performed on a single machine [8]. To significantly reduce training time, distributed algorithms leveraging multiple parallel computing nodes are a natural choice [9]. For instance, DeepMind developed a massively distributed architecture for training deep Q-networks (DQNs) for reinforcement learning [10], which achieves comparable performance but ten times faster than single-GPU DQN implementations for most Atari games [11]. Decentralized optimization algorithms have also been employed in data centers to accelerate training [12].

In recent years, research on decentralized optimization and learning algorithms has thus flourished, and we reference the surveys [13, 14, 15, 16] for a comprehensive overview. However, alongside these advancements, the risk of cyber-attacks on decentralized systems has significantly increased. This is especially relevant in sensitive applications such as healthcare and transportation. Referencing the computer security literature, there are three fundamental security properties of information technology systems: *confidentiality*, *integrity*, and *availability*, often referred to as the CIA triad [17, 18]. However, *employing the decentralized paradigm is not enough to satisfy the CIA properties*, showing, e.g., a vulnerability to data privacy attacks despite not disclosing data [19]. The situation is further complicated by ubiquitous cyber-channels in large-scale systems, which

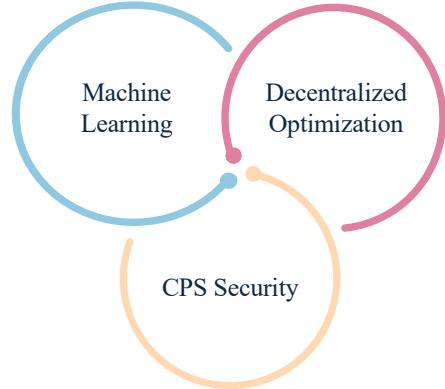


Figure 1: The intersection of machine learning, decentralized optimization, and CPS security.

are inherently susceptible to adversarial attacks compromising parts of the system [20]. These attacks can leak sensitive information from datasets and impair the performance of decision-making or machine learning models. As more centralized learning systems transition to decentralized models, the importance of secure decentralized optimization and learning algorithms becomes increasingly critical.

Addressing this pressing issue requires efforts across three domains: machine learning, decentralized optimization, and cyber-physical system (CPS) security, as illustrated in Figure 1. Recently, significant advancements have been made in developing secure decentralized optimization and learning algorithms. In this paper, we provide a comprehensive review of recent developments in the field.

1.1. Organization of the paper

The survey paper is organized around the two main classes of attacks that secure decentralized optimization and learning aim to address:

- *Attacks on data privacy*: In these attacks, the attackers attempt to steal sensitive information from other honest participants. The attackers, whether inside or outside the system, may record intermediate model updates and gradients. These attacks can be further classified as feature or attribute inference attacks [21, 22], membership inference attacks [23, 24], data reconstruction attacks [19]. Particularly for data reconstruction attacks, advanced inference techniques based on the minimization of gradient difference measured in distance [19] and cosine similarity [25] were reported.

- *Attacks on decision or model security:* In these attacks, the attackers aim to impair the performance of the final decision or model by maliciously manipulating local data and updates. Examples include data poisoning attacks [26] and model poisoning attacks [27]. While arbitrarily behaved poisoning attacks are easily detectable, strategies have been developed to make the poisoning stealthier [28]. Note that the free-riding attack [29], where attackers get the global decision or model without contributing to the process (*e.g.*, by uploading random updates), can also be considered a form of the model poisoning attack.

The outline of the paper, depicted in Figure 2, is as follows. We begin by presenting preliminaries on decentralized optimization and learning over networks in Section 2. Section 3 focuses on privacy-preserving decentralized algorithms designed to protect against data privacy attacks, providing a tutorial on three cryptographic tools and their integration into decentralized optimization and learning algorithms. Then, in Section 4, we survey resilient decentralized optimization and learning algorithms. To offer an in-depth understanding of the underlying principles, we elaborate on the design and analysis of resilient aggregation and consensus protocols, which are key enablers of resilient decentralized algorithms. Section 5 concludes the survey with discussions on potential future directions.

1.2. Related survey papers

There are other surveys dedicated to the secure implementation of decentralized optimization and learning in CPS. Different from this survey, they primarily focus on a limited range of attacks, such as privacy-preserving decentralized optimization [30] and resilient decentralized optimization [31, 32]. We comprehensively address adversarial attacks on both data privacy and model security. Moreover, the aim of this survey is not only to document the state-of-the-art secure solutions to these problems, but also to provide a tutorial on their development. This approach will facilitate an in-depth understanding of the underlying principles of secure decentralized optimization and learning. We also highlight upcoming trends in the field.

Notation. \mathbb{R} , \mathbb{R}^d , and $\mathbb{R}^{d \times d}$, for some given $d > 0$, denote the set of real numbers, vectors, and matrices, respectively. \mathbb{N} and \mathbb{Z} denote the set of natural numbers and integers, respectively. Given $n \in \mathbb{N}$, we denote by $[n]$ the set of integers $\{1, 2, \dots, n\}$. Given $q \in \mathbb{N}$ and $v \in \mathbb{Z}$, the modulo operation is defined as $v \bmod q :=$

$v - \lfloor v/q \rfloor q$, where $\lfloor \cdot \rfloor$ denotes the floor function. Column vectors are considered as the default orientation unless otherwise stated. The symbol \geq is element-wise when applied to vectors. All norms $\|\cdot\|$ are 2-norms unless otherwise stated. For a vector $x \in \mathbb{R}^d$ and a closed convex set $X \subseteq \mathbb{R}^d$, the orthogonal projection mapping is defined as $\text{proj}_X[x] := \arg\min_{y \in X} \|y - x\|$. Let \otimes be the Kronecker product. The cardinality of a set S is denoted as $|S|$.

The notation $O(\cdot)$ is used to represent the asymptotic upper bound on the growth rate of a function. Specifically, $h(n) = O(g(n))$ means that there exist positive constants C and n_0 such that $0 \leq h(n) \leq Cg(n)$ for all $n \geq n_0$. The notation $\Theta(\cdot)$ represents the tightest asymptotic bound. $h(n)$ is said to be $\Theta(g(n))$ if $h(n)$ is bounded above and below by constant multiples of $g(n)$ for sufficiently large values of n .

To describe the network topology, we denote by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ a *directed graph* (in short, *digraph*), where $\mathcal{V} = [n]$ denotes the set of n nodes (or agents) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents the set of edges. For $i, j \in \mathcal{V}$, the ordered pair $(i, j) \in \mathcal{E}$ denotes an edge from i to j . Node j is said to be an in-neighbor of i if $(j, i) \in \mathcal{E}$, and the set of i 's in-neighbors is denoted by $\mathcal{N}_i^- = \{j \in \mathcal{V} | (j, i) \in \mathcal{E}\}$. Similarly, the set of i 's out-neighbors is defined as $\mathcal{N}_i^+ = \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$. A directed path in a graph is an ordered sequence of nodes such that any ordered pair of nodes appearing consecutively in the sequence is an edge. If there exists a directed path between i and j , then j is said to be reachable from i . A graph is *strongly connected* if every node is reachable from any other node. A graph \mathcal{G} is *undirected* if $(i, j) \in \mathcal{E}$ implies that $(j, i) \in \mathcal{E}$.

2. Decentralized optimization and learning

The traditional centralized optimization and learning paradigm, depicted in Figure 3a, entails collecting data in a single location, where it is then processed. The effectiveness and performance of this set-up has been demonstrated extensively. However, in many applications these data are sensitive in nature, and transmitting them would potentially expose them to malicious agents. Moreover, recent technological advances have enabled the aggregation of huge volumes of data, whose transmission would be resource intensive and impractical. A paradigm shift is therefore needed to overcome the limitations of the centralized approach. The objective then becomes that of *enabling cooperative optimization and learning without the agents sharing raw data*.

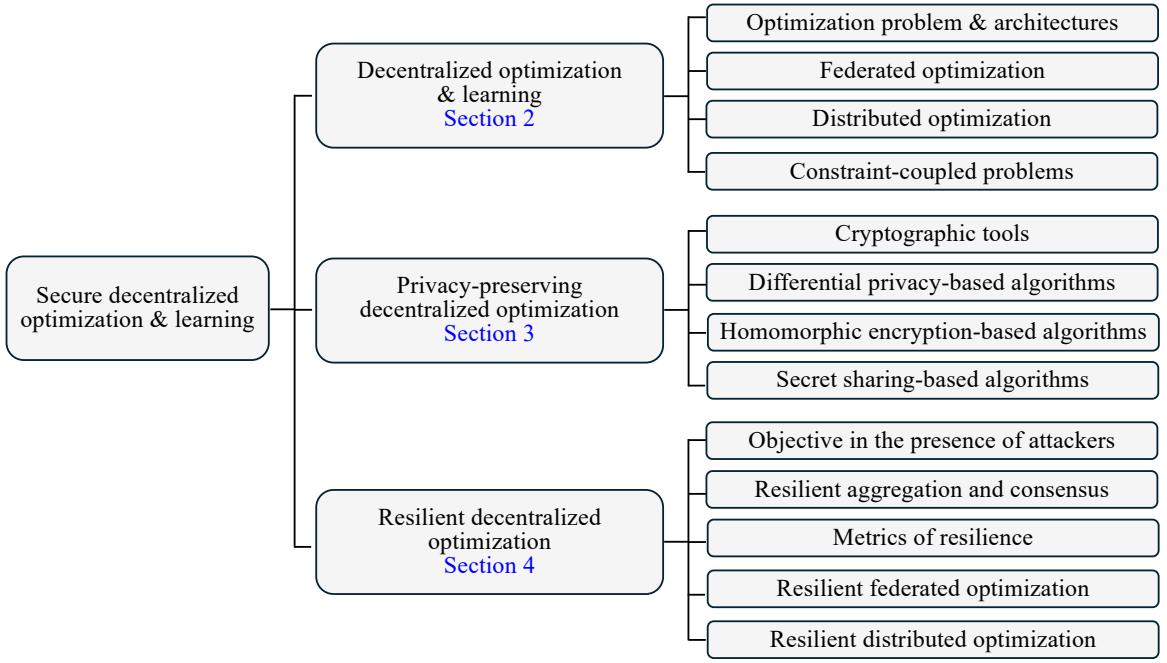


Figure 2: Organization of the paper.

2.1. Optimization problem and system architectures

This section provides a concise introduction to decentralized optimization and learning. The objective is to highlight fundamental concepts and challenges, while for an in-depth literature review we reference [15, 16, 33, 13, 34, 14, 35]. We use several interchangeable terms commonly found in the learning and optimization community, such as objective, cost, or loss, and solution, model, or decision.

We start formulating the problem at hand through empirical risk minimization, a foundational technique in learning.

Example 1 (Empirical risk minimization). Consider a set of $n \in \mathbb{N}$ agents, each locally storing a private dataset $\{\xi_i^h\}_{h=1}^{m_i}$, $i \in \{1, \dots, n\}$, $m_i \in \mathbb{N}$, $\xi_i^h \in \mathbb{R}^q$. Given a loss function $\ell : \mathbb{R}^d \times \mathbb{R}^q \rightarrow \mathbb{R} : (x, \xi) \mapsto \ell(x, \xi)$, which identifies the accuracy of a model, we can define the *empirical risk minimization* (ERM) problem [36]

$$\min_{x \in \mathbb{R}^d} \left\{ \frac{1}{n} \sum_{i=1}^n \sum_{h=1}^{m_i} \ell(x, \xi_i^h) \right\}. \quad (1)$$

As an example, we can take $\xi_i^h = (a_i^h, b_i^h) \in \mathbb{R}^{1 \times d} \times \{-1, 1\}$ and $\ell(x, \xi_i^h) = \log(1 + \exp(-b_i^h a_i^h x))$ for a classification task. In this task, we aim to minimize the fitting error of a logistic regression model parameterized by x . The goal is to optimize the model's parameters

so that it can effectively classify input data into discrete categories. If the local datasets are collected in a single location according to the centralized model of Figure 3a, the problem can be solved as a traditional ERM [36]. However, as discussed above, we want to resort to decentralized architectures to avoid sharing raw data.

Abstracting now from the example of ERM, our objective is to enable the cooperative solution of the class of problems

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x) \right\}, \quad (2)$$

without sharing the local cost (or loss) functions $f_i : \mathbb{R}^d \rightarrow \mathbb{R}$, which may otherwise disclose private data. The idea is to reformulate (2) into the following *consensus optimization* problem [37, 14]

$$\min_{x \in \mathbb{R}^d, \forall i} \left\{ \frac{1}{n} \sum_{i=1}^n f_i(x_i) \right\} \quad \text{s.t. } x_1 = \dots = x_n, \quad (3)$$

whose solutions are denoted

$$x^* = [x^*]_{i=1}^n, \quad x^* \in \operatorname{argmin}_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n f_i(x).$$

Let $x = [x_i]_{i=1}^n$ and define the *consensus constraint set* $C = \{x \in \mathbb{R}^{nd} \mid x_1 = \dots = x_n\}$. We make the following standing assumption to simplify our discussion.

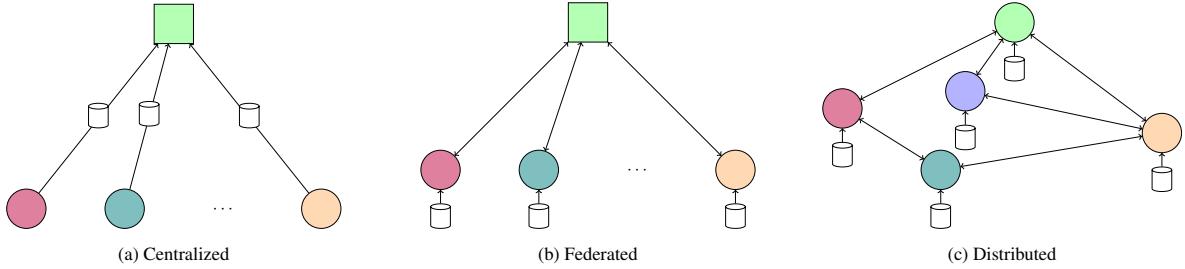


Figure 3: Depictions of the three main architectures for optimization and learning.

Assumption 1 (Smoothness). *The local cost functions are L -smooth:*

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leq L \|x - y\|, \quad \forall x, y \in \mathbb{R}^d, i \in [n].$$

Let us now discuss the solution of (3) under Assumption 1. By smoothness, we can apply the *projected gradient descent* [38], which yields $\mathbf{x}^{(t+1)} = \text{proj}_C(\mathbf{x}^{(t)} - \alpha \nabla f(\mathbf{x}^{(t)}))$, where $\nabla f(\mathbf{x})$ is a vector stacking the local gradients $\nabla f_i(x_i)$. Since the projection of a vector \mathbf{x} onto the consensus set corresponds to averaging its components, the projected gradient descent can be rewritten as the following updates, $t \in \mathbb{N}, i \in [n]$:

$$y_i^{(t+1)} = x_i^{(t)} - \alpha \nabla f_i(x_i^{(t)}) \quad (4a)$$

$$x_i^{(t+1)} = \frac{1}{n} \sum_{j=1}^n y_j^{(t+1)}. \quad (4b)$$

Update (4a) is a *local training* step, during which each agent performs a gradient step on the local cost. Agents only need access to local information (*e.g.*, their dataset $\{\xi_i^{h,m_i}\}_{h=1}^{m_i}$) to perform this update. Update (4b) instead performs *aggregation* of the results of local training, and requires cooperation from all agents. Depending on the architecture deployed to solve (3), there are different ways of implementing this aggregation, which we will discuss in Sections 2.2 and 2.3.

We distinguish two main architectures to enable decentralized optimization and learning, namely *federated*, depicted in Figure 3b, and *distributed*, depicted in Figure 3c. The main difference lies in how the agents that store data are interconnected. In federated architectures, the agents are aided by a *coordinator*, which communicates with all of them and aggregates their local results into a more accurate solution or model. In distributed learning, instead, agents only rely on peer-to-peer communications to converge to a global solution or model.

More specifically, we use the following terminology:

- *Centralized*: to reference traditional set-ups in which data is collected at a single location for processing;
- *Decentralized*: to reference any set-up in which data is privately stored by a set of agents, which process it locally and collaboratively share their results.

Under the decentralized umbrella, we further differentiate:

- *Federated*: in which the coordinator serves the special role of aggregating the results of local computations by communicating with all agents;
- *Distributed* (also called *peer-to-peer* or *decentralized federated*): in which the agents are connected to each other according to some communication graph topology, and no agent serves a privileged role.

2.2. Federated optimization

The federated architecture of Figure 3b is composed of a coordinator connected to each of the n agents. Exploiting this structure, it is therefore possible to implement (4) as detailed in Algorithm 1. In principle, Algorithm 1 is sufficient to cooperatively solve (3). However, deploying this algorithm in practice faces several challenges [13, 14], which require us to refine its design. In the following, we discuss these challenges one by one, and the design solutions that have been proposed to address them. We defer the treatment of the challenge of security to Sections 3 and 4.

Communications bottleneck. Each iteration of Algorithm 1 includes a round of communications from all agents to the coordinator (to transmit $\{y_i^{(t+1)}\}_{i=1}^n$), and then back from the coordinator to the agents ($y^{(t+1)}$). However, the network carrying these communications may act as a bottleneck. On the one hand, the available bandwidth may be limited when employing wireless (or

Algorithm 1 Naïve federated gradient descent

Require: initial conditions $x_i^{(0)}$, $i \in [n]$, step-size $\alpha > 0$.

- 1: **for** $t = 0, 1, \dots$ every agent i **do**
- // local training
- 2: each agent i performs (4a) $y_i^{(t+1)} = x_i^{(t)} - \alpha \nabla f_i(x_i^{(t)})$
- 3: and transmits the result to the coordinator
- // aggregation
- 4: the coordinator collects $\{y_i^{(t+1)}\}_{i=1}^n$, aggregates them $y^{(t+1)} = \frac{1}{n} \sum_{i=1}^n y_i^{(t+1)}$ and broadcasts $y^{(t+1)}$ to all agents, which set $x_i^{(t+1)} = y^{(t+1)}$
- 5: **end for**

over-the-air) communications [39]. On the other, the size of the packets being shared may be large ($d \gg 1$) especially when training high-dimensional models such as (deep) neural networks. The goal therefore is to *reduce the communication requirements* of Algorithm 1.

Different techniques have been leveraged to this end, which we summarize in the following:

- Reducing the size of communications: by applying *quantization or compression*. A host of different quantization/communication techniques have been proposed [40], and we reference [41] for a comprehensive overview.
- Reducing the number of communications: via partial participation, local training, or event-triggered communication. *Partial participation*, or *client selection*, reduces the number of agents that communicate with the coordinator at each iteration. The complementary approach of *local training*, instead, consists in performing multiple steps (or epochs) of local training per each communication round [42]. Another approach is *event-triggered* communication, which initiates communication only when necessary [43, 44].

Resources heterogeneity. In many applications, the agents cooperating towards the solution of (3) are equipped with highly heterogeneous resources [7]. For example, they may rely on different computational powers, communication and storage capabilities, as well as different battery powers. We can thus identify two consequences of resource heterogeneity:

- *Asynchronous local computations*: agents with different computational powers, or different power consumption rates, may conclude a round of local training with different speeds, thus communicating

Algorithm 2 FedAvg

Require: initial conditions $x_i^{(0)}$, $i \in [n]$, step-size $\alpha > 0$, number of participating clients n_p , local training epochs n_e .

- 1: **for** $t = 0, 1, \dots$ **do**
- // local training
- 2: randomly pick n_p participating agents
- 3: **for** each participating agent i **do**
- 4: set $w_{i,0}^{(t)} = x_i^{(t)}$ and
- 5: **for** $k = 0, 1, \dots, n_e$ **do**
- 6: $w_{i,k+1}^{(t)} = w_{i,k}^{(t)} - \alpha \hat{\nabla} f_i(w_{i,k}^{(t)})$
- 7: **end for**
- 8: set $y_i^{(t+1)} = w_{i,n_e}^{(t)}$ and transmit it to the coordinator
- 9: **end for**
- // aggregation
- 10: the coordinator collects $\{y_i^{(t+1)}\}$ from the participating agents, aggregates them

$$y^{(t+1)} = \frac{1}{n} \left(\sum_{i \text{ part.}} y_i^{(t+1)} + \sum_{i \text{ not part.}} y_i^{(t)} \right)$$

and broadcasts $y^{(t+1)}$ to all agents, which set $x_i^{(t+1)} = y^{(t+1)}$

- 11: **end for**
-

more or less frequently with the coordinator [7]. This results in partial participation, albeit due to external constraints rather than by design choice.

- *Inexact local training*: to account for their limited resources, the agents may resort to inexact local training techniques. The foremost example is the computation of *stochastic gradients* $\hat{\nabla} f_i$ rather than full gradients ∇f_i [45]. This may however result in loss of accuracy.

We are now ready to re-design Algorithm 1 by incorporating these techniques. The resulting Algorithm 2 coincides with *FedAvg*, the foundational federated algorithm proposed in [46].

One last challenge: statistical heterogeneity. Recall the empirical risk minimization problem (1), where each agent collects and stores a dataset $\{\xi_i^h\}_{h=1}^{m_i}$, which defines the local cost $f_i(x) = \sum_{h=1}^{m_i} \ell(x, \xi_i^h)$. The datapoints of each agent can be modeled as randomly drawn from a given distribution $\xi_i^h \sim \mathcal{D}_i$. However, in most applications these *data distributions are heterogeneous* [14].

As discussed above, the design of Algorithm 2 includes the use of multiple local training steps to reduce

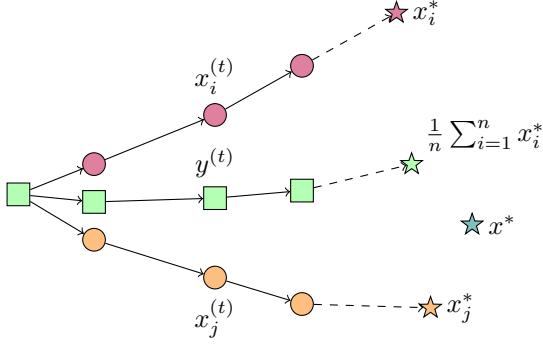


Figure 4: An illustration of client drift.

the number of communications. However, this heuristic turns out to be detrimental when the data distributions $\{\mathcal{D}_i\}_{i=1}^n$ are heterogeneous (or non-i.i.d.). Indeed, during local training an agent's solution is skewed by the biased perspective that its data offers, leading to *client drift* [47], depicted in Figure 4. In particular, the trajectory $\{x_i^{(t)}\}_{t \in \mathbb{N}}$ of each agent i converges towards a local solution $x_i^* \in \operatorname{argmin}_{x \in \mathbb{R}^d} f_i(x)$. And the trajectory $\{y^{(t)}\}_{t \in \mathbb{N}}$ generated by the coordinator converges towards $\frac{1}{n} \sum_{i=1}^n x_i^*$ rather than the true solution x^* .

Research into solutions to statistical heterogeneity has proliferated in the past years, with [42] detailing a short history. The main goal is the design of novel federated algorithms which employ multiple local training steps without undergoing client drift. In other words, local training is no longer used as a heuristic, with provable guarantees on the algorithms' convergence.

2.3. Distributed optimization

The algorithms discussed in Section 2.2 were designed on the blueprint of the projected gradient descent in (4), where the coordinator implements the projection (4b) onto the consensus set. However, distributed architectures in general lack a central node capable of averaging local computations. The question then is: *how can a projection onto the consensus set be implemented in a distributed fashion?*

In the following we briefly discuss the research stemming from this question, and discuss how the challenges of Section 2.2 also impact the design of distributed algorithms. To simplify our notation, we consider undirected graphs only.

Distributed gradient descent. Research on the design of distributed protocols that converge to a consensus on the average of local quantities $\{u_i\}_{i=1}^n$ has matured significantly in the past decades [48]. The central protocol

that has been developed and analyzed is that of *average consensus*:

$$z_{i,0} = u_i, \quad z_{i,k+1} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} z_{j,k} \quad k = 0, 1, \dots \quad (5)$$

where \mathcal{N}_i is the set of agents that exchange information with i , and w_{ij} are positive coefficients that sum to one.

Therefore, one of the first attempts at designing a distributed version of (4) has been to try and implement the projection (4b) with average consensus (5). The resulting distributed gradient descent algorithm is [49]

$$y_i^{(t+1)} = x_i^{(t)} - \alpha \nabla f_i(x_i^{(t)}) \quad (6a)$$

$$x_i^{(t+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} y_j^{(t+1)}. \quad (6b)$$

However, this algorithm does not converge to the optimal solution: in fact, it can only reach a neighborhood thereof, whose radius depends on the magnitude of the step-size α [49], see also [50, 51]. As such, exact – but sub-linear – convergence can only be achieved by employing a diminishing step-size [37].

Gradient tracking. The issue with (6) is that the consensus projection (4b) is applied on a dynamic quantity, which changes at every local gradient step. It is therefore necessary to resort to distributed protocols that can track the dynamic average of time-varying signals. This gave rise to the class of *gradient tracking* algorithms.

Let $\{u_{i,k}\}_{i=1}^n, k \in \mathbb{N}$ be local signals, then one example of dynamic average consensus is [48]

$$z_{i,k+1} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} (z_{j,k} + u_{j,k+1} - u_{j,k}) \quad k = 0, 1, \dots \quad (7)$$

Applying this scheme to approximate the consensus projection (4b) thus yields the algorithm

$$y_i^{(t+1)} = x_i^{(t)} - \alpha \nabla f_i(x_i^{(t)}) \quad (8a)$$

$$x_i^{(t+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} (x_j^{(t)} + y_j^{(t+1)} - y_j^{(t)}) \quad (8b)$$

which converges to the exact solution with a constant step-size [52]. For a comprehensive overview of gradient tracking algorithms we reference [37, 53].

Distributed dual averaging. Let $d(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}$ be a strongly convex proximal function, then applying the (centralized) dual averaging to (2) yields the update [54]:

$$x^{(t+1)} = \operatorname{argmin}_{x \in \mathbb{R}^d} \left\{ \alpha_t \left\langle \sum_{\tau=0}^t g^{(\tau)}, x \right\rangle + d(x) \right\} \quad (9)$$

where $g^{(t)}$ is a subgradient of $\frac{1}{n} \sum_{i \in \mathcal{V}} f_i$ at $x^{(t)}$, and α_t is a non-increasing sequence of positive parameters. In particular, we can choose

$$d(x) = \tilde{d}(x) - \tilde{d}(x^{(0)}) - \langle \nabla \tilde{d}(x^{(0)}), x - x^{(0)} \rangle,$$

where \tilde{d} is any 1-strongly convex function, such as $\tilde{d}(x) = \|x\|^2/2$.

Similarly to the consensus projection (4b), we need to design a protocol that can compute $z^{(t)} = \sum_{\tau=0}^t g^{(\tau)}$ in a distributed fashion. The idea proposed in [55] is to employ a consensus-based mechanism:

$$z_i^{(t+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} z_j^{(t)} + g_i^{(t)}, \quad (10)$$

which is followed by a local dual averaging step

$$x_i^{(t+1)} = \operatorname{argmin}_{x \in \mathcal{X}} \left\{ \alpha_t \langle z_i^{(t+1)}, x \rangle + d(x) \right\}. \quad (11)$$

The resulting distributed dual averaging, however, has sub-linear convergence [55, 56], although a modified version achieved linear convergence in [57].

Having explored gradient-based algorithms in the previous sections, we now shift our focus to *primal-dual methods*.

Alternating direction method of multipliers. Let us start by reformulating the consensus optimization problem (3) as the equivalent

$$\min_{x_i \in \mathbb{R}^d, \forall i} \left\{ \frac{1}{n} \sum_{i=1}^n f_i(x_i) \right\} \quad \text{s.t.} \quad x_i = y_{ij}, \quad x_j = y_{ji}, \quad y_{ij} = y_{ji}, \quad (12)$$

where the *bridge* variables y_{ij}, y_{ji} , two for each edge, are introduced to decouple the consensus constraints. We define the *augmented Lagrangian* of (12):

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{w}) = \frac{1}{n} \sum_{i=1}^n f_i(x_i) + g(\mathbf{y}) + \langle \mathbf{w}, \mathbf{A}\mathbf{x} - \mathbf{y} \rangle + \frac{\rho}{2} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2 \quad (13)$$

where $\rho > 0$ is a *penalty* parameter, \mathbf{w} the Lagrange multipliers, \mathbf{A} a matrix encoding the constraints $x_i = y_{ij}$, $x_j = y_{ji}$, and $g(\mathbf{y}) = 0$ if $y_{ij} = y_{ji} \forall i, j \in \mathcal{N}_i$, $g(\mathbf{y}) = +\infty$ otherwise. The alternating direction method of multipliers (ADMM) [58] is then defined by applying alternating minimization to the augmented Lagrangian

$$\mathbf{x}^{(t+1)} = \operatorname{argmin}_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{y}^{(t)}, \mathbf{w}^{(t)}) \quad (14a)$$

$$\mathbf{y}^{(t+1)} = \operatorname{argmin}_{\mathbf{y}} \mathcal{L}(\mathbf{x}^{(t+1)}, \mathbf{y}, \mathbf{w}^{(t)}) \quad (14b)$$

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \rho (\mathbf{A}\mathbf{x}^{(t+1)} - \mathbf{y}^{(t+1)}), \quad (14c)$$

which, exploiting the distributed structure of the problem yields Algorithm 3 [59].

Algorithm 3 Distributed ADMM

Require: initial conditions $y_{ij}^{(0)}, w_{ij}^{(0)}, i \in [n], j \in \mathcal{N}_i$, penalty $\rho > 0$.

- 1: **for** $t = 0, 1, \dots$ **do**
- // local training
- 2: **for** each active agent i **do**
- 3: compute
- $x_i^{(t+1)} = \operatorname{argmin}_{x_i \in \mathbb{R}^d} \left\{ f_i(x_i) + \langle x_i, \sum_{j \in \mathcal{N}_i} w_{ij}^{(t)} - \rho y_{ij}^{(t)} \rangle + \frac{\rho |\mathcal{N}_i|}{2} \|x_i\|^2 \right\}$
- 4: transmit $x_i^{(t+1)} - w_{ij}^{(t)}$ to each neighbor $j \in \mathcal{N}_i$
- 5: **end for**
- // consensus
- 6: **for** each agent i and active neighbor $j \in \mathcal{N}_i$ **do**
- 7: receive $x_j^{(t+1)} - w_{ji}^{(t)}$ and compute
- $y_{ij}^{(t+1)} = \frac{1}{2\rho} (x_i^{(t+1)} - w_{ij}^{(t)} + x_j^{(t+1)} - w_{ji}^{(t)})$
- $w_{ij}^{(t+1)} = w_{ij}^{(t)} - \rho (x_i^{(t+1)} - y_{ij}^{(t+1)})$
- 8: **end for**
- 9: **end for**

Revisiting the challenges of decentralized architectures. The previous sections illustrated how reaching consensus in a distributed architecture poses a significant design challenge. Additionally, distributed architectures are also subject to the challenges of *communications bottlenecks*, *resources heterogeneity*, and *statistical heterogeneity*, discussed in Section 2.2.

First of all, both gradient-based algorithms and dual methods require peer-to-peer communications, which might need to be quantized/compressed to reduce their burden. Interestingly, while ADMM is robust to quantization [60], gradient tracking in general is not [61], and requires specific design changes. The weakness of gradient tracking descends from its reliance on (dynamic) average consensus, which is not robust to communication errors. Similarly, average consensus is not robust to the partial participation [62], usually called *asynchrony* in this context, resulting from resources heterogeneity – while ADMM is [59]. The solution, therefore, is to robustify gradient tracking by replacing average consensus with a different protocol, see e.g., [63, 64, 65]. While inherently robust to asynchrony and communication errors, ADMM (and dual averaging as well) presents a different challenge: the agents are required to solve an optimization problem to update $x_i^{(t)}$. However, given their limited (and heterogeneous) resources,

the agents may be able to only compute an inexact update [60].

2.4. Constraint-coupled problems

The previous sections focused on the consensus optimization problem of (3) which models a broad class of decentralized learning and optimization problems. This problem is also referred to as *cost-coupled*, as the agents' data is combined through the sum of local loss functions. Another class of problems with relevance in several decentralized applications is that of *constraint-coupled* [33], characterized by

$$\min_{x_i \in \mathbb{R}^{d_i}, \forall i} \quad \sum_{i=1}^n f_i(x_i) \quad \text{s.t.} \quad \sum_{i=1}^n g_i(x_i) \leq 0, \quad x_i \in X_i, \quad \forall i \quad (15)$$

where the local decisions $\{x_i \in \mathbb{R}^{d_i}\}_{i=1}^n$ are coupled through the global constraint $\sum_{i=1}^n g_i(x_i) \leq 0$, $g_i : \mathbb{R}^d \rightarrow \mathbb{R}^M$, and additionally are subject to the local constraints $x_i \in X_i$. Problem (15) finds application in resource allocation [66, 67], distributed charging control of electric vehicles [68], distributed control of multi-agent systems [69, 70], etc. The algorithms discussed in Sections 2.2 and 2.3 cannot be directly applied to solve (15) and tailored approaches need to be designed, discussed briefly in the following. Sections 3 and 4 will further illustrate how privacy and robustness mechanisms can be tailored to this class of problems.

A widely used approach to solving (15) is to leverage the fact that its dual is a cost-coupled problem [33]. In particular, the dual optimization problem of (15) is

$$\max_{w \geq 0} \left\{ \sum_{i=1}^n \psi_i(w) \right\}, \quad (16)$$

where $\psi_i(w) := \min_{x_i \in X_i} \{f_i(x_i) + \langle w, g_i(x_i) \rangle\}$ and $w \in \mathbb{R}^M$ is the vector of dual variables. Clearly, (16) is a cost-coupled problem and – in principle – the algorithms of Sections 2.2 and 2.3 can now be applied to it. However, notice that the local costs ψ_i are defined through a minimization problem, and may not be smooth. Different approaches can be taken to solve this issue, and we reference [71, 72, 73, 74] for some examples.

Solving the cost-coupled dual of (15), however, only guarantees asymptotic satisfaction of the constraint. This approach is thus not suitable for safety-critical systems, in which the constraints need to be verified at all times. The alternative is to address directly a suitably reformulated version of the primal problem (15), see [75, 76, 77, 78] for some examples.

3. Privacy-preserving decentralized optimization

In the decentralized algorithms discussed in Section 2, the agents cooperate by sharing locally trained models (either with the coordinator or their neighbors). However, sharing these models potentially exposes the agents' private data. This section will start by reviewing the prevalent cryptographic tools employed to avoid data leakage. This will be followed by a comprehensive review of privacy-preserving decentralized optimization and consensus algorithms.

In this section, we consider the following three types of privacy attacks. Some examples are provided in Table 1.

- *Honest-but-curious agents or central coordinator*: These agents or the central coordinator follow the algorithm to perform communication and computation. However, they may record intermediate results to infer sensitive information about other agents.
- *Colluding agents*: Certain agents may collude with the central coordinator or amongst themselves to deduce private information about a specific victim agent.
- *Outside eavesdropper*: This attacker can intercept all shared messages during the execution of the training protocol but will not actively inject false messages or interrupt message transmissions.

3.1. Cryptographic tools

In the field of optimization and learning, key privacy-preserving techniques include differential privacy [79], homomorphic encryption [80], and the secret-sharing protocol [81].

3.1.1. Differential privacy

In recent years, differential privacy (DP) has become prevalent in cryptography and machine learning [82, 83], owing to its precise mathematical formulation and computational tractability. Informally, DP ensures that the output of an algorithm remains insensitive to the presence or absence of any individual's data in the dataset, thereby protecting individuals from privacy attacks that aim to extract their personal information.

Next, we review some key concepts in DP. Let $\mathcal{D} = \{\xi^1, \dots, \xi^q\}$ denote a dataset of size q with records drawn from a universe \mathbb{X} (e.g., \mathbb{R}^d). Two datasets \mathcal{D} and \mathcal{D}' are referred to as *neighboring* if they are of the same size and differ in one data point. This is particularly relevant

Table 1: Examples of privacy attacks.

Attack vectors	Description	Representative references
feature/attribute inference	derive an individual's characteristics from the intermediate computation results	[21, 22]
membership inference	determine whether a sample was used to train the model (tracing attack)	[23, 24]
data construction	reconstruct training samples and associated labels accurately	[19, 25]

to machine learning where the optimization problem is defined over datasets, see Example 1.

Definition 1 (Differential privacy). Given $\varepsilon, \delta \geq 0$, a randomized function $\mathcal{M} : \mathbb{X}^q \rightarrow \mathbb{Y}$ is said to be (ε, δ) -DP, if for every pair of neighboring datasets $\mathcal{D}, \mathcal{D}' \in \mathbb{X}^q$ and every subset $O \subseteq \mathbb{Y}$, we have

$$\mathbb{P}[\mathcal{M}(\mathcal{D}) \in O] \leq e^\varepsilon \mathbb{P}[\mathcal{M}(\mathcal{D}') \in O] + \delta. \quad (17)$$

In addition, when $\delta = 0$ in (17), \mathcal{M} is said to be ε -DP.

Definition 1 highlights the indistinguishability of the datasets \mathcal{D} and \mathcal{D}' based on the output of the function \mathcal{M} . To see this, we consider the case $\delta = 0$, for which it holds

$$\frac{\mathbb{P}[\mathcal{M}(\mathcal{D}) \in O]}{\mathbb{P}[\mathcal{M}(\mathcal{D}') \in O]} \leq e^\varepsilon.$$

The numerator represents the probability of observing the output O when the function \mathcal{M} is applied to the dataset \mathcal{D} , while the denominator represents the probability of observing the same output O when \mathcal{M} is applied to the neighboring dataset \mathcal{D}' . The requirement that the ratio of these two probabilities be bounded by e^ε implies that the outputs of \mathcal{M} on the two datasets are indistinguishable, as their probabilities of occurrence cannot differ significantly.

We present two fundamental mechanisms that are widely used to achieve DP in the field of optimization and learning: the Laplace and Gaussian mechanisms.

Lemma 1 (Laplace mechanism). *Consider the Laplace mechanism for answering the query $r : \mathcal{D} \rightarrow \mathbb{R}^d$:*

$$\mathcal{M}_L(\mathcal{D}) = r(\mathcal{D}) + (\nu_1, \dots, \nu_d),$$

where $\nu_i, \forall i$ are independent and have the following probability density function $e^{-\frac{x}{\lambda}}/(2\lambda)$ where $\lambda > 0$ is a scale parameter. The mechanism \mathcal{M}_L is $(\Delta_1/\lambda, 0)$ -DP where Δ_1 denotes the ℓ_1 -sensitivity of r , i.e., $\Delta_1 = \sup_{\mathcal{D}, \mathcal{D}'} \|r(\mathcal{D}) - r(\mathcal{D}')\|_1$ where \mathcal{D} and \mathcal{D}' are neighboring datasets.

Lemma 2 (Gaussian mechanism). *Consider the Gaussian mechanism for answering the query $r : \mathcal{D} \rightarrow \mathbb{R}^d$:*

$$\mathcal{M}_G(\mathcal{D}) = r(\mathcal{D}) + \nu,$$

where $\nu \sim \mathcal{N}(0, \sigma^2 I_d)$. The mechanism \mathcal{M}_G is $(\sqrt{2 \log(2/\delta)} \Delta_2 / \sigma, \delta)$ -DP where Δ_2 denotes the ℓ_2 -sensitivity of r , i.e., $\Delta_2 = \sup_{\mathcal{D}, \mathcal{D}'} \|r(\mathcal{D}) - r(\mathcal{D}')\|$ where \mathcal{D} and \mathcal{D}' are neighboring datasets.

Next, we take the projected gradient descent in (4) applied to the ERM problem in (1) as an example to illustrate the Gaussian mechanism.

Example 2 (Differentially private gradient descent). Denote by $\mathcal{D}_i = \{\xi_i^h\}_{h=1}^m$ the local dataset to agent i . Then, the local gradient descent update in (4a) can be rewritten as

$$y_i^{(t+1)} = x_i^{(t)} - \alpha r^{(t)}(\mathcal{D}_i)$$

where α is the step-size and

$$r^{(t)}(\mathcal{D}_i) = \sum_{h=1}^m \nabla \ell(x_i^{(t)}, \xi_i^h).$$

To make the gradient query $r^{(t)}$ differentially private, we employ the Gaussian mechanism

$$\mathcal{M}_G^{(t)} = r^{(t)}(\mathcal{D}_i) + \nu,$$

where $\nu \sim \mathcal{N}(0, \sigma^2 I_d)$. As elaborated in Lemma 2, the privacy level of $\mathcal{M}_G^{(t)}$ is determined by σ and the sensitivity of $r^{(t)}(\mathcal{D}_i)$, defined by

$$\Delta_2 = \sup_{\xi_i \in \mathcal{D}_i, \xi'_i \in \mathcal{D}'_i} \|\nabla \ell(x_i^{(t)}, \xi_i) - \nabla \ell(x_i^{(t)}, \xi'_i)\|.$$

We note that the privacy guarantees provided in Lemmas 1 and 2 are for a single query, i.e., a single iteration of the optimization algorithm. The privacy leakage can accumulate when multiple queries are made to a differentially private mechanism. Finding tighter bounds on the cumulative privacy leakage for multiple queries is

an active area of research [84]. In the simplest case, the DP parameter grows linearly with respect to the number of queries [85]. Advanced composition of multiple queries and the corresponding privacy leakage have been studied in [86]. Tight composition bounds were also provided along with other statistical privacy definitions, such as Rényi DP [87].

DP can be achieved through other mechanisms as well, *e.g.*, the exponential mechanism [88] and the median mechanism [89]. For special scenarios where the query outputs are bounded, such as interval observer design for dynamical systems [90], the *truncated* Laplace mechanism [91] and the *truncated* Gaussian mechanism [92] can be employed. Furthermore, while both the Laplace and Gaussian mechanisms have analytic expressions for the noise, numerically optimized additive noise mechanisms have been discussed in [93]. Finally, we remark that the DP constraint typically induces a trade-off between privacy and utility in learning algorithms [94, 95, 96], where the utility quantifies the accuracy of a trained model.

3.1.2. Homomorphic encryption

Homomorphic encryption (HE) is a classical encryption scheme [97, 98]. The goal in cryptography is to take a message or data, called plaintext, and provide an encrypted version, called ciphertext, which obscures the original text. Indeed, without the correct decryption key or algorithm, it is computationally infeasible to recover the original plaintext from the ciphertext, thereby ensuring the confidentiality of the information. A fully homomorphic encryption scheme (FHE) refers to a scheme, where *any* computations on the plaintexts can be obtained by directly manipulating ciphertexts so that the privacy of the underlying plaintexts is preserved. Such property for the encryption and decryption operations is referred to as *homomorphic*, which describes the preservation of the algebraic structure with respect to a certain arithmetic function. FHE is achievable if and only if addition and multiplication operations can be homomorphically performed [99]. More formally, let $x \in \mathbb{Z}$ and $y \in \mathbb{Z}$ be the plaintexts. Their ciphertexts are denoted as $\text{Enc}(x)$ and $\text{Enc}(y)$, respectively, where $\text{Enc}(\cdot)$ represents the encryption operation. The encryption of $x + y$ (or $x \cdot y$) can be obtained by simply adding or multiplying the corresponding ciphertexts, *i.e.*,

$$\text{Dec}(\text{Enc}(x) \triangle \text{Enc}(y)) = x \triangle y \quad (18)$$

where \triangle represents either the addition or multiplication operation, and $\text{Dec}(\cdot)$ is the decryption operation.

We remark that when dealing with non-integer plaintexts, quantization techniques can be employed to convert them into integer representations. However, the quantization process introduces approximation errors, and these quantization effects on the overall system performance should be properly handled.

It is important to note that the encryption procedure introduces random noise, and the error accumulates as more homomorphic operations are performed, which may eventually prevent the correct decryption of ciphertexts. Consequently, the homomorphic property holds only for a bounded number of operations. To address this issue, the *bootstrapping* technique has been employed [100], which outputs a new ciphertext with reduced noise based on a ciphertext with high noise and a bootstrapping or refreshing key. This process enables an unlimited number of homomorphic operations to be performed. Importantly, this procedure is carried out without decrypting the input ciphertext, meaning that it is a homomorphic evaluation of the decryption procedure. Bootstrapping is typically the most sophisticated and computationally demanding component of an FHE scheme [101].

A cryptosystem is called *additively homomorphic* if (18) only holds for sum, *e.g.*, Paillier cryptosystem [98], and *multiplicatively homomorphic* if (18) holds for product, *e.g.*, RSA cryptosystem [80]. Any additively homomorphic cryptosystems also support encrypted multiplications with (un-encrypted) constant numbers, that is, given $k \in \mathbb{N}$

$$k \cdot \text{Enc}(y) = \underbrace{\text{Enc}(y) + \text{Enc}(y) + \cdots + \text{Enc}(y)}_{k \text{ times}}.$$

Extensions can be made to the case with $k \in \mathbb{Z}$ and multiplications with integer matrices. Therefore, although being less powerful than FHE, additively homomorphic encryption offers a computationally more tractable approach for designing privacy-preserving optimization and control algorithms [102, 103].

Example 3 (Paillier cryptosystem and its application in privacy-preserving optimization). The Paillier cryptosystem [98] can be summarized in three key steps:

- 1) *Key generation*: Pick two large prime numbers p and q satisfying $\text{gcd}(pq, (p-1)(q-1)) = 1$ where gcd represents the greatest common divisor. Let $N = pq$. Choose $1 \leq g \leq N^2$ randomly and compute $\lambda = \text{lcm}(p-1, q-1)$ where lcm is the least common multiple. Compute μ as a modular multiplicative inverse of $L(g^\lambda \bmod N^2)$, *i.e.*, $\mu \cdot L(g^\lambda \bmod N^2) \equiv 1 \bmod N$, where $L(x) = \frac{x-1}{N}$. The public key is (N, g) . The private key is (λ, μ) .

2) *Encryption*: To encrypt an (integer) message m :

- i) Choose a random integer $0 < x < N$,
- ii) Produce an encryption $\text{Enc}(m) = g^m x^N \bmod N^2$.

Through arithmetic operations, it can be demonstrated that the Paillier cryptosystem exhibits an additive homomorphic property, which is expressed as follows:

$$\begin{array}{ll} \text{Enc}(m_1) & = g^{m_1} x_1^N \bmod N^2 \\ \text{Enc}(m_2) & = g^{m_2} x_2^N \bmod N^2 \\ \text{Enc}(m_1) \cdot \text{Enc}(m_2) & = g^{m_1+m_2} (x_1 x_2)^N \bmod N^2 \\ & = \text{Enc}(m_1 + m_2) \end{array}$$

3) *Decryption*: Let c be the ciphertext to decrypt. The plaintext message can be exactly recovered as

$$m = L(c^\lambda \bmod N^2) \cdot \mu \bmod N$$

The correctness of the algorithm was proved in [98]. Thanks to its additively homomorphic property, Paillier's cryptosystem can be conveniently employed to implement the optimization algorithm. Recall the gradient descent update step given by (4a):

$$y_i^{(t+1)} = x_i^{(t)} - \alpha \nabla f_i(x_i^{(t)}).$$

To perform this update step in an encrypted manner, we can utilize Paillier cryptosystem as shown in [104]:

$$\begin{aligned} \text{Enc}(y_i^{(t+1)}) &= \text{Enc}(x_i^{(t)} - \alpha \nabla f_i(x_i^{(t)})) \\ &= \text{Enc}(x_i^{(t)}) \cdot \text{Enc}(-\alpha \nabla f_i(x_i^{(t)})). \end{aligned}$$

3.1.3. Secret sharing

Secret sharing (SS) protocols are encryption methods for distributing a confidential message in the form of “shares” to multiple parties. The message remains confidential as long as the attacker can only intercept a limited number of shares. Specifically, an (n, k) -threshold SS scheme consists of two sub-algorithms: *Share* and *Reconstruct*. The *Share* algorithm takes a private message S as input and outputs n secret shares $\{S_1, \dots, S_n\}$. The *Reconstruct* algorithm takes k secret shares $\{S_i\}_{i \in \mathcal{I}}$ with $|\mathcal{I}| = k$ and $\mathcal{I} \subseteq [n]$, and recovers S as the output. Each share S_i is distributed to one party $i \in [n]$. The requirement is that any collusion of less than k parties cannot reconstruct the private message S . Each party is able to perform computations on its share and generate an outcome, from which the data owner can reconstruct the computation result.

Shamir's scheme [81] is a well-known SS scheme in which the secret shares are generated by evaluating a polynomial of degree $(k - 1)$ over n different points. In particular, consider a polynomial of degree $(k - 1)$:

$$p(\theta) = \sum_{\ell=1}^{k-1} a_\ell \theta^\ell + x$$

where x denotes an integer secret, a_1, \dots, a_{k-1} are random coefficients that are uniformly distributed in the finite field of $\{0, 1, \dots, M - 1\}$, and M is a prime number larger than x . By setting $\theta = 1, \dots, n$, the data owner obtains $\{p_1^\Pi, \dots, p_n^\Pi\}$, where

$$p_\theta^\Pi = p(\theta) \bmod M.$$

At least k secret shares are required to reconstruct the secret x by

$$x = \sum_{\ell=1}^k p_\ell^\Pi \prod_{\nu=0}^{k-1} \frac{\nu}{\nu - \ell}.$$

SS offers a scalable, robust, and secure approach to safeguarding sensitive information. One of its key advantages is the ability to reconstruct the original secret data from a predetermined number of, not necessarily n , shares. This feature becomes particularly valuable in scenarios where nodes fail due to various reasons, as the secret can still be recovered from the remaining valid shares. Additionally, SS is often combined with other cryptographic techniques, such as encryption [105], thereby providing an additional layer of security.

Next, we use an example to illustrate how SS can be used to implement the model aggregation step, *i.e.*, (4b), in a privacy-preserving manner.

Example 4 (Privacy-preserving aggregation based on secret sharing [106]). Assume there are four agents, each with a secret value x_1, x_2, x_3 , and x_4 , respectively – for example, in (4) we have $x_i := x_i^{(t)} - \alpha \nabla f_i(x_i^{(t)})$. They want to compute the sum of their secret values, $\sum_{i=1}^4 x_i$, without revealing their individual secrets. To achieve this, they agree on a polynomial degree $k = 3$ and the number of shares $n = 4$.

The protocol works as follows: Each agent i constructs a random polynomial $p_i(\theta)$ of degree 3 in the following form:

$$p_i(\theta) = a_{i,3}\theta^3 + a_{i,2}\theta^2 + a_{i,1}\theta + x_i$$

where $a_{i,3}, a_{i,2}$, and $a_{i,1}$ are randomly chosen coefficients, and x_i is the agent's secret value. Each agent i computes the values of their polynomial $p_i(\theta)$ for

$\theta = 1, 2, 3, 4$, and shares $p_i(j)$ with agent j , respectively. After receiving the shares $p_j(i)$ from all other agents j , each agent i computes the sum of the received shares: $\sum_{j=1}^n p_j(i)$. Each agent i sends the sum of shares $\sum_{j=1}^n p_j(i)$ to all other agents. At this point, each agent knows the values of the sum polynomial $p(\theta) = \sum_{i=1}^n p_i(\theta)$ at points $\theta = 1, 2, 3, 4$, which are:

$$p(\theta) = \left(\sum_{i=1}^n a_{i,3} \right) \theta^3 + \left(\sum_{i=1}^n a_{i,2} \right) \theta^2 + \left(\sum_{i=1}^n a_{i,1} \right) \theta + \sum_{i=1}^n x_i.$$

Since each agent knows four points on the sum polynomial $p(\theta)$, which has degree 3, they can use polynomial interpolation to recover the coefficients of $p(\theta)$, including the constant term $\sum_{i=1}^n x_i$, which is the desired sum of their secret values.

A comparison of the three cryptographic tools is presented in Table 2. DP is designed to prevent data leakage from trained models that are disclosed. Encryption and secret sharing, instead, have the goal of obscuring the disclosed model itself. It is important to note that these tools are complementary and can be combined to further enhance privacy protection; see Section 3.5 for more discussions.

In the following sections, we review how the cryptographic tools of the previous section have been applied to prevent data leakage in decentralized optimization and learning.

3.2. Differential privacy-based algorithms

We categorize the existing DP-based decentralized optimization algorithms based on their communication topologies. Following this categorization, we present a discussion on privacy amplification measures for DP-based algorithms, which leverage specific components of an algorithm, *e.g.*, quantization, to enhance privacy.

3.2.1. Federated topology

For federated learning with heterogeneous data, the authors in [107] developed a personalized linear model training algorithm with DP. In [108], general models were considered. In particular, the subsampling of users' local data was explicitly used to amplify the DP guarantee and improve utility. [109] proposed locally differentially private algorithms for federated learning with strongly convex and composite objectives. These algorithms employed the Gaussian mechanism, with noise variance dynamically adjusted over time to enhance the trade-off between accuracy and privacy. For optimization problems where each agent holds its own constraint and collaboratively minimizes an objective

that is a function of the average of individual variables, [110] perturbed the gradients used in the projected gradient descent step performed by each agent. The algorithm featured a step-size of $\Theta(1/\sqrt{\epsilon})$, and the utility loss was on the order of $O(\sqrt[4]{dn^2/\epsilon})$, where n denotes the number of agents and d denotes the dimension of the decision variable.

3.2.2. Peer-to-peer topology

We review DP-based distributed algorithms operated over peer-to-peer topologies, dividing them based on the objective that they accomplish: consensus or optimization. It is worth mentioning that, in the consensus literature, existing works usually aim to protect the initial states of agents, while in the distributed optimization literature, the focus is different. Some privacy-preserving distributed optimization works concentrate on protecting datasets, which is particularly relevant in machine learning problems. Alternatively, other works in the distributed optimization domain emphasize protecting the individual cost functions of the agents.

Consensus. [111] studied the private iterative consensus problem, where agents are required to converge while protecting the privacy of their initial values from honest but curious adversaries. The Laplace mechanism was used to achieve DP. Similarly, [112] injected Laplace noise into the consensus-seeking process to attain DP. However, they adjusted the consensus protocol so that the proposed algorithm converges almost surely to an unbiased estimate of the average of agents' initial states. [113] proposed a privacy-preserving distributed dynamic average consensus with noise-adding, which ensures DP. [114] developed a differentially private bipartite consensus algorithm operated over signed networks. The noise variance and step-size¹ were jointly designed to achieve asymptotically unbiased bipartite consensus. [115] introduced pairwise network differential privacy, a relaxation of local differential privacy that captures the notion that the privacy leakage from one node to another may depend on their relative positions in the graph. The authors analyzed the combination of local noise injection with gossip averaging protocols on fixed and random communication graphs, providing theoretical guarantees for the privacy and utility trade-offs achieved by these algorithms. In the context of average consensus of positive systems, [116] proposed a novel differentially private randomized mechanism that

¹The step-size refers to the weight assigned to neighboring information in each round.

Table 2: A comparison of main cryptographic tools.

Cryptographic tools	Utility	Privacy guarantee	Computation and communication load	Feature	Representative references
DP	inexact	statistical indistinguishability	low	easy implementation	[82, 83]
HE	exact	computational indistinguishability	high	operations on encrypted data	[99]
SS	exact	computational indistinguishability	medium	resilient to dropouts	[81]

perturbs the value using truncated Gaussian noise in a multiplicative manner. They analyzed the algorithm's performance in terms of an accuracy metric and quantified its DP guarantee. A comparison of DP-based consensus algorithms is provided in Table 3.

Distributed optimization. For distributed optimization problems, [117] proposed a differentially private distributed gradient descent algorithm that perturbs the local output with Laplace noise. The algorithm employed a linearly decaying step-size, ensuring that the sensitivity also decreases linearly. The prescribed differential privacy parameter ε was decomposed into a sequence $\{\varepsilon_t\}_{t \geq 1}$ such that $\sum_{t=1}^{\infty} \varepsilon_t = \varepsilon$, allowing each iteration t to be ε_t -DP. However, this choice of a linearly decaying step-size significantly slowed down the convergence rate and led to a utility loss of order $O(d/\varepsilon^2)$, where d represents the dimension of the decision variable. Along this line of research, several works extended differentially private distributed optimization algorithms to handle time-varying objective functions [118, 119, 120].

[121] incorporated a gradient-tracking mechanism into the differentially private distributed optimization algorithm, enabling it to achieve a linear convergence rate. [122] developed a privacy-preserving scheme based on the robust gradient-tracking distributed optimization algorithm introduced in [123]. The exchanged variable was updated with carefully calibrated Laplace noise to achieve DP. Moreover, [122] employed the SS technique, where each individual state was split into two shares. Only the share whose update does not directly depend on the local gradient was exchanged with neighbors, thereby enhancing privacy protection. Other existing works based on SS will be reviewed later in Section 3.4. Inspired also by [123], [124] developed a privacy-preserving distributed algorithm that supports both constant and decreasing step-sizes. To attenuate the noise effect while ensuring DP, the authors in [125] constructed topology-aware noise. With this method, each agent perturbed the messages to its neighbors (includ-

ing itself) with different perturbations whose weighted sum was zero. [126] presented an impossibility result regarding the simultaneous achievement of DP and exact accuracy in distributed optimization. The authors demonstrated that $(\varepsilon, 0)$ -DP cannot be achieved by the Laplace mechanism when employing diminishing step-sizes. Furthermore, they quantified the accuracy loss and DP parameters when utilizing linearly decaying step-sizes and noise variances. In [127], a novel algorithm was developed where each agent maintains two models at each step: a private model and a proxy model. Communication with neighboring agents involved only the proxy model, which was trained with DP noise. The training losses included a KL divergence loss between the private and proxy models, allowing the private model to benefit from the proxy model's updates.

Apart from perturbing the transmitted information, there are also works focusing on perturbing cost functions. In contrast to cases involving private data samples, this setup considers the cost as the private information. Therefore, to define DP according to Definition 1, sets of cost functions are considered. In this context, [128] demonstrated the impossibility of achieving DP by perturbing the inter-agent messages with noise when the underlying noise-free dynamics are asymptotically stable. Motivated by this limitation, this work established a general framework for handling functional data, decomposing the objective functions into an infinite sequence of coefficients corresponding to the elements of an orthogonal basis in a separable Hilbert space and injecting noise into the infinite coefficient sequence. DP can be achieved by solving the distributed optimization problem defined by the perturbed local costs.

For composite empirical risk minimization problems, the ADMM and dual averaging were used to design distributed algorithms with DP in [129, 130] and [131], respectively. In particular, the authors in [131] incorporated an agent subsampling procedure, *i.e.*, partial participation, to enhance the privacy-accuracy trade-off in distributed optimization. For distributed stochastic

Table 3: A comparison of DP-based consensus algorithms.

[Ref.]	Problem	Perturbed term	Noise & variance change	Step-size [†]	Composition [‡] range	Utility loss
[111]	average consensus	states	Laplace, decreasing	constant	∞	$O(1/\sqrt{n}\epsilon)$ in accuracy
[112]	average consensus	states	Laplace, decreasing	constant	∞	exact expected value of convergence, $O(1/\sqrt{n}\epsilon)$ in accuracy
[113]	dynamic average consensus	states	Laplace, increasing	decreasing	∞	-
[114]	bipartite consensus	states	Laplace, increasing	decreasing	∞	-
[116]	positive consensus	states	multiplicative Gaussian, constant	decreasing	T	-
[115] (Algs. 1-2)	gossip averaging	states	Gaussian constant	constant	T	$O(1/n^2\epsilon)$ in consensus error

[†] The step-size refers to the weight assigned to neighboring information in each round.

[‡] The composition range indicates the number of iterations considered when calculating privacy loss.

optimization with convex or non-convex costs, [132] employed ternary quantization for the communication among agents and proved that the quantized distributed optimization algorithm preserves $(0, \delta)$ -DP per iteration. [133] considered non-convex problems and developed a distributed optimization algorithm with DP that avoids convergence to local maxima and saddle points. [134] proposed two gradient-based algorithms for differentially private distributed optimization. These algorithms can guarantee a finite privacy budget, *i.e.*, ϵ in Definition 1, even when the number of iterations approaches infinity, under certain conditions. A comparison of DP-based distributed optimization algorithms is given in Table 4.

Resource allocation. For a class of resource allocation problems with equality constraints, [135] developed a privacy-preserving distributed algorithm based on the Laplace mechanism. Considering a similar setup, [136] proposed a differentially private deviation tracking algorithm. The authors established the linear convergence of their algorithm under suitable assumptions.

Game. Inspired by [117], [137] perturbed the transmitted messages in distributed Nash equilibrium seeking with linearly decaying step-size and Laplacian noise. The authors in [138] developed a distributed algorithm with DP for stochastic aggregative games. However, (ϵ, δ) -DP was proved only for each iteration. [139] generalized function perturbation to the game setting and

proposed a Laplace linear quadratic functional perturbation algorithm. This mechanism maintains the concavity property of the perturbed payoff functions and thus ensures the existence of Nash equilibrium of the perturbed game. Recently, [140] developed a fully distributed differentially private algorithm to achieve both rigorous DP and guaranteed computation accuracy of the Nash equilibrium.

Privacy amplification for DP-based algorithms. The privacy guarantees offered by DP-based algorithms that rely on noise injection can be further enhanced through the incorporation of other (randomized) mechanisms. For instance, the synergistic effect of combining noise injection with compressed communication has been explored in [141, 142, 143, 144]. Another direction is the integration with agent sampling procedures, as proposed in [131]. By randomly selecting a subset of agents to participate in each iteration, this approach can effectively amplify the DP guarantee. Furthermore, the local training scheme (see Section 2.2), a common component in federated learning settings, has been shown to improve privacy guarantees, as discussed in [145].

3.3. Homomorphic encryption-based algorithms

In this section, we present an overview of privacy-preserving decentralized algorithms based on HE.

3.3.1. Federated topology

For quadratic optimization problems, where the linear term in the objective and the constant in the con-

Table 4: A comparison of DP-based distributed optimization algorithms.

[Ref.]	Private information	Perturbed term	Privacy-preserving mechanism	Step-size	Composition range	Utility loss [‡]
[115] (Alg. 3)	datasets	variables	Gaussian with constant VAR [†]	constant	T	$O(d/\mu^2\epsilon)$ for S.C. [*]
[117]	costs	variables	Laplace with decreasing VAR	decreasing	∞	$O(d/\epsilon^2)$ for S.C.
[121]	costs	variables	Laplace with decreasing VAR	constant	∞	$O(d/\epsilon^2)$ for S.C.
[122]	costs	decomposed variables	Laplace with constant VAR	constant	T	-
[124]	costs	variables	Laplace with constant VAR	constant or decreasing	∞	-
[125]	costs	variables	Laplace with constant VAR	constant	T	-
[126]	costs	variables	Laplace with decreasing VAR	summable	∞	-
[132]	gradients	variables	ternary quantization	decreasing	1	-
[133]	datasets	gradients	Gaussian with constant VAR	decreasing	1	-
[134]	costs	variables	Laplace with increasing VAR	decreasing	∞	-
[131]	datasets	gradients	Gaussian with constant VAR	decreasing	T	$O(\sqrt{d}/\epsilon)$ for C. $O(d^2/\epsilon^2)$ for S.C.
[128]	costs	costs	Laplace with decreasing VAR	square-summable	∞	$O(n/\epsilon)$ for S.C.

[†] VAR represents variance.

[‡] For utility loss, the abbreviations C. and S.C. refer to convex and strongly convex functions, respectively.

* The variable ι represents the rate for agent sampling [131].

straints are private, [146] designed a privacy-preserving cloud-based optimization algorithm based on additively homomorphic encryption. A comprehensive theoretical analysis of this algorithm was provided later by [147]. In the presence of a coordinator, [148] proposed two privacy-preserving optimization algorithms with private and public keys, respectively, ensuring privacy security against input-output inference for quadratic functions. Considering a similar setup, [149] developed a privacy-preserving optimization algorithm for problems with coupled costs and constraints.

HE has been actively utilized to protect privacy in machine learning over federated topologies. In [104], the authors employed additive homomorphic encryption for exchanging models and gradients between the coordinator and agents. This approach enables privacy preservation while retaining the training accuracy. The authors in [150] proposed to encrypt a batch of quan-

tized gradients from individual agents to boost computation speed and communication efficiency for federated learning with HE.

3.3.2. Peer-to-peer topology

Early attempts for multi-agent consensus on encrypted values were reported in [151, 152]. Specifically, [152] proposed a distributed gossip algorithm that operates on encrypted values consistent with a cryptosystem initiated by a trusted third-party. An encrypted consensus scheme enabling individual agents to recover the decrypted consensus value in a fully distributed manner was later introduced in [102]. For directed networks, [153] presented a variant, while [154] developed a fundamental consensus scheme on integer values and leveraged it to design a homomorphically encrypted consensus protocol. [155] addressed the scenario where all immediate neighbors are untrusted, potentially compromising the initial states of agents even with HE-based

consensus protocol in [102]. To mitigate this risk, they proposed the use of the Paillier cryptosystem to replace the initial states with virtual ones. A specific application of privacy-preserving consensus in directed networks to the economic dispatch problem in power grids was reported in [156].

[157] proposed a privacy-preserving distributed optimization algorithm based on ADMM and an additively homomorphic encryption scheme. This work was extended to constrained problems in [158], where the privacy-preserving algorithm was based on the distributed projected gradient descent [159]. The authors in [160] adopted the Paillier encryption scheme to generate zero-sum perturbations for local objectives, and developed a privacy-preserving distributed algorithm with DP guarantee. [161] proposed a privacy-preserving distributed ADMM algorithm for constrained quadratic problems, where a special key switching functionality was employed to enable secure computations. The authors also discussed the application of their proposed algorithm to the formation control problem.

To provide a comprehensive overview, a detailed comparison of HE-based distributed algorithms is presented in Table 5.

3.4. Secret sharing-based algorithms

Next, we review SS-based privacy-preserving decentralized algorithms by categorizing them based on their underlying communication topologies.

3.4.1. Federated topology

[105] proposed a privacy-preserving aggregation method for federated learning, combining masking and the (n, p) -threshold SS scheme. Specifically, each node adds a pairwise additive mask to its state for obfuscation, while ensuring that the sum of the original states is preserved. The masks are generated randomly using a shared seed, which is protected by the (n, p) -threshold SS scheme. The incorporation of the (n, p) -threshold SS scheme serves two purposes: first, it protects the pairwise seed used to generate the masks, and second, it provides resilience against node failures. [162] developed a multi-secret sharing scheme that could simultaneously share multiple secrets based on the finite field Fourier transform [163], easing the computation cost in massive-scale federated learning.

3.4.2. Peer-to-peer topology

For general peer-to-peer communication topologies, SS schemes have been successfully employed in designing privacy-preserving algorithms for average consensus and distributed optimization.

Consensus. [164] proposed a privacy-preserving average consensus approach where each agent splits its state into two shares and transmits only one of them to its neighbors. The key to preserving average consensus lies in locally updating the untransmitted share in accordance with an augmented graph topology. Extensions to dynamic average consensus and push-sum consensus can be found in [165] and [166], respectively. Similar methods that mask the true state by adding deterministic offsets were discussed in [167, 168, 169]. A more comprehensive study on SS-based private consensus was reported in [170], where polynomials were used to generate shares and conditions on the privacy degree for achieving average consensus were investigated. With a common privacy degree lower than the number of neighbors, the consensus protocol exhibits resilience to node failures, a feature unique to this approach.

Distributed optimization. In the realm of general distributed optimization, the authors of [171] proposed a novel approach to obfuscate individual objective functions by adding deliberately designed affine functions, thereby preserving the global objective function unchanged. For this algorithm, a formal statistical guarantee was subsequently provided in [172]. Alternatively, [173] considered a more general decomposition of local objective functions and enforced equivalence with the original problem by introducing consensus constraints. In the context of distributed optimization with applications to energy resource control, [174] combined SS and the projected gradient method to design a privacy-preserving algorithm, demonstrating the versatility of these techniques across diverse domains.

To summarize, a comparison of SS-based algorithms is provided in Table 6.

3.5. Discussions

In this section, we discuss the fundamental trade-off among privacy, accuracy, and efficiency of privacy-preserving decentralized algorithms. Then, we review a few notable works that leverage the combination of multiple cryptographic techniques and also that protect privacy without relying on any of these three cryptographic tools.

Privacy-accuracy-efficiency trade-off. There exists a trade-off among privacy, accuracy, and efficiency when leveraging different cryptographic techniques for privacy-preserving decentralized algorithm design. DP-based algorithms offer rigorous privacy guarantees and are computationally and communication-wise efficient. However, they inherently degrade the accuracy of the

Table 5: A comparison of HE-based algorithms.

[Ref.]	Peer-to-peer topology	Problem	Private information	HE properties	
				Homomorphism	Key for agents
[146]	✗	cloud-based QP	problem parameters	additive	common
[148] (Alg. 1)	✗	cloud-based constrained optimization	variables and constraints	full	common
[148] (Alg. 2)	✗	cloud-based optimization for affine objectives	variables and constraints	additive	individual
[149]	✗	cloud-based constrained optimization	primal and dual variables	additive	common
[104]	✗	federated learning	gradients	additive	common
[150]	✗	federated learning	quantized gradients	additive	common
[151]	✓	consensus	variables	additive	individual
[152]	✓	consensus	variables	additive	common
[102]	✓	consensus	initial variables	additive	individual
[153]	✓	consensus over directed networks	initial variables	additive	individual
[156]	✓	economic dispatch	exchanged variables	additive	individual
[157]	✓	distributed optimization	variables and function	additive	individual
[158]	✓	distributed constrained optimization	variables and function	additive	individual
[161]	✓	distributed constrained QP and cooperative control	variables and control objective	additive	individual

decision or model, as there is a fundamental trade-off between DP and accuracy [94]. Conversely, HE and SS-based algorithms can preserve accuracy while providing computational indistinguishability as the privacy guarantee. Nevertheless, decentralized algorithms with (fully) HE incur significant additional computational overhead. SS-based algorithms, on the other hand, typically require agreement on the degree of polynomials and the number of shares, which can lead to high communication costs in practical scenarios.

Synthesis of cryptographic tools in designing distributed algorithms. The cryptographic tools introduced in Section 3.1 are orthogonal, and they can be combined to strengthen the privacy protection. Indeed, research studies that employ multiple cryptographic tools were reported in [160] and [122]. In particular, [160] employed the Paillier cryptosystem to secretly construct zero-sum functional perturbations for privacy-preserving distributed optimization. This approach enables the overall algorithm to achieve DP with guarantees. As discussed in Section 3.2, the work by [122] presented an approach that seamlessly integrated SS and DP techniques for enhancing privacy protection.

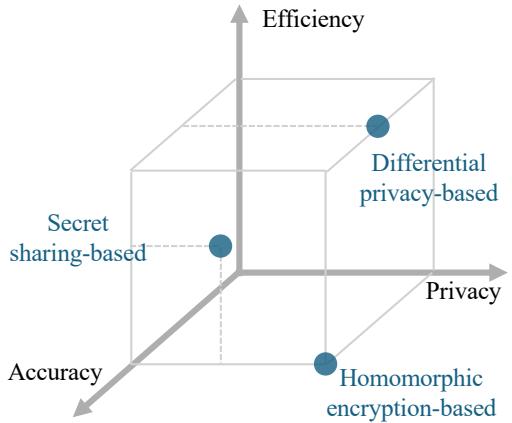


Figure 5: An illustration of the privacy-accuracy-efficiency trade-off.

Distributed algorithms with other privacy-preserving mechanisms. Several notable privacy-preserving distributed algorithms used other measures to prevent privacy leakage. For example, [175] proposed a privacy-preserving average consensus protocol that involves adding random noise to the states. The noise is generated by taking the difference between two realizations of

Table 6: A comparison of SS-based algorithms.

[Ref.]	Peer-to-peer topology	Problem	SS properties		Resilience to dropouts
			Number of shares	Construction	
[105]	✗	federated learning	$n \geq 2$	polynomial	✓
[162]	✗	federated learning	$n \geq 2$	fast Fourier transform	✓
[169]	✓	average consensus	$ \mathcal{N}_i $	zero-sum deterministic offsets	✗
[164]	✓	average consensus	2	state decomposition	✗
[165]	✓	dynamic consensus	2	state decomposition	✗
[166]	✓	directed consensus	2	state decomposition	✗
[170]	✓	average consensus	$n \geq 2$	polynomial	✓
[171]	✓	distributed optimization	$ \mathcal{N}_i $	zero-sum deterministic objective perturbation	✗
[173]	✓	distributed optimization	2	objective decomposition	✗
[122]	✓	distributed optimization	2	state decomposition	✗
[174]	✗	distributed optimization	$n \geq 2$	polynomial	✗

a Gaussian distribution with linearly decaying weights at consecutive time instants. [176] designed two distributed optimization algorithms, one with a constant step-size and another with a diminishing step-size. They derived theoretical conditions under which a certain degree of privacy could be achieved. [177] developed a novel and general subspace perturbation method for distributed optimization. This work maintains a certain level of privacy by bounding the local information leakage, as measured by mutual information. It ensures exact convergence to the optimum without compromising privacy.

4. Resilient decentralized optimization

The goal of designing resilient decentralized algorithms is to prevent significant accuracy degradation of the decision or model in the presence of any attacks on the algorithm. The possible threat models depend on the architecture of the network. In this section, we begin by presenting the attack model and introducing some notations. Then, we discuss how the objectives in decentralized optimization and consensus-seeking problems need to be reformulated when attackers are present. This is followed by resilient aggregation and consensus mechanisms, which serve as the key module to achieve resilience in federated and distributed algorithms, respectively. Next, we review some metrics of resilience: aggregation resilience, graph robustness, and cost redundancy. Following this, we provide an overview and a detailed comparison of resilient decentralized optimization algorithms in the literature.

Attackers in practical scenarios may have different objectives, such as degrading the model’s accuracy or manipulating the collaborative decision (*e.g.* biasing it). To comprehensively address these threats, we consider two general attack models: the Byzantine attack model [178] and the malicious attack model. In the Byzantine attack model, only the honest agents correctly follow the algorithm, while the faulty agents may exhibit arbitrary and potentially *different* behaviors to different neighbors. On the other hand, in the malicious attack model, the faulty agents are restricted to send the *same* message to all of their neighbors, although this message may be arbitrary and differ from the intended algorithm. In both models, the faulty agents could collude among themselves to decide on deceptive values to be communicated, further complicating the mitigation of such attacks. We highlight that a minimal set of requirements are imposed in these two models to constrain the adversaries. Therefore, such attack models have been widely deployed in consensus [179, 180, 181], federated learning [182], and distributed optimization [183], to ensure broad applicability of the established results.

We denote by \mathcal{F} the set of faulty agents, and by \mathcal{H} the collection of honest agents. It is clear that $\mathcal{H} \cap \mathcal{F} = \emptyset$ and $\mathcal{H} \cup \mathcal{F} = \mathcal{V}$, where \mathcal{V} represents the set of all agents. As mentioned above, the attack model depends on the network architecture, *i.e.*, federated or peer-to-peer topology; thus we distinguish the following models²:

- *n_f-total attack*: There are at most n_f faulty agents

²The n_f -total attack model has been used also in the peer-to-peer scenario to simplify analysis.

in the network. That is, $|\mathcal{F}| \leq n_f$.

- n_f -local attack: There are at most n_f faulty agents in the in-neighborhood of any agent. That is, $|\mathcal{F} \cap \mathcal{N}_i^-| \leq n_f$, for any agent $i \in \mathcal{V}$.

4.1. Objective in the presence of attackers

In the presence of adversarial agents in multi-agent systems, the goals for honest agents need to be adjusted.

Consensus. In the multi-agent consensus problem, agreeing on the mean of the initial states from all agents, including the adversarial ones, is neither feasible nor sensible. Therefore, in resilient distributed consensus, the goal for the honest agents is to reach a consensus value within the convex hull formed by the initial states of the regular (non-adversarial) agents. Formally, the goal of the resilient consensus is as follows. Design a decentralized protocol which outputs $x_i^{(t)}$, $i \in \mathcal{H}$, $t \in \mathbb{N}$ starting from $x_i^{(0)}$, and such that:

- *Safety:* Let $X^{(0)} = \{x_1^{(0)}, \dots, x_{|\mathcal{H}|}^{(0)}\} \subset \mathbb{R}^d$ be the set of initial states of honest agents, then at each time t , and for any honest agent i , its value $x_i^{(t)}$ should be in the convex hull of $X^{(0)}$;
- *Agreement:* As t goes to infinity, it holds for any honest agent i that $\lim_{t \rightarrow \infty} x_i^{(t)} = \bar{x}$ for some $\bar{x} \in \mathbb{R}^d$.

Notice that if the safety objective is guaranteed, then \bar{x} belongs to the convex hull of $X^{(0)}$.

Decentralized optimization over federated or peer-to-peer topology. For the finite-sum optimization problem (2), solving the original problem in the presence of adversarial agents is generally not possible. Instead, a more reasonable goal is to approximate a minimizer of the following cost function:

$$f_{\mathcal{H}}(x) = \frac{1}{|\mathcal{H}|} \sum_{i \in \mathcal{H}} f_i(x). \quad (19)$$

This objective function considers only the cost functions of the honest agents, effectively excluding the adversarial agents' contributions.

Resource allocation. Similarly, for the resource allocation problem in (15), the problem for honest agents becomes

$$\min_{x_i \in \mathbb{R}^{d_i}, \forall i \in \mathcal{H}} \sum_{i \in \mathcal{H}} f_i(x_i) \text{ s.t. } \sum_{i \in \mathcal{H}} g_i(x_i) \leq 0 \quad x_i \in X_i, \quad \forall i \in \mathcal{H}.$$

In this new problem formulation, the cost functions and coupling constraints are modified to only account for contributions from honest agents.

Throughout Section 4, we assume the above shifted objectives as default for resilient decentralized optimization and consensus. We note that solving the modified problems remains challenging when the identities of Byzantine agents are unknown. Consequently, existing works have typically focused on developing approximate solution methods rather than seeking exact solutions. It is noteworthy that a few recent resilient algorithms incorporate mechanisms to proactively identify attackers based on the information received during each communication round, such as reputation scoring [184]. As a result, these algorithms can potentially achieve improved performance.

4.2. Resilient aggregation and consensus

Averaging is a major step in decentralized optimization and learning algorithms over both federated and peer-to-peer topologies. To achieve resilience against Byzantine agents, standard averaging can be replaced by resilient aggregation and consensus rules. This section begins by introducing a set of common resilient aggregation rules applicable to federated topologies, followed by several useful pre-aggregation techniques. Subsequently, it reviews resilient consensus protocols over peer-to-peer topologies, emphasizing how the safety condition can be guaranteed. Table 7 presents a comparison of resilient aggregation and consensus rules. Finally, it presents various metrics of resilience, including aggregation resilience expressed as a concentration inequality, as well as the notions of graph robustness and cost redundancy.

4.2.1. Resilient aggregation over federated topology

We begin with the basic aggregation rules and then introduce two pre-aggregation techniques.

Coordinate-wise median (CWMed) [185]. Given a set of n vectors $X = \{x_i \in \mathbb{R}^d\}_{i=1}^n$, the CWMed of x_1, \dots, x_n , denoted by $\text{CWMed}(X)$, is defined as

$$\text{CWMed}(X) = \operatorname{argmin}_{x \in \mathbb{R}^d} \sum_{i=1}^n \|x - x_i\|_1.$$

Coordinate-wise trimmed mean (CWTM) [185]. Given a set of n vectors $X = \{x_i \in \mathbb{R}^d\}_{i=1}^n$, and the largest number of faulty nodes $n_f < n/2$. Denote by $[x_i]_k$ the k -th coordinate of x_i , and by τ_k the permutation on $[n]$ that sorts the k -th coordinate of x_i , $\forall i$ in a non-decreasing order, i.e., $[x_{\tau_k(1)}]_k \leq [x_{\tau_k(2)}]_k \leq \dots \leq [x_{\tau_k(n)}]_k$. The

Table 7: A comparison of resilient aggregation and consensus rules.

Aggregation rules	Dimension	Safety guarantee	Complexity	Number of values removed
coordinate-wise median	≥ 1	✗	$O(nd)$	0
coordinate-wise trimmed mean	≥ 1	✗	$O(dn \log(n))$	$2n_f$
Krum	≥ 1	✗	$O(n^2(d + \log(n)))$	n_f
geometric median	≥ 1	✗	-	0
centered clipping	≥ 1	✗	$O(nd)$ per step	0
mean subsequence reduced	1	✓	$O(n \log(n))$	up to $2n_f$
Tverberg partition-based	≥ 1	✓	NP	-
centerpoint-based	≥ 1	✓	$O(n)$ for $n = 2$ $O(n^2)$ for $n = 3$	-

CWTM of x_1, \dots, x_n , denoted by CWTM(X), is a vector in \mathbb{R}^d whose k -th coordinate is

$$[\text{CWTM}(X)]_k = \frac{1}{n - 2n_f} \sum_{j \in [n_f+1, n-n_f]} [x_{\tau_k(j)}]_k.$$

Krum [186]. Given a set of n vectors $X = \{x_i \in \mathbb{R}^d\}_{i=1}^n$, and the largest number of faulty nodes $n_f < n$. The output of Krum is a vector $x \in X$ that is closest to its neighbors upon pruning the n_f farthest ones. Let C_i be the set of $n - n_f - 1$ closest vectors to x_i . Then, Krum outputs the vector that has the smallest sum of distances to its $n - n_f$ closest neighbors, *i.e.*,

$$\text{Krum}(X) = x_{i^*} \quad \text{where} \quad i^* = \operatorname{argmin}_{i \in [n]} \sum_{j \in C_i} \|x_i - x_j\|^2.$$

Geometric median (GM) [187]. Given a set of n vectors $X = \{x_i \in \mathbb{R}^d\}_{i=1}^n$, their geometric median, denoted by GM(X), is defined to be a vector that minimizes the sum of Euclidean distances to all the vectors, *i.e.*,

$$\text{GM}(X) = \operatorname{argmin}_{x \in \mathbb{R}^d} \sum_{i=1}^n \|x - x_i\|.$$

In general, there exists no closed form for GM. However, an approximate solution can be derived by multiple iterations of the smoothed Weiszfeld's algorithm [188], each of which has a time complexity of $O(nd)$.

Centered clipping (CC) [189]. Let $X = \{x_i \in \mathbb{R}^d\}_{i=1}^n$ be the set of vectors to be aggregated. Starting from some point v_0 , one iteratively computes a sequence of vectors $v_m, m \in [M]$ according to

$$v_m = v_{m-1} + \frac{1}{n} \sum_{i=1}^n (x_i - v_{m-1}) \min \left\{ 1, \frac{\tau_m}{\|x_i - v_{m-1}\|} \right\}$$

where $\tau_m \geq$ is a clipping parameter. Then, CC(X) = v_M .

CC is an iterative algorithm, and it can be considered scalable if the number of iterations M is small enough [189]. In particular, each CC update has a time complexity of $O(nd)$. However, the value of M that leads to a satisfactory aggregation depends on the input vectors X and the clipping parameter.

The aggregation mechanisms listed are not exhaustive. There are other aggregation rules, such as the Minimum Enclosing Ball with Outliers [190], as well as variants that are modifications or combinations of the typical ones listed above.

We remark that average aggregation, which is shown to be fragile to attacks, can be formulated as the following optimization problem

$$\text{Avg}(X) = \operatorname{argmin}_{x \in \mathbb{R}^d} \sum_{i=1}^n \|x - x_i\|^2.$$

We can see that Avg, CWMed, and GM solve a similar problem but use different norms to define the objective function. The robustness of CWMed and GM stems from the resilience of their norms to outliers, in contrast to the squared norm used by Avg, which is not robust to outliers.

To further improve the performance of resilient aggregation rules, pre-processing techniques can be employed. The idea is to transform the vectors to be aggregated into a new set, to which resilient aggregation is then applied.

Bucketing [191]. The bucketing procedure finds its root in the median-of-means [192]. Given a set of n vectors $X = \{x_i \in \mathbb{R}^d\}_{i=1}^n$ and an integer parameter $s \geq 1$, the s -bucketing procedure generates $\lceil n/s \rceil$ vectors, where each vector is produced by the following steps:

- 1) Pick random permutation π of $[n]$;

- 2) Compute the following quantity

$$y_i = \frac{1}{s} \sum_{j=(i-1)s+1}^{\min\{n, i \cdot s\}} x_{\pi(j)}.$$

Nearest neighbor mixing (NNM) [193]. Given a set of n vectors $X = \{x_i \in \mathbb{R}^d\}_{i=1}^n$, and the largest number of faulty nodes $n_f < n/2$. NNM outputs $Y = \{y_i \in \mathbb{R}^d\}_{i=1}^n$, where each y_i is generated according to the steps:

- 1) Sort the vectors in X to get $(x_{i:1}, \dots, x_{i:n})$ such that

$$\|x_{i,1} - x_i\| \leq \dots \leq \|x_{i,n} - x_i\|;$$

- 2) Average the $n - n_f$ nearest neighbors of x_i , i.e.,

$$y_i = \frac{1}{n - n_f} \sum_{j=1}^{n-n_f} x_{i:j}.$$

Pre-aggregation techniques show improvements in both theory and practice when combined with most resilient aggregation mechanisms. Among these, NNM requires a higher computational load compared to bucketing.

4.2.2. Resilient consensus over peer-to-peer topology

Although resilient consensus in a peer-to-peer topology also requires each agent to aggregate messages from its neighbors, the resilient aggregation rules for federated topology reviewed in the previous section are not directly applicable for two reasons. First, in federated topology, resilient aggregation is typically performed by a coordinator that does not produce a model based on its own data. In contrast, agents in resilient consensus generate updates independently, which must be utilized. Second, the resilient aggregation rules do not meet the safety requirements in consensus as described in Section 4.1. In this section, we introduce resilient consensus protocols designed to address these two issues.

Mean subsequence reduced (MSR). Given a set of n scalars $X = \{x_i\}_{i=1}^n \subset \mathbb{R}$, a reference point $x_j \in X$, and the maximum number of faulty nodes $n_f < n/2$, MSR aims to remove some values from X based on their relationship with the reference point x_j and the value of n_f . The process is as follows:

- 1) Sort all values in X in a descending order;

- 2) If there are less than n_f values in X strictly larger than x_j , remove all values larger than x_j from X . If there are at least n_f values in X strictly larger than x_j , remove the n_f largest values from X (breaking ties arbitrarily);
- 3) If there are less than n_f values in X strictly smaller than x_j , remove all values smaller than x_j from X . If there are at least n_f values in X strictly smaller than x_j , remove the n_f smallest values from X (breaking ties arbitrarily).

Denote the set of removed values as R_j when taking x_j as the reference point. Then, MSR outputs the following scalar

$$\text{MSR}(X, x_j) = \frac{1}{|X \setminus R_j|} \sum_{i \in X \setminus R_j} x_i.$$

Safe point-based aggregation. The extension of MSR to the multi-dimensional case is highly nontrivial, as running MSR for each coordinate does not guarantee the safety condition in Section 4.1, as illustrated by the following example.

Example 5 (Resilient consensus of probability vectors [194]). Consider a group of 4 agents, where the honest agents 1, 2, 3 have initial states represented by the following probability vectors:

$$x_1^{(0)} = \begin{bmatrix} a \\ b \\ b \end{bmatrix}, x_2^{(0)} = \begin{bmatrix} b \\ a \\ b \end{bmatrix}, x_3^{(0)} = \begin{bmatrix} b \\ b \\ a \end{bmatrix},$$

where $1 \geq a > b \geq 0$ and $a + 2b = 1$. The fourth agent is Byzantine. By running the MSR algorithm independently for each coordinate of the state, the honest agents end up agreeing on $\hat{x} = [b; b; b]$. Although each coordinate of \hat{x} satisfies the scalar safety condition along each dimension separately, \hat{x} does not fulfill the safety condition in the multi-dimensional case. To see this, observe that \hat{x} is not a probability vector because $3b < a + 2b = 1$. Consequently, \hat{x} is not contained within the convex hull of $x_1^{(0)}, x_2^{(0)}$ and $x_3^{(0)}$.

Motivated by this, the safe point-based aggregation strategy was developed. To proceed, we recall the definition of n_f -safe point.

Definition 2 (n_f -safe point). Given a set of n points in \mathbb{R}^d , of which at most n_f are adversarial, then a point in \mathbb{R}^d that is guaranteed to lie in the interior of the convex hull of $n - n_f$ normal points is called an n_f -safe point.

The idea of safe point-based aggregation mechanisms is that, at time step t , once an n_f -safe point $s_i^{(t)}$ is secured, each agent i updates its state by

$$x_i^{(t+1)} = x_i^{(t)} + \alpha_i^{(t)}(s_i^{(t)} - x_i^{(t)}),$$

where $\alpha_i^{(t)}$ is a possibly time-varying weight chosen from $(0, 1)$. Since both $x_i^{(t)}$ and $s_i^{(t)}$ are in the convex hull of states of honest agents, safety is guaranteed. Next, two strategies are introduced to obtain an n_f -safe point.

- *Tverberg partition*: The idea is to partition a set of n points in \mathbb{R}^d into $n_f + 1$ subsets such that the convex hulls of any two subsets intersect. The intersection of the convex hulls of all $n_f + 1$ subsets is called a Tverberg region. Any point lying in the interior of this Tverberg region is a n_f -safe point by construction [194]. However, computing a Tverberg partition is NP-hard in general [195]. Approximate algorithms can be employed to compute a partition [196], albeit at the cost of a stricter bound on the number of malicious agents [197]. Figure 6 provides an illustration of this technique.
- *Centerpoint*: The notion of a centerpoint can be viewed as a generalization of the median in higher-dimensional Euclidean space [198]. Formally, given a set X of n points in \mathbb{R}^d , a centerpoint is a point (not necessarily in X) such that any closed half-space of \mathbb{R}^d containing this point also contains at least $n/d+1$ points from X . Figure 7 provides an illustration of this technique. It was shown that an n_f -safe point is equivalent to an interior centerpoint if $n > n_f(d+1)$ [198]. For the special cases of $d=2$ and $d=3$, efficient computation methods for centerpoints were reported in [199] and [200], respectively. Their algorithms have time complexities of $O(n)$ and $O(n^2)$ for $d=2$ and $d=3$, respectively. For higher dimensions, [201] provided a general approximate method. In all cases, these centerpoint-based approaches achieve a better trade-off between resilience and computational complexity than Tverberg partition-based ones.

4.3. Metrics of resilience

Analyzing the resilience of decentralized algorithms from a theoretical standpoint requires the definition of suitable metrics, which we review in the following.

4.3.1. Aggregation resilience

To quantify aggregation resilience, we introduce the concept of concentration, which refers to the property that, for any subset of inputs of size $n - n_f$, the output of the aggregation rule is in close proximity to the average of these inputs [202, 203, 32].

Definition 3 (Concentration). For $n_f < n$, a resilient aggregation rule A is said to satisfy concentration if there exists a real-valued function Φ such that for any set of n vectors $X := \{x_i\}_{i=1}^n$ and any subset $S \subseteq [n]$ with $|S| = n - n_f$,

$$\mathbb{E} [\|A(X) - \bar{x}_S\|^2] \leq \mathbb{E} [\Phi(x_i, i \in S)], \quad (20)$$

where $\bar{x}_S := (n - n_f)^{-1} \sum_{i \in S} x_i$ and the expectation is taken over the randomness of the input vectors.

The concentration criteria in the literature differ from each other in the definition of proximity, *i.e.*, Φ . It serves as quantitative conditions and can therefore be used for analytical purposes. Next, we introduce three common concentration conditions, that is, (n_f, λ) -resilient averaging [202], (δ_{\max}, ζ) -agnostic robust aggregator (ARAgg) [203], and (n_f, κ) -robustness [193].

- (n_f, λ) -resilient averaging: $\mathbb{E} [\Phi(x_i, i \in S)] = \lambda \max_{i,j \in S} \|x_i - x_j\|$ where $\lambda \geq 0$ is a parameter.
- (δ_{\max}, ζ) -ARAgg: Suppose $n_f/n \leq \delta_{\max} < 0.5$ and $\mathbb{E} [\|x_i - x_j\|^2] \leq \rho^2$ for all $i, j \in S$. Let $\mathbb{E} [\Phi(x_i, i \in S)] = \zeta \rho^2 n_f/n$, where $\zeta \geq 0$ is a parameter.
- (n_f, κ) -robustness: $\mathbb{E} [\Phi(x_i, i \in S)] = \kappa |S|^{-1} \sum_{i \in S} \|x_i - \bar{x}_S\|^2$ where $\kappa \geq 0$ is a parameter.

4.3.2. Graph robustness

For algorithms applied over sparse network topologies, there is a close relationship between the topology and the maximum number of tolerable faulty agents. Graph robustness is a useful tool to reveal their coupling. It was first formalized in [181] and is recalled below:

Definition 4. (Graph robustness): A digraph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is r -robust, if for any pair of disjoint and nonempty subsets $\mathcal{V}_1, \mathcal{V}_2 \subsetneq \mathcal{V}$, at least one of the following statements hold:

1. There exists an agent in \mathcal{V}_1 such that it has at least r in-neighbors outside \mathcal{V}_1 ;
2. There exists an agent in \mathcal{V}_2 such that it has at least r in-neighbors outside \mathcal{V}_2 .

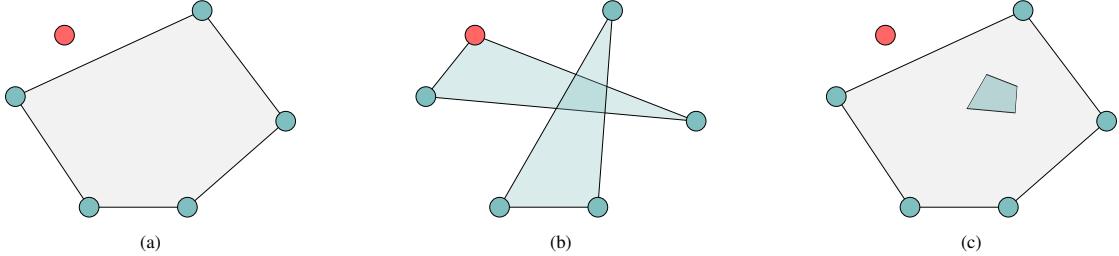


Figure 6: A depiction of Tverberg partition for the case with $n = 6$, $n_f = 1$, and $d = 2$. (a) plots the convex hull of the five points from honest agents; (b) presents a Tverberg partition consisting of two subsets, where only one of them contains adversarial agent and the intersection of the subsets is a Tverberg region; (c) illustrates that such Tverberg region is a subset of the convex hull of honest agents.

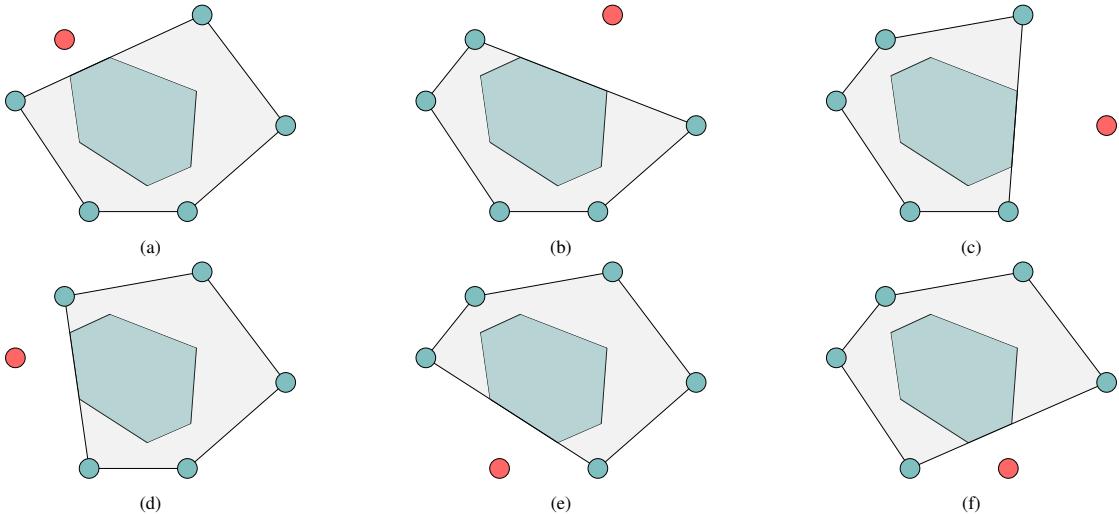


Figure 7: A depiction of the centerpoint region for the case where $n = 6$, $n_f = 1$, and $d = 2$. The region of centerpoints is shaded in teal. Any point within this teal region is also a 1-safe point, meaning that regardless of which one of the six agents is adversarial, every point in the teal region is guaranteed to lie within the convex hull formed by the remaining five honest agents.

Additionally, the digraph \mathcal{G} is (r, s) -robust if any statement among 1., 2. and 3. (below) holds:

3. There are no less than s agents in $\mathcal{V}_1 \cup \mathcal{V}_2$ such that each of them has at least r in-neighbors outside their respective sets.

Intuitively speaking, graph robustness is a measure of connectivity in graphs. It asserts that for any two disjoint and non-empty subsets of nodes, there exist a significant number of nodes within these subsets that have a sufficient number of incoming connections from nodes outside the subsets. In other words, graph robustness ensures that each subset of nodes is adequately connected to the rest of the graph, reducing the likelihood of disconnected components or isolated clusters. This requirement, along with resilient consensus protocols that eliminate a number of suspicious links, helps maintain connectivity and ultimately achieve agreement [181, 204].

4.3.3. Cost redundancy

Because the identities of Byzantine agents are unknown, solving (19) is generally impossible. To circumvent this issue, researchers have explored additional conditions on the cost functions. A line of work has proposed the notion of redundancy in cost functions for the solvability of (19), as introduced below.

Definition 5 ($2n_f$ -redundancy). For a given set of honest agents \mathcal{H} , their costs are said to satisfy $2n_f$ -redundancy if for any subset $S \subset \mathcal{V}$ with $|S| \geq n - 2n_f$,

$$\operatorname{argmin}_x \sum_{i \in S} f_i(x) = \operatorname{argmin}_x \sum_{i \in \mathcal{H}} f_i(x).$$

The $2n_f$ -redundancy property implies that the sum of the costs of any $n - 2n_f$ honest agents and the sum of the costs of all honest agents share common minimizers. [205] demonstrated that this property is a necessary condition for exactly solving (19) in the presence

of up to n_f Byzantine agents. However, this condition is challenging to fulfill. Motivated by this difficulty, [206] generalized it to the notion of $(2n_f, \epsilon)$ -redundancy.

Definition 6 $((2n_f, \epsilon)$ -redundancy). For $\epsilon \geq 0$, the agents' local cost functions are said to satisfy the (n_f, ϵ) -redundancy property if and only if for every pair of subsets of agents $S_1, S_2 \subseteq [n]$, where $|S_1| = n - f$, $|S_2| \geq n - 2f$ and $S_2 \subseteq S_1$,

$$\text{dist}\left(\arg\min_x \sum_{i \in S_1} f_i(x), \arg\min_x \sum_{i \in S_2} f_i(x)\right) \leq \epsilon.$$

where $\text{dist}(X, Y)$ denotes the Hausdorff distance between two sets X and Y , defined as

$$\text{dist}(X, Y) := \max \left\{ \sup_{x \in X} \inf_{y \in Y} \|x - y\|, \sup_{y \in Y} \inf_{x \in X} \|y - x\| \right\}.$$

[206] demonstrated that $(2n_f, \epsilon)$ -redundancy is both necessary and sufficient for a deterministic algorithm to output a point within a neighborhood around a true minimum point with radius ϵ .

In the following sections we survey the Byzantine-resilient algorithms that have been proposed in recent years. These methods employ the resilient aggregation techniques discussed in section 4.2, and their performance is analyzed through the lens of the metrics defined in section 4.3.

4.4. Resilient federated optimization

4.4.1. Homogeneous data

Early Byzantine-resilient federated learning algorithms, such as those discussed in [207, 208], which address convex and non-convex costs respectively, were built on stochastic gradient descent. In particular, a detection and isolation process guided by concentration inequalities was developed to reduce the sample and time complexity. The work in [186] focused on the development of the resilient aggregation rule, Krum, to alleviate the effect of faulty agents. [209] discovered that reducing the variance of honest gradients is beneficial to defending faulty agents, and proposed to apply aggregation rules to momentums to improve the performance. The paper [202] coined the concept of (f, λ) -resilient averaging aggregators, and quantified the performance of several common aggregators under this framework.

4.4.2. Heterogeneous data

Resilient federated learning with heterogeneous data distributions is arguably more challenging, since the distribution difference among agents makes the faulty

agents harder to identify [189, 203]. [189] motivated the use of momentums in the heterogeneous case, and developed the concept of (δ_{\max}, ζ) -ARAgg and the CC rule for resilient aggregation. Further, [203] presented a pre-aggregation technique called bucketing, which enhances the performance of aggregators within the framework of (δ_{\max}, ζ) -ARAgg. Other criteria for evaluating resilient aggregators include the (f, κ) -robustness [193] and the (f, ξ) -robust averaging [210], with their relationships also discussed. [211] highlighted that variance reduction techniques lower the stochastic noise in honest gradients, which in turn improves the practical performance of resilient aggregators. For online learning problems, [212] showed that a linear regret in the fully adversarial case may be unavoidable, and proved a sublinear regret for an algorithm based on GM and the momentum method when honest agents have i.i.d. losses. [185, 213] discussed the optimal statistical rates achievable by resilient federated learning algorithms and presented nearly optimal algorithms for strongly convex loss functions. [214] adopted another inequality to more accurately capture the heterogeneity in practice, under which a tighter convergence result can be stated.

Considering the case with both Byzantine attacks and communication compression, [215] demonstrated that the Byzantine attacks and stochastic gradients make the compression noise more challenging to handle than attack-free cases. Motivated by this, the authors proposed to compress the gradient difference, which reduces both the compression and stochastic noises. This idea was further explored in [216], where improved convergence guarantees were established under relaxed assumptions. For similar problem setups, [217] devised new algorithms with error feedback to accommodate biased compressors, e.g., Top_k sparsification, to improve the practical performance. [218, 219] investigated the behavior of federated learning with partial participation in the presence of Byzantine agents, and tailored aggregation mechanisms to tackle the challenge.

In another line of research, [205] coined the notion of redundancy in objective functions (later extended and relaxed in [206]) and used it to develop the condition that ensures full resilience in decentralized optimization, that is, convergence to the consensual optimum of regular agents.

4.5. Resilient distributed consensus and optimization

We review resilient distributed algorithms, categorizing them into two primary objectives: consensus and optimization.

Table 8: A comparison of resilient consensus algorithms.

[Ref.]	Vector consensus	Attack [†] model	Aggregation rule	Sparse network	Sufficient conditions	
					Synchronous	Asynchronous
[181]	✗	B	weighted MSR	✓	The network of honest agents is $(n_f + 1)$ -robust	-
[220, 221]	✗	B	weighted MSR	✓	The network of honest agents is $(n_f + 1)$ -robust	
[184]	✗	M	reputation-based consensus	✓	$n_{f_i} < \frac{ \mathcal{N}_i^- }{2}$ and $ \mathcal{N}_i^- > 2$	
[194, 222]	✓	B	Tverberg partition	✗	$n > \max\{3n_f, (d+1)n_f\}$	$n > (d+2)n_f$
[223, 197]	✓	B	Tverberg partition	✓	$n_{f_i} \leq \frac{ \mathcal{N}_i^- }{d+1} - 1$	-
[224]	✓	B	auxiliary point	✓	$n_{f_i} \leq \frac{ \mathcal{N}_i^- }{d+1} - 1$	-
[198]	✓	B	centerpoint	✓	$n_{f_i} \leq \lfloor \frac{ \mathcal{N}_i^- }{d+1} \rfloor$	-

[†] M and B represents the malicious and Byzantine attack model, respectively.

4.5.1. Resilient consensus

The resilient consensus algorithms are reviewed for both scalar and vector scenarios. Table 8 provides an overview of these algorithms.

Scalar consensus. The early works on resilient *scalar* consensus date back to [179, 180]. However, these were restricted to complete networks. Extensions to general sparse networks were developed in [181, 225, 226], where the analysis was based on graph robustness. In these frameworks, the key module for achieving resilience against false data injection (FDI) attacks is the MSR. Under reasonable conditions on the graph, the honest agents using the MSR algorithm reach asymptotic consensus [181].

For asynchronous networks, [220] derived necessary and sufficient topological conditions for achieving approximate consensus. Notably, the conditions in [220] are not more restrictive than those required for synchronous networks. The rate of convergence was examined in [221]. [184] proposed a novel approach to resilient consensus-seeking by integrating a reputation system into the algorithm. In their framework, each agent dynamically evaluates the reputation of its neighbors based on the information received from them over time. This reputation scoring mechanism enables agents to gradually identify and mitigate the influence of malicious neighbors. Under a set of well-defined conditions, including the assumptions that malicious agents do not perturb the initial variables and that the number of malicious neighbors for each agent is less than half, the proposed algorithm exhibits linear convergence to consensus. Notably, this convergence occurs while simultaneously identifying the malicious agents within

the network. Based on evidence theory, [227] developed a methodology where each agent computes reputation values for its neighbors, which is combined with a weight correction scheme to mitigate the influence of malicious agents during the consensus-seeking process. Note that the aforementioned works considered the compromised agents to be static throughout the consensus-seeking process. In contrast, [228] explored a mobile attack model in which a fixed number of adversaries can dynamically switch their targets and developed modified MSR algorithms to mitigate the effects caused by such switching behavior.

Vector consensus. Extending the approach to the *vector* case (*i.e.*, $d \geq 2$) is highly non-trivial, as running MSR independently for each coordinate might cause honest agents to converge to a point outside the convex hull of their initial states [194]. Considering complete communication networks, [194] developed a resilient algorithm based on Tverberg partition [229], for which necessary and sufficient conditions were provided in both synchronous and asynchronous systems. The scenario with incomplete networks was examined in [223], where a sufficient condition, and a (different) necessary condition were given. [222] introduced the concept of safe area, which is guaranteed to be within the convex hull of non-faulty agents. They developed a resilient vector consensus scheme for complete networks. [197] extended the safe point framework to incomplete networks and proposed a resilient rendezvous algorithm for multirobot systems, where the safe point is computed using an approximate algorithm for calculating a Tverberg partition. Another efficient algorithm to compute the safe point was developed in [224], where the

time complexity is $\mathcal{O}((pr)^3)$ with $p = dn_f + 1$ and $r = \binom{(d+1)n_f + 1}{n_f}$. [198] interpreted the safe point as an interior centerpoint, based on which they achieved an improved trade-off between resilience and computation for safe point-based approaches. [230] reported an extension of this method to handle cases with imprecise input data.

4.5.2. Distributed optimization

Similar to consensus, we classify resilient distributed optimization algorithms according to their problem dimensions. Table 9 summarizes these algorithms.

Scalar optimization. [204] studied the distributed scalar optimization problem in the presence of malicious agents. A resilient distributed optimization algorithm was developed based on the distributed subgradient method and the MSR. The authors proved that the estimates by regular agents converge to the convex hull of their local minimizers, under assumptions on the attack model and communication graph. Considering the Byzantine attack model and scalar problems, [231] proposed a resilient distributed optimization algorithm based on trimmed mean. This algorithm guarantees convergence to a region characterized by the minimizers of the convex combinations of local objective functions of the honest agents, under proper conditions. With the help of a set of identifiable trusted nodes, [232] developed an improved distributed scalar optimization algorithm against Byzantine attackers. To address scenarios with mobile malicious attackers, [233] devised a distributed optimization algorithm based on the resilient consensus strategy proposed in their previous work [228]. For constrained optimization, [246] developed a resilient distributed projected subgradient method based on MSR, which guarantees convergence to a bound characterized by minimizers of costs from a subset of regular agents and convergence to the exact minimum when a certain cost redundancy condition is satisfied.

Vector optimization. To address multi-dimensional problems, [234] employed the framework of coordinate descent to decompose the problem into multiple one-dimensional subproblems. They proposed a multi-loop algorithm, where the one outer loop iterates over the dimensions, and the inner loop performs multiple iterations for each coordinate within the current dimension. To reduce sample complexity, [235] proposed a loopless algorithm compatible with several resilient aggregation rules. They established convergence guarantees

for this algorithm, particularly when using the CWTM rule for general non-convex problems. [236, 237] introduced an auxiliary variable to each agent. This auxiliary variable is updated by resilient consensus with coordinate-wise MSR, and is used as a reference in the update of decision variables to defend Byzantine attackers. For stochastic optimization problems under the Byzantine attack model, [238] proposed a general robust aggregation rule. In this rule, each agent iteratively discards the message from its neighbors that deviates the most from the mean in this iteration. To more effectively filter Byzantine agents in distributed learning with homogeneous data, [247] devised a two-stage aggregation strategy. The first stage employs a principled neighbor selection approach, where each agent identifies a subset of neighbors based on the similarity between their respective states and those of their peers. In the subsequent stage, this neighbor set is refined through a discerning evaluation of the loss over the selected states. Recently, [239] proposed another two-stage aggregation strategy for Byzantine-resilient distributed learning. In the first stage, each agent filters its neighboring set by discarding nodes whose states significantly deviate from its own state. The second stage involves clipping the variables in the remaining neighborhood set before performing the aggregation. Assuming agents share a common minimizer, [240] proposed a resilient distributed optimization algorithm that incorporates momentum methods and gradient clipping techniques. Leveraging inter-agent trust, [241] proposed a resilient distributed optimization algorithm with two key features: 1) enabling each agent to identify its malicious neighbors, and 2) ensuring convergence to the exact optimum despite the presence of malicious agents. Based on the notions of cost redundancy [205] and graph robustness, [242] developed a resilient algorithm that converges to the exact optimum.

Resource allocation. For a class of resource allocation problems where the coupled constraint is a function of the average of local decision variables, [67] developed a federated primal-dual optimization algorithm that employs a CWMed-based mean estimation rule to achieve resilience. For a class of resource allocation problems with simple inequality constraints, [244] proposed a fully distributed algorithm based on the primal-dual algorithm with a regularized Lagrangian and CWTM. The algorithm linearly converges to a neighborhood of the optimum due to the presence of Byzantine agents and the regularization. For coupling equality constraints, [245] designed a fully distributed resilient resource allocation algorithm that properly aggregates dual variables

Table 9: A comparison of resilient distributed optimization algorithms for peer-to-peer topologies.

[Ref.]	Vector optimization	Attack [†] model	Aggregation rule	Aggregated quantity	Assumption
[204]	✗	B and M	MSR	decision variables	$(2n_f + 1)$ -robust \mathcal{G}
[231]	✗	B	MSR	decision variables and gradients	every reduced graph has a nonempty source component
[232]	✗	B	averaging based on honest neighbors' info	decision variables	A known, connected network of honest agents
[233]	✗	mobile M	MSR	decision variables	$n \geq 4n_f + 4$ and $ \mathcal{N}_i^- \geq 2n_f + 1 + \frac{n}{2}$
[234]	✓	B	coordinate-wise MSR	decision variables	every reduced graph has a nonempty source component
[235]	✓	B	general rules	decision variables	every reduced graph has a nonempty source component
[236, 237]	✓	B	trimmed mean and distance-based rules	decision and auxiliary variables	$(2n_f + 1)$ -robust \mathcal{G}
[238]	✓	B	iterative outlier scissor (IOS)	decision variables	the network resulted from IOS is strongly connected
[239]	✓	B	remove-then-clip	decision variables	bounded weights for faulty agents
[240]	✓	gradient attack	clip	decision variables	common individual minimizer
[241]	✓	B	averaging among trusted neighborhood	decision variables	observations of trust and honest agents form a connected graph
[242]	✓	B	resilient convex combination [243]	decision variables	cost and graph redundancy
[244]	✓	B	CWTM	decision variables	every reduced graph has a nonempty source component
[245]	✓	B	general rules	dual variables	honest agents form a connected graph

[†] M and B represents the malicious and Byzantine attack model, respectively.

to achieve resilience. The algorithm was shown to converge as long as the aggregation rule satisfies a concentration property.

4.6. Discussions

Achieving resilience under data heterogeneity. In the context of federated learning with heterogeneous datasets, convergence to the exact solution of (19) is not possible. This is because the inherent heterogeneity among honest agents makes the Byzantine updates more difficult to exclude. Indeed, [203] presented a lower bound result, indicating that the optimization error is proportional to the variance of individual gradients. However, this heterogeneity model is restrictive and may not cover certain basic learning problems, such as least-squares regression. To address this issue, [214] developed another gradient dissimilarity notion to characterize heterogeneity and established lower bound ac-

cordingly. The notion of cost redundancy is another interesting research direction related to this issue, which serves a condition for the solvability of (19), as detailed in Section 4.3.

The curse of dimensionality. Solving multi-dimensional resilient decentralized optimization problems is challenging for two main reasons. First, in resilient consensus problems, running scalar resilient algorithms for each coordinate can lead to violations of safety requirements. Second, resilient aggregation and consensus algorithms that are efficient in multi-dimensional settings typically require heavy computational loads (*e.g.* sorting vectors), as discussed in previous sections. Aggregation mechanisms that are computationally lighter tend to be less resilient, often exhibiting larger parameters in the concentration criterion for aggregation resilience. This highlights an

efficiency-resilience trade-off in resilient decentralized optimization and learning.

5. Conclusion and Outlook

We have presented a comprehensive survey of existing privacy-preserving and resilient decentralized optimization and learning algorithms, highlighting their strengths and limitations.

5.1. Conclusion

This survey underscores the critical importance of a systematic design approach that prioritizes resilience and privacy preservation for secure decentralized optimization and learning systems. While a fruitful set of existing results has partially addressed this by integrating defense modules like differential privacy, encryption, and secret sharing protocols for privacy preservation, as well as resilient aggregation mechanisms for resilience into standard decentralized frameworks, incorporating additional defense mechanisms inevitably impacts the system’s overall utility. Notably, the interdependencies between these mechanisms and the dynamic systems can be leveraged to optimize the security-utility trade-off. Furthermore, different attacks pose varying consequences, necessitating distinct defense measures. Consequently, the deployment of defense mechanisms should be tailored to the specific criteria and practical requirements of the systems.

5.2. Topics not covered in this survey

In Section 1, we discuss the CIA triad in information technology systems: confidentiality, integrity, and availability. The first two, confidentiality and integrity, pertain to data privacy and model security in decentralized optimization and learning, which are the focus of this work. However, the third property, *availability* (e.g., protection against denial-of-service attacks), is not covered in this study.

Adversarial machine learning, which involves incorporating adversarial samples into the training set to enhance robustness [248, 249], is not covered in this survey. While this approach has shown excellent performance, it may not effectively address model security attacks that can behave arbitrarily, a key focus of this survey.

Finally, this survey does not address the design of attacks, such as optimization-based data privacy attacks [19] and gradient variance-based model security attacks [27]. However, the threat model we consider is general enough to encompass these attacks as special cases, and

the secure decentralized optimization and learning algorithms reviewed in this survey are applicable to them.

5.3. Future research directions

Several important and challenging future research directions have recently garnered significant attention. Some of the most notable areas are summarized below.

Achieving privacy and resilience simultaneously. The majority of existing secure decentralized algorithms consider either privacy preservation or resilience, but not both. However, decentralized optimization and learning algorithms may face threats targeting both data and the algorithm itself. This challenging yet important problem has recently drawn attention from various communities, such as consensus [250, 251] and decentralized machine learning [210, 252]. In particular, [250] combined the Laplace mechanism and MSR to make the consensus algorithm privacy-preserving and resilient, while [251] incorporated the Gaussian mechanism into a reputation-based resilient consensus algorithm [184] to achieve DP. For secure federated learning, [210] showed that there is a fundamental accuracy cost when simultaneously enforcing DP and resilience. This is primarily because DP mechanisms introduce artificial data heterogeneity among workers through noise injection, making identifying malicious agents more challenging. However, this research direction is far from complete. For example, combining other privacy-preserving mechanisms, such as HE and SS, with resilient aggregation to achieve privacy and resilience simultaneously is worthy of exploration. Investigating the fundamental performance limits of secure distributed optimization and learning is also an interesting avenue. Another intriguing direction is to combine privacy and resilience when considering weaker yet meaningful adversaries, such as the honest non-curious coordinator scenario [253].

Securing decentralized algorithms in asynchronous environment. Existing secure decentralized optimization algorithms implicitly assume synchronous communication. This assumption facilitates the implementation of resilient aggregation and provides a basis for theoretical justification of the algorithms. However, achieving global synchronization over a network can be challenging in practice. Furthermore, synchronized updates are inefficient and unreliable, as the time taken per iteration is determined by the slowest node and the optimization process is vulnerable to single-node failures [254, 255]. Consequently, defending against attackers in an asynchronous environment has emerged as an important yet

challenging task. A few recent attempts have been made in [256, 257] for resilient federated learning, where their approaches to deal with asynchrony involve a dampening scheme that scales each gradient based on its staleness and creating a buffer at the coordinator while waiting for enough gradients to arrive. Interesting future research directions include the consideration of peer-to-peer topologies and more general models of asynchrony [254].

Detecting and isolating faulty agents to improve accuracy. A natural approach to improving the security-utility trade-off is to incorporate proactive fault detection and isolation mechanisms [258] into decentralized optimization and learning systems. Recent works, such as [184], implemented reputation scoring systems within consensus-seeking algorithms to address this. In federated learning, [259] adjusted the weights assigned to each agent based on their contributions, as measured by the Shapley value. Despite these advancements, it remains unclear how to effectively combine proactive mechanisms with passive robust aggregation techniques to achieve theoretical guarantees on system performance and resilience. Additionally, there is a need to explore whether these strategies can be extended to distributed optimization problems over peer-to-peer topologies.

Exploiting additional side information to enhance resilience. Most resilient decentralized optimization approaches rely on the transmitted data between agents to infer the presence of anomalies. This approach can be further strengthened by exploiting additional side information from the CPS, such as arriving directions of WiFi signals [260], to provide supplementary avenues for resilience. For example, [261, 241] developed an inter-agent trust evaluation framework that ensures convergence to the exact optimum in distributed optimization. Along this line of research, future efforts may include: 1) formalizing more characterizable notions of trust in multi-agent networks and 2) developing novel decentralized optimization algorithms that seamlessly integrate and exploit trust values.

Leveraging blockchain technology to secure decentralized learning. Blockchain technology, originally developed for the digital currency Bitcoin, combines cryptographic algorithms and decentralized consensus protocols to enable secure data sharing and storage among agents, even in the presence of faulty agents [262]. By design, it is a suitable architecture for secure decentralized optimization. Indeed, this architecture has

paved the way for innovative applications in various domains. For example, [263] developed a secure federated learning system, [264] studied blockchain-based robot swarms, and [265] seamlessly combined blockchain and energy dispatch tasks through replacing the mathematical puzzle in Proof of Work by Proof of Solution to a task-related optimization problem. While the integration of blockchains into decentralized optimization and related problems holds significant promise, several technological challenges need to be addressed. For example, the computing, storage, and communication limitations of mobile multi-agent systems must be considered, as they may pose constraints on the effective implementation of blockchain-based solutions.

Decentralized optimization and learning for control. Decentralized optimization and learning are closely tied to the two primary tasks in control: system identification [266, 267] and control design [268, 269]. Recent advancements in decentralized optimization have laid the foundations for advanced control design. For example, developing violation-free distributed optimization algorithms under coupling constraints for safe distributed control, as recently attempted in [270, 271], is a key direction. Another promising future direction is secure distributed control based on decentralized optimization methodologies. Finally, leveraging fast distributed optimization algorithms to enable the distributed implementation of data-enabled predictive control (DeePC) [272] is also worthy of exploration.

Acknowledgements

This work was partially supported by the European Union’s Horizon Research and Innovation Actions programme under grant agreement No. 101070162, partially by the Swedish Research Council Distinguished Professor Grant 2017-01078, the Knut and Alice Wallenberg Foundation Wallenberg Scholar Grant, and partially by the NSERC Postdoctoral Fellowship PDF-577876-2023.

References

- [1] European Council, Regulation of the European Parliament and of the Council on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), <https://data.consilium.europa.eu/doc/document/ST-11744-2020-REV-2/en/pdf>.

- [2] European Council, Regulation of the European Parliament and of the Council laying down harmonised rules on artificial intelligence (Artificial Intelligence Act) and amending certain Union legislative acts, <https://data.consilium.europa.eu/doc/document/ST-5662-2024-INIT/en/pdf>.
- [3] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, D. Ramage, Federated learning for mobile keyboard prediction, arXiv preprint arXiv:1811.03604 (2018).
- [4] A. Rauniyar, D. H. Hagos, D. Jha, J. E. Häkegård, U. Bagci, D. B. Rawat, V. Vlassov, Federated learning for medical applications: A taxonomy, current trends, challenges, and future research directions, IEEE Internet of Things Journal (2023) 1–1.
- [5] S. Zhang, J. Li, L. Shi, M. Ding, D. C. Nguyen, W. Tan, J. Weng, Z. Han, Federated learning in intelligent transportation systems: Recent applications and open problems, IEEE Transactions on Intelligent Transportation Systems (2024) 1–27.
- [6] Y. Qu, H. Dai, Y. Zhuang, J. Chen, C. Dong, F. Wu, S. Guo, Decentralized federated learning for uav networks: Architecture, challenges, and opportunities, IEEE Network 35 (6) (2021) 156–162.
- [7] L. Li, Y. Fan, M. Tse, K.-Y. Lin, A review of applications in federated learning, Computers & Industrial Engineering 149 (2020) 106854.
- [8] C. Li, Demystifying gpt-3 language model: A technical overview, <https://lambdalabs.com/blog/demystifying-gpt-3>.
- [9] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, K. He, Accurate, large mini-batch sgd: Training imagenet in 1 hour, arXiv preprint arXiv:1706.02677 (2017).
- [10] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen, et al., Massively parallel methods for deep reinforcement learning, arXiv preprint arXiv:1507.04296 (2015).
- [11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, nature 518 (7540) (2015) 529–533.
- [12] H. Mikami, H. Suganuma, Y. Tanaka, Y. Kageyama, et al., Massively distributed sgd: Imagenet/resnet-50 training in a flash, arXiv preprint arXiv:1811.05233 (2018).
- [13] T. Li, A. K. Sahu, A. Talwalkar, V. Smith, Federated learning: Challenges, methods, and future directions, IEEE Signal Processing Magazine 37 (3) (2020) 50–60.
- [14] T. Gafni, N. Shlezinger, K. Cohen, Y. C. Eldar, H. V. Poor, Federated learning: A signal processing perspective, IEEE Signal Processing Magazine 39 (3) (2022) 14–41.
- [15] A. Nedić, J. Liu, Distributed optimization for control, Annual Review of Control, Robotics, and Autonomous Systems 1 (2018) 77–103.
- [16] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, K. H. Johansson, A survey of distributed optimization, Annual Reviews in Control 47 (2019) 278–305.
- [17] B. Matt, Computer security: art and science (2002).
- [18] A. Teixeira, K. C. Sou, H. Sandberg, K. H. Johansson, Secure control systems: A quantitative risk management approach, IEEE Control Systems Magazine 35 (1) (2015) 24–45.
- [19] L. Zhu, S. Han, Deep leakage from gradients, in: Federated Learning, Springer, 2020, pp. 17–31.
- [20] L. Su, N. H. Vaidya, Fault-tolerant multi-agent optimization: optimal iterative distributed algorithms, in: Proceedings of the 2016 ACM Symposium on Principles of Distributed Comput-
- ing, 2016, pp. 425–434.
- [21] B. Hitaj, G. Ateniese, F. Perez-Cruz, Deep models under the gan: information leakage from collaborative deep learning, in: Proceedings of the 24th ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 603–618.
- [22] L. Melis, C. Song, E. De Cristofaro, V. Shmatikov, Exploiting unintended feature leakage in collaborative learning, in: 40th IEEE Symposium on Security and Privacy (SP), 2019, pp. 691–706.
- [23] R. Shokri, M. Stronati, C. Song, V. Shmatikov, Membership inference attacks against machine learning models, in: 38th IEEE Symposium on Security and Privacy (SP), 2017, pp. 3–18.
- [24] M. Nasr, R. Shokri, A. Houmansadr, Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning, in: 40th IEEE Symposium on Security and Privacy (SP), 2019, pp. 739–753.
- [25] J. Geiping, H. Bauermeister, H. Dröge, M. Moeller, Inverting gradients—how easy is it to break privacy in federated learning?, Advances in Neural Information Processing Systems 33 (2020) 16937–16947.
- [26] A. N. Bhagoji, S. Chakraborty, P. Mittal, S. Calo, Analyzing federated learning through an adversarial lens, in: International conference on machine learning, PMLR, 2019, pp. 634–643.
- [27] G. Baruch, M. Baruch, Y. Goldberg, A little is enough: Circumventing defenses for distributed learning, Advances in Neural Information Processing Systems 32 (2019).
- [28] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, V. Shmatikov, How to backdoor federated learning, in: International conference on artificial intelligence and statistics, PMLR, 2020, pp. 2938–2948.
- [29] J. Lin, M. Du, J. Liu, Free-riders in federated learning: Attacks and defenses, arXiv preprint arXiv:1911.12560 (2019).
- [30] Z. Chen, Y. Wang, Privacy-preserving distributed optimization and learning, arXiv preprint arXiv:2403.00157 (2024).
- [31] Z. Yang, A. Gang, W. U. Bajwa, Adversary-resilient distributed and decentralized statistical inference and machine learning: An overview of recent advances under the byzantine threat model, IEEE Signal Processing Magazine 37 (3) (2020) 146–159.
- [32] R. Guerraoui, N. Gupta, R. Pinot, Byzantine machine learning: A primer, ACM Computing Surveys (2023).
- [33] G. Notarstefano, I. Notaricola, A. Camisa, Distributed optimization for smart cyber-physical networks, Foundations and Trends® in Systems and Control 7 (3) (2019) 253–383.
- [34] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, Y. Gao, A survey on federated learning, Knowledge-Based Systems 216 (2021) 106775.
- [35] H. Gao, M. T. Thai, J. Wu, When decentralized optimization meets federated learning, IEEE Network 37 (5) (2023) 233–239.
- [36] M. Mohri, A. Rostamizadeh, A. Talwalkar, Foundations of machine learning, second edition Edition, Adaptive computation and machine learning, The MIT Press, Cambridge, Massachusetts, 2018.
- [37] A. Nedić, A. Olshevsky, M. G. Rabbat, Network topology and communication-computation tradeoffs in decentralized optimization, Proceedings of the IEEE 106 (5) (2018) 953–976.
- [38] H. H. Bauschke, P. L. Combettes, Convex analysis and monotone operator theory in Hilbert spaces, 2nd Edition, CMS books in mathematics, Springer, Cham, 2017.
- [39] L. Qian, P. Yang, M. Xiao, O. A. Dobre, M. D. Renzo, J. Li, Z. Han, Q. Yi, J. Zhao, Distributed learning for wireless communications: Methods, applications and challenges, IEEE

- Journal of Selected Topics in Signal Processing 16 (3) (2022) 326–342.
- [40] L. Xu, X. Yi, C. Deng, Y. Shi, T. Chai, T. Yang, Quantized zeroth-order gradient tracking algorithm for distributed nonconvex optimization under polyak–lojasiewicz condition, IEEE Transactions on Cybernetics (2024).
- [41] Z. Zhao, Y. Mao, Y. Liu, L. Song, Y. Ouyang, X. Chen, W. Ding, Towards efficient communications in federated learning: A contemporary survey, Journal of the Franklin Institute (2023) S0016003222009346.
- [42] M. Grudzień, G. Malinovsky, P. Richtárik, Can 5th generation local training methods support client sampling? yes!, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2023, pp. 1055–1092.
- [43] L. Xu, X. Yi, Y. Shi, K. H. Johansson, T. Chai, T. Yang, Distributed nonconvex optimization with event-triggered communication, IEEE Transactions on Automatic Control (2023).
- [44] C. Liu, H. Li, Y. Shi, D. Xu, Distributed event-triggered gradient method for constrained convex minimization, IEEE Transactions on Automatic Control 65 (2) (2019) 778–785.
- [45] E. Gorbunov, F. Hanzely, P. Richtarik, Local SGD: Unified theory and new efficient methods, in: A. Banerjee, K. Fukumizu (Eds.), Proceedings of The 24th International Conference on Artificial Intelligence and Statistics, Vol. 130 of Proceedings of Machine Learning Research, PMLR, 2021, pp. 3556–3564.
- [46] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y. Arcas, Communication-Efficient Learning of Deep Networks from Decentralized Data, in: A. Singh, J. Zhu (Eds.), Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Vol. 54 of Proceedings of Machine Learning Research, PMLR, Fort Lauderdale, FL, USA, 2017, pp. 1273–1282.
- [47] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, A. T. Suresh, SCAFFOLD: Stochastic Controlled Averaging for Federated Learning, in: H. D. III, A. Singh (Eds.), Proceedings of the 37th International Conference on Machine Learning, Vol. 119 of Proceedings of Machine Learning Research, PMLR, 2020, pp. 5132–5143.
- [48] S. S. Kia, B. V. Scov, J. Cortes, R. A. Freeman, K. M. Lynch, S. Martinez, Tutorial on dynamic average consensus: The problem, its applications, and the algorithms, IEEE Control Systems Magazine 39 (3) (2019) 40–72.
- [49] J. Chen, A. H. Sayed, Diffusion adaptation strategies for distributed optimization and learning over networks, IEEE Transactions on Signal Processing 60 (8) (2012) 4289–4305.
- [50] K. Yuan, Q. Ling, W. Yin, On the convergence of decentralized gradient descent, SIAM Journal on Optimization 26 (3) (2016) 1835–1854.
- [51] A. Sundararajan, B. V. Scov, L. Lessard, A canonical form for first-order distributed optimization algorithms, in: American Control Conference (ACC), 2019, pp. 4075–4080.
- [52] K. Yuan, B. Ying, X. Zhao, A. H. Sayed, Exact diffusion for distributed optimization and learning—parts I: Algorithm development, IEEE Transactions on Signal Processing 67 (3) (2019) 708–739.
- [53] R. Xin, S. Pu, A. Nedić, U. A. Khan, A general framework for decentralized optimization with first-order methods, Proceedings of the IEEE 108 (11) (2020) 1869–1889.
- [54] Y. Nesterov, Primal-dual subgradient methods for convex problems, Mathematical Programming 120 (1) (2009) 221–259.
- [55] J. C. Duchi, A. Agarwal, M. J. Wainwright, Dual averaging for distributed optimization: Convergence analysis and network scaling, IEEE Transactions on Automatic control 57 (3) (2011) 592–606.
- [56] C. Liu, X. Wu, X. Yi, Y. Shi, K. H. Johansson, Rate analysis of dual averaging for nonconvex distributed optimization, IFAC-PapersOnLine 56 (2) (2023) 5209–5214.
- [57] C. Liu, Z. Zhou, J. Pei, Y. Zhang, Y. Shi, Decentralized composite optimization in stochastic networks: A dual averaging approach with linear convergence, IEEE Transactions on Automatic Control 68 (8) (2022) 4650–4665.
- [58] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Machine Learning 3 (1) (2010) 1–122.
- [59] N. Bastianello, R. Carli, L. Schenato, M. Todisco, Asynchronous distributed optimization over lossy networks via relaxed admm: Stability and linear convergence, IEEE Transactions on Automatic Control 66 (6) (2021) 2620–2635.
- [60] N. Bastianello, D. Deplano, M. Franceschelli, K. H. Johansson, Online distributed learning over random networks, arXiv:2309.00520 [cs, eess, math] (Sep. 2023).
- [61] M. Bin, I. Notarnicola, T. Parisini, Stability, linear convergence, and robustness of the Wang–Elia algorithm for distributed consensus optimization, in: IEEE 61st Conference on Decision and Control (CDC), 2022, pp. 1610–1615.
- [62] N. Bof, R. Carli, L. Schenato, Average consensus with asynchronous updates and unreliable communication, IFAC-PapersOnLine 50 (1) (2017) 601–606.
- [63] N. Bof, R. Carli, G. Notarstefano, L. Schenato, D. Varagno, Multiagent Newton–Raphson optimization over lossy networks, IEEE Transactions on Automatic Control 64 (7) (2019) 2983–2990.
- [64] Y. Tian, Y. Sun, G. Scutari, Achieving linear convergence in distributed asynchronous multiagent optimization, IEEE Transactions on Automatic Control 65 (12) (2020) 5264–5279.
- [65] G. Carnevale, N. Bastianello, G. Notarstefano, R. Carli, ADMM-tracking gradient for distributed optimization over asynchronous and unreliable networks, arXiv:2309.14142 [math] (Sep. 2023).
- [66] L. Xiao, S. Boyd, Optimal scaling of a gradient method for distributed resource allocation, Journal of optimization theory and applications 129 (2006) 469–488.
- [67] B. Turan, C. A. Uribe, H.-T. Wai, M. Alizadeh, Resilient primal–dual optimization algorithms for distributed resource allocation, IEEE Transactions on Control of Network Systems 8 (1) (2020) 282–294.
- [68] M. Liu, P. K. Phanivong, Y. Shi, D. S. Callaway, Decentralized charging control of electric vehicles in residential distribution networks, IEEE Transactions on Control Systems Technology 27 (1) (2017) 266–281.
- [69] Z. Wang, C.-J. Ong, Accelerated distributed mpc of linear discrete-time systems with coupled constraints, IEEE Transactions on Automatic Control 63 (11) (2018) 3838–3849.
- [70] X. Fang, L. Xie, Distributed formation maneuver control using complex laplacian, IEEE Transactions on Automatic Control (2023).
- [71] D. Mateos-Núñez, J. Cortés, Distributed saddle-point subgradient algorithms with laplacian averaging, IEEE Transactions on Automatic Control 62 (6) (2016) 2720–2735.
- [72] A. Falsone, K. Margellos, S. Garatti, M. Prandini, Dual decomposition for multi-agent distributed optimization with coupling constraints, Automatica 84 (2017) 149–158.
- [73] I. Notarnicola, G. Notarstefano, Constraint-coupled distributed optimization: a relaxation and duality approach, IEEE Transactions on Control of Network Systems 7 (1) (2019) 483–492.
- [74] S. Liang, G. Yin, et al., Distributed smooth convex optimization with coupled constraints, IEEE Transactions on Automatic Control 65 (1) (2019) 347–353.
- [75] X. Tan, D. V. Dimarogonas, Distributed implementation of

- control barrier functions for multi-agent systems, *IEEE Control Systems Letters* 6 (2021) 1879–1884.
- [76] P. Mestres, J. Cortés, Distributed and anytime algorithm for network optimization problems with separable structure, in: 62nd IEEE Conference on Decision and Control (CDC), 2023, pp. 5463–5468.
- [77] X. Wu, S. Magnússon, M. Johansson, A new family of feasible methods for distributed resource allocation, in: 60th IEEE Conference on Decision and Control (CDC), 2021, pp. 3355–3360.
- [78] X. Wu, S. Magnússon, M. Johansson, Distributed safe resource allocation using barrier functions, *Automatica* 153 (2023) 111051.
- [79] C. Dwork, F. McSherry, K. Nissim, A. Smith, Calibrating noise to sensitivity in private data analysis, in: Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4–7, 2006. Proceedings 3, Springer, 2006, pp. 265–284.
- [80] R. L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM* 21 (2) (1978) 120–126.
- [81] A. Shamir, How to share a secret, *Communications of the ACM* 22 (11) (1979) 612–613.
- [82] C. Dwork, Differential privacy, in: International Colloquium on Automata, Languages, and Programming, Springer, 2006, pp. 1–12.
- [83] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 308–318.
- [84] J. Altschuler, K. Talwar, Privacy of noisy stochastic gradient descent: More iterations without more privacy loss, *Advances in Neural Information Processing Systems* 35 (2022) 3788–3800.
- [85] C. Dwork, A. Roth, The algorithmic foundations of differential privacy, *Foundations and Trends® in Theoretical Computer Science* 9 (3–4) (2014) 211–407.
- [86] P. Kairouz, S. Oh, P. Viswanath, The composition theorem for differential privacy, in: International Conference on Machine Learning, PMLR, 2015, pp. 1376–1385.
- [87] I. Mironov, Rényi differential privacy, in: IEEE 30th Computer Security Foundations Symposium (CSF), 2017, pp. 263–275.
- [88] F. McSherry, K. Talwar, Mechanism design via differential privacy, in: 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS’07), 2007, pp. 94–103.
- [89] A. Roth, T. Roughgarden, Interactive privacy via the median mechanism, in: Proceedings of the forty-second ACM symposium on Theory of computing, 2010, pp. 765–774.
- [90] K. H. Degue, J. Le Ny, Differentially private interval observer design with bounded input perturbation, in: American Control Conference (ACC), 2020, pp. 1465–1470.
- [91] Q. Geng, W. Ding, R. Guo, S. Kumar, Tight analysis of privacy and utility tradeoff in approximate differential privacy, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 89–99.
- [92] F. Liu, Generalized gaussian mechanism for differential privacy, *IEEE Transactions on Knowledge and Data Engineering* 31 (4) (2018) 747–756.
- [93] D. M. Sommer, L. Abfalsterer, S. Zingg, E. Mohammadi, Learning numeric optimal differentially private truncated additive mechanisms, arXiv preprint arXiv:2107.12957 (2021).
- [94] R. Bassily, A. Smith, A. Thakurta, Private empirical risk minimization: Efficient algorithms and tight error bounds, in: 2014 IEEE 55th Annual Symposium on Foundations of Computer Science, IEEE, 2014, pp. 464–473.
- [95] D. Kifer, A. Smith, A. Thakurta, Private convex empirical risk minimization and high-dimensional regression, in: Proceedings of the 25th Annual Conference on Learning Theory, Vol. 23, PMLR, 2012, pp. 25.1–25.40.
- [96] K. Chaudhuri, A. Sarwate, K. Sinha, Near-optimal differentially private principal components, *Advances in Neural Information Processing Systems* 25 (2012) 989–997.
- [97] R. L. Rivest, L. Adleman, M. L. Dertouzos, et al., On data banks and privacy homomorphisms, *Foundations of secure computation* 4 (11) (1978) 169–180.
- [98] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: International conference on the theory and applications of cryptographic techniques, Springer, 1999, pp. 223–238.
- [99] C. Marcolla, V. Sucasas, M. Manzano, R. Bassoli, F. H. Fitzek, N. Aaraj, Survey on fully homomorphic encryption, theory, and applications, *Proceedings of the IEEE* 110 (10) (2022) 1572–1609.
- [100] C. Gentry, A fully homomorphic encryption scheme, Stanford university, 2009.
- [101] A. Al Badawi, Y. Polyakov, Demystifying bootstrapping in fully homomorphic encryption, *Cryptology ePrint Archive* (2023).
- [102] M. Ruan, H. Gao, Y. Wang, Secure and privacy-preserving consensus, *IEEE Transactions on Automatic Control* 64 (10) (2019) 4035–4049.
- [103] J. Kim, D. Kim, Y. Song, H. Shim, H. Sandberg, K. H. Johansson, Comparison of encrypted control approaches and tutorial on dynamic systems using learning with errors-based homomorphic encryption, *Annual Reviews in Control* 54 (2022) 200–218.
- [104] Y. Aono, T. Hayashi, L. Wang, S. Moriai, et al., Privacy-preserving deep learning via additively homomorphic encryption, *IEEE Transactions on Information Forensics and Security* 13 (5) (2017) 1333–1345.
- [105] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, K. Seth, Practical secure aggregation for privacy-preserving machine learning, in: proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 1175–1191.
- [106] F. Emekci, O. D. Sahin, D. Agrawal, A. El Abbadi, Privacy preserving decision tree learning over multiple parties, *Data & Knowledge Engineering* 63 (2) (2007) 348–361.
- [107] R. Hu, Y. Guo, H. Li, Q. Pei, Y. Gong, Personalized federated learning with differential privacy, *IEEE Internet of Things Journal* 7 (10) (2020) 9530–9539.
- [108] M. Noble, A. Bellet, A. Dieuleveut, Differentially private federated learning on heterogeneous data, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2022, pp. 10110–10145.
- [109] J. Zhang, D. Fay, M. Johansson, Dynamic privacy allocation for locally differentially private federated learning with composite objectives, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2024, pp. 9461–9465.
- [110] S. Han, U. Topcu, G. J. Pappas, Differentially private distributed constrained optimization, *IEEE Transactions on Automatic Control* 62 (1) (2016) 50–64.
- [111] Z. Huang, S. Mitra, G. Dullerud, Differentially private iterative synchronous consensus, in: Proceedings of the ACM Workshop on Privacy in the Electronic Society, 2012, pp. 81–90.
- [112] E. Nozari, P. Tallapragada, J. Cortés, Differentially private average consensus: Obstructions, trade-offs, and optimal algorithm design, *Automatica* 81 (2017) 221–231.
- [113] Y. Wang, A robust dynamic average consensus algorithm that

- ensures both differential privacy and accurate convergence, in: 62nd IEEE Conference on Decision and Control (CDC), 2023, pp. 1130–1137.
- [114] J. Wang, J. Ke, J.-F. Zhang, Differentially private bipartite consensus over signed networks with time-varying noises, *IEEE Transactions on Automatic Control* (2024).
- [115] E. Cyffers, M. Even, A. Bellet, L. Massoulié, Mufflato: Peer-to-peer privacy amplification for decentralized optimization and averaging, *Advances in Neural Information Processing Systems* 35 (2022) 15889–15902.
- [116] Y. Wang, J. Lam, H. Lin, Differentially private average consensus for networks with positive agents, *IEEE Transactions on Cybernetics* (2023).
- [117] Z. Huang, S. Mitra, N. Vaidya, Differentially private distributed optimization, in: Proceedings of the 16th International Conference on Distributed Computing and Networking, 2015, pp. 1–10.
- [118] J. Zhu, C. Xu, J. Guan, D. O. Wu, Differentially private distributed online algorithms over time-varying directed networks, *IEEE Transactions on Signal and Information Processing over Networks* 4 (1) (2018) 4–17.
- [119] Y. Xiong, J. Xu, K. You, J. Liu, L. Wu, Privacy-preserving distributed online optimization over unbalanced digraphs via sub-gradient rescaling, *IEEE Transactions on Control of Network Systems* 7 (3) (2020) 1366–1378.
- [120] D. Han, K. Liu, Y. Lin, Y. Xia, Differentially private distributed online learning over time-varying digraphs via dual averaging, *International Journal of Robust and Nonlinear Control* 32 (5) (2022) 2485–2499.
- [121] T. Ding, S. Zhu, J. He, C. Chen, X. Guan, Differentially private distributed optimization via state and direction perturbation in multiagent systems, *IEEE Transactions on Automatic Control* 67 (2) (2021) 722–737.
- [122] X. Chen, L. Huang, L. He, S. Dey, L. Shi, A differentially private method for distributed optimization in directed networks via state decomposition, *IEEE Transactions on Control of Network Systems* (2023).
- [123] S. Pu, A robust gradient tracking method for distributed optimization over directed networks, in: 59th IEEE Conference on Decision and Control (CDC), 2020, pp. 2335–2341.
- [124] Y. Xuan, Y. Wang, Gradient-tracking based differentially private distributed optimization with enhanced optimization accuracy, *Automatica* 155 (2023) 111150.
- [125] S. Vlaski, A. H. Sayed, Graph-homomorphic perturbations for private decentralized learning, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 5240–5244.
- [126] L. Huang, J. Wu, D. Shi, S. Dey, L. Shi, Differential privacy in distributed optimization with gradient tracking, *IEEE Transactions on Automatic Control* (2024).
- [127] S. Kalra, J. Wen, J. C. Cresswell, M. Volkovs, H. R. Tizhoosh, Decentralized federated learning through proxy model sharing, *Nature Communications* 14 (1) (2023) 2899.
- [128] E. Nozari, P. Tallapragada, J. Cortés, Differentially private distributed convex optimization via functional perturbation, *IEEE Transactions on Control of Network Systems* 5 (1) (2016) 395–408.
- [129] T. Zhang, Q. Zhu, Dynamic differential privacy for ADMM-based distributed classification learning, *IEEE Transactions on Information Forensics and Security* 12 (1) (2016) 172–187.
- [130] X. Zhang, M. M. Khalili, M. Liu, Improving the privacy and accuracy of ADMM-based distributed algorithms, in: International Conference on Machine Learning, PMLR, 2018, pp. 5796–5805.
- [131] C. Liu, K. H. Johansson, Y. Shi, Distributed empirical risk minimization with differential privacy, *Automatica* 162 (2024) 111514.
- [132] Y. Wang, T. Başar, Quantization enabled privacy protection in decentralized stochastic optimization, *IEEE Transactions on Automatic Control* (2022).
- [133] Y. Wang, T. Başar, Decentralized nonconvex optimization with guaranteed privacy and accuracy, *Automatica* 150 (2023) 110858.
- [134] Y. Wang, A. Nedić, Tailoring gradient methods for differentially-private distributed optimization, *IEEE Transactions on Automatic Control* (2023).
- [135] F. Chen, X. Chen, L. Xiang, W. Ren, Distributed economic dispatch via a predictive scheme: heterogeneous delays and privacy preservation, *Automatica* 123 (2021) 109356.
- [136] T. Ding, S. Zhu, C. Chen, J. Xu, X. Guan, Differentially private distributed resource allocation via deviation tracking, *IEEE Transactions on Signal and Information Processing over Networks* 7 (2021) 222–235.
- [137] M. Ye, G. Hu, L. Xie, S. Xu, Differentially private distributed nash equilibrium seeking for aggregative games, *IEEE Transactions on Automatic Control* 67 (5) (2021) 2451–2458.
- [138] J. Wang, J.-F. Zhang, X. He, Differentially private distributed algorithms for stochastic aggregative games, *Automatica* 142 (2022) 110440.
- [139] Y. Chen, G. Shi, Differentially private games via payoff perturbation, in: American Control Conference (ACC), 2023, pp. 429–434.
- [140] Y. Wang, A. Nedić, Differentially-private distributed algorithms for aggregative games with guaranteed convergence, *IEEE Transactions on Automatic Control* (2024).
- [141] G. Yan, T. Li, K. Wu, L. Song, Killing two birds with one stone: Quantization achieves privacy in distributed learning, *Digital Signal Processing* 146 (2024) 104353.
- [142] N. Lang, E. Sofer, T. Shaked, N. Shlezinger, Joint privacy enhancement and quantization in federated learning, *IEEE Transactions on Signal Processing* 71 (2023) 295–310.
- [143] K. Chaudhuri, C. Guo, M. Rabbat, Privacy-aware compression for federated data analysis, in: Uncertainty in Artificial Intelligence, PMLR, 2022, pp. 296–306.
- [144] S. Amiri, A. Belloum, S. Klous, L. Gommans, Compressive differentially private federated learning through universal vector quantization, in: AAAI Workshop on Privacy-Preserving Artificial Intelligence, 2021, pp. 2–9.
- [145] N. Bastianello, C. Liu, K. H. Johansson, Enhancing privacy in federated learning through local training, arXiv preprint arXiv:2403.17572 (2024).
- [146] Y. Shoukry, K. Gatsis, A. Alanwar, G. J. Pappas, S. A. Seshaia, M. Srivastava, P. Tabuada, Privacy-aware quadratic optimization using partially homomorphic encryption, in: IEEE 55th Conference on Decision and Control (CDC), 2016, pp. 5053–5058.
- [147] A. B. Alexandru, K. Gatsis, Y. Shoukry, S. A. Seshaia, P. Tabuada, G. J. Pappas, Cloud-based quadratic optimization with partially homomorphic encryption, *IEEE Transactions on Automatic Control* 66 (5) (2020) 2357–2364.
- [148] Y. Lu, M. Zhu, Privacy preserving distributed optimization using homomorphic encryption, *Automatica* 96 (2018) 314–325.
- [149] X. Huo, M. Liu, Privacy-preserving distributed multi-agent cooperative optimization—paradigm design and privacy analysis, *IEEE Control Systems Letters* 6 (2021) 824–829.
- [150] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, Y. Liu, Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning, in: 2020 USENIX annual technical conference (USENIX ATC 20), 2020, pp. 493–506.
- [151] R. Lazzeretti, S. Horn, P. Braca, P. Willett, Secure multi-

- party consensus gossip algorithms, in: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2014, pp. 7406–7410.
- [152] N. M. Freris, P. Patrinos, Distributed computing over encrypted data, in: 2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton), IEEE, 2016, pp. 1116–1122.
- [153] H. Gao, C. Zhang, M. Ahmad, Y. Wang, Privacy-preserving average consensus on directed graphs using push-sum, in: 2018 IEEE Conference on Communications and Network Security (CNS), IEEE, 2018, pp. 1–9.
- [154] C. N. Hadjicostis, A. D. Domínguez-García, Privacy-preserving distributed averaging via homomorphically encrypted ratio consensus, *IEEE Transactions on Automatic Control* 65 (9) (2020) 3887–3894.
- [155] N. M. Hung, Y.-U. Kim, H.-S. Ahn, A novel security method for exact average consensus, *IFAC-PapersOnLine* 56 (2) (2023) 8387–8392.
- [156] Y. Yan, Z. Chen, V. Varadharajan, M. J. Hossain, G. E. Town, Distributed consensus-based economic dispatch in power grids using the paillier cryptosystem, *IEEE Transactions on Smart Grid* 12 (4) (2021) 3493–3502.
- [157] C. Zhang, M. Ahmad, Y. Wang, Admm based privacy-preserving decentralized optimization, *IEEE Transactions on Information Forensics and Security* 14 (3) (2018) 565–580.
- [158] C. Zhang, Y. Wang, Enabling privacy-preservation in decentralized optimization, *IEEE Transactions on Control of Network Systems* 6 (2) (2018) 679–689.
- [159] A. Nedic, A. Ozdaglar, P. A. Parrilo, Constrained consensus and optimization in multi-agent networks, *IEEE Transactions on Automatic Control* 55 (4) (2010) 922–938.
- [160] Y. Zhou, S. Pu, Private and accurate decentralized optimization via encrypted and structured functional perturbation, *IEEE Control Systems Letters* 7 (2022) 1339–1344.
- [161] P. Bindef, J. Adamek, N. Schlüter, M. Schulze Darup, Towards privacy-preserving cooperative control via encrypted distributed optimization, *at-Automatisierungstechnik* 71 (9) (2023) 736–747.
- [162] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, K. Ramchandran, Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning, *arXiv preprint arXiv:2009.11248* (2020).
- [163] J. M. Pollard, The fast fourier transform in a finite field, *Mathematics of computation* 25 (114) (1971) 365–374.
- [164] Y. Wang, Privacy-preserving average consensus via state decomposition, *IEEE Transactions on Automatic Control* 64 (11) (2019) 4711–4716.
- [165] K. Zhang, Z. Li, Y. Wang, A. Louati, J. Chen, Privacy-preserving dynamic average consensus via state decomposition: Case study on multi-robot formation control, *Automatica* 139 (2022) 110182.
- [166] X. Chen, L. Huang, K. Ding, S. Dey, L. Shi, Privacy-preserving push-sum average consensus via state decomposition, *IEEE Transactions on Automatic Control* (2023).
- [167] C. Altafini, A dynamical approach to privacy preserving average consensus, in: 2019 IEEE 58th Conference on Decision and Control (CDC), IEEE, 2019, pp. 4501–4506.
- [168] N. Gupta, J. Katz, N. Chandra, Privacy in distributed average consensus, *IFAC-PapersOnLine* 50 (1) (2017) 9515–9520.
- [169] N. E. Manitara, C. N. Hadjicostis, Privacy-preserving asymptotic average consensus, in: European Control Conference (ECC), 2013, pp. 760–765.
- [170] S. Zhang, T. O. Timoudas, M. Dahleh, Network consensus with privacy: A secret sharing method, *arXiv preprint arXiv:2103.03432* (2021).
- [171] S. Gade, N. H. Vaidya, Private learning on networks, *arXiv preprint arXiv:1612.05236* (2016).
- [172] N. Gupta, S. Gade, N. Chandra, N. H. Vaidya, Preserving statistical privacy in distributed optimization, *IEEE Control Systems Letters* 5 (3) (2020) 779–784.
- [173] C. Zhang, H. Gao, Y. Wang, Privacy-preserving decentralized optimization via decomposition, *arXiv preprint arXiv:1808.09566* (2018).
- [174] X. Huo, M. Liu, Privacy-preserving distributed energy resource control with decentralized cloud computing, *arXiv preprint arXiv:2301.02198* (2023).
- [175] Y. Mo, R. M. Murray, Privacy preserving average consensus, *IEEE Transactions on Automatic Control* 62 (2) (2016) 753–765.
- [176] Y. Wang, A. Nedić, Decentralized gradient methods with time-varying uncoordinated stepsizes: Convergence analysis and privacy design, *IEEE Transactions on Automatic Control* (2023).
- [177] Q. Li, R. Heusdens, M. G. Christensen, Privacy-preserving distributed optimization via subspace perturbation: A general framework, *IEEE Transactions on Signal Processing* 68 (2020) 5983–5996.
- [178] L. Lamport, The weak byzantine generals problem, *Journal of the ACM (JACM)* 30 (3) (1983) 668–676.
- [179] D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, W. E. Weihl, Reaching approximate agreement in the presence of faults, *Journal of the ACM (JACM)* 33 (3) (1986) 499–516.
- [180] R. Kieckhafer, M. Azadmanesh, Fault-tolerant convergent voting in large sparsely connected networks, in: Proc. Complex Syst Eng'g Synth. and Assessment Tech. Workshop, 1992, pp. 31–51.
- [181] H. J. LeBlanc, H. Zhang, X. D. Koutsoukos, S. Sundaram, Resilient asymptotic consensus in robust networks, *IEEE Journal on Selected Areas in Communications* 31 (2013) 766–781.
- [182] J. Feng, H. Xu, S. Mannor, Distributed robust learning, *arXiv preprint arXiv:1409.5937* (2014).
- [183] S. Sundaram, B. Gharesifard, Distributed optimization under adversarial nodes, *IEEE Transactions on Automatic Control* 64 (3) (2018) 1063–1076.
- [184] G. Ramos, D. Silvestre, C. Silvestre, A discrete-time reputation-based resilient consensus algorithm for synchronous or asynchronous communications, *IEEE Transactions on Automatic Control* (2023).
- [185] D. Yin, Y. Chen, R. Kannan, P. Bartlett, Byzantine-robust distributed learning: Towards optimal statistical rates, in: Proceedings of the 35th International Conference on Machine Learning (ICML'18), Vol. 80 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 5650–5659.
- [186] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, J. Stainer, Machine learning with adversaries: Byzantine tolerant gradient descent, in: Advances in Neural Information Processing Systems (NeurIPS'17), Vol. 30, Curran Associates, Inc., 2017.
- [187] K. Pillutla, S. M. Kakade, Z. Harchaoui, Robust aggregation for federated learning, *IEEE Transactions on Signal Processing* 70 (2022) 1142–1154.
- [188] K. Pillutla, S. M. Kakade, Z. Harchaoui, Robust aggregation for federated learning, *IEEE Transactions on Signal Processing* 70 (2022) 1142–1154.
- [189] S. P. Karimireddy, L. He, M. Jaggi, Learning from history for byzantine robust optimization, in: Proceedings of the 38th International Conference on Machine Learning (ICML'21), Vol. 139 of Proceedings of Machine Learning Research, PMLR, 2021, pp. 5311–5319.
- [190] Y. Yi, R. You, H. Liu, C. Liu, Y. Wang, J. Lv, Near-optimal resilient aggregation rules for distributed learning using 1-center

- and 1-mean clustering with outliers, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 38, 2024, pp. 16469–16477.
- [191] S. P. Karimireddy, L. He, M. Jaggi, Byzantine-robust learning on heterogeneous datasets via bucketing, in: International Conference on Learning Representations, 2021.
- [192] G. Lugosi, S. Mendelson, Sub-gaussian estimators of the mean of a random vector, *The Annals of Statistics* 47 (2) (2019) 783–794.
- [193] Y. Allouah, S. Farhadkhani, R. Guerraoui, N. Gupta, R. Pinot, J. Stephan, Fixing by mixing: A recipe for optimal byzantine ml under heterogeneity, in: Proceedings of The 26th International Conference on Artificial Intelligence and Statistics (AISTATS’23), Vol. 206 of Proceedings of Machine Learning Research, PMLR, 2023, pp. 1232–1300.
- [194] N. H. Vaidya, V. K. Garg, Byzantine vector consensus in complete graphs, in: Proceedings of the 2013 ACM symposium on Principles of distributed computing, 2013, pp. 65–73.
- [195] S. Har-Peled, T. Zhou, Improved approximation algorithms for Tverberg partitions, arXiv preprint arXiv:2007.08717 (2020).
- [196] W. Mulzer, D. Werner, Approximating Tverberg points in linear time for any fixed dimension, *Discrete & Computational Geometry* 2 (50) (2013) 520–535.
- [197] H. Park, S. A. Hutchinson, Fault-tolerant rendezvous of multirobot systems, *IEEE Transactions on Robotics* 33 (3) (2017) 565–582.
- [198] W. Abbas, M. Shabbir, J. Li, X. Koutsoukos, Resilient distributed vector consensus using centerpoint, *Automatica* 136 (2022) 110046.
- [199] A. Mukhopadhyay, S. Jadhav, Computing a centerpoint of a finite planar set of points in linear time., *Discrete & computational geometry* 12 (3) (1994) 291–312.
- [200] T. M. Chan, An optimal randomized algorithm for maximum tukey depth, in: Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2004.
- [201] G. L. Miller, D. R. Sheehy, Approximate centerpoints with proofs, *Computational Geometry: Theory and Applications* 8 (43) (2010) 647–654.
- [202] S. Farhadkhani, R. Guerraoui, N. Gupta, R. Pinot, J. Stephan, Byzantine machine learning made easy by resilient averaging of momentums, in: Proceedings of the 39th International Conference on Machine Learning (ICML’22), Vol. 162 of Proceedings of Machine Learning Research, PMLR, 2022, pp. 6246–6283.
- [203] S. P. Karimireddy, L. He, M. Jaggi, Byzantine-robust learning on heterogeneous datasets via bucketing, in: International Conference on Learning Representations (ICLR’22), 2022.
- [204] S. Sundaram, B. Gharesifard, Distributed optimization under adversarial nodes, *IEEE Transactions on Automatic Control* 64 (2019) 1063–1076.
- [205] N. Gupta, N. H. Vaidya, Fault-tolerance in distributed optimization: The case of redundancy, in: Proceedings of the 39th Symposium on Principles of Distributed Computing, 2020, pp. 365–374.
- [206] S. Liu, N. Gupta, N. H. Vaidya, Approximate Byzantine fault-tolerance in distributed optimization, in: Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing (PODC’21), ACM, 2021, pp. 379–389.
- [207] D. Alistarh, Z. Allen-Zhu, J. Li, Byzantine stochastic gradient descent, *Advances in neural information processing systems* 31 (2018).
- [208] Z. Allen-Zhu, F. Ebrahimianghzani, J. Li, D. Alistarh, Byzantine-resilient non-convex stochastic gradient descent, in: International Conference on Learning Representations (ICLR’21), 2021.
- [209] E. M. El Mhamdi, R. Guerraoui, S. L. A. Rouault, Distributed momentum for byzantine-resilient stochastic gradient descent, in: 9th International Conference on Learning Representations (ICLR), 2021.
- [210] Y. Allouah, R. Guerraoui, N. Gupta, R. Pinot, J. Stephan, On the privacy-robustness-utility trilemma in distributed learning, in: Proceedings of the 40th International Conference on Machine Learning (ICML’23), Vol. 202 of Proceedings of Machine Learning Research, PMLR, 2023.
- [211] Z. Wu, Q. Ling, T. Chen, G. B. Giannakis, Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks, *IEEE Transactions on Signal Processing* 68 (2020) 4583–4596.
- [212] X. Dong, Z. Wu, Q. Ling, Z. Tian, Byzantine-robust distributed online learning: Taming adversarial participants in an adversarial environment, *IEEE Transactions on Signal Processing* (2023).
- [213] B. Zhu, L. Wang, Q. Pang, S. Wang, J. Jiao, D. Song, M. I. Jordan, Byzantine-robust federated learning with optimal statistical rates, in: Proceedings of The 26th International Conference on Artificial Intelligence and Statistics (AISTATS’23), Vol. 206 of Proceedings of Machine Learning Research, PMLR, 2023, pp. 3151–3178.
- [214] Y. Allouah, R. Guerraoui, N. Gupta, R. Pinot, G. Rizk, Robust distributed learning: tight error bounds and breakdown point under data heterogeneity, *Advances in Neural Information Processing Systems* 36 (2024).
- [215] H. Zhu, Q. Ling, Byzantine-robust distributed learning with compression, *IEEE Transactions on Signal and Information Processing over Networks* (2023).
- [216] E. Gorbunov, S. Horváth, P. Richtárik, G. Gidel, Variance reduction is an antidote to byzinates: Better rates, weaker assumptions and communication compression as a cherry on the top, in: The Eleventh International Conference on Learning Representations, 2022.
- [217] A. Rammal, K. Gruntkowska, N. Fedin, E. Gorbunov, P. Richtárik, Communication compression for byzantine robust learning: New efficient algorithms and improved rates, arXiv preprint arXiv:2310.09804 (2023).
- [218] Y. Allouah, S. Farhadkhani, R. Guerraoui, N. Gupta, R. Pinot, G. Rizk, S. Voitovych, Tackling byzantine clients in federated learning, arXiv preprint arXiv:2402.12780 (2024).
- [219] G. Malinovsky, P. Richtárik, S. Horváth, E. Gorbunov, Byzantine robustness and partial participation can be achieved simultaneously: Just clip gradient differences, arXiv preprint arXiv:2311.14127 (2023).
- [220] A. Haseltalab, M. Akar, Approximate byzantine consensus in faulty asynchronous networks, in: American Control Conference (ACC), 2015, pp. 1591–1596.
- [221] A. Haseltalab, M. Akar, Convergence rate analysis of a fault-tolerant distributed consensus algorithm, in: 54th IEEE Conference on Decision and Control (CDC), 2015, pp. 5111–5116.
- [222] H. Mendes, M. Herlihy, N. Vaidya, V. K. Garg, Multidimensional agreement in byzantine systems, *Distributed Computing* 28 (6) (2015) 423–441.
- [223] N. H. Vaidya, Iterative byzantine vector consensus in incomplete graphs, in: Distributed Computing and Networking: 15th International Conference, ICDCN 2014, Coimbatore, India, January 4–7, 2014. Proceedings 15, Springer, 2014, pp. 14–28.
- [224] J. Yan, X. Li, Y. Mo, C. Wen, Resilient multi-dimensional consensus in adversarial environment, *Automatica* 145 (2022) 110530.
- [225] H. Zhang, E. Fata, S. Sundaram, A notion of robustness in complex networks, *IEEE Transactions on Control of Network Systems* 2 (3) (2015) 310–320.

- [226] S. M. Dibaji, H. Ishii, R. Tempo, Resilient randomized quantized consensus, *IEEE Transactions on Automatic Control* 63 (8) (2017) 2508–2522.
- [227] V. Bonagura, C. Fioravanti, G. Oliva, S. Panzieri, Resilient consensus based on evidence theory and weight correction, in: American Control Conference (ACC), 2023, pp. 393–398.
- [228] Y. Wang, H. Ishii, F. Bonnet, X. Défago, Resilient real-valued consensus in spite of mobile malicious agents on directed graphs, *IEEE Transactions on Parallel and Distributed Systems* 33 (2022) 586–603.
- [229] P. Soberón, R. Strausz, A generalisation of Tverberg's theorem, *Discrete & Computational Geometry* 47 (2012) 455–460.
- [230] C. A. Lee, W. Abbas, A geometric approach to resilient distributed consensus accounting for state imprecision and adversarial agents, *arXiv preprint arXiv:2403.09009* (2024).
- [231] L. Su, N. H. Vaidya, Byzantine-resilient multiagent optimization, *IEEE Transactions on Automatic Control* 66 (5) (2020) 2227–2233.
- [232] C. Zhao, J. He, Q.-G. Wang, Resilient distributed optimization algorithm against adversarial attacks, *IEEE Transactions on Automatic Control* 65 (10) (2019) 4308–4315.
- [233] Y. Wang, C. Liu, H. Ishii, K. H. Johansson, Resilient distributed optimization under mobile malicious attacks, *IFAC-PapersOnLine* 56 (2) (2023) 997–1002.
- [234] Z. Yang, W. U. Bajwa, Byrdie: Byzantine-resilient distributed coordinate descent for decentralized learning, *IEEE Transactions on Signal and Information Processing over Networks* 5 (4) (2019) 611–627.
- [235] C. Fang, Z. Yang, W. U. Bajwa, Bridge: Byzantine-resilient decentralized gradient descent, *IEEE Transactions on Signal and Information Processing over Networks* 8 (2022) 610–626.
- [236] K. Kuwaranancharoen, L. Xin, S. Sundaram, Byzantine-resilient distributed optimization of multi-dimensional functions, in: Proc. American Control Conference (ACC), 2020, pp. 4399–4404.
- [237] K. Kuwaranancharoen, L. Xin, S. Sundaram, Scalable distributed optimization of multi-dimensional functions despite byzantine adversaries, *IEEE Transactions on Signal and Information Processing over Networks* (2024).
- [238] Z. Wu, T. Chen, Q. Ling, Byzantine-resilient decentralized stochastic optimization with robust aggregation rules, *IEEE Transactions on Signal Processing* (2023).
- [239] C. Yang, J. Ghaderi, Byzantine-robust decentralized learning via remove-then-clip aggregation, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 38, 2024, pp. 21735–21743.
- [240] S. Yu, S. Kar, Secure distributed optimization under gradient attacks, *IEEE Transactions on Signal Processing* (2023).
- [241] M. Yemini, A. Nedić, S. Gil, A. J. Goldsmith, Resilience to malicious activity in distributed optimization for cyberphysical systems, in: IEEE 61st Conference on Decision and Control (CDC), 2022, pp. 4185–4192.
- [242] J. Zhu, Y. Lin, A. Velasquez, J. Liu, Resilient distributed optimization, in: American Control Conference (ACC), 2023, pp. 1307–1312.
- [243] X. Wang, S. Mou, S. Sundaram, A resilient convex combination for consensus-based distributed algorithms, *arXiv preprint arXiv:1806.10271* (2018).
- [244] R. Wang, Y. Liu, Q. Ling, Byzantine-resilient resource allocation over decentralized networks, *IEEE Transactions on Signal Processing* 70 (2022) 4711–4726.
- [245] R. Wang, Q. Ling, Z. Tian, D3: Dual-domain defenses for byzantine-resilient decentralized resource allocation, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2024, pp. 9331–9335.
- [246] M. Kaheni, E. Usai, M. Franceschelli, Resilient constrained optimization in multi-agent systems with improved guarantee on approximation bounds, *IEEE Control Systems Letters* 6 (2022) 2659–2664.
- [247] S. Guo, T. Zhang, H. Yu, X. Xie, L. Ma, T. Xiang, Y. Liu, Byzantine-resilient decentralized stochastic gradient descent, *IEEE Transactions on Circuits and Systems for Video Technology* 32 (6) (2021) 4096–4106.
- [248] I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, *arXiv preprint arXiv:1412.6572* (2014).
- [249] A. Kurakin, I. Goodfellow, S. Bengio, Adversarial machine learning at scale, *arXiv preprint arXiv:1611.01236* (2016).
- [250] D. Fiore, G. Russo, Resilient consensus for multi-agent systems subject to differential privacy requirements, *Automatica* 106 (2019) 18–26.
- [251] G. Ramos, S. Pequito, D. Silvestre, Reputation-based resilient consensus with differential privacy guarantees, Available at SSRN 4733736 (2024).
- [252] Y. Allouah, R. Guerraoui, J. Stephan, Can machines learn robustly, privately, and efficiently?, *arXiv preprint arXiv:2312.14712* (2023).
- [253] P. Kairouz, Z. Liu, T. Steinke, The distributed discrete gaussian mechanism for federated learning with secure aggregation, in: International Conference on Machine Learning, PMLR, 2021, pp. 5201–5212.
- [254] X. Wu, C. Liu, S. Magnússon, M. Johansson, Delay-agnostic asynchronous distributed optimization, in: 62nd IEEE Conference on Decision and Control (CDC), 2023, pp. 1082–1087.
- [255] X. Wu, C. Liu, S. Magnússon, M. Johansson, Delay-agnostic asynchronous coordinate update algorithm, in: International Conference on Machine Learning, PMLR, 2023, pp. 37582–37606.
- [256] G. Damaskinos, R. Guerraoui, R. Patra, M. Taziki, et al., Asynchronous byzantine machine learning (the case of sgd), in: International Conference on Machine Learning, PMLR, 2018, pp. 1145–1154.
- [257] Y.-R. Yang, W.-J. Li, Buffered asynchronous sgd for byzantine learning, *Journal of Machine Learning Research* 24 (204) (2023) 1–62.
- [258] G. Bianchin, A. Cenedese, M. Luvisotto, G. Michieletto, Distributed fault detection in sensor networks via clustering and consensus, in: 2015 54th IEEE Conference on Decision and Control (CDC), IEEE, 2015, pp. 3828–3833.
- [259] Q. Sun, X. Li, J. Zhang, L. Xiong, W. Liu, J. Liu, Z. Qin, K. Ren, Shapleyfl: Robust federated learning based on shapley value, in: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023, pp. 2096–2108.
- [260] J. Xiong, K. Jamieson, Securearray: Improving wifi security with fine-grained physical-layer information, in: Proceedings of the 19th annual international conference on Mobile computing & networking, 2013, pp. 441–452.
- [261] M. Yemini, A. Nedić, A. J. Goldsmith, S. Gil, Characterizing trust and resilience in distributed consensus for cyberphysical systems, *IEEE Transactions on Robotics* 38 (1) (2021) 71–91.
- [262] M. Belotti, N. Božić, G. Pujolle, S. Secci, A vademecum on blockchain technologies: When, which, and how, *IEEE Communications Surveys & Tutorials* 21 (4) (2019) 3796–3838.
- [263] Y. Miao, Z. Liu, H. Li, K.-K. R. Choo, R. H. Deng, Privacy-preserving byzantine-robust federated learning via blockchain systems, *IEEE Transactions on Information Forensics and Security* 17 (2022) 2848–2861.
- [264] V. Strobel, A. Pacheco, M. Dorigo, Robot swarms neutralize harmful byzantine robots using a blockchain-based token econ-

- omy, *Science Robotics* 8 (79) (2023).
- [265] S. Chen, H. Mi, J. Ping, Z. Yan, Z. Shen, X. Liu, N. Zhang, Q. Xia, C. Kang, A blockchain consensus mechanism that uses proof of solution to optimize energy dispatch and trading, *Nature Energy* 7 (6) (2022) 495–502.
 - [266] H. Wang, L. F. Toso, J. Anderson, Fedsysid: A federated approach to sample-efficient system identification, in: *Learning for Dynamics and Control Conference*, PMLR, 2023, pp. 1308–1320.
 - [267] I. A. Azzolini, M. Bin, L. Marconi, T. Parisini, Robust and scalable distributed recursive least squares, *Automatica* 158 (2023) 111265.
 - [268] H. Wang, L. F. Toso, A. Mitra, J. Anderson, Model-free learning with heterogeneous dynamical systems: A federated lqr approach, *arXiv preprint arXiv:2308.11743* (2023).
 - [269] L. Wang, K. Zhang, A. Zhou, M. Simchowitz, R. Tedrake, Robot fleet learning via policy merging, in: *The Twelfth International Conference on Learning Representations*.
 - [270] C. Liu, X. Tan, X. Wu, D. V. Dimarogonas, K. H. Johansson, Achieving violation-free distributed optimization under coupling constraints, *arXiv preprint arXiv:2404.07609* (2024).
 - [271] X. Tan, C. Liu, K. H. Johansson, D. V. Dimarogonas, A continuous-time violation-free multi-agent optimization algorithm and its applications to safe distributed control, *arXiv preprint arXiv:2404.07571* (2024).
 - [272] J. Coulson, J. Lygeros, F. Dörfler, Data-enabled predictive control: In the shallows of the deepc, in: *2019 18th European Control Conference (ECC)*, IEEE, 2019, pp. 307–312.