Privacy-preserving Decentralized Federated Learning over Time-varying Communication Graph

YANG LU, Lancaster University, UK
ZHENGXIN YU, Lancaster University, UK
NEERAJ SURI, Lancaster University, UK

Establishing how a set of learners can provide privacy-preserving federated learning in a fully decentralized (peer-to-peer, no coordinator) manner is an open problem. We propose the first privacy-preserving consensus-based algorithm for the distributed learners to achieve decentralized global model aggregation in an environment of high mobility, where participating learners and the communication graph between them may vary during the learning process. In particular, whenever the communication graph changes, the Metropolis-Hastings method [69] is applied to update the weighted adjacency matrix based on the current communication topology. In addition, the Shamir's secret sharing scheme [61] is integrated to facilitate privacy in reaching consensus of the global model. The paper establishes the correctness and privacy properties of the proposed algorithm. The computational efficiency is evaluated by a simulation built on a federated learning framework with a real-world dataset.

CCS Concepts: • **Security and privacy** → **Privacy-preserving protocols**; *Information-theoretic techniques*; Usability in security and privacy; • **Computer systems organization** → Peer-to-peer architectures.

Additional Key Words and Phrases: federated learning, decentralized aggregation, privacy, mobility

1 INTRODUCTION

1.1 Background and motivation

Federated learning is a collaborative machine learning technique providing privacy preservation of the individual learners' local training data [33, 45]. Each learner downloads the current global model from a centralized server, updates it by incorporating its local training data, and then sends the updated model back to the server. The server then aggregates the local models of all the individual learners to update the global model. Thus, only local training models can be observed during the training process, while raw training data do not leave their owners' devices. Given this significant feature of privacy preservation, federated learning has been applied to a wide range of applications, including wireless communications [51], autonomous driving [15], multi-access edge computing [74], smart manufacturing [10], and healthcare [70].

The traditional federated learning paradigm has two major issues. First, it requires a centralized server such that it is connected to all the local learners. In some scenarios, the learners are geographically dispersed over a large area and may lack such a connect-to-all server. In addition, the paradigm is not robust, since if the single centralized server fails, then the whole learning task cannot proceed. Second, the local training models are directly uploaded to the centralized server. As has been recently pointed out [16, 50], it is possible that private local training data can be reconstructed from local training models via model inference or inversion attacks.

Authors' addresses: Yang Lu, Lancaster University, InfoLab21, Lancaster, UK, y.lu44@lancaster.ac.uk; Zhengxin Yu, Lancaster University, InfoLab21, Lancaster, UK, z.yu8@lancaster.ac.uk; Neeraj Suri, Lancaster University, InfoLab21, Lancaster, UK, neeraj.suri@lancaster.ac.uk.

2471-2566/2023/4-ART $15.00
https://doi.org/10.1145/3591354

The above two issues necessitate new mechanisms that can achieve federated learning in a decentralized and privacy-preserving manner.

1.2 Related works

Multiple recent works have addressed the issue with fixed centralized server. Based on the technologies in achieving model aggregation, these works can be mainly categorized into two classes. The first class of works dynamically selects a learner to take the role of the centralized server [7, 56, 73]. Informally, for each round of model updates, a learner is first selected, either randomly or by following specific rules. All the other learners send their local models, possibly relayed via in-between learners, to the selected learner, who then performs model aggregation to update the global model. This approach requires all the learners to coordinate to select the learner for performing model aggregation in each round of model update. Another class of works adopts consensus-based algorithms, where the learners iteratively update their local models to reach consensus on the desired global model [37, 41, 59]. At each iteration, the learners exchange their local models only with their one-hop neighbors. In contrast to the first approach, the consensus-based approach does not require global coordination[1] between the learners and hence is easier for practical implementation. However, all these works only consider a fixed communication topology and not applicable to an environment of high mobility where the communication topology may change between successive rounds of model aggregation. In addition, all the aforementioned works directly exchange local models between the learners and thus still suffer from model inference and inversion attacks.

In this work, we develop the first privacy-preserving consensus-based decentralized federated learning algorithm that considers mobility. This is closely related to the problem of privacy-preserving consensus, where the target is to protect the privacy of the participants' initial states in the process of reaching consensus.

Existing works on privacy-preserving multi-agent consensus and machine learning can be categorized into four classes.

The first class of works uses perturbation-based approaches. An important branch of works in this class uses the technique of differential privacy [28, 29, 52, 75]. Noticeably, instead of considering the consensus-based framework for the case of a single global model, the work [4] considered the case where learners have different learning objectives and developed a decentralized differentially private machine learning scheme. Differentially private schemes add random perturbations into individuals' private data such that the participation of an individual cannot be inferred via perturbed data by an adversary with access to arbitrary auxiliary information [14]. Due to the usage of persistent random noises, there is a fundamental trade-off between privacy and utility [22, 43]. The very recent work [30] proposed a different perturbation-based approach, which, inspired by the combinatorial block design theory, partitioned learners into disjoint groups so as to minimize communications between different groups during an Alternating Direction Method of Multiplier (ADMM)-based iterative algorithm for decentralized aggregation. This approach has two limitations. First, as revealed by Theorem III.2 (and explicitly mentioned by the second bullet of the contribution statement in page 1) therein, this approach also has a fundamental trade-off between privacy and accuracy. In particular, this approach can only support a limited number of iterations, while privacy will be compromized if going beyond the limit. Second, this approach only heuristically reduces privacy leakage between different groups by reducing their communications, but does not address the privacy issue between learners in a same group, or the case where adversarial learners in different groups can collaborate with each other.

The second class of works obfuscates exchanged data by adding decaying or correlated noises, which can guarantee consensus accuracy [19, 20, 27, 44, 47]. This approach ensures that the private data cannot be uniquely

[1]By global coordination, we mean that all the learners need to participate to make certain network-wide decision.

determined. However, it still causes privacy leakage in the sense of information entropy of the private data, and the level of privacy leakage is determined by the magnitude of the noises [27].

The third class of works adopts the technique of homomorphic encryption [32, 36, 42, 63]. Informally speaking, homomorphic encryption allows certain algebraic operations to be carried out on ciphertexts, thus generating an encrypted result which, when decrypted, matches the result of operations performed on plaintexts [72]. Existing homomorphic encryption-based works require the existence of a centralized third party to carry out aggregation over ciphertexts. Hence, they are not applicable to the decentralized setting. Additionally, homomorphic encryption schemes usually incur heavy computational overheads. It is worth noting that several papers, e.g., [3, 8, 49], developed multiparty homomorphic encryption (MHE) schemes, which allow multiple parties to cooperatively generate a common public key whose private key is distributed among the parties. This enables the parties to cooperatively decrypt a ciphertext without learning anything beyond the plaintext. Please refer to Section III-A of [49] for a detailed discussion of MHE. MHE schemes have been applied to distributed learning settings where the homomorphic evaluation is carried out cooperatively by all the parties [18, 58]. While these works do not employ a peer-to-peer setting, it is promising that they can be extended to such a setting by incorporting, e.g., threshold secret sharing techniques [48].

The fourth class of works leverages state decomposition to achieve privacy-preserving consensus in a decentralized setting [57, 65]. In this approach, a scalar step size shared between two neighboring agents is constructed as a product of two scalar numbers, each randomly generated by one of the two agents and kept unknown to the other one. During the consensus algorithm, the agents exchange the product of their states and the randomly generated step size splits. Without knowing their step size splits, one agent cannot determine the values of the states of its neighbors. However, to guarantee convergence of the underlying consensus algorithm, the step size splits have to be restricted in a small interval. This will cause privacy degradation, as one can have a good estimate of the value of an agent's state by knowing the admissible interval and observing the product of the state and the step size split.

Positioning our research

To overcome the above limitations of existing works, we propose a new algorithm which integrates Shamir's secret sharing (SSS) to achieve privacy-preserving consensus-based decentralized federated learning. Informally speaking, SSS distributes a secret among a group of participants, each of whom is allocated a share of the secret. As established by Shamir [61], the secret can be reconstructed only when a sufficient number of shares are combined together, while a smaller number of shares contain no information of the secret. This technique has been widely applied to secure multiparty computation (SMC) on complete graphs [11], where each participant can communicate with each other participant. Roughly speaking, each participant sends one share of its secret to each other participant. Each participant then computes an aggregation of the shares it receives from all the other participants. When a sufficient number of aggregated results are combined, the desired aggregation of the secrets of all the participants can be reconstructed. While these approaches work well in fully connected graphs, most real-world applications entail sparse graphs, e.g., optimal resource allocation in power systems [9], multi-robot formation control [1], and distributed environmental monitoring [2]. In addition, in an environment of high mobility, the communication topology may change over time. For SMC over time-varying sparse graphs, where, at each round of computation, each participant can only communicate with its current neighbors, the above mechanisms cannot be applied. Few research has been conducted to SMC over sparse graphs. An exception is the recent work [39], which applied SSS to achieve privacy-preserving average consensus over sparse graphs. The work [39] has three major limitations. First, the approach of [39] needs to randomly activate one learner at each iteration, which requires global coordination between the learners. Second, the approach of [39] can only deal with the case where each learner has at least two neighbors and the three learners form a fully connected graph.

Third, rigorous correctness and privacy analysis are absent in [39]. The paucity of SMC research on time-varying sparse graphs motivates our work to establish fundamental results therein.

1.3 Overview of approach and contributions

This paper considers the problem of privacy-preserving decentralized federated learning over a time-varying communication graph. Specifically, we consider the case where the global training model is updated as a weighted average of the learners' local training models, and an average consensus algorithm is adopted to achieve decentralized aggregation. First, a simplified problem setting is considered, where the participating learners are fixed and the communication topology between can only change between successive training rounds. In each round of model aggregation, the Metropolis-Hastings method [69] is applied to update the weighted adjacency matrix based on the current communication topology to ensure convergence of average consensus. To protect the privacy of local training models against semi-honest learners, the learners use the Shamir's secret sharing scheme [61] to distribute their local models to their one-hop neighbors. Upon receiving the shares from its neighbors, each learner updates its model by inputting the sum of the shares it holds to the consensus algorithm. The algorithm is then extended to deal with the issues of change of participating learners and time-varying communication topology within a training round. Whenever the communication graph changes, the Metropolis-Hastings method is applied to update the weighted adjacency matrix. When a learner leaves the network, its current state is sent to one of its neighbors. The usage of the Shamir's secret sharing scheme guarantees that there is no additional privacy leakage in these operations. A proper scaling operation is exerted at the end of the consensus process to ensure correctness. The contributions of our work are fourfold.

- First, the proposed algorithm is the first that can achieve federated learning over a time-varying communication graph in a fully decentralized (without any global coordination between the learners during the iterative training process) and provably privacy-preserving manner.
- Second, in terms of privacy-preserving consensus, the proposed algorithm, for the first time, simultaneously achieves the following properties: (i) applicable to an arbitrary undirected connected communication graph without the need of a third party; (ii) no additional loss on accuracy of consensus (model aggregation) other than that caused by quantization error; (iii) no additional privacy leakage beyond the learners' own inputs (the local training models) and outputs (the updated global models); (iv) no privacy-convergence trade-off; (v) allow collaborations between adversarial learners.
- Third, the correctness and privacy properties of the proposed algorithm are rigorously analyzed. In particular, the correctness analysis addresses new challenges brought by signed real-valued models and termination of consensus iteration, and the privacy analysis addresses new challenges in potential additional privacy leakage caused by consensus process and time-varying communication topology. Please refer to Section 4.2 for detailed discussions.
- Fourth, the correctness and computational efficiency of the proposed algorithm are demonstrated by a simulation on a federated learning framework using a real-world dataset.

1.4 Organization

The rest of this paper is organized as follows. Section 2 introduces the problem statement for a simplified setting, where participating learners are fixed and the communication graph between them only varies between successive training rounds, while keeping fixed within a training round. Section 3 provides some necessary technical preliminaries. New challenges in algorithm design and analysis are identified in Section 4. The proposed algorithm for the problem setting of Section 2 is detailed in Section 5. Its correctness and privacy properties are analyzed in Section 6. An extended algorithm is developed in Section 7 to further deal with change of learners

and time-varying communication topologies within a training round. In Section 8, case studies are presented to test the performance of the proposed algorithms. Conclusions and future works are found in Section 9.

2 PROBLEM STATEMENT

In this section, we first review the framework of centralized federated learning. Next, we formulate the problem of decentralized federated learning over a time-varying communication graph and identify its privacy issue. Subsequently, we introduce the adopted attacker model and privacy definition. Finally, we clarify the objectives of the paper.

2.1 Centralized federated learning

Consider a set of N learners $\mathcal{V} \triangleq \{1, \cdots, N\}$. Each learner i holds a set of $m_i \in \mathbb{N}$ local data samples, denoted by D_i. The learners aim to collaboratively train a common global model $\theta \in \mathbb{R}^n$ over all the D_i's, where n is the dimension of the model to be trained. In federated learning, for the purpose of preserving privacy of individual D_i's, in each round t of model update, each learner i first trains a local model $\theta_i^{(t)} \in \mathbb{R}^n$ over D_i. This can be expressed as

$$\theta_i^{(t)} = \mathcal{F}_i(\theta_i^{(t,0)}, D_i), \tag{1}$$

where $\theta_i^{(t,0)} \in \mathbb{R}^n$ is the initial model for learner i's local training in round t, and $\mathcal{F}_i$ is its local training algorithm, e.g., a stochastic gradient descent-based algorithm [46].

The global model $\theta^{(t)} \in \mathbb{R}^n$ is derived by performing a weighted aggregation over all the $\theta_i^{(t)}$'s as

$$\theta^{(t)} = \sum_{i \in \mathcal{V}} w_i \theta_i^{(t)}, \tag{2}$$

where $w_i > 0$ is the weight on $\theta_i^{(t)}$. A popular choice of w_i is given by $w_i = \frac{m_i}{m}$ with $m = \sum_{i \in \mathcal{V}} m_i$, i.e., w_i is the proportion of learner i's training data in the overall training data. Notice that in an execution of Eq. (2), only the local training models $\theta_i^{(t)}$'s can be observed, while the raw training data never leave their owners' devices.

In the centralized setting, as shown by Fig. 1, each learner i uploads $\theta_i^{(t)}$ to a centralized server. Upon receiving the local models from all the learners, the centralized server updates the global model $\theta^{(t)}$ by Eq. (2) and sends $\theta^{(t)}$ to all the learners. Each learner i then sets $\theta_i^{(t+1,0)} = \theta^{(t)}$ and $t \leftarrow t + 1$, and progresses to Eq. (1) for the next round of local training.

2.2 Decentralized federated learning over a time-varying communication graph

The above centralized federated learning paradigm with a fixed centralized server is not robust and suffers from the issue of single-point failure, i.e., if the single centralized server fails, then the whole learning task cannot proceed. A popular approach to mitigate this issue is to dynamically select a learner to play the role of the centralized server for each new round of training, where it requires that each newly selected learner must be able to directly communicate with all the other learners. This approach in general works well for the settings with a fixed set of learners connected by a complete communication topology, i.e., each learner can communicate with each other learner. However, this approach may not be suitable for the following two important settings:

(i) A fixed set of learners connected by a fixed sparse communication topology where a centralized server does not exist;

(ii) A set of mobile learners, leading to time-varying communication topologies, where it is difficult or even impractical to establish a connect-to-all centralized server, due to, e.g., learners' limited communication range and high mobility.

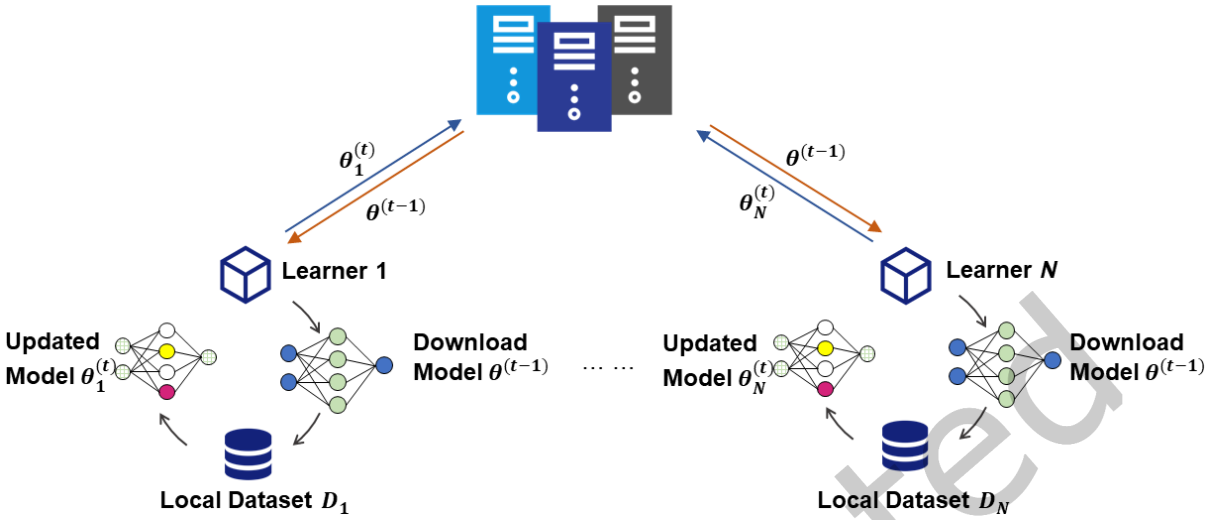


Fig. 1. Centralized federated learning.

A motivating application for setting (i) is load forecasting of distributed energy resources (DERs) in the smart grids [64]. In particular, a set of DERs, e.g., solar and wind power generators, aim to collaboratively predict future energy consumption at the consumer level. This task can be effectively formulated as a machine learning problem where future load is learned from historical data of smart meters installed at the side of consumers of participating DERs. However, data stored at smart meters must be kept confidential to the corresponding consumers, as energy-use information attached to the data act as an information-rich side channel, exposing consumer habits and behaviors. It has been shown that power load profiles at a granularity of 15 minutes may reveal whether a child is left alone at home and at a finer granularity may reveal the daily routines of consumers [24]. Federated learning is therefore a promising candidate to achieve load prediction while protecting privacy of individual consumers' smart meter data. However, in the modern smart grids, DERs are usually geographically dispersed over a large area and connected via a quite sparse communication topology without a connect-to-all entity, rendering existing federated learning schemes that rely on a centralized server (either fixed or dynamically updated) inapplicable.

A motivating application for setting (ii) is vehicular ad-hoc networks (VANET)-based high-definition (HD) mapping in autonomous driving [71]. Specifically, a fleet of vehicles aim to leverage the underlying VANET to collaboratively update the HD map of their surrounding area. This task can be formulated as a machine learning problem where the updated HD map can be learned from the image sensing data of participating vehicles. However, such image sensing data contain sensitive location trace information and must be kept confidential to their owners. The contextual information attached to location traces may significantly reveal individuals' habits, interests, activities and relationships [34]. It can also reveal their personal or corporate secrets, expose them to unwanted advertisement and location-based spams, cause social reputation or economic damage, make them victims of blackmail or even physical violence. Again, due to the privacy concern, federated learning is a promising machine learning candidate to be applied. However, the aforementioned federated learning paradigms may not be suitable because it is difficult or even impractical to establish a centralized server in VANET. First, each vehicle has a limited wireless (e.g., Wi-Fi and Bluetooth) communication range and can only talk to its neighboring vehicles which are within the range. Hence, while all the vehicles in the network may form time-varying connected

graphs, usually there is no one that is close enough to all the other vehicles at any time instant. Additionally, even for the cases where communication range is not a critical constraint (e.g., the concerned geographic area is small enough such that the vehicles' communication range is beyond the diameter of the area), establishing a stable centralized server is difficult due to vehicles' high mobility. Since individual vehicles can freely join and leave the network, the centralized server may need to be updated frequently, and, to guarantee performance, a complete communication topology should be maintained at any time instant among all the currently participating vehicles. This might be difficult due to, e.g., constraints on communication bandwidth and energy.

Decentralized model aggregation. The above discussions indicate that, in some scenarios, especially when the learners are geographically dispersed over a large area, there may not exist a centralized server that is connected to all the learners; please see Fig. 2 as an illustration. In such cases, the learners need to carry out the model aggregation Eq. (2) in a decentralized manner over the underlying communication graph between them.

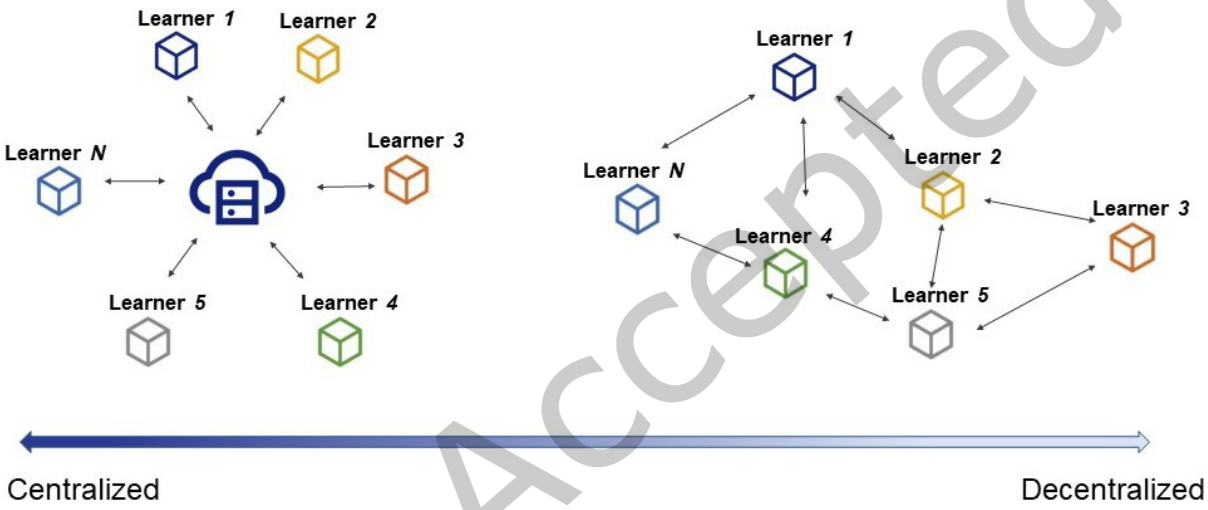


Fig. 2. Centralized aggregation vs decentralized aggregation.

Time-varying communication graph. In an environment of mobile learners, the communication topology between the learners may vary between successive rounds of model aggregation as depicted in Fig. 3.

Denote $\mathcal{G}^{(t)} = (\mathcal{V}, \mathcal{E}^{(t)})$ as the communication graph between the learners during the t-th round of model aggregation, where $\mathcal{E}^{(t)} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of communication links such that $(i, j)^{(t)} \in \mathcal{E}^{(t)}$ if and only if learner i can receive messages from learner j during the t-th round of model aggregation. Denote $\mathcal{N}_i^{(t)} \subseteq \mathcal{V}$ as the set of neighbors of learner i in $\mathcal{G}^{(t)}$, i.e., $\mathcal{N}_i^{(t)} = \{j \in \mathcal{V} \setminus \{i\} : (i, j)^{(t)} \in \mathcal{E}^{(t)}\}$. Denote $\bar{\mathcal{N}}_i^{(t)} = \mathcal{N}_i^{(t)} \cup \{i\}$. In this paper, we first study the case characterized by the following assumption on $\mathcal{G}^{(t)}$. This assumption will be relaxed in Section 7.

Assumption 2.1. *For any $t \in \mathbb{N}$, $\mathcal{G}^{(t)}$ is undirected, connected, and time invariant within the t-th round of model aggregation.*

Remark 2.1. *Assumption 2.1 covers a wide range of applications with fixed or slowly changing participating learners and communication topologies. For example, this is the case for many applications in the smart grids, where the participating learners are power generators. In such applications, the update of the participating power generators (i.e., joining of new power generators and leaving of existing ones) and the underlying communication topology*

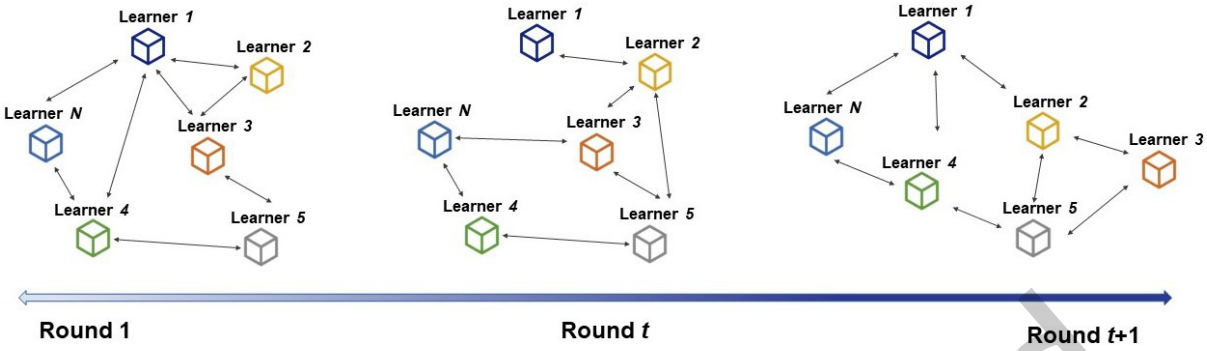


Fig. 3. Communication topology change between successive rounds of model aggregation.

usually change at a very slow time scale, because such updates will incur significant budgets on new constructions as well as many other strategic considerations. A concrete example is the problem of load forecasting of DERs mentioned in Section 2.2.

Privacy issue. During each round t of model aggregation, for each learner i, its local model $\theta_i^{(t)}$ must be kept private to itself, as breach of $\theta_i^{(t)}$ may enable an attacker to reconstruct learner i's local training data by inference or inversion attacks.

2.3 Attacker model

We consider the semi-honest attacker model, i.e., an adversarial learner correctly follows the designed algorithm but attempts to use its received data to infer others' private data ([26], pp-20). Moreover, the adversarial learners can collaborate with each other to infer the benign learners' local models. This attacker model has been widely used in various applications, e.g., privacy-preserving linear programming, dataset process and consensus [13, 17, 29]. We assume that the communication links between the learners are secure[2].

2.4 Privacy definition

As discussed above, our concerned problem is how all the learners can collaboratively compute the correct global model $\theta^{(t)}$ without disclosing their local models $\theta_i^{(t)}$'s to other learners. This is a secure multiparty computation (SMC) problem. Perfect secrecy, which will be adopted in this paper, is a standard privacy notion for SMC. Roughly speaking, an algorithm provides perfect secrecy if, after executing the algorithm, the adversarial entities only know their own inputs and outputs, but do not know anything beyond them, even if they have unlimited computing power [62]. It is worth noting that, unlike perturbation-related privacy notions, e.g., differential privacy, perfect secrecy does not induce the issue of utility-privacy trade-off.

We next provide the formal definition of perfect secrecy in the general context of SMC, where, given a set of entities $\mathcal{V}$, each entity $i \in \mathcal{V}$ has a secret input x_i and aims to compute the value of $f_i(\{x_j\}_{j\in\mathcal{V}})$. To do that, we need to introduce the notions of *perfect indistinguishability* and *view*. First, the following definition states that two distributions are perfectly indistinguishable if they follow the same distribution.

Definition 2.1 ([11]). *Let $\mathcal{X} = \{\mathcal{X}(\kappa)\}_{\kappa\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}(\kappa)\}_{\kappa\in\mathbb{N}}$ be two distribution ensembles, where, for each $\kappa \in \mathbb{N}$, $\mathcal{X}(\kappa)$ and $\mathcal{Y}(\kappa)$ are two random variables with the same probability space and the same range $R(\kappa)$. We say*

[2]Secure communication links can be enforced by cryptographic technologies such as encryption schemes.

that $\mathcal{X}$ and $\mathcal{Y}$ are perfectly indistinguishable, denoted by $\mathcal{X} \overset{p}{\equiv} \mathcal{Y}$, if the following holds

$$\sum_{r \in R(\kappa)} |\Pr[\mathcal{X}(\kappa) = r] - \Pr[\mathcal{Y}(\kappa) = r]| = 0, \quad \forall \kappa \in \mathbb{N}.$$

Next, we introduce the notion of *view*. Informally, the view of an entity is the set of all the messages the entity can see after the execution of the algorithm.

Definition 2.2 ([11, 23]). *Let Π be an algorithm for computing $f = \{f_i\}_{i \in \mathcal{V}}$. For an execution of Π on a joint input $x = \{x_i\}_{i \in \mathcal{V}}$, the view of entity i, denoted by $\text{VIEW}_i^{\Pi}(x)$, is $\text{VIEW}_i^{\Pi}(x) \triangleq \{x_i, m_1^i, \cdots, m_{t_i}^i\}$, where t_i is the total number of messages received by entity i, and for each $\ell \in \{1, \cdots, t_i\}$, m_ℓ^i is the ℓ-th message it receives.*

This provides the basis to define perfect secrecy.

Definition 2.3 ([11]). *Let Π be an algorithm for computing $f = \{f_i\}_{i \in \mathcal{V}}$. Given a joint input $x = \{x_i\}_{i \in \mathcal{V}}$, denote the joint view of the entities in a set $\mathcal{I} \subseteq \mathcal{V}$ by $\text{VIEW}_{\mathcal{I}}^{\Pi}(x)$. Let $\mathcal{A}$ be the set of adversarial learners. We say that Π provides perfect secrecy against $\mathcal{A}$ if there exists a probabilistic polynomial-time algorithm S, such that for any admissible x, it holds that*

$$S(\mathcal{A}, \{x_i\}_{i \in \mathcal{A}}, \{f_i\}_{i \in \mathcal{A}}) \overset{p}{\equiv} \text{VIEW}_{\mathcal{A}}^{\Pi}(x). \tag{3}$$

The condition Eq. (3) implies that whatever can been seen by $\mathcal{A}$ after the execution of Π can be simulated by an algorithm S using only $\mathcal{A}$'s own inputs and outputs, and $\mathcal{A}$ cannot distinguish $S(\mathcal{A}, \{x_i\}_{i \in \mathcal{A}}, \{f_i\}_{i \in \mathcal{A}})$ and $\text{VIEW}_{\mathcal{A}}^{\Pi}(x)$ even if it has unlimited computing power. In other words, the execution of Π does not provide $\mathcal{A}$ any additional information beyond what it must know, i.e., $\mathcal{A}$'s own inputs and outputs.

2.5 Design objectives

In this paper, we aim to design a privacy-preserving decentralized algorithm for the model aggregation Eq. (2) over a time-varying sparse communication graph satisfying Assumption 2.1, such that the following properties are simultaneously guaranteed:

- Correctness: For every round $t \in \mathbb{N}$, all the learners derive the correct global model $\theta^{(t)}$ given by Eq. (2), up to a quantization error (which is introduced due to usage of fixed-point arithmetic so as to apply SSS; please refer to part (i) in Section 4.2 and Remark 5.1).
- Privacy: The proposed algorithm protects the privacy of benign learners' local models $\theta_i^{(t)}$'s against semi-honest learners in the sense of perfect secrecy.

3 TECHNICAL PRELIMINARIES

In this paper, we achieve the objectives stated in Section 2.5 by integrating average consensus and Shamir's secret sharing. This section provides necessary technical preliminaries of the two techniques.

3.1 Consensus-based decentralized model aggregation

Average consensus is an effective method to achieve decentralized aggregation over sparse communication graphs. This subsection first provides preliminaries on average consensus-based decentralized model aggregation, then introduces the Metropolis-Hastings method to deal with time-varying communication graphs. More detailed discussions can be found in [60, 68, 69].

Average consensus. Roughly speaking, this method enables a set of entities over a sparse connected communication graph, each with an initial state, to iteratively interact with their neighbors and update their states, such that all the entities' states will asymptotically converge to the average of their initial states.

To apply the average consensus method, for each round t of model aggregation, the communication graph $\mathcal{G}^{(t)}$ needs to be equipped with a weighted adjacency matrix $A^{(t)} = [a_{ij}^{(t)}] \in \mathbb{R}^{N\times N}$ such that $a_{ij}^{(t)} > 0$ if $(i,j)^{(t)} \in \mathcal{E}^{(t)}$ and $a_{ij}^{(t)} = 0$ otherwise. For now, we assume that $A^{(t)}$ is given and provide the average consensus update rule and its convergence property. The construction of $A^{(t)}$ will be illustrated afterwords.

With $A^{(t)}$, to carry out the model aggregation Eq. (2) in a decentralized manner, each learner i iteratively constructs a sequence of weighted local models $\{\bar{\theta}_i^{(t)}(k)\}$, where k is the iteration index for the consensus algorithm below, such that $\bar{\theta}_i^{(t)}(0) = w_i\theta_i^{(t)}$, and the update rule is given by

$$\bar{\theta}_i^{(t)}(k+1) = a_{ii}^{(t)}\bar{\theta}_i^{(t)}(k) + \sum_{j\in\mathcal{N}_i^{(t)}} a_{ij}^{(t)}\bar{\theta}_j^{(t)}(k). \tag{4}$$

For any $k \in \mathbb{N}$, let $\bar{\theta}^{(t)}(k) = \{\bar{\theta}_i^{(t)}(k)\}_{i\in\mathcal{V}}$ be the learners' joint state at iteration k. Given an initial joint state $\bar{\theta}^{(t)}(0)$, we say that the learners asymptotically reach average consensus if all the learners' states converge to the average of their initial states as k tends to infinity, i.e.,

$$\lim_{k\to\infty} \bar{\theta}_i^{(t)}(k) = \frac{1}{N}\sum_{j\in\mathcal{V}} \bar{\theta}_j^{(t)}(0),\ \forall i \in \mathcal{V}. \tag{5}$$

If Eq. (5) is true, then each learner i's state $\bar{\theta}_i^{(t)}(k)$ will asymptotically converge to $\bar{\theta}_i^{(t)}(\infty) = \frac{1}{N}\sum_{j\in\mathcal{V}} \bar{\theta}_j^{(t)}(0) = \frac{1}{N}\sum_{j\in\mathcal{V}} w_j\theta_j^{(t)} = \frac{1}{N}\theta^{(t)}$, and hence each learner i can derive the global model $\theta^{(t)}$ by computing $N\bar{\theta}_i^{(t)}(\infty)$.

The following lemma provides a sufficient and necessary condition for reaching average consensus.

Lemma 3.1 ([68]). *With $A^{(t)}$ in each round t of model aggregation, the learners can achieve asymptotic average consensus Eq. (5) by the update rule Eq. (4) from any initial joint state $\bar{\theta}^{(t)}(0)$ if and only if the following conditions are simultaneously satisfied*

$$\rho(A^{(t)} - \frac{1}{N}1_N 1_N^T) < 1, \tag{6}$$

$$1_N^T A^{(t)} = 1_N^T, \tag{7}$$

$$A^{(t)}1_N = 1_N, \tag{8}$$

where 1_N is the N-dimensional column vector with all ones, and $\rho(\cdot)$ denotes the spectral radius[3] of a square matrix.

The intuition of Lemma 3.1 lies in that condition (6) guarantees asymptotic consensus, while conditions (7) and (8) ensure that the convergence is to the desired average point $\frac{1}{N}\sum_{j\in\mathcal{V}} \bar{\theta}_j^{(t)}(0)$.

The next question is how to construct $A^{(t)}$ that satisfies all the conditions (6)–(8). One efficient approach is the Metropolis-Hastings method, illustrated next.

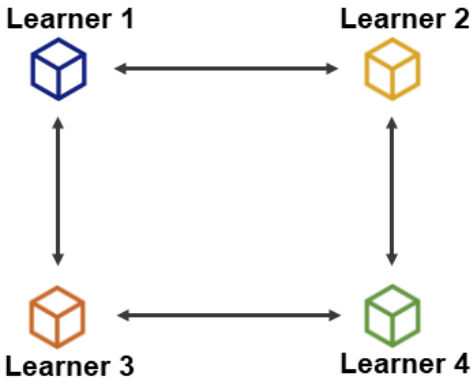
Metropolis-Hastings method. For a time-varying communication graph, the Metropolis-Hastings method [69] can be applied to update $A^{(t)}$ to ensure asymptotic average consensus. In particular, for each round t, based on its current local communication topology, each learner i constructs weights $a_{ij}^{(t)}$'s for all $j \in \bar{\mathcal{N}}_i^{(t)}$ as follows

$$a_{ij}^{(t)} = \begin{cases} \frac{1}{\max\{|\mathcal{N}_i^{(t)}|, |\mathcal{N}_j^{(t)}|\}+1} & \text{if } j \in \mathcal{N}_i^{(t)} \\ 1 - \sum\limits_{j\in\mathcal{N}_i^{(t)}} \frac{1}{\max\{|\mathcal{N}_i^{(t)}|, |\mathcal{N}_j^{(t)}|\}+1} & \text{if } j = i, \end{cases} \tag{9}$$

where $|\cdot|$ denotes the cardinality of a set.

[3]The spectral radius of a square matrix is the largest absolute value of its eigenvalues.

As an illustrative example, in Fig. 4, the figure on the left shows the communication topology between four learners, and the matrix on the right is the corresponding weighted adjacency matrix $A^{(t)}$ constructed by (9).



$$A^{(t)} = \begin{bmatrix} 1/3 & 1/3 & 1/3 & 0 \\ 1/3 & 1/3 & 0 & 1/3 \\ 1/3 & 0 & 1/3 & 1/3 \\ 0 & 1/3 & 1/3 & 1/3 \end{bmatrix}$$

Fig. 4. An example of matrix $A^{(t)}$ constructed by the Metropolis-Hastings method (9).

The following lemma states that $A^{(t)}$ constructed by (9) satisfies all the conditions of Lemma 3.1.

Lemma 3.2 ([60]). *Under Assumption 2.1, in each round t, if $A^{(t)}$ is constructed by* (9)*, then the conditions* (6)–(8) *are all satisfied.*

Under Assumption 2.1, by Lemmas 3.1 and 3.2, the update rule (4) with $A^{(t)}$ constructed by (9) ensures asymptotic average consensus (5). Notice that in running both (4) and (9), each learner only needs information from its neighbors. Hence, the aforementioned consensus-based method realizes the model aggregation (2) over a time-varying communication graph in a fully decentralized manner.

On the other hand, in implementing (4), each learner i directly sends its state $\bar{\theta}_i^{(t)}(k)$ at each consensus iteration k to its neighbors. This causes the breach of its initial state $\bar{\theta}_i^{(t)}(0)$ and of its local training model $\theta_i^{(t)}$. Hence, the privacy issue remains to be addressed for the implementation of (4).

Remark 3.1. *Notice that the construction of $A^{(t)}$ by* (9) *is completely local. That is, to construct $a_{ij}^{(t)}$, learner i only has to communicate with its neighbor $j \in \mathcal{N}_i^{(t)}$, while the construction of $a_{ij}^{(t)}$ is completely independent of all the other learners and thus does not need a global coordination. This is in contrast to the works [7, 39, 56, 73], which need to perform global coordinations to enable all the learners to collaboratively select a new learner (as the centralized server in [7, 56, 73], and as the single update entity of the current training round in [39]) at the beginning of each training round.*

3.2 Shamir's secret sharing

In the paper, we will adopt SSS to facilitate privacy in the implementation of (4). This subsection provides some preliminaries on how to use SSS to distribute a secret over a finite set of entities. More detailed discussions can be found in [11, 61].

Shares generation. To distribute a secret s over a set of entities $\mathcal{V}$, SSS uses a polynomial of degree smaller than $|\mathcal{V}|$ to generate $|\mathcal{V}|$ shares of s, one share for one entity of $\mathcal{V}$. Formally, given a prime number $p > |\mathcal{V}|$ and a positive integer $\tau < |\mathcal{V}|$, a secret $s \in \mathbb{Z}_p$ is split into $|\mathcal{V}|$ shares $\{\mathcal{H}^i\}_{i\in\mathcal{V}}$ by Algorithm 1. In the algorithm, p is the parameter to set the underlying finite field for SSS, and τ is the degree of the polynomial used to generate shares of s. After shares generation, the share $\mathcal{H}^i$ is sent to entity i for all $i \in \mathcal{V}$.

Algorithm 1: Shamir's secret shares generation

Syntax: $\{\mathcal{H}^i\}_{i\in\mathcal{V}} = \text{Alg}_{\text{ssg}}(s, p, \tau, \mathcal{V})$.
The executor selects τ scalars $c_1, \cdots, c_\tau \in \mathbb{Z}_p$ uniformly at random with $c_\tau \neq 0$, defines a polynomial $\mathcal{H}$ as $\mathcal{H}(\eta) = s + c_1\eta + \cdots + c_\tau\eta^\tau$, and computes $\mathcal{H}^i = \mathcal{H}(i) \mod p$ for all $i \in \mathcal{V}$.

Reconstruction. As given by the following lemma, the secret s can be reconstructed by collecting arbitrary $\tau + 1$ or more shares via the technique of Lagrange interpolation. This property directly follows the fact that a polynomial of degree τ can be uniquely determined by any $\tau + 1$ or more points of the polynomial.

Lemma 3.3 ([11]). *Let $(s, p, \tau, \mathcal{V})$ and $\{\mathcal{H}^i\}_{i\in\mathcal{V}}$ be a set of inputs and corresponding outputs of Algorithm 1, respectively. Then for any set $C \subseteq \mathcal{V}$ with $|C| \geq \tau + 1$, s can be reconstructed as $s = \sum_{i\in C} \mathcal{H}^i \delta_{C,i} \mod p$, where*

$$\delta_{C,i} = \prod_{j\in C, j\neq i} \frac{j}{j-i} \mod p, \ \forall i \in C. \tag{10}$$

Privacy. The privacy property of SSS is given by the following lemma, which states that the collection of any τ or less shares generated by Algorithm 1 contains no information of s. This property follows the fact that it takes at least $\tau + 1$ points to define a polynomial of degree τ.

Lemma 3.4 ([11]). *SSS provides perfect secrecy against any set $\mathcal{I} \subseteq \mathcal{V}$ such that $|\mathcal{I}| \leq \tau$.*

4 NEW CHALLENGES IN ALGORITHM DESIGN AND ANALYSIS

In this section, we first provide the high-level idea of algorithm design based on integrating SSS with average consensus. After that, we identify new challenges to a trivial integration brought by the nature of the concerned problem setting.

4.1 High-level description

As mentioned, in this paper, we achieve privacy-preserving decentralized federated learning by integrating SSS with the average consensus update rule (4). Informally speaking, to protect the privacy of $\bar{\theta}_i^{(t)}(0)$, each learner i uses a new state $s_i^{(t)}(0)$ as the initial state in executing (4). The new states $s_i^{(t)}(0)$'s need to simultaneously satisfy:

- Correctness: The average consensus point under the initial states $s_i^{(t)}(0)$'s is or can be used to locally derive the desired global model $\theta^{(t)}$.
- Privacy: The observation and the derivation process of $s_i^{(t)}(0)$'s do not disclose any information of $\bar{\theta}_i^{(t)}(0)$'s.

To this end, the learners generate $s_i^{(t)}(0)$'s via SSS, as informally illustrated as follows. First, by Algorithm 1, each learner i uses a polynomial of degree $|\mathcal{N}_i^{(t)}|$ to generate $|\mathcal{N}_i^{(t)}| + 1$ shares of $\bar{\theta}_i^{(t)}(0)$, distributes $|\mathcal{N}_i^{(t)}|$ shares to its corresponding neighbors, while keeping one share private to itself. After the exchange of shares, each learner i aggregates the $|\mathcal{N}_i^{(t)}|$ shares received from its neighbors and the one share generated and held secretly by itself to form $s_i^{(t)}(0)$, and uses it as the initial state in executing (4).

We next informally discuss the correctness and privacy intuitions of the above procedure.

- Correctness: By the convergence property of (4), all the learners can derive $\sum_{i\in\mathcal{V}} s_i^{(t)}(0)$, which is the aggregation of all the shares of all the learners' local models $\{\bar{\theta}_i^{(t)}(0)\}_{i\in\mathcal{V}}$. Notice that, by the reconstruction property of SSS, each individual $\bar{\theta}_i^{(t)}(0)$ can be reconstructed by aggregating all of its $|\mathcal{N}_i^{(t)}| + 1$ shares. Hence, $\sum_{i\in\mathcal{V}} s_i^{(t)}(0)$ can be used to reconstruct $\sum_{i\in\mathcal{V}} \bar{\theta}_i^{(t)}(0)$, which is the global model $\theta^{(t)}$.

- Privacy: By the privacy property of SSS, $\tilde{\theta}_i^{(t)}(0)$ is perfectly secret if and only if not all of its $|\mathcal{N}_i^{(t)}| + 1$ shares are known to the adversarial learners. It can be perceived that a necessary condition for this is that learner i has at least one benign neighbor.

4.2 New challenges

The last subsection presents a high-level framework based on the integration of the consensus method and SSS. However, for the concerned problem setting, a trivial integration is far from enough. In this subsection, we identify new challenges in terms of design and analysis which are critical for establishing rigorous correctness and privacy properties. Besides, we also briefly illustrate how these challenges are addressed in this paper, while the details are provided in Section 5 and Section 6.

There are four major challenges, as detailed next. Specifically, the first two are due to the real-valued setting of our problem of interest, and bring new challenges to correctness guarantee. The last two stem from more complicated information flow caused by the iterative nature of the consensus process as well as time-varying communication topology between successive training rounds, and bring new challenges to privacy analysis.

(i) Signed real-valued models. The standard SSS scheme involves modular operations and has to be implemented over non-negative integers. However, the training models in federated learning usually take signed real values. To address this mismatch, we propose a transformation between non-negative integers and signed real numbers (given by (15)). Roughly speaking, the learners transform their local models into integers and apply the procedure described in the last subsection. After the final non-negative integer-valued consensus model is derived, each agent then uses the proposed transformation to turn it back to a signed real-valued model. If the parameter p in Algorithm 1 is sufficiently large (a sufficient lower bound of p is provided by (16)), then it is guaranteed that the transformed real-valued model is the correct global model.

(ii) Termination of consensus iteration. As mentioned in the last paragraph, the learners input integer-valued models into the procedure described in Section 4.1. Hence, theoretically, the asymptotic consensus result is a non-negative integer-valued model. To reconstruct the global model by SSS, before the integer-to-real transformation, this model needs to be exerted a modulo p operation (please refer to Lemma 3.3 and (14)). However, since the convergence of the consensus update rule is only asymptotic and the weights $a_{ij}^{(t)}$'s in (4) are decimals, the intermediate results of (4) may also be decimals. Due to the subsequent modulo p operation, even if the terminating result is close to the theoretical integer-valued result, there could be a large deviation in the remainder after the modulo operation. To see this, consider the case where the terminating result is 99.4 and rounded to 99, the theoretical result is 100, and the value of p is 50. After the modulo p operation, the remainders for the terminating result and the theoretical result are 49 and 0, respectively. This shows that, compared to usual consensus applications, we need a more careful control on the termination condition. To address this challenge, we identify a sufficient lower bound of the number of consensus iterations (given by (17)) that guarantees that the absolute difference between the terminating and the theoretical results is strictly smaller than 0.5, and hence the result after the rounding operation is just the theoretical result.

(iii) Privacy leakage during consensus process. For the standard SSS-based secure sum computation over a complete communication graph, each entity receives all the other entities' shares just once and then performs an aggregation. For this standard scheme, as long as there is an honest majority (more specifically, the number of adversarial entities is no greater than the degree of the polynomial used to generated shares), then the adversarial entities cannot gain anything beyond the sum of all the entities' private inputs. However, in our case, since the communication graph is sparse, secure sum computation is further facilitated by a consensus process, where the shares need to be iteratively exchanged and aggregated according to the consensus update rule and the underlying communication topology. Such multiple rounds of communications may cause additional privacy leakage, e.g., partial sum (the sum of the local models of a subset of learners). This indicates that new privacy

analysis is needed for the consensus process. To address this challenge, we identify a graph-oriented condition, which can be used to characterize the view of the adversarial learners throughout the whole consensus process (please refer to Lemma 6.1).

(iv) Privacy property under time-varying communication topology. Besides the privacy issue caused by the consensus process, the time-varying communication topology further induces new challenges to privacy preservation. Specifically, due to time-varying communication topology, one-shot privacy preservation (privacy for one round of training) is not enough. Instead, we must establish a privacy condition with respect to the evolution of the communication topology. To address this challenge, we further extend the graph-oriented condition mentioned in the last paragraph to derive a sufficient and necessary condition under which perfect secrecy is achieved throughout the evolution of the communication topology (please refer to Theorem 6.2).

5 PRIVACY-PRESERVING DECENTRALIZED ALGORITHM DESIGN

In this section, the proposed privacy-preserving decentralized federated learning algorithm for the problem setting characterized by Assumption 2.1 is developed. First, we illustrate the design details and highlight how the challenges identified in the last section are addressed. A summary of the whole design is provided afterwards.

5.1 Design details

In this paper, we use finite precision to cope with transformations between real numbers and integers. In particular, throughout the paper, the precision level is set by $\sigma \in \mathbb{N}$, that is, for any real number, only the first σ fraction digits are kept while rest ones are dropped.

The overall design has three phases, secret shares generation of local models, consensus iteration, and global model reconstruction. The design is detailed next.

Secret shares generation of local models. All the learners first agree on a positive prime number p, which can be realized by a maximum consensus algorithm offline. We next fix a training round t and a learner i, and illustrate the secret shares generation of $\bar{\theta}_i^{(t)}(0)$.

First, for each $j \in \bar{\mathcal{N}}_i^{(t)}$, learner i computes $\delta^{(t)}_{\bar{\mathcal{N}}_i^{(t)},j}$ by (10). Then, for each $l \in \{1, \cdots, n\}$ (recall that n is the dimension of the model to be trained), learner i first transforms $\bar{\theta}_{il}^{(t)}(0)$ into an integer via multiplying by 10^σ, and then applies Algorithm 1 to use a polynomial of degree $|\mathcal{N}_i^{(t)}|$ to generate $|\bar{\mathcal{N}}_i^{(t)}| = |\mathcal{N}_i^{(t)}| + 1$ shares of integer $10^\sigma \bar{\theta}_{il}^{(t)}(0)$, denoted as $\{\mathcal{H}_{il}^{j(t)}\}_{j\in\bar{\mathcal{N}}_i^{(t)}} = \mathrm{Alg}_{\mathrm{ssg}}(10^\sigma \bar{\theta}_{il}^{(t)}(0), p, |\mathcal{N}_i^{(t)}|, \bar{\mathcal{N}}_i^{(t)})$. In light of Lemma 3.3, to facilitate later reconstruction of $10^\sigma \bar{\theta}_{il}^{(t)}(0)$, learner i further computes $\{\mathcal{S}_{il}^{j(t)}\}_{j\in\bar{\mathcal{N}}_i^{(t)}}$ as

$$\mathcal{S}_{il}^{j(t)} = \mathcal{H}_{il}^{j(t)} \delta^{(t)}_{\bar{\mathcal{N}}_i^{(t)},j} \mod p, \ \forall j \in \bar{\mathcal{N}}_i^{(t)}. \tag{11}$$

For each $j \in \bar{\mathcal{N}}_i^{(t)}$, with $\mathcal{S}_{il}^{j(t)}$ ready for all $l \in \{1, \cdots, n\}$, learner i forms $\mathcal{S}_i^{j(t)} = \{\mathcal{S}_{il}^{j(t)}\}_{l\in\{1,\cdots,n\}}$. Then learner i sends $\mathcal{S}_i^{j(t)}$ to learner j for each $j \in \mathcal{N}_i^{(t)}$, while keeping $\mathcal{S}_i^{i(t)}$ private to itself.

Consensus iteration. The learners agree on a positive integer K, which is the number of iterations for running the average consensus algorithm. Again, this can be realized by a maximum consensus algorithm offline. In each training round t, upon receiving $\mathcal{S}_j^{i(t)}$ generated as above from all of its neighbors $j \in \mathcal{N}_i^{(t)}$, each learner i constructs its new initial state $s_i^{(t)}(0)$ as

$$s_i^{(t)}(0) = \sum_{j\in\bar{\mathcal{N}}_i^{(t)}} \mathcal{S}_j^{i(t)} \mod p, \tag{12}$$

and sends $s_i^{(t)}(0)$ to learner j for all $j \in \mathcal{N}_i^{(t)}$. Then, from $k = 0$ to $k = K-1$, each learner i iteratively updates its state $s_i^{(t)}(k)$ by

$$s_i^{(t)}(k+1) = a_{ii}^{(t)} s_i^{(t)}(k) + \sum_{j \in \mathcal{N}_i^{(t)}} a_{ij}^{(t)} s_j^{(t)}(k), \tag{13}$$

and sends $s_i^{(t)}(k+1)$ to learner j for all $j \in \mathcal{N}_i^{(t)}$.

Global model reconstruction. At the end of the consensus iteration, each learner i first performs the following roundness[4] and modular operations over $s_i^{(t)}(K)$

$$z_{il}^{(t)} = \lfloor N s_{il}^{(t)}(K) \rceil \mod p, \ \forall l \in \{1, \cdots, n\}. \tag{14}$$

In (14), the rounding operation is needed to ensure perfect correctness. Specifically, with $A^{(t)}$ generated by (9), the update rule (13) ensures that $s_i^{(t)}(k)$ asymptotically converges to the point $\frac{1}{N}\sum_{j\in\mathcal{V}} s_j^{(t)}(0)$. Hence, for each $l \in \{1, \cdots, n\}$, $N s_{il}^{(t)}(k)$ asymptotically converges to the point $\sum_{j\in\mathcal{V}} s_{jl}^{(t)}(0)$, which is a non-negative integer. However, since the convergence is asymptotic, there could be a difference between $N s_{il}^{(t)}(K)$ and $\sum_{j\in\mathcal{V}} s_{jl}^{(t)}(0)$. If K is large enough such that the condition $|N s_{il}^{(t)}(K) - \sum_{j\in\mathcal{V}} s_{jl}^{(t)}(0)| < 0.5$ holds, then it is guaranteed that the rounded integer in (14) is equal to the correct consensus point, i.e., $\lfloor N s_{il}^{(t)}(K) \rceil = \sum_{j\in\mathcal{V}} s_{jl}^{(t)}(0)$. Based on this condition, a sufficient lower bound of K is given by (17) in Section 6.1.

Notice that each $z_{il}^{(t)}$ is a non-negative integer smaller than p (because it is a remainder of modulo p operation). Each learner i then transforms $z_{il}^{(t)}$ for every $l \in \{1, \cdots, n\}$ back to a signed real number as follows

$$\tilde{\theta}_{il}^{(t)} = \begin{cases} z_{il}^{(t)}/10^{\sigma}, \text{ if } 0 \le z_{il}^{(t)} \le (p-1)/2, \\ (z_{il}^{(t)} - p)/10^{\sigma}, \text{ if } (p+1)/2 \le z_{il}^{(t)} < p. \end{cases} \tag{15}$$

In (15), the divide by 10^{σ} operation transforms the integer $z_{il}^{(t)}$ into a real number $\tilde{\theta}_{il}^{(t)}$ with σ fraction digits, while the sign of $\tilde{\theta}_{il}^{(t)}$ is determined by the location of $z_{il}^{(t)}$ in the range of $[0, p)$. For sufficiently large p, the sign correctness is guaranteed. Roughly, p needs to be larger than twice of $10^{\sigma}|\theta_l^{(t)}|$, as informally explained next. Following the reconstruction property of SSS and the convergence of the consensus update rule, we should have $10^{\sigma}\theta_l^{(t)} \mod p = z_{il}^{(t)}$. The question is, given the remainder $z_{il}^{(t)}$, how to use it to reconstruct $10^{\sigma}\theta_l^{(t)}$ with the correct sign. Under the condition $p > 2 \times 10^{\sigma}|\theta_l^{(t)}|$, if $\theta_l^{(t)} \ge 0$, then the remainder $z_{il}^{(t)}$ must locate in the left half of $[0, p)$, while if $\theta_l^{(t)} < 0$, then $z_{il}^{(t)}$ must locate in the right half of $[0, p)$. Hence, conversely, as given by (15), the location of $z_{il}^{(t)}$ in $[0, p)$ can be used to correctly reconstruct the sign of $\theta_l^{(t)}$. A rigorous sufficient lower bound of p is given by (16) in Section 6.1.

Remark 5.1. *Notice that, to enable usage of SSS, fixed-point arithmetic is applied by setting the precision level σ. This will cause a quantization error, as, for each learner's local model, only the first σ fraction digits are kept while the rest are dropped. Consequently, with N learners, the quantization error in one round of model aggregation is upper bounded by $N10^{-\sigma}$. Notice that this quantization error is fundamentally different from the privacy-utility trade-off in differentially private schemes. There, the accuracy loss is caused by usage of random noises and a sufficient amount of accuracy loss is necessary to have predefined privacy level. In contrary, in our case, the precision level does not affect privacy and it can be tuned to reduce the quantization error arbitrarily small.*

[4]Given $a \in \mathbb{R}$, denote by $\lfloor a \rfloor$ the greatest integer less than or equal to a; by $\lceil a \rceil$ the least integer greater than or equal to a; and by $\lfloor a \rceil$ the roundness of a, such that $\lfloor a \rceil = \lfloor a \rfloor$ if $a - \lfloor a \rfloor < 0.5$, and $\lfloor a \rceil = \lceil a \rceil$ if $\lceil a \rceil - a \le 0.5$.

Algorithm 2: Privacy-preserving decentralized federated learning

The learners agree on a positive prime number p and two positive integers T and K; // Agreement of hyper-parameters
foreach $i \in \mathcal{V}$ **do**
  Learner i arbitrarily sets $\theta_i^{(1,0)} \in \mathbb{R}^n$; // Local training initialization
for $t = 1; t \leq T; t = t + 1$ **do**
  foreach $i \in \mathcal{V}$ **do**
    Learner i trains $\theta_i^{(t)}$ by (1); // Local training
    foreach $j \in \bar{\mathcal{N}}_i^{(t)}$ **do**
      Learner i constructs $a_{ij}^{(t)}$ by (9); // Construction of local weights in $A^{(t)}$
      Learner i constructs $\delta^{(t)}_{\bar{\mathcal{N}}_i^{(t)},j}$ by (10); // Construction of polynomials for Lagrange interpolation
    foreach $l \in \{1, \cdots, n\}$ **do**
      Learner i generates $\{\mathcal{H}_{il}^{j(t)}\}_{j\in\bar{\mathcal{N}}_i^{(t)}} = \mathrm{Alg}_{\mathrm{ssg}}(10^{\sigma}\bar{\theta}_{il}^{(t)}(0), p, |\mathcal{N}_i^{(t)}|, \bar{\mathcal{N}}_i^{(t)})$ by Algorithm 1; // Generation of secret shares of local models
      foreach $j \in \bar{\mathcal{N}}_i^{(t)}$ **do**
        Learner i computes $\mathcal{S}_{il}^{j(t)}$ by (11); // Construction of secret shares after Lagrange interpolation
    foreach $j \in \bar{\mathcal{N}}_i^{(t)}$ **do**
      Learner i forms $\mathcal{S}_i^{j(t)} = \{\mathcal{S}_{il}^{j(t)}\}_{l\in\{1,\cdots,n\}}$ and sends $\mathcal{S}_i^{j(t)}$ to learner j; // Distribution of secret shares
  foreach $i \in \mathcal{V}$ **do**
    Learner i constructs $s_i^{(t)}(0)$ by (12) and sends it to learner $j, \forall j \in \mathcal{N}_i^{(t)}$; // Construction of initial states for consensus iteration
  for $k = 0; k < K; k = k + 1$ **do**
    foreach $i \in \mathcal{V}$ **do**
      Learner i constructs $s_i^{(t)}(k+1)$ by (13) and sends it to learner $j, \forall j \in \mathcal{N}_i^{(t)}$; // Average consensus
  foreach $i \in \mathcal{V}$ **do**
    foreach $l \in \{1, \cdots, n\}$ **do**
      Learner i constructs $z_{il}^{(t)}$ by (14); // Roundness operation
      Learner i constructs $\tilde{\theta}_{il}^{(t)}$ by (15); // Transformation back to signed real numbers
    Learner i forms $\tilde{\theta}_i^{(t)} = \{\tilde{\theta}_{il}^{(t)}\}_{l\in\{1,\cdots,n\}}$; // Form the trained model as a vector
    Learner i sets $\theta_i^{(t+1,0)} = \tilde{\theta}_i^{(t)}$. // Update initial model for the next round's local training

5.2 Overall algorithm design summary

Algorithm 2 presents our overall design, with its operational steps summarized next.

At step 1, all the learners agree on three parameters. In particular, p is the parameter to set the finite field for SSS, T is the number of training rounds, and K is the number of consensus iterations in each training round. As mentioned in the last subsection, these parameters can be realized by a maximum consensus algorithm offline. At step 2, each learner i sets the initial model $\theta_i^{1,0}$ for its local training in the first round. At step 3, each learner i trains its local model $\theta_i^{(t)}$ by $\mathcal{F}_i$ with its initial model $\theta_i^{t,0}$ and dataset D_i. At step 4, based on its current local communication topology, each learner i constructs its local weights $a_{ij}^{(t)}$ in $A^{(t)}$ by the Metropolis-Hastings method. At steps 5–8, each learner i applies SSS to generate shares of $10^{\sigma}\bar{\theta}_i^{(t)}(0)$ and distributes the shares $\{\mathcal{S}_i^{j(t)}\}_{j\in\mathcal{N}_i^{(t)}}$ to its neighbors. At step 9, each learner i constructs the initial state $s_i^{(t)}(0)$ for the consensus iteration as the sum of all the shares assigned to it. At step 10, each learner i updates its state $s_i^{(t)}(k)$ by the average consensus algorithm with $A^{(t)}$. At steps 11–13, each learner i transforms the consensus model back to a signed real-valued model $\tilde{\theta}_i^{(t)}$. At step 14, each learner i sets $\tilde{\theta}_i^{(t)}$ as the initial model $\theta_i^{(t+1,0)}$ for its local training in round $t+1$.

In Algorithm 2, communications between learners incur at steps 4, 8, 9 and 10. Particularly, at step 4, each learner i sends the scalar $|\mathcal{N}_i^{(t)}|$ to each of its neighbors. Thus, at this step, each learner i sends (and also receives) $|\mathcal{N}_i^{(t)}|$ scalars in total. At each of steps 8, 9 and 10, each learner i sends an n-dimensional vector to each of its neighbors ($\mathcal{S}_i^{j(t)}$, $s_i^{(t)}(0)$, and $s_i^{(t)}(k+1)$, respectively). Thus, at each of these three steps, each learner i sends (and also receives) $|\mathcal{N}_i^{(t)}|$ n-dimensional vectors in total.

Remark 5.2. *It is worth noting that Algorithm 2 requires a one-time global coordination for determining the hyper-parameters, p, T and K, during the initialization step (step 1 of Algorithm 2), and this one-time coordination can be carried out offline. However, Algorithm 2 does not have any repeated global coordinations throughout the loop of the training phase (steps 2–14 of Algorithm 2). This is in contrast to the schemes in [7, 39, 56, 73], which incur repeated global coordinations at each round of training. The frequency of such global coordinations should be minimized as they introduce additional computational burdens and communication delays, especially for sparse communication topologies without a connect-to-all entity, where a global coordination itself usually has to be implemented via an additional consensus process.*

6 CORRECTNESS, PRIVACY, AND VULNERABILITY ANALYSIS

This section first establishes the correctness and privacy properties for Algorithm 2. After that, based on the privacy property, vulnerability analysis is further provided.

6.1 Correctness analysis

The correctness property of Algorithm 2 is established by the following theorem, which states that each learner $i \in \mathcal{V}$ derives the correct aggregated global model $\theta^{(t)}$ for each round t.

Theorem 6.1. *Suppose that Assumption 2.1 holds. By Algorithm 2, with sufficiently large p and K such that*

$$p > \max\{N, 1 + 2\times 10^{\sigma} N \max_{t,i,l} |\theta_{il}^{(t)}|\}, \tag{16}$$

$$\max_t 2p\sqrt{N}\|N(A^{(t)})^K - 1_N 1_N^T\| < 1, \tag{17}$$

where $\|\cdot\|$ the ℓ_2 norm of a matrix, it holds that $\tilde{\theta}_i^{(t)} = \theta^{(t)}$ for all $i \in \mathcal{V}$ and all $t \in \{1, \cdots, T\}$.

Proof: By Lemma 3.3, we have

$$\sum_{j\in\bar{\mathcal{N}}_i} \mathcal{S}_{il}^{j(t)} \equiv 10^{\sigma}\bar{\theta}_{il}^{(t)}(0) \mod p. \tag{18}$$

By (12), we have

$$\sum_{i\in\mathcal{V}} s_i^{(t)}(0) \equiv \sum_{i\in\mathcal{V}}\sum_{j\in\bar{\mathcal{N}}_i^{(t)}} \mathcal{S}_j^{i(t)} \mod p. \tag{19}$$

Notice that $\sum_{i\in\mathcal{V}}\sum_{j\in\bar{\mathcal{N}}_i^{(t)}} \mathcal{S}_j^{i(t)}$ is just the sum of all shares generated by all the N learners. Hence, by a rearrangement of the summation order, we have

$$\sum_{i\in\mathcal{V}}\sum_{j\in\bar{\mathcal{N}}_i^{(t)}} \mathcal{S}_j^{i(t)} = \sum_{i\in\mathcal{V}}\sum_{j\in\bar{\mathcal{N}}_i^{(t)}} \mathcal{S}_i^{j(t)}. \tag{20}$$

By (18), (19) and (20), we have

$$\sum_{i\in\mathcal{V}} s_i^{(t)}(0) \equiv \sum_{i\in\mathcal{V}} 10^{\sigma}\bar{\theta}_i^{(t)}(0) \mod p. \tag{21}$$

Fix any $l \in \{1, \cdots, n\}$. Let $s^{l(t)}(k) = \{s_{il}^{(t)}(k)\}_{i\in\mathcal{V}}$. By (13), we obtain

$$s^{l(t)}(k+1) = A^{(t)}s^{l(t)}(k), \tag{22}$$

which further leads to

$$s^{l(t)}(k) = (A^{(t)})^k s^{l(t)}(0). \tag{23}$$

Under Assumption 2.1, by Lemmas 3.1 and 3.2, $s_{il}^{(t)}(k)$ asymptotically converges to $\frac{1}{N}1_N^T s^{l(t)}(0)$, and hence $Ns_{il}^{(t)}(k)$ asymptotically converges to $1_N^T s^{l(t)}(0)$ for all $i \in \mathcal{V}$. Notice that, for all $i \in \mathcal{V}$, it holds that $0 \le s_{il}^{(t)}(0) < p$, because $s_{il}^{(t)}(0)$ is a remainder of modulo p operation derived by (12). Since $s^{l(t)}(0)$ is an N-dimensional vector, we then have

$$\|s^{l(t)}(0)\| < p\sqrt{N}. \tag{24}$$

With a slight abuse of notation, let $A_i^{(t)K}$ be the i-th row of $(A^{(t)})^K$. By (17), (23) and (24), we have

$$\begin{aligned}
&|Ns_{il}^{(t)}(K) - 1_N^T s^{l(t)}(0)| \\
&= |NA_i^{(t)K}s^{l(t)}(0) - 1_N^T s^{l(t)}(0)| && \text{(25a)}\\
&\le \|NA_i^{(t)K} - 1_N^T\|\|s^{l(t)}(0)\| && \text{(25b)}\\
&\le \|N(A^{(t)})^K - 1_N 1_N^T\|\|s^{l(t)}(0)\| && \text{(25c)}\\
&< \|N(A^{(t)})^K - 1_N 1_N^T\| p\sqrt{N} && \text{(25d)}\\
&< 0.5, && \text{(25e)}
\end{aligned}$$

where the equality (25a) is due to (23); the inequality (25b) is a well-known relationship for norm operators; the inequality (25c) is because $NA_i^{(t)K} - 1_N^T$ is the i-th row of $N(A^{(t)})^K - 1_N 1_N^T$, and the ℓ_2 norm of any one row of a matrix is no greater than that of the whole matrix; the inequality (25d) is due to (24); and the inequality (25e) is

due to (17). Notice that $1_N^T s^{l(t)}(0)$ is a non-negative integer. By (25), we then have $\lfloor N s_{il}^{(t)}(K) \rceil = 1_N^T s^{l(t)}(0) = \sum_{j\in\mathcal{V}} s_j^{(t)}(0)$. By (21), we then have

$$\lfloor N s_{il}^{(t)}(K) \rceil \equiv \sum_{j\in\mathcal{V}} 10^\sigma \bar{\theta}_{jl}^{(t)}(0) = \sum_{j\in\mathcal{V}} 10^\sigma w_j \theta_{jl}^{(t)} \mod p. \tag{26}$$

By (16), noticing that $w_j \leq 1$ for all $j \in \mathcal{V}$, we have

$$\begin{aligned} p &> 1 + 2\times 10^\sigma N \max_{t,i,l} |\theta_{il}^{(t)}| \\ &\geq 1 + 2\times 10^\sigma \sum_{j\in\mathcal{V}} |w_j \theta_{jl}^{(t)}| \\ &\geq 1 + 2\times 10^\sigma |\sum_{j\in\mathcal{V}} w_j \theta_{jl}^{(t)}|. \end{aligned} \tag{27}$$

By (27), it is either

$$0 \leq 10^\sigma \sum_{j\in\mathcal{V}} w_j \theta_{jl}^{(t)} < (p-1)/2 \tag{28}$$

or

$$-(p-1)/2 < 10^\sigma \sum_{j\in\mathcal{V}} w_j \theta_{jl}^{(t)} < 0. \tag{29}$$

In the case of (28), by (14) and (26), we have

$$z_{il}^{(t)} = 10^\sigma \sum_{j\in\mathcal{V}} w_j \theta_{jl}^{(t)} \mod p = 10^\sigma \sum_{j\in\mathcal{V}} w_j \theta_{jl}^{(t)} \tag{30}$$

By (28) and (30), we have $0 \leq z_{il}^{(t)} < (p-1)/2$. By (30) and (15), we then have

$$\tilde{\theta}_{il}^{(t)} = z_{il}^{(t)}/10^\sigma = \sum_{j\in\mathcal{V}} w_j \theta_{jl}^{(t)}. \tag{31}$$

In the case of (29), by (14) and (26), we have

$$z_{il}^{(t)} = 10^\sigma \sum_{j\in\mathcal{V}} w_j \theta_{jl}^{(t)} \mod p = p + 10^\sigma \sum_{j\in\mathcal{V}} w_j \theta_{jl}^{(t)} \tag{32}$$

By (29) and (32), we have $(p+1)/2 < z_{il}^{(t)} < p$. By (32) and (15), we then have

$$\tilde{\theta}_{il}^{(t)} = (z_{il}^{(t)} - p)/10^\sigma = \sum_{j\in\mathcal{V}} w_j \theta_{jl}^{(t)}. \tag{33}$$

By (31) and (33), we have that $\tilde{\theta}_{il}^{(t)} = \sum_{j\in\mathcal{V}} w_j \theta_{jl}^{(t)}$ always holds. The above analysis holds for all $t \in \{1,\cdots,T\}$, all $i \in \mathcal{V}$, and all $l \in \{1,\cdots,n\}$. Therefore, by (2), we have that $\tilde{\theta}_i^{(t)} = \theta^{(t)}$ for all $i \in \mathcal{V}$ and all $t \in \{1,\cdots,T\}$. This completes the proof. ■

Remark 6.1. *By the analysis above, we can see that perfect average consensus is reached after a finite K number of iterations. We note that this finite average consensus is only due to the usage of finite precision. By* (16) *and* (17)*, we can see that the bound of K increases with the value of the precision level σ. When σ tends to infinity, then K also tends to infinity, which indicates asymptotic average consensus.*

6.2 Privacy analysis

To develop the privacy property of Algorithm 2, we first introduce the following notions.

Let $\mathcal{B}$ and $\mathcal{A}$ be the sets of benign and adversarial learners, respectively. Notice that $\mathcal{B} \cup \mathcal{A} = \mathcal{V}$. Given any round $t \in \{1, \cdots, T\}$, we say that a subset $\mathcal{B}_s \subseteq \mathcal{B}$ of benign learners are surrounded by $\mathcal{A}$ in $\mathcal{G}^{(t)}$ if there exists a *connected* subgraph of $\mathcal{G}^{(t)}$ consisting of all the benign learners in $\mathcal{B}_s$ but no benign learners in $\mathcal{B}\backslash\mathcal{B}_s$ and no adversarial learners in $\mathcal{A}$, such that for each $i \in \mathcal{B}_s$, it holds that $\mathcal{N}_i^{(t)} \cap (\mathcal{B}\backslash\mathcal{B}_s) = \emptyset$. That is, for every benign learner in $\mathcal{B}_s$, all of its benign neighbors, if any, are inside $\mathcal{B}_s$. Let $\hat{\mathcal{B}}^{(t)}$ be the set containing all such sets $\mathcal{B}_s$'s in round t, i.e., $\hat{\mathcal{B}}^{(t)} = \{\mathcal{B}_s \subseteq \mathcal{B} : \text{the learners in } \mathcal{B}_s \text{ are surrounded by } \mathcal{A} \text{ in } \mathcal{G}^{(t)}\}$.

First, the following lemma establishes the *view* of the adversarial learners throughout the execution of Algorithm 2.

Lemma 6.1. *By Algorithm 2, in each round $t \in \{1, \cdots, T\}$, the adversarial learners in $\mathcal{A}$ can obtain the value of $\{\sum_{i\in\mathcal{B}_s} \tilde{\theta}_i^{(t)}(0)\}_{\mathcal{B}_s\in\hat{\mathcal{B}}^{(t)}}$, but nothing beyond it.*

Proof: Fix any $t \in \{1, \cdots, T\}$ for concreteness of illustration. Consider any $\mathcal{B}_s \in \hat{\mathcal{B}}^{(t)}$. Let $\bar{\mathcal{B}}_s$ be the complementary set of $\mathcal{B}_s$ in $\mathcal{V}$, i.e., $\bar{\mathcal{B}}_s = \mathcal{V}\backslash\mathcal{B}_s$. For each $l \in \{1, \cdots, n\}$, let $s_{\mathcal{B}_s}^{l(t)}(k) = \{s_{il}^{(t)}(k)\}_{i\in\mathcal{B}_s}$ and $s_{\bar{\mathcal{B}}_s}^{l(t)}(k) = \{s_{il}^{(t)}(k)\}_{i\in\bar{\mathcal{B}}_s}$. With a slight abuse of notation, let $A_{\mathcal{B}_s}^{(t)k}$ be the rows of $(A^{(t)})^k$ corresponding to $s_{\mathcal{B}_s}^{l(t)}(k)$. Moreover, let $A_{\mathcal{B}_s,\mathcal{B}_s}^{(t)k}$ and $A_{\mathcal{B}_s,\bar{\mathcal{B}}_s}^{(t)k}$ be the columns of $A_{\mathcal{B}_s}^{(t)k}$ corresponding to $s_{\mathcal{B}_s}^{l(t)}(k)$ and $s_{\bar{\mathcal{B}}_s}^{l(t)}(k)$, respectively. By (23), we have

$$s_{\mathcal{B}_s}^{l(t)}(k) = A_{\mathcal{B}_s}^{(t)k} s^{l(t)}(0) = A_{\mathcal{B}_s,\mathcal{B}_s}^{(t)k} s_{\mathcal{B}_s}^{l(t)}(0) + A_{\mathcal{B}_s,\bar{\mathcal{B}}_s}^{(t)k} s_{\bar{\mathcal{B}}_s}^{l(t)}(0). \tag{34}$$

By the definition of $\hat{\mathcal{B}}^{(t)}$, for any $i \in \bar{\mathcal{B}}_s$, $s_{il}^{(t)}(0)$ can only reach $s_{\mathcal{B}_s}^{l(t)}(k)$ either directly from or relayed by some learner in $\mathcal{A}$. Therefore, by knowing $A^{(t)}$, the learners in $\mathcal{A}$ can compute the value of $A_{\mathcal{B}_s,\bar{\mathcal{B}}_s}^{(t)k} s_{\bar{\mathcal{B}}_s}^{l(t)}(0)$. For any $i \in \mathcal{B}_s$ such that $\mathcal{N}_i^{(t)} \cap \mathcal{A} \neq \emptyset$, by (34), the learners in $\mathcal{A}$ can derive the value of $A_{i,\mathcal{B}_s}^{(t)k} s_{\mathcal{B}_s}^{l(t)}(0)$ as

$$A_{i,\mathcal{B}_s}^{(t)k} s_{\mathcal{B}_s}^{l(t)}(0) = s_{il}^{(t)}(k) - A_{i,\bar{\mathcal{B}}_s}^{(t)k} s_{\bar{\mathcal{B}}_s}^{l(t)}(0). \tag{35}$$

Notice that $A_{i,\mathcal{B}_s}^{(t)k} s_{\mathcal{B}_s}^{l(t)}(0)$ asymptotically converges to $\sum_{i\in\mathcal{B}_s} s_{il}^{(t)}(0)$. This implies that the learners in $\mathcal{A}$ can derive the value of $\sum_{i\in\mathcal{B}_s} s_{il}^{(t)}(0)$. For each $i \in \mathcal{B}_s$, by (12) and the definition of $\hat{\mathcal{B}}^{(t)}$, $s_{il}^{(t)}(0)$ can be written as

$$s_{il}^{(t)}(0) = \sum_{j\in\bar{\mathcal{N}}_i^{(t)}\cap\mathcal{B}_s} \mathcal{S}_{jl}^{i(t)} + \sum_{j\in\mathcal{N}_i^{(t)}\cap\mathcal{A}} \mathcal{S}_{jl}^{i(t)} \mod p. \tag{36}$$

Notice that in (36), for each $j \in \mathcal{N}_i^{(t)} \cap \mathcal{A}$, $\mathcal{S}_{jl}^{i(t)}$ is generated by the adversarial learner j. Hence, the learners in $\mathcal{A}$ know the value of $\sum_{j\in\mathcal{N}_i^{(t)}\cap\mathcal{A}} \mathcal{S}_{jl}^{i(t)}$. By also knowing the value of $\sum_{i\in\mathcal{B}_s} s_{il}^{(t)}(0)$, by (36), the learners in $\mathcal{A}$ can derive

$$\sum_{i\in\mathcal{B}_s} \sum_{j\in\bar{\mathcal{N}}_i^{(t)}\cap\mathcal{B}_s} \mathcal{S}_{jl}^{i(t)} \equiv \sum_{i\in\mathcal{B}_s} s_{il}^{(t)}(0) - \sum_{i\in\mathcal{B}_s} \sum_{j\in\mathcal{N}_i^{(t)}\cap\mathcal{A}} \mathcal{S}_{jl}^{i(t)} \mod p. \tag{37}$$

Notice that in (37), $\sum_{i\in\mathcal{B}_s} \sum_{j\in\bar{\mathcal{N}}_i^{(t)}\cap\mathcal{B}_s} \mathcal{S}_{jl}^{i(t)}$ is the sum of all those shares generated by the learners in $\mathcal{B}_s$ that are assigned to the learners in $\mathcal{B}_s$ themselves. For each $i \in \mathcal{B}_s$ and for each $j \in \mathcal{N}_i^{(t)} \cap \mathcal{A}$, $\mathcal{S}_{il}^{j(t)}$ is the share generated by learner i and assigned to the adversarial learner j. Hence, the learners in $\mathcal{A}$ know the value of $\sum_{i\in\mathcal{B}_s} \sum_{j\in\mathcal{N}_i^{(t)}\cap\mathcal{A}} \mathcal{S}_{il}^{j(t)}$, which is the sum of all those shares generated by the learners in $\mathcal{B}_s$ that are assigned to

the learners in $\mathcal{A}$. Therefore, given the definition of $\hat{\mathcal{B}}^{(t)}$, the learners in $\mathcal{A}$ can derive the sum of all the shares generated by the learners in $\mathcal{B}_s$ as

$$\sum_{i\in\mathcal{B}_s}\sum_{j\in\bar{\mathcal{N}}_i^{(t)}}\mathcal{S}_{il}^{j(t)}=\sum_{i\in\mathcal{B}_s}\sum_{j\in\bar{\mathcal{N}}_i^{(t)}\cap\mathcal{B}_s}\mathcal{S}_{jl}^{i(t)}+\sum_{i\in\mathcal{B}_s}\sum_{j\in\mathcal{N}_i^{(t)}\cap\mathcal{A}}\mathcal{S}_{il}^{j(t)}. \tag{38}$$

By the analysis below (20) in the proof of Theorem 6.1, we conclude that the learners in $\mathcal{A}$ can then derive the value of $\sum_{i\in\mathcal{B}_s}\bar{\theta}_{il}^{(t)}(0)$. The above analysis holds for any $t\in\{1,\cdots,T\}$, any $\mathcal{B}_s\in\hat{\mathcal{B}}^{(t)}$ and any $l\in\{1,\cdots,n\}$. Therefore, in each round $t\in\{1,\cdots,T\}$, the adversarial learners in $\mathcal{A}$ can obtain the value of $\{\sum_{i\in\mathcal{B}_s}\bar{\theta}_i^{(t)}(0)\}_{\mathcal{B}_s\in\hat{\mathcal{B}}^{(t)}}$.

Next we show that the learners in $\mathcal{A}$ do not gain anything beyond the value of $\{\sum_{i\in\mathcal{B}_s}\bar{\theta}_i^{(t)}(0)\}_{\mathcal{B}_s\in\hat{\mathcal{B}}^{(t)}}$. Let $\mathcal{D}_s\subseteq\mathcal{B}$ be a subset of benign learners that form a connected subgraph within themselves. It suffices to show that if $\mathcal{D}_s$ is not surrounded by $\mathcal{A}$ in $\mathcal{G}^{(t)}$, then the learners in $\mathcal{A}$ do not obtain any information about $\{\bar{\theta}_i^{(t)}(0)\}_{i\in\mathcal{D}_s}$. Since $\mathcal{D}_s\notin\hat{\mathcal{B}}^{(t)}$, there exists at least one learner $d\in\mathcal{D}_s$ such that $\mathcal{N}_d^{(t)}\cap(\mathcal{B}\backslash\mathcal{D}_s)\neq\emptyset$. Let $d'\in\mathcal{N}_d^{(t)}\cap(\mathcal{B}\backslash\mathcal{D}_s)$. We only need to consider the worst case where $(\mathcal{D}_s\cup\{d'\})\in\hat{\mathcal{B}}^{(t)}$. Similar to the derivation of (37), the learners in $\mathcal{A}$ can derive

$$\sum_{i\in\mathcal{D}_s\cup\{d'\}}\sum_{j\in\bar{\mathcal{N}}_i^{(t)}\cap(\mathcal{D}_s\cup\{d'\})}\mathcal{S}_{jl}^{i(t)}\equiv\sum_{i\in\mathcal{D}_s\cup\{d'\}}s_{il}^{(t)}(0)-\sum_{i\in\mathcal{D}_s\cup\{d'\}}\sum_{j\in\mathcal{N}_i^{(t)}\cap\mathcal{A}}\mathcal{S}_{jl}^{i(t)}\mod p. \tag{39}$$

Write the sum $\sum_{i\in\mathcal{D}_s\cup\{d'\}}\sum_{j\in\bar{\mathcal{N}}_i^{(t)}\cap(\mathcal{D}_s\cup\{d'\})}\mathcal{S}_{jl}^{i(t)}$ as

$$\sum_{i\in\mathcal{D}_s\cup\{d'\}}\sum_{j\in\bar{\mathcal{N}}_i^{(t)}\cap(\mathcal{D}_s\cup\{d'\})}\mathcal{S}_{jl}^{i(t)}=\sum_{i\in\mathcal{D}_s}\sum_{j\in\bar{\mathcal{N}}_i^{(t)}\cap\mathcal{D}_s}\mathcal{S}_{jl}^{i(t)}+\sum_{i\in\mathcal{D}_s\cap\mathcal{N}_{d'}^{(t)}}\mathcal{S}_{il}^{d'(t)}+\sum_{i\in\mathcal{D}_s\cap\mathcal{N}_{d'}^{(t)}}\mathcal{S}_{d'l}^{i(t)}+\mathcal{S}_{d'l}^{d'(t)}. \tag{40}$$

In the right-hand side of (40), the sum of the first two terms is the sum of all those shares generated by the learners in $\mathcal{D}_s$ that are assigned to the learners in $\mathcal{D}_s$ themselves and to learner d', while the sum of the last two terms is the sum of the shares generated by d' that are assigned to the learners in $\mathcal{D}_s$ and to d' itself. In order to derive the sum $\sum_{i\in\mathcal{D}_s}\bar{\theta}_{il}(0)$, the learners in $\mathcal{A}$ need to obtain the sum of the first two terms, i.e., $\sum_{i\in\mathcal{D}_s}\sum_{j\in\bar{\mathcal{N}}_i^{(t)}\cap\mathcal{D}_s}\mathcal{S}_{jl}^{i(t)}+\sum_{i\in\mathcal{D}_s\cap\mathcal{N}_{d'}^{(t)}}\mathcal{S}_{il}^{d'(t)}$. By (39) and (40), the learners in $\mathcal{A}$ know the value of the modular sum

$$\sum_{i\in\mathcal{D}_s}\sum_{j\in\bar{\mathcal{N}}_i^{(t)}\cap\mathcal{D}_s}\mathcal{S}_{jl}^{i(t)}+\sum_{i\in\mathcal{D}_s\cap\mathcal{N}_{d'}^{(t)}}\mathcal{S}_{il}^{d'(t)}+\sum_{i\in\mathcal{D}_s\cap\mathcal{N}_{d'}^{(t)}}\mathcal{S}_{d'l}^{i(t)}+\mathcal{S}_{d'l}^{d'(t)}\mod p. \tag{41}$$

However, since learner d' is benign, the learners in $\mathcal{A}$ do not know the value of $\sum_{i\in\mathcal{D}_s\cap\mathcal{N}_{d'}^{(t)}}\mathcal{S}_{d'l}^{i(t)}+\mathcal{S}_{d'l}^{d'(t)}$. Since the learners in $\mathcal{A}$ cannot split the modular sum (41), they cannot learn anything about the value of $\sum_{i\in\mathcal{D}_s}\sum_{j\in\bar{\mathcal{N}}_i^{(t)}\cap\mathcal{D}_s}\mathcal{S}_{jl}^{i(t)}+\sum_{i\in\mathcal{D}_s\cap\mathcal{N}_{d'}^{(t)}}\mathcal{S}_{il}^{d'(t)}$. By Lemma 3.4, this implies that the learners in $\mathcal{A}$ do not gain any information about $\{\bar{\theta}_i^{(t)}(0)\}_{i\in\mathcal{D}_s}$. This completes the proof. ■

Based on Lemma 6.1, the perfect secrecy property of Algorithm 2 is established by the following theorem. It states that the algorithm provides perfect secrecy if and only if all the benign learners in $\mathcal{B}$ form a connected subgraph within themselves for every round t. In other words, there is no proper subset of benign learners that are surrounded by $\mathcal{A}$ in any round t.

Theorem 6.2. *Algorithm 2 provides perfect secrecy against $\mathcal{A}$ if and only if $\hat{\mathcal{B}}^{(t)}=\{\mathcal{B}\}$ for all $t\in\{1,\cdots,T\}$.*

Proof: By Definition 2.3, if the algorithm provides perfect secrecy against $\mathcal{A}$, then, in each round t, the learners in $\mathcal{A}$ must only gain the value of $\sum_{i\in\mathcal{V}}\bar{\theta}_i^{(t)}(0)$. Notice that the learners in $\mathcal{A}$ know the sum of their own local

models, i.e., $\sum_{i\in\mathcal{A}} \bar{\theta}_i^{(t)}(0)$. Hence, they definitely can infer the sum of all the benign learners' local models $\sum_{i\in\mathcal{B}} \bar{\theta}_i^{(t)}(0)$ by computing $\sum_{i\in\mathcal{B}} \bar{\theta}_i^{(t)}(0) = \sum_{i\in\mathcal{V}} \bar{\theta}_i^{(t)}(0) - \sum_{i\in\mathcal{A}} \bar{\theta}_i^{(t)}(0)$. Therefore, the algorithm is perfectly secret if and only if the learners in $\mathcal{A}$ do not gain anything about $\{\bar{\theta}_i^{(t)}(0)\}_{i\in\mathcal{B}}$ beyond the value of $\sum_{i\in\mathcal{B}} \bar{\theta}_i^{(t)}(0)$ for all $t \in \{1, \cdots, T\}$.

First, if $\hat{\mathcal{B}}^{(t)} \neq \{\mathcal{B}\}$ for some round t, then there exists a proper subset $\mathcal{B}_s \subsetneq \mathcal{B}$ of benign learners such that $\mathcal{B}_s \in \hat{\mathcal{B}}^{(t)}$. By Lemma 6.1, the learners in $\mathcal{A}$ can then obtain the value of $\sum_{i\in\mathcal{B}_s} \bar{\theta}_i^{(t)}(0)$, which is an additional piece of information beyond $\sum_{i\in\mathcal{B}} \bar{\theta}_i^{(t)}(0)$. Hence, the algorithm is not perfectly secret.

Next, consider the case where $\hat{\mathcal{B}}^{(t)} = \{\mathcal{B}\}$ for all $t \in \{1, \cdots, T\}$. By Lemma 6.1, in each round t, the learners in $\mathcal{A}$ gain nothing beyond the value of $\{\sum_{i\in\mathcal{B}_s} \bar{\theta}_i^{(t)}(0)\}_{\mathcal{B}_s\in\hat{\mathcal{B}}^{(t)}} = \sum_{i\in\mathcal{B}} \bar{\theta}_i^{(t)}(0)$. Therefore, the algorithm provides perfect secrecy against $\mathcal{A}$. This completes the proof. ■

Remark 6.2. *The condition of Theorem 6.2, i.e., $\hat{\mathcal{B}}^{(t)} = \{\mathcal{B}\}$ for all $t \in \{1, \cdots, T\}$, ensures the strong privacy property of perfect secrecy such that the adversarial learners in $\mathcal{A}$ do not even know partial sums of the local models of any proper subset of the benign learners. It would be worth noting that, if we only target on the weaker privacy property such that each individual benign learner's local model is not disclosed to the learners in $\mathcal{A}$, then by Lemma 6.1, the condition becomes that each benign learner has at least one benign neighbor in $\mathcal{G}^{(t)}$ for all $t \in \{1, \cdots, T\}$.*

6.3 Vulnerability analysis

Lemma 6.1 and Theorem 6.2 provide generic theoretical privacy foundations for Algorithm 2. This subsection uses these results to perform vulnerability analysis for arbitrary connected communication topologies.

In light of Lemma 6.1 and Theorem 6.2, in general, a denser connection is favorable for a higher privacy level, because it is more likely that a subset of benign learners are directly connected to more other benign learners and thus less likely to be completely surrounded by adversarial learners. In terms of individual privacy, the above theoretical results indicate that the more neighbors a benign learner has, the less likely its local model will be disclosed to adversarial learners. Therefore, in a general connected (sparse) communication topology, the leaf benign learners (a leaf learner is a learner that only has one neighbor) are most vulnerable to local model leakage, while the benign learners with most neighbors are least vulnerable from a likelihood perspective. The above discussion indicates that, in terms of application, while the proposed algorithm can be applied to arbitrary connected communication topologies to enhance privacy, it is less vulnerable (more likely to achieve a high privacy level) for dense communication topologies with a small number of leaf learners.

We next apply Lemma 6.1 and Theorem 6.2 to several representative communication topologies to provide more insights on the connection between the privacy analysis and the number and position of adversarial learners.

- Complete topology. In a complete topology, any learner can communicate with any other learner. By Lemma 6.1, for any benign learner, its individual local model is not disclosed to the adversarial learners if and only if there exists at least one different benign learner in the network. In general, the adversarial learners will know the partial sum of all the benign learners' local models, but nothing else. Notice that, for a complete topology, perfect secrecy always holds. This can be seen by applying the condition of Theorem 6.2, as all the benign learners are fully connected and there is no proper subset of benign learners that are surrounded by adversarial learners. Therefore, for the extreme case where there is only one benign learner while all the other learners are adversarial, in which case the adversarial learners will know the local model of the benign learner, it does not violate the notion of perfect secrecy. Intrinsically, this is because the adversarial learners can derive the benign learner's local model solely from what they must know, i.e., their own inputs (their local models) and output (the aggregation of all local models). This is in consistent with the classical result of SMC on complete topologies.

- Star topology. In a star topology, there is one centralized learner that is connected to all the other learners, for which we call plain learners, while all the plain learners cannot communicate with each other. By Lemma 6.1, any benign plain learner's local model is not disclosed to adversarial learners if and only if the centralized learner is benign. If the centralized learner is benign, then its local model is not disclosed if and only if there exists at least one benign plain learner. In general, if the centralized learner is adversarial, then, no matter whether there are plain adversarial learners, the adversarial learners will know the local model of every benign learner. If the centralized learner is benign, then the (plain) adversarial learners will know the partial sum of the local models of all the other learners (i.e., the centralized learner and all the benign plain learners). By Theorem 6.2, perfect secrecy holds if and only if the centralized learner is benign.
- Line topology. In a line topology, all the learners are connected as a line. By Lemma 6.1, for a benign learner on one edge of the line, its local model is disclosed if and only if its only neighbor is benign. For a non-edge benign learner, its local model is not disclosed if and only if at least one of its two neighbors is benign. In general, the adversarial learners will know the partial sums of the local models of the benign learners that are either in between of two adversarial learners or on one side (till the edge) of one adversarial learner. By Theorem 6.2, perfect secrecy holds if and only if there is no non-edge adversarial learner.

Remark 6.3. *The above analysis indicates that, in some cases, if the total number of benign learners is very small, then the disclosed partial sum only consists of the local models of very few benign learners and this may be taken as unacceptable. For the extreme case where there is only one benign learner while all the other learners are adversarial, then, even under perfect secrecy, the adversarial learners will know the local model of the single benign learner. However, we first note that this is not a new issue caused by decentralized computing on sparse communication topologies, but exists for standard SMC on complete communication topologies. Moreover, this issue is not due to poor design of SMC algorithms, but is a fundamental limitation for SMC that targets perfect correctness. In particular, under the requirement of perfect correctness, the adversarial learners will finally know the correct global model, which is the sum of the local models of all the learners. They can then derive the sum of the local models of all the benign learners by subtracting the sum of their own local models from the global model. Therefore, for applications where perfect correctness is critical, such fundamental limitation on privacy is unavoidable. In practice, the above vulnerability analysis can be used to guide evaluation of suitableness of the proposed algorithm based on application-specific privacy requirements.*

7 TIME-VARYING COMMUNICATION TOPOLOGY WITHIN A TRAINING ROUND

In Algorithm 2, we assume that, within each training round, the participating learners and the communication topology between them are both fixed; please refer to Assumption 2.1. However, this assumption could be restrictive for applications with highly mobile learners, especially when K needs to be large to ensure consensus convergence. In such cases, during the consensus process (step 10 of Algorithm 2), the communication topology may change and some learners may leave the network. This will degrade the correctness of consensus convergence. Especially for the problems where the learners hold non-IID (independent and identically distributed) data, one learner's leaving may cause a significant drift of training data distribution, which may cause a huge deviation on the derived global model. In this section, we provide an extension of Algorithm 2 to relax Assumption 2.1.

7.1 Relaxed problem setting

The relaxed problem setting is stated as follows. In each training round, the communication topology can change at any iteration k during the consensus process. Moreover, participating learners may leave the network and drop the training task at certain consensus iterations. We assume that, at each consensus iteration, the current communication topology between the remaining learners is undirected and connected. Additionally, we require that, new learners can only join in between successive training rounds, while once a training round begins, no

new learners can join in before this training round terminates. This is in consistent with many robust learning works, which only consider potential dropping of existing learners. A further discussion on joining of new learners is provided at the end of this section.

7.2 Extended algorithm

We next illustrate how to extend Algorithm 2 for the above problem setting. In the following, for a time-varying quantity d, we use the notation $d^{(t)}(k)$ to denote its value at consensus iteration k of training round t.

- At step 10 of Algorithm 2, for each consensus iteration k:
  - (i) Each learner i updates its local weights $a_{ij}^{(t)}(k)$ by (9) based on the current communication topology;
  - (ii) Each learner i sends $s_i^{(t)}(k)$ to all of its current neighbors $\mathcal{N}_i^{(t)}(k)$;
  - (iii) Each learner i updates its state by $s_i^{(t)}(k+1) = a_{ii}^{(t)}(k)s_i^{(t)}(k) + \sum_{j\in\mathcal{N}_i^{(t)}(k)} a_{ij}^{(t)}(k)s_j^{(t)}(k)$;
  - (iv) If a learner i will leave the network and drop the training task in the next iteration $k+1$, then, learner i arbitrarily picks one of its current neighbors $j \in \mathcal{N}_i^{(t)}(k)$ and sends $s_i^{(t)}(k+1)$ together with a message, e.g., "leaving", to learner j. Learner j then sets $s_j^{(t)}(k+1) = s_j^{(t)}(k+1) + s_i^{(t)}(k+1)$.
- At step 11 of Algorithm 2:
  - (v) Each learner i constructs $z_{il}^{(t)}$ by $z_{il}^{(t)} = \lfloor N^{(t)}(K)s_{il}^{(t)}(K)\rceil \mod p$, $\forall l \in \{1,\cdots,n\}$.

The key ideas of the above steps are summarized as follows. Steps (i)–(iii): Whenever the communication topology changes, the learners update the weighted adjacency matrix and use the new matrix to update their states. For a fixed set of learners, this strategy guarantees that each learner's state will asymptotically converge to the correct point, i.e., the average of all the learners' initial states. Step (iv): Whenever a learner leaves, its state is added into one of its neighbor's state and thus maintained in the remaining consensus process. By this strategy, it guarantees that, if the consensus process converges at iteration K, the converging point is $\frac{1}{N^{(t)}(K)}\sum_{i\in\mathcal{V}^{(t)}(0)} s_i^{(t)}(0)$, i.e., the sum of the initial states of all the learners at iteration 0 divided by the number of learners at iteration K. Step (v): After the consensus process terminates, each learner scales its final state by $N^{(t)}(K)$ to obtain the desired aggregation point $\sum_{i\in\mathcal{V}^{(t)}(0)} s_i^{(t)}(0)$.

Remark 7.1. *The above step (iv) can be easily adjusted to cope with different practical situations. For example, if the picked neighbor will also leave the network in the next iteration, learner i can then arbitrarily pick another neighbor. This is repeated until a neighbor that will stay in the next iteration is found. Moreover, if all of learner i's neighbors will also leave the network in the next iteration, then learner i first sends its state $s_i^{(t)}(k+1)$ to one of its neighbor j, and then learner j sends the updated state $s_j^{(t)}(k+1) + s_i^{(t)}(k+1)$ to one of its neighbors different from learner i. This process proceeds until a learner that will stay in the next iteration is found.*

The above extended algorithm assumes that, whenever a learner leaves the network, it is able to take proper actions as required by step (iv). These actions guarantee that the converging point of the remaining learners is not affected by its leaving. In practice, there could be scenarios where a learner unintentionally drops out of the network without being able to take the required actions, or an adversarial learner purposely leaves the network without taking the actions. Such scenarios need to be investigated under an active attacker model and addressed by robust and resilient learning technologies, and are thus beyond the scope of this paper. We leave the study of these scenarios to our future works.

7.3 Correctness and privacy analysis

We next provide the correctness and privacy analysis for the above extended algorithm.

7.3.1 Correctness. The correctness property is characterized by the following theorem.

Theorem 7.1. *Consider the problem setting of Section 7.1. By the extended algorithm of Section 7.2, if, at each training round t, there exists $\bar{K}^{(t)}$ such that $\mathcal{V}^{(t)}(k) \equiv \mathcal{V}^{(t)}(\bar{K}^{(t)})$ for all $k \geq \bar{K}^{(t)}$, i.e., the remaining learners keep constant from consensus iteration $\bar{K}^{(t)}$ (while the communication topology between them can still change), then, with sufficiently large p and K such that*

$$p > \max\{N^{(t)}(0), 1 + 2 \times 10^{\sigma} N^{(t)}(0) \max_{t,i,l} |\theta_{il}^{(t)}|\}, \tag{42}$$

$$K > \max_t \bar{K}^{(t)} + \max_t \hat{K}^{(t)}, \tag{43}$$

where, for each training round t, $\hat{K}^{(t)}$ satisfies

$$\max_t 2p\sqrt{N^{(t)}(\bar{K}^{(t)})} \| N^{(t)}(\bar{K}^{(t)}) \prod_{k=0}^{\hat{K}^{(t)}-1} A^{(t)}(\bar{K}^{(t)} + k) - 1_{N^{(t)}(\bar{K}^{(t)})} 1_{N^{(t)}(\bar{K}^{(t)})}^T \| < 1, \tag{44}$$

it holds that $\tilde{\theta}_i^{(t)} = \theta^{(t)}$ for all $i \in \mathcal{V}^{(t)}(K)$ and all $t \in \{1, \cdots, T\}$.

Proof: We fix a training round t. At each consensus iteration k, by the update rule given above at step (iii), we have

$$\begin{aligned}
\sum_{i \in \mathcal{V}^{(t)}(k)} s_i^{(t)}(k+1) &= \sum_{i \in \mathcal{V}^{(t)}(k)} [a_{ii}^{(t)}(k) s_i^{(t)}(k) + \sum_{j \in \mathcal{N}_i^{(t)}(k)} a_{ij}^{(t)}(k) s_j^{(t)}(k)] \\
&= \sum_{i \in \mathcal{V}^{(t)}(k)} a_{ii}^{(t)}(k) s_i^{(t)}(k) + \sum_{i \in \mathcal{V}^{(t)}(k)} \sum_{j \in \mathcal{N}_i^{(t)}(k)} a_{ij}^{(t)}(k) s_j^{(t)}(k) \\
&= \sum_{i \in \mathcal{V}^{(t)}(k)} a_{ii}^{(t)}(k) s_i^{(t)}(k) + \sum_{i \in \mathcal{V}^{(t)}(k)} [\sum_{j \in \mathcal{N}_i^{(t)}(k)} a_{ij}^{(t)}(k)] s_i^{(t)}(k) \\
&= \sum_{i \in \mathcal{V}^{(t)}(k)} [a_{ii}^{(t)}(k) + \sum_{j \in \mathcal{N}_i^{(t)}(k)} a_{ij}^{(t)}(k)] s_i^{(t)}(k) \\
&= \sum_{i \in \mathcal{V}^{(t)}(k)} s_i^{(t)}(k),
\end{aligned} \tag{45}$$

where the last equality is due to the property of $A^{(t)}(k)$ given by (8).

By the above step (iv), at the end of each consensus iteration k, if a learner i will leave the network in the next iteration $k+1$, then, it picks an arbitrary neighbor j and $s_j^{(t)}(k+1)$ is updated by $s_j^{(t)}(k+1) = s_j^{(t)}(k+1) + s_i^{(t)}(k+1)$. This operation guarantees that, at the beginning of iteration $k+1$, it holds that

$$\sum_{i \in \mathcal{V}^{(t)}(k+1)} s_i^{(t)}(k+1) = \sum_{i \in \mathcal{V}^{(t)}(k)} s_i^{(t)}(k+1). \tag{46}$$

By (45) and (45), we have

$$\sum_{i \in \mathcal{V}^{(t)}(k+1)} s_i^{(t)}(k+1) = \sum_{i \in \mathcal{V}^{(t)}(k)} s_i^{(t)}(k). \tag{47}$$

Since (47) holds for any k, we obtain

$$\sum_{i \in \mathcal{V}^{(t)}(k)} s_i^{(t)}(k) = \sum_{i \in \mathcal{V}^{(t)}(0)} s_i^{(t)}(0), \quad \forall k. \tag{48}$$

By Theorem 8.3 of [53], if the dimension of $A^{(t)}(k)$ is fixed from some iteration $k = \bar{K}^{(t)}$, then, it holds that

$$\lim_{K\to\infty} N^{(t)}(\bar{K}^{(t)}) \prod_{k=\bar{K}^{(t)}}^{K} A^{(t)}(k) = 1_{N^{(t)}(\bar{K}^{(t)})} 1^{T}_{N^{(t)}(\bar{K}^{(t)})}. \tag{49}$$

By (49), if p and K satisfy (42), (43) and (44), by following a similar procedure of the proof of Theorem 6.1, we can derive

$$s_i^{(t)}(K) = \frac{1}{N^{(t)}(\bar{K}^{(t)})} \sum_{j\in\mathcal{V}^{(t)}(\bar{K}^{(t)})} s_j^{(t)}(\bar{K}^{(t)}), \quad \forall i \in \mathcal{V}^{(t)}(\bar{K}^{(t)}). \tag{50}$$

By (48) and (50), noticing that $N^{(t)}(K) = N^{(t)}(\bar{K}^{(t)})$ and $\mathcal{V}^{(t)}(K) = \mathcal{V}^{(t)}(\bar{K}^{(t)})$, we obtain

$$s_i^{(t)}(K) = \frac{1}{N^{(t)}(K)} \sum_{j\in\mathcal{V}^{(t)}(0)} s_j^{(t)}(0), \quad \forall i \in \mathcal{V}^{(t)}(K). \tag{51}$$

By the above step (v), we then have

$$z_i^{(t)} = \lfloor N^{(t)}(K) s_i^{(t)}(K) \rceil \mod p = \sum_{j\in\mathcal{V}^{(t)}(0)} s_j^{(t)}(0) \mod p, \quad \forall i \in \mathcal{V}^{(t)}(K). \tag{52}$$

The remaining proof follows the proof of Theorem 6.1. ■

7.3.2 Privacy. The privacy property is characterized by the following theorem.

Theorem 7.2. *The results of Lemma 6.1 and Theorem 6.2 hold for the extended algorithm of Section 7.2, where the conditions are exerted on the communication topology at the phase of secret shares generation and sharing for each training round.*

Proof: The proof of Lemma 6.1 indicates that the privacy property is fully determined by the communication topology at the phase of secret shares generation and sharing. This is because, if a benign learner (or a set of benign learners) has at least one benign neighbor in this phase, then, one of its shares is coupled with one of this benign neighbor's shares and cannot be split in any subsequent operations. Since the extensions of Section 7.2 only modify the consensus process (more specifically, steps 10 and 11 of Algorithm 2), they do not change the property property. ■

7.4 Discussion on joining of new learners within a training round

For the case where new learners are allowed to join in within a training round, a straightforward extension is to have all the learners, including both existing and new ones, to generate shares of their current data and use the new shares to start a new consensus process. More specifically, in a training round t, if new learners join in at consensus iteration k, then, these new learners generate shares of their local models (i.e., $10^{\sigma}\bar{\theta}_i^{(t)}(0)$), while the existing learners generate shares of their current states (i.e., $s_i^{(t)}(k)$). The learners then exchange the shares with their current neighbors and form the initial states for the new consensus process. It is easy to see that convergence correctness is guaranteed. In particular, in the proof of Theorem 7.1, we have shown that, for any consensus iteration k, the sum of $s_i^{(t)}(k)$ over all the current learners is always equal to the sum of $s_i^{(t)}(0)$ over all the learners at the beginning of the training round. Hence, with new learners joining in, the new converging point will be the average of the sum of all the current learners' new shares, which is just the sum of all the current learners' local models, i.e., the new desired global model. However, this straightforward extension does not provide desired privacy guarantee. This is because, for the existing learners, their states $s_i^{(t)}(k)$'s are directly shared with their neighbors and thus are not secrets any more. Therefore, even if a new learner i's neighbors at

the phase of secret shares generation and sharing are all benign, it is still possible that the adversarial learners can first obtain all its neighbors' other shares (those not for learner i), and then infer its neighbors' shares for learner i (by knowing its neighbors' states), and further infer all of learner i's shares and reconstruct its local model. In this paper, for change of participating learners within a training round, we focus on the issue of learner leaving, which is usually more critical in practice, and leave the issue of learner joining to our future works.

8 PERFORMANCE EVALUATION

This section provide comprehensive simulations to test the performance of Algorithm 2 and the extended algorithm of Section 7.2. It is worth noting that simulations of this section are mainly for the purpose of proof-of-concept, i.e., experimentally verifying the convergence results established by Theorem 6.1 and Theorem 7.1.

8.1 Simulation setup

Environment. The simulation environment is as follows. On the hardware side, the simulation is performed on a Lenovo ThinkPad laptop computer with Intel(R) Core(TM) i5-1135G7 CPU at 2.40 GHz. On the software side, the simulation is performed on MATLAB R2021b.

Dataset. The dataset we use in the simulation is MNIST [12], which is a large-scale dataset of handwritten digits that is broadly used for training various image processing systems. It has a training set of 60000 samples and a testing set of 10000 samples. Each data sample has 784 features and 1 label. The simulation uses the set of 60000 training data samples and evenly distributes them over 100 learners. For each data sample, the label is removed. Hence, each learner has 600 local training data samples and each data sample consists of 784 features.

ML model for local training. In each round t, each learner i uses an autoencoder to train its local model $\theta_i^{(t)}$. An autoencoder is an unsupervised learning algorithm for neural networks to learn efficient codings of unlabeled data. A significant advantage of autoencoder is that it can greatly reduce the noise of input data, leading to much more efficient creation of deep learning models. Due to this advantage, it has been applied to various problems, e.g., dimensionality reduction [66], feature detection [25], image recognition [40], defense against backdoor attacks [35], and anomaly detection [76]. An autoencoder has two parts, an encoder that compresses the input into a latent space representation, and a decoder that maps this representation to a reconstruction of the input. With h hidden layers, in each round t, each learner i's encoder consists of an $h \times 784$ weight matrix $W_{i(e)}^{(t)}$ and an $h \times 1$ bias vector $b_{i(e)}^{(t)}$, and its decoder consists of a $784 \times h$ weight matrix $W_{i(d)}^{(t)}$ and a 784×1 bias vector $b_{i(d)}^{(t)}$. The local model $\theta_i^{(t)}$ is constructed by stacking all the entries of $W_{i(e)}^{(t)}$, $b_{i(e)}^{(t)}$, $W_{i(d)}^{(t)}$ and $b_{i(d)}^{(t)}$ into a single column vector. Therefore, the dimension of $\theta_i^{(t)}$ is $n = h \times 784 + h + 784 \times h + 784 = 1569h + 784$. Notice that verification of correctness can also be conducted on other neural network architectures, e.g., feed-forward neural networks and convolutional neural networks, in a similar manner. To avoid redundancy, in the simulations, we stick to autoencoder for learners' local training.

8.2 Simulation results for Algorithm 2

We first test the performance of Algorithm 2. For the simulations in this subsection, the learners are fixed throughout the training process and the communication topology between can only change between successive training rounds. In the simulation, we set $\sigma = 2$, $p = 1020431$, and $T = 6$. For the correctness verification, we set $K = 10$. We have verified that the conditions given by (16) and (17) are both satisfied.

We first verify the correctness property of Algorithm 2. Here we use one hidden layer for each learner's local training, which leads to $n = 2353$. To simulate time-varying communication topology, for each round t, an arbitrary communication topology satisfying Assumption 2.1 is applied. First, we verify correct average consensus at each training round. To this end, we pick an arbitrary $l \in \{1, \cdots, n\}$ for the illustration. In each

training round t, for each $i \in \mathcal{V}$ and each $k \in \{0, \cdots, K\}$, with $s_{il}^{(t)}(k)$ generated by (13), we construct $z_{il}^{(t)}(k)$ by (14) with $s_{il}^{(t)}(K)$ replaced by $s_{il}^{(t)}(k)$, and then construct $\tilde{\theta}_{il}^{(t)}(k)$ by (15) with $z_{il}^{(t)}$ replaced by $z_{il}^{(t)}(k)$. Notice that $z_{il}^{(t)} = z_{il}^{(t)}(K)$ and $\tilde{\theta}_{il}^{(t)} = \tilde{\theta}_{il}^{(t)}(K)$. For each $t = 1, \cdots, 6$, the trajectories of $\tilde{\theta}_{il}^{(t)}(k)$ for all $i \in \mathcal{V}$ are sequentially shown in Fig. 5. We can see that, for each round t, all the 100 trajectories converge to a same value. Indeed, for each $t = 1, \cdots, 6$, we have verified that **all the 100 trajectories converge to the correct value of the desired global sum** $\theta^{l(t)} = \sum_{i \in \mathcal{V}} w_i \theta_{il}^{(t)}$, i.e., at $k = K = 10$, $\tilde{\theta}_{il}^{(t)}(10) = \theta^{l(t)}$ for all $i \in \mathcal{V}$. To further verify correctness of convergence, we also plot in Fig. 6 the trajectories of $\max_{i \in \mathcal{V}, l \in \{1, \cdots, n\}} |\tilde{\theta}_{il}^{(t)} - \theta^{l(t)}|$ for $t = 1, \cdots, 6$. In Fig. 6, for each training round t, the trajectory of $\max_{i \in \mathcal{V}, l \in \{1, \cdots, n\}} |\tilde{\theta}_{il}^{(t)} - \theta^{l(t)}|$ converges to zero. This verifies that $\tilde{\theta}_{il}^{(t)}$ **converges to the correct value of the desired global model** $\theta^{l(t)}$ **for all** $i \in \mathcal{V}$ **and all** $l \in \{1, \cdots, n\}$. Moreover, we pick an arbitrary $i \in \mathcal{V}$ and plot the trajectories of $\tilde{\theta}_{il}^{(t)}$ and $|\tilde{\theta}_{il}^{(t)} - \theta^{l(t)}|$ for three arbitrarily picked l's ($l = 1569, 2033, 2040$); as shown by Fig. 7 (notice that these two trajectories are the same for all learners as $\tilde{\theta}_{il}^{(t)}(k)$'s for all $i \in \mathcal{V}$ converge to a same value). In each sub-figure of Fig. 7, the red dashed curve is the trajectory of $|\tilde{\theta}_{il}^{(t)} - \theta^{l(t)}|$, i.e., the absolute difference between the consensus value and the ground-truth value at each round t. Notice that this curve is constant at 0, which indicates that $\tilde{\theta}_{il}^{(t)}$ is equal to $\theta^{l(t)}$ at all training rounds. This verifies that the global model computed by Algorithm 2 is correct under time-varying communication topology. The blue solid curve in Fig. 7 is the trajectory of $\tilde{\theta}_{il}^{(t)}$. It illustrates the convergence of the plain federated learning scheme. Fig. 5 and Fig. 7 together verify the correctness property of Algorithm 2.

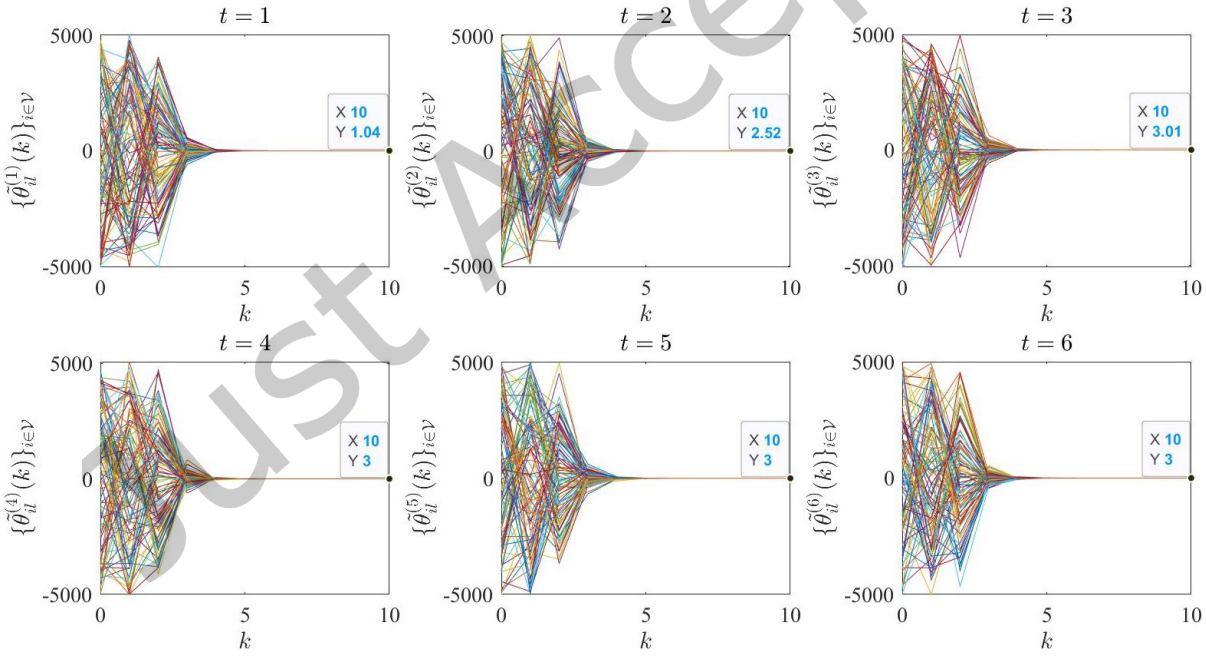


Fig. 5. Trajectories of $\{\tilde{\theta}_{il}^{(t)}(k)\}_{i \in \mathcal{V}}$ for $t = 1, \cdots, 6$, where $\{\tilde{\theta}_{il}^{(t)}(k)\}$ is the set of $\tilde{\theta}_{il}^{(t)}(k)$ for $k = 0, 1, \cdots, 10$, and $\{\tilde{\theta}_{il}^{(t)}(k)\}_{i \in \mathcal{V}}$ is the collection of $\{\tilde{\theta}_{il}^{(t)}(k)\}$ for all $i \in \mathcal{V}$.

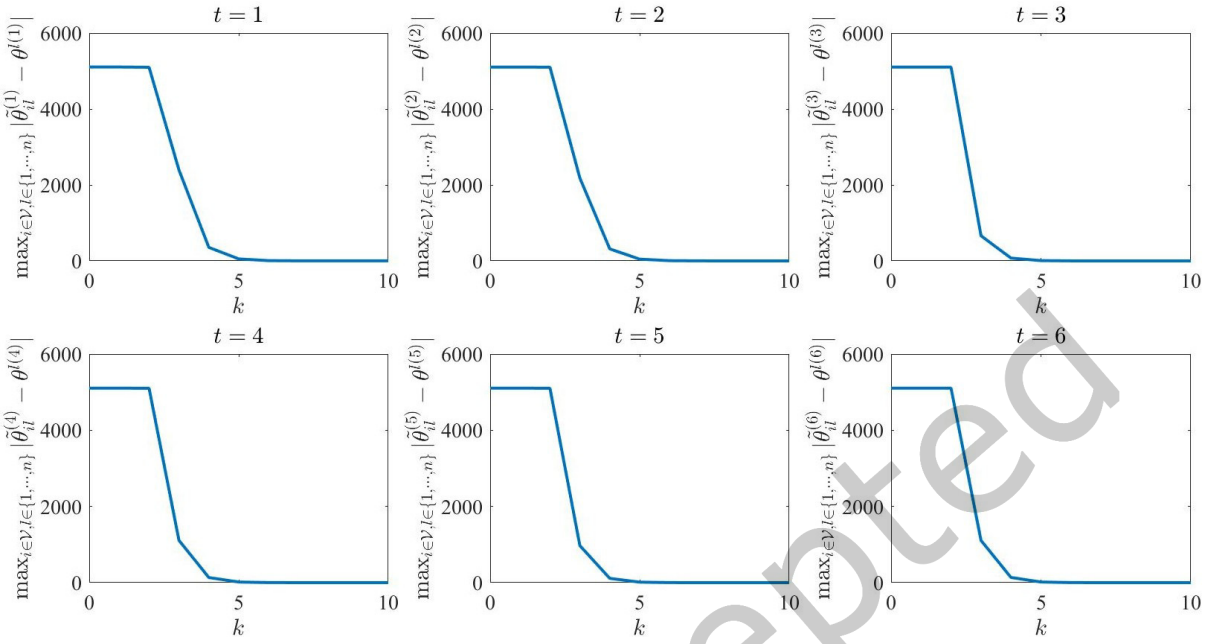


Fig. 6. Trajectories of $\max_{i\in\mathcal{V},l\in\{1,\cdots,n\}} |\tilde{\theta}_{il}^{(t)} - \theta^{l(t)}|$ for $t = 1, \cdots, 6$.

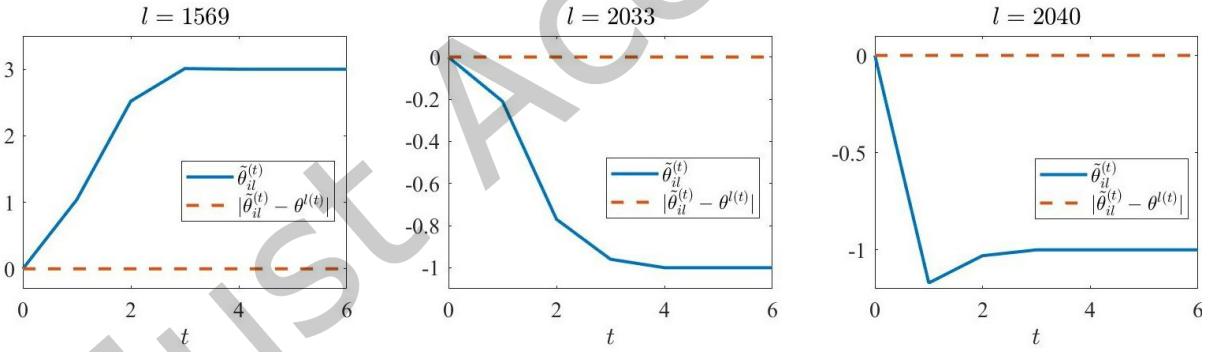


Fig. 7. Trajectories of $\tilde{\theta}_{il}^{(t)}$ and $|\tilde{\theta}_{il}^{(t)} - \theta^{l(t)}|$ for three arbitrarily picked l's.

We next use simulations to show the impact of the communication topology on the convergence rate of the consensus process. It is well known that the convergence rate is dependent on the overall connectivity degree of the communication topology. Roughly speaking, for a given number of learners, **a denser communication topology usually exhibits a higher convergence rate**. More specifically, the second largest eigenvalue of the underlying weighted adjacency matrix ($A^{(t)}$) is an important indicator of topology connectivity. In our problem setting, a smaller second largest eigenvalue of $A^{(t)}$ indicates a denser connectivity of the communication topology and a better convergence rate [53]. To visually show the impact of the communication topology on the

convergence rate, for the same l as above, we generate the sequence of $\{\tilde{\theta}_{il}^{(1)}(k)\}_{i\in\mathcal{V}}$ under six communication topologies. The first one is the complete topology, i.e., $(i,j)\in\mathcal{E}^{(1)}$ for all $i,j\in\mathcal{V}$ with $i\neq j$. The second one is a sparse topology where each learner has 40 neighbors. The third and fourth ones are sparser topologies where each learner has 20 and 10 neighbors, respectively. The fifth one is the star topology, i.e., there exists one learner i such that $j\in\mathcal{N}_i^{(1)}$ for all $j\in\mathcal{V}\backslash\{i\}$, while $(j,\ell)\notin\mathcal{E}^{(1)}$ for any $j,\ell\in\mathcal{V}\backslash\{i\}$. The sixth one is the line topology, i.e., the connection of the learners forms a line. Intuitively, the six topologies have descending connectivity degrees. Indeed, their corresponding matrix $A^{(1)}$ have ascending second largest eigenvalues: 0, 0.3259, 0.8181, 0.9555, 0.9900, and 0.9997, respectively. The complete, star and line topologies are three representative communication topologies and have broad applications. In particular, the complete topology has the largest possible connectivity degree (densest) and is typical for, e.g., secure mulitparty computation tasks [11]; the star topology depicts the (sparse) spoke–hub distribution paradigm and is common in, e.g., cloud computing [6]; and the line topology has the smallest possible connectivity degree (sparsest) for connected graphs and is widely used in, e.g., power systems [21]. The other three cases depict three different connectivity degrees in between and are used to simulate general sparse graphs covering a wider range of connectivity degrees. The trajectories of $\tilde{\theta}_{il}^{(1)}(k)$ for all $i\in\mathcal{V}$ under these six communication topologies are shown in Fig. 8. We can see that under all the six communication topologies, all the 100 trajectories converge to the value of the desired global sum 1.04, but clearly with descending convergence rates, which matches discussion above. By (25), we can see that the convergence rate can be estimated by the decaying rate of $\|N(A^{(t)})^k-1_N1_N^T\|$. The trajectories of $\|N(A^{(1)})^k-1_N1_N^T\|$ under the above six communication topologies are shown in Fig. 9 (the small figure shows the convergence under the line topology). It matches the convergence rates observed in Fig. 8. Notice that the trajectory of $\|N(A^k-1_N1_N^T\|$ for a given matrix A can be generated offline. Hence, if the learners have prior knowledge of average connectivity degree of $\mathcal{G}^{(t)}$, then based on the decaying rate of $\|N(A^{(t)})^k-1_N1_N^T\|$ for possible communication topologies, they may be able to choose a less conservative value of K.

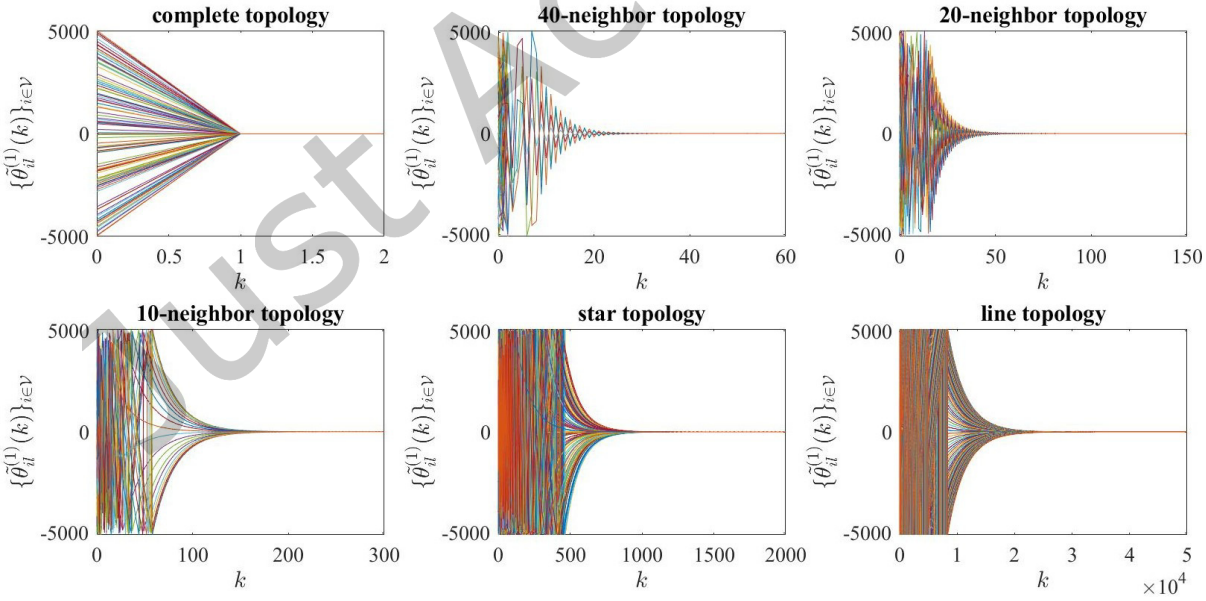


Fig. 8. Trajectories of $\{\tilde{\theta}_{il}^{(1)}(k)\}_{i\in\mathcal{V}}$ under different communication topologies.

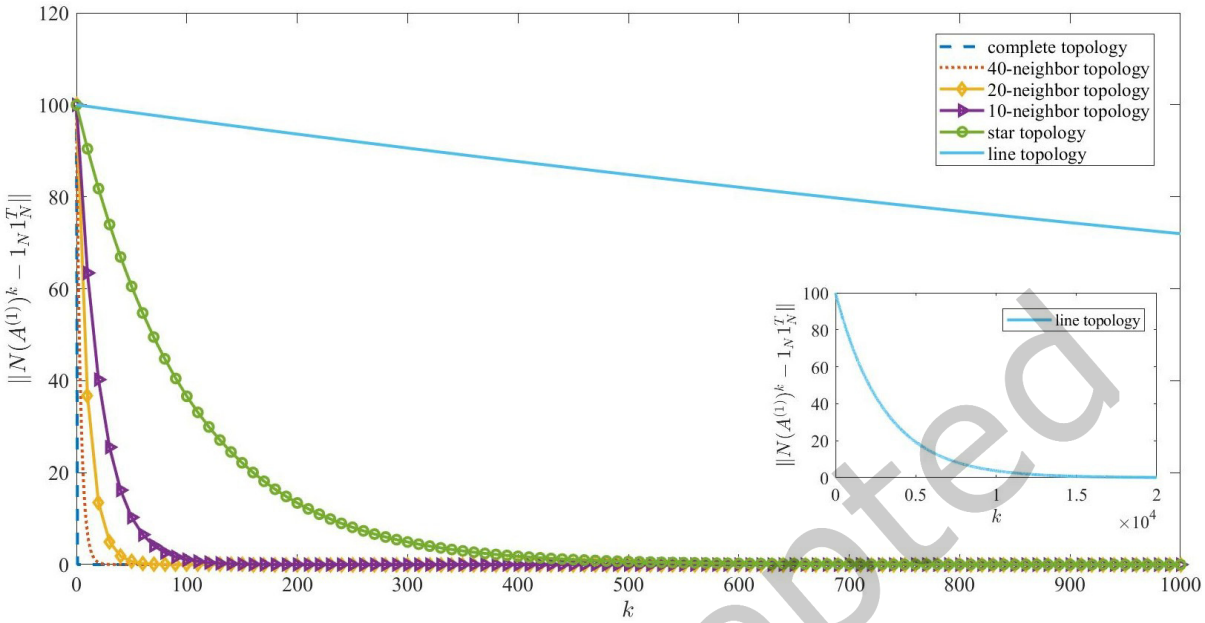


Fig. 9. Trajectories of $\|N(A^{(1)})^k - 1_N 1_N^T\|$ under different communication topologies.

Finally we verify the computational efficiency of Algorithm 2. Breakdowns of computational overhead under different communication topologies and autoencoder hidden layers are shown in Table 1. In the table, h is the number of autoencoder hidden layers, n is the number of model features (recall that $n = 1569h + 784$), t_1 is the time of constructing the weighted adjacency matrix $A^{(t)}$ per learner per round, t_2 is the time of secret shares generation and sharing per learner per round, and t_3 is the time of consensus process per learner per round. For each communication topology, the number of consensus iterations K is fixed at a value where convergence is achieved. The table shows that, for each communication topology, as h and n increase, t_1 does not change, while t_2 and t_3 increase correspondingly. The unchange of t_1 is due to that t_1 only depends on the number of the learners' neighbors and independent from h or n. The increase of t_2 and t_3 is because the average number of shares increases. In particular, the average number of shares per learner per round is $|\mathcal{N}_{avg}|n$, where $|\mathcal{N}_{avg}|$ denotes the average number of neighbors per learner per round. Therefore, for a given communication topology, $|\mathcal{N}_{avg}|$ is fixed, and t_2 and t_3 will increase as n increases. Moreover, for fixed h and n, for the communication topologies from left to right in the table, t_1 and t_2 decrease, while t_3 increases. The decrease of t_1 and t_2 is due to that, from left to right, the communication topology is sparser and sparser, and thus $|\mathcal{N}_{avg}|$ is smaller and smaller, so that, in average, the learners need fewer operations in constructing $A^{(t)}$ and also generate fewer shares. The increase of t_3 is due to that, for a sparser communication topology, more iterations are needed for consensus convergence. The table indicates that a learner's computational overhead is mainly determined by the total number of shares it needs to generate and the number of iterations of the consensus process. We next further examine these relationships. First, we examine the relationship between computational overhead and the total number of shares. As mentioned, this number depends on two factors, one is $\bar{\mathcal{N}}_i^{(t)}$, the number of its neighbors including itself, and the other is n, the dimension of $\theta^{(t)}$. Notice that the first factor is related to the size of learners. In each training round t, a learner's total number of shares is $\bar{\mathcal{N}}_i^{(t)}n$. To examine the relationship

between the computational overhead and $\bar{N}_i^{(t)} n$, an easy way is to tune the values of n by tuning the values of h according to the equation $n = 1569h + 784$. Without loss of generality, the following simulations adopt a fixed communication topology, where the learners' connectivity degrees are well balanced (i.e., they have similar number of neighbors). We use an arbitrary such communication topology where the average number of neighbors for one learner is 87. For each value of n, Algorithm 2 is run for $T = 10$ rounds, and in each round t, the consensus algorithm is run for $K = 10$ iterations. The results are shown in the left sub-figure of Fig. 10, where the x-axis is the average number of total shares per learner per round (in this case, $87n$), and the y-axis is the average time per learner per training round for the phase of global model aggregation (steps 4–14 of Algorithm 2). From the left sub-figure of Fig. 10, we can see that when the average total number of shares per learner per round is 10^6, the average time per learner per training round is merely around 2.4 seconds. In addition, this sub-figure illustrates that the average time per learner per training round grows linearly with the average total number of shares per learner per round, where the growth rate is very slow, approximately 2.315×10^{-6}. This verifies that **our algorithm is computationally efficient and scales well with large-size dense networks and high dimensional training models**. Next we examine the relationship between the running time for the consensus process (step 10 of Algorithm 2) and the number of consensus iterations K. The above communication topology is adopted and fixed. For each value of K, Algorithm 2 is run for $T = 10$ rounds, and in each round t, the consensus algorithm is run for K iterations. The results are shown in the right sub-figure of Fig. 10, where the x-axis is the value of K, and the y-axis is the average consensus time per learner per training round. From the right sub-figure of Fig. 10, we can see that when $K = 10^5$, the average consensus time per learner per training round is merely around 0.6153 seconds. In addition, this sub-figure shows that the average consensus time per learner per training round grows linearly with the value of K, where the growth rate is also very slow, approximately 6.155×10^{-6}. This verifies that **our algorithm is also computationally efficient for sparse networks which may need a large number of consensus iterations**.

Table 1. Breakdowns of computational overhead under different communication topologies and autoencoder hidden layers, where h is the number of autoencoder hidden layers, n is the number of model features, K is the number of consensus iterations, t_1 is the time of constructing the weighted adjacency matrix $A^{(t)}$, t_2 is the time of secret shares generation and sharing, and t_3 is the time of consensus process, all times measured per learner per round

			complete	40-neighbor	20-neighbor	10-neighbor	star	line
h	n	t (s)	$K = 2$	$K = 60$	$K = 150$	$K = 300$	$K = 2000$	$K = 50000$
1	2353	t_1	0.00009	0.00008	0.00006	0.00005	0.00004	0.00002
		t_2	0.74	0.12	0.034	0.016	0.012	0.0012
		t_3	0.00016	0.00059	0.0015	0.0022	0.015	0.34
3	5491	t_1	0.00009	0.00008	0.00006	0.00005	0.00004	0.00002
		t_2	1.72	0.33	0.082	0.020	0.018	0.0016
		t_3	0.00043	0.0014	0.0028	0.0049	0.032	0.76
5	8629	t_1	0.00009	0.00008	0.00006	0.00005	0.00004	0.00002
		t_2	3.36	0.50	0.14	0.036	0.035	0.0021
		t_3	0.00053	0.0018	0.0048	0.0083	0.061	1.18
7	11767	t_1	0.00009	0.00008	0.00006	0.00005	0.00004	0.00002
		t_2	4.35	0.62	0.18	0.040	0.038	0.0031
		t_3	0.00060	0.0025	0.0068	0.0095	0.063	1.65
9	14905	t_1	0.00009	0.00008	0.00006	0.00005	0.00004	0.00002
		t_2	5.44	0.74	0.24	0.052	0.045	0.0037
		t_3	0.00068	0.0028	0.0084	0.012	0.079	1.97

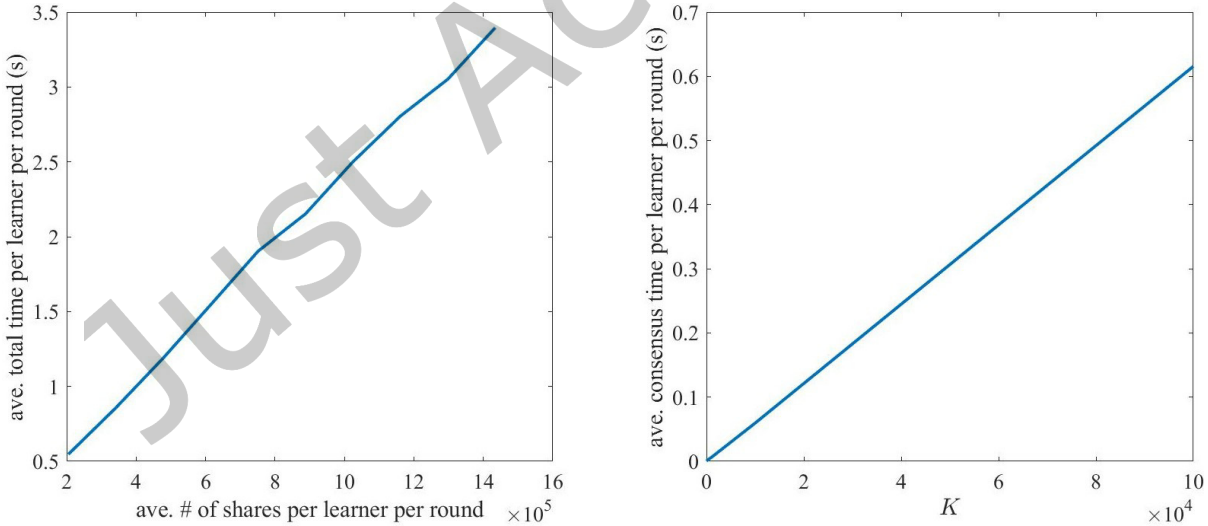


Fig. 10. Relationship between computational overhead and total number of shares (left) and number of consensus iterations K (right).

8.3 Simulation results for the extended algorithm of Section 7.2

We next test the performance of the extended algorithm of Section 7.2. In particular, the simulations in this subsection focus on consensus convergence under time-varying communication topologies and leaving of learners within a training round. To see the effect of non-IID data, we purposely modify the distributions of the learners' local models such that 10 learners' local model distributions are significantly different from those of the remaining 90 learners. Without loss of generality, we choose these 10 learners to be the last 10 learners in the index system, i.e., learners 91∼100.

First, we simulate the case where $K = 600$ and the 10 learners whose data distributions are significantly different from the majority leave the network at iteration 100. The simulation results are shown in Fig. 11. In all the three sub-figures, the red dashed curve depicts the correct value of the desired global model $\sum_{i=1}^{100} \bar{\theta}_{il}^{(t)}(0)$, which is -122.53. The first sub-figure applies the restart method. That is, at iteration 100, the remaining 90 learners regenerate shares of their local models, $\{10^{\sigma}\bar{\theta}_{il}^{(t)}(0)\}_{i=1}^{90}$, based on the current communication topology and start a new consensus process with the new shares. The remaining 90 learners' converging point is thus $\sum_{i=1}^{90} \bar{\theta}_{il}^{(t)}(0) = 431.88$. The deviation between the converging point and the correct global model is $431.88 - (-122.53) = 554.41$. The second sub-figure applies the trivial extension method where, at iteration 100, learners 91∼100 just leave the network without taking any actions, and the remaining 90 learners just use their current states $\{s_i^{(t)}(100)\}_{i=1}^{90}$ to proceed to the next consensus iteration under the new communication topology. As shown by the second sub-figure, the remaining 90 learners' converging point is at the value of 1781.66. The deviation between this converging point and the correct global model is $1781.66 - (-122.53) = 1904.19$, which is even larger than the deviation under the restart method. In fact, by the proof of Theorem 7.1, the deviation of $\{s_i^{(t)}(K)\}_{i=1}^{90}$ under the trivial extension method is equal to $\sum_{i=91}^{100} s_i^{(t)}(100)$, and this value can be very large. The second sub-figure shows that Algorithm 2 with the above trivial extension method can perform poorly if the learners hold non-IID data and some learners leave the network within a training round. The third sub-figure applies our extended algorithm of Section 7.2. We can see that, **by our proposed extended method, all the remaining 90 learners' local models converge to the correct global model at the value of** -122.53. This verifies the correctness of the proposed extended algorithm.

We next simulate the proposed extended algorithm for the general case where **both change of communication topology and leaving of learners happen at multiple iterations within a training round**. In particular, we set that changes of communication topology happen at 50 randomly chosen iterations, and leaving of learners happens at iterations 100, 200, 300, 400, and 500, where, each time, 10 learners leave the network. The trajectories of $\{\tilde{\theta}_{il}^{(t)}(k)\}_{i\in\mathcal{V}^{(t)}(K)}$ are shown in Fig. 12. We can see that **all the remaining 50 learners' local models converge to the correct global model at the value of** -122.53. Moreover, Fig. 12 shows that, after convergence is achieved or nearly achieved, if leaving of learners happen again, there will be a new transient process by which the remaining learners' local models re-converge to the correct global model. Therefore, in theory, to guarantee correctness of consensus convergence, we need that no further leaving of learners happens after some iteration. In practice, the learners can terminate the consensus process if their states keep constant for a few consecutive iterations.

9 CONCLUSIONS AND FUTURE WORKS

This paper develops a new algorithm for privacy-preserving decentralized federated learning over a time-varying communication graph. First, a simplified problem setting is considered, where the participating learners are fixed and the communication topology between can only change between successive training rounds. A consensus-based framework is adopted to enable decentralized global model aggregation. In each round of model aggregation, the Metropolis-Hastings method is applied to update the weighted adjacency matrix based on the

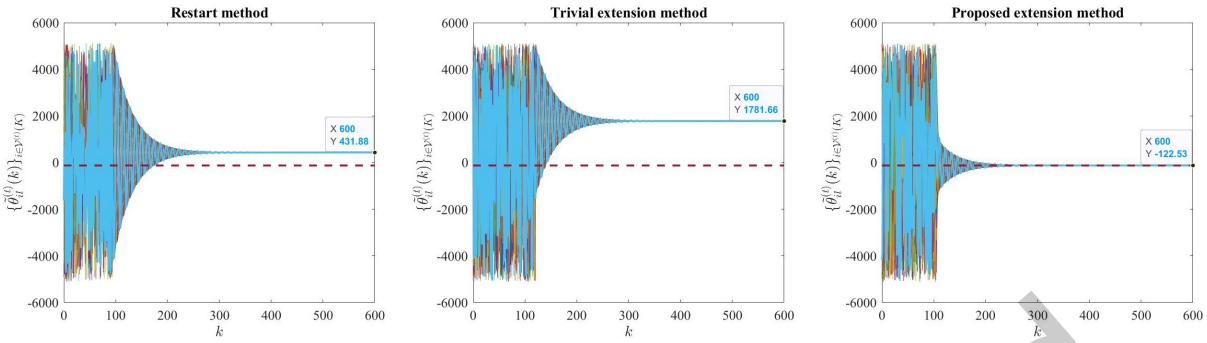


Fig. 11. Trajectories of $\{\tilde{\theta}_{il}^{(t)}(k)\}_{i\in\mathcal{V}^{(t)}(K)}$ for the case where 10 learners whose data distributions are significantly different from the majority leave the network at iteration 100.

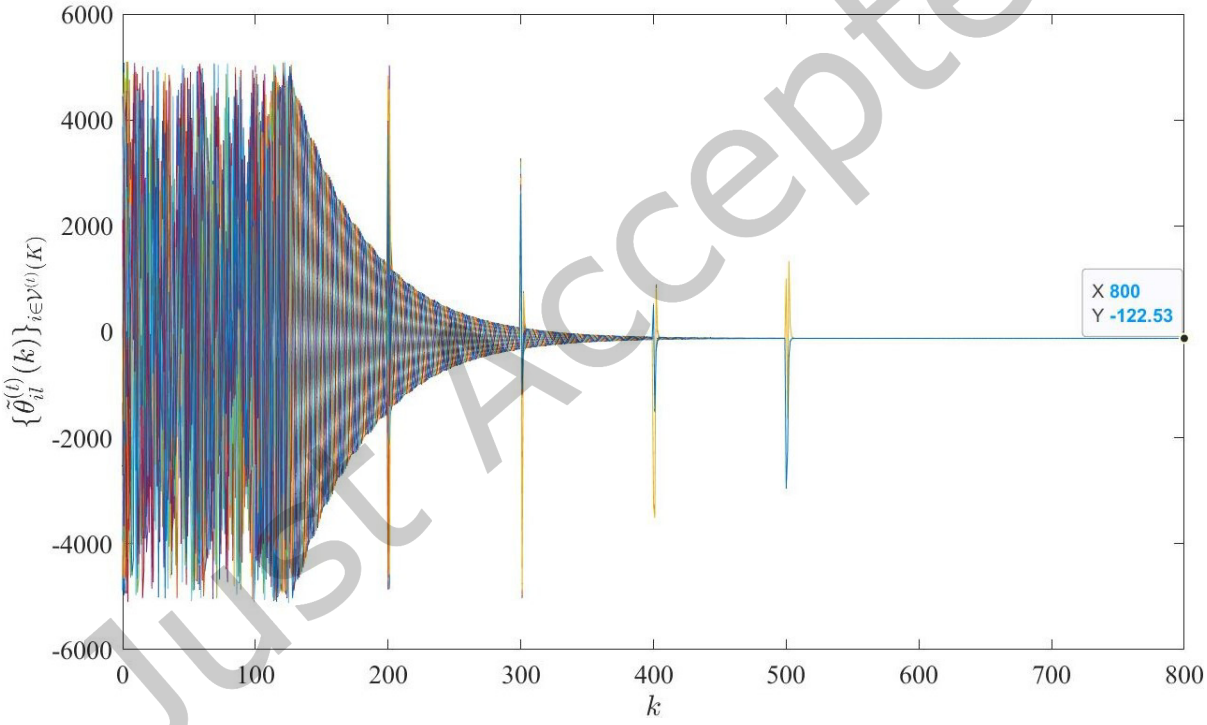


Fig. 12. Trajectories of $\{\tilde{\theta}_{il}^{(t)}(k)\}_{i\in\mathcal{V}^{(t)}(K)}$ for the general case where both change of communication topology and leaving of learners happen multiple times within a training round.

current communication topology so as to ensure convergence of average consensus. The technique of Shamir's secret sharing scheme is further integrated with the consensus-based framework to facilitate privacy preservation. The algorithm is then extended to deal with the issues of change of participating learners and time-varying communication topology within a training round. The correctness and privacy properties of the proposed

algorithm are both analyzed. Its correctness, convergence rate and computational overhead are examined by a case study on a federated learning application using the MNIST dataset. Beyond global model aggregation in federated learning, the proposed algorithm can be readily applied to general secure aggregation tasks over sparse time-varying communication graphs, e.g., decentralized opinions agreement, multi-vehicle rendezvous, and energy supply/consumption aggregation. Moreover, it can also be applied to facilitate privacy for more complicated problems that are solvable by consensus-based approaches [31], e.g., distributed formation control, state estimation, unconstrained convex optimization, and resource allocation.

An interesting future work topic is to extend the attacker model to also include active (malicious) inference attacks targeting on inferring benign learners' local models. Several papers, e.g., [5, 38, 54, 67], have shown that, in a centralized federated learning setting, a malicious centralized server can elude SMC protocols to infer benign learners' local models by maliciously modifying the global model shared with the learners. A very recent paper [55] also pointed out this issue as an open question for federated learning in a decentralized setting. Compared to centralized setting, decentralized setting in general has a broader attack surface. On the other hand, in a centralized setting, if the centralized server is malicious, then it may be able to leverage the aforementioned active attack to infer local models of all the benign learners. A decentralized setting may be able to mitigate this issue as a benign learner's individual privacy may only depend on its direct neighbors, in which case, if a benign learner does not have malicious direct neighbors, it can survive such active attacks. However, it still remains to be investigated whether a malicious learner can devise a sophisticated yet computationally feasible strategy to modify the models it shares with its direct neighbors so as to infer the local models of its multi-hop neighbors. In any case, how to configure SMC protocols to resist such active inference attacks is a critical challenge for both centralized and decentralized federated learning. Another interesting future work topic is to investigate active data poisoning attacks targeting on failing the learning task. For example, with external data poisoning attacks, data transmitted over communication links may be tampered by external attackers; and with Byzantine attacks, the learners themselves may be corrupted to maliciously deviate from the designed algorithm. These attacks may cause a significant distort in the global model aggregation, leading to very poor learning performance or even learning failure. In the presence of such active data poisoning attacks, a resilient algorithm needs to be developed which can maintain a satisfactory learning performance.

REFERENCES

[1] J. Alonso-Mora, E. Montijano, M. Schwager, and D. Rus. 2016. Distributed multi-robot formation control among obstacles: A geometric and optimization approach with consensus. In *2016 IEEE International Conference on Robotics and Automation*. 5356–5363.

[2] R. Aragues, C. Sagues, and Y. Mezouar. 2013. Feature-based map merging with dynamic consensus on information increments. In *2013 IEEE International Conference on Robotics and Automation*. 2725–2730.

[3] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs. 2012. Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE. In *Advances in Cryptology – EUROCRYPT 2012*. 483–501.

[4] A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi. 2018. Personalized and Private Peer-to-Peer Machine Learning. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 84)*. PMLR, 473–481.

[5] F. Boenisch, A. Dziedzic, R. Schuster, A. S. Shamsabadi, I. Shumailov, and N. Papernot. 2021. When the Curious Abandon Honesty: Federated Learning Is Not Private. *arXiv preprint* (2021). [Online] https://arxiv.org/abs/2112.02918.

[6] L. Chao. 2015. *Cloud computing networking: Theory, practice, and development.* CRC Press.

[7] C. Che, X. Li, C. Chen, X. He, and Z. Zheng. 2021. A Decentralized Federated Learning Framework via Committee Mechanism with Convergence Guarantee. *arXiv preprint* (2021). [Online] https://arxiv.org/pdf/2108.00365.pdf.

[8] H. Chen, W. Dai, M. Kim, and Y. Song. 2019. Efficient Multi-Key Homomorphic Encryption with Packed Ciphertexts with Application to Oblivious Neural Network Inference. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 395–412.

[9] A. Cherukuri and J. Cortés. 2015. Distributed Generator Coordination for Initialization and Anytime Optimization in Economic Dispatch. *IEEE Transactions on Control of Network Systems* 2, 3 (2015), 226–237.

[10] R. Cioffi, M. Travaglioni, G. Piscitelli, A. Petrillo, and F. De Felice. 2020. Artificial Intelligence and Machine Learning Applications in Smart Production: Progress, Trends, and Directions. *Sustainability* 12, 2 (2020), 1–26.
[11] R. Cramer, I. Damgård, and J. B. Nielsen. 2015. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press.
[12] L. Deng. 2012. The MNIST Database of Handwritten Digit Images for Machine Learning Research. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.
[13] J. Dreier and F. Kerschbaum. 2011. Practical privacy-preserving multiparty linear programming based on problem transformation. In *Proceedings of the 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*. 916–924.
[14] C. Dwork and A. Roth. 2014. The Algorithm Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3–4 (August 2014), 211–407.
[15] A. M. Elbir, B. Soner, and S. Coleri. 2020. Federated Learning in Vehicular Networks. *arXiv preprint* (2020). [Online] https://arxiv.org/pdf/2006.01412.pdf.
[16] M. Fredrikson, S. Jha, and T. Ristenpart. 2015. Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 1322–1333.
[17] M. J. Freedman, K. Nissim, and B. Pinkas. 2004. Efficient Private Matching and Set Intersection. In *Proceedings of the 2004 International Conference on the Theory and Applications of Cryptographic Techniques*. 1–19.
[18] D. Froelicher, J. R. Troncoso-Pastoriza, A. Pyrgelis, S. Sav, J. Sousa, J. Bossuat, and J. Hubaux. 2021. Scalable Privacy-Preserving Distributed Learning. *Proceedings on Privacy Enhancing Technologies* 2021, 323–347.
[19] S. Gade and N. H. Vaidya. 2018. Privacy-Preserving Distributed Learning via Obfuscated Stochastic Gradients. In *2018 IEEE Conference on Decision and Control*. 184–191.
[20] S. Gade and N. H. Vaidya. 2018. Private Optimization on Networks. In *2018 Annual American Control Conference*. 1402–1409.
[21] B. Gäde, A. M. Lehmann, J. Deutschmann, and J. B. Huber. 2017. A power line communication topology module for ns-3 and DCE. In *2017 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. 295–301.
[22] Q. Geng and P. Viswanath. 2014. The Optimal Mechanism in Differential Privacy. In *2014 IEEE International Symposium on Information Theory*. 2371–2375.
[23] O. Goldreich. 2004. *Foundations of Cryptography: Volume 2-Basic Applications*. Cambridge University Press.
[24] Y. Gong, Y. Cai, Y. Guo, and Y. Fang. 2016. A privacy-preserving scheme for incentive-based demand response in the smart grid. *IEEE Transactions on Smart Grid* 7, 3 (2016), 1304–1313.
[25] A. Géron. 2019. *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems*. O'Reilly.
[26] C. Hazay and Y. Lindell. 2010. *Efficient Secure Two-Party Protocols–Techniques and Constructions*. Springer.
[27] J. He, L. Cai, P. Cheng, J. Pan, and L. Shi. 2019. Distributed Privacy-Preserving Data Aggregation Against Dishonest Nodes in Network Systems. *IEEE Internet of Things Journal* 6, 2 (2019), 1462–1470.
[28] J. He, L. Cai, and X. Guan. 2020. Differential Private Noise Adding Mechanism and Its Application on Consensus Algorithm. *IEEE Transactions on Signal Processing* 68 (2020), 4069–4082.
[29] Z. Huang, S. Mitra, and G. Dullerud. 2012. Differentially Private Iterative Synchronous Consensus. In *ACM workshop on privacy in the electronic society*. 81–90.
[30] B. Jeon, S. M. Ferdous, M. R. Rahman, and A. Walid. 2021. Privacy-Preserving Decentralized Aggregation for Federated Learning. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 1–6.
[31] S. S. Kia, B. Van Scoy, J. Cortes, R. A. Freeman, K. M. Lynch, and S. Martinez. 2019. Tutorial on Dynamic Average Consensus: The Problem, Its Applications, and the Algorithms. *IEEE Control Systems Magazine* 39, 3 (2019), 40–72.
[32] K. Kogiso and T. Fujita. 2015. Cyber-security enhancement of networked control systems using homomorphic encryption. In *Proceedings of 2015 IEEE 54th Annual Conference on Decision and Control (CDC)*. 6836–6843.
[33] J. Konečný, H. McMahan, D. Ramage, and P. Richtárik. 2016. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. *arXiv preprint* (2016). [Online] https://arxiv.org/pdf/1610.02527.pdf.
[34] I. Krontiris, F. C. Freiling, and T. Dimitriou. 2010. Location privacy in urban sensing networks: research challenges and directions. *IEEE Wireless Communications* 17, 5 (2010), 30–35.
[35] H. Kwon. 2021. Defending Deep Neural Networks against Backdoor Attack by Using De-trigger Autoencoder. *IEEE Access* (2021), 1–1.
[36] R. L. Lagendijk, Z. Erkin, and M. Barni. 2013. Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation. *IEEE Signal Processing Magazine* 30, 1 (2013), 82–105.
[37] A. Lalitha, O. Cihan Kilinc, T. Javidi, and F. Koushanfar. 2019. Peer-to-Peer Federated Learning on Graphs. *arXiv preprint* (2019). [Online] https://arxiv.org/pdf/1901.11173.pdf.
[38] M. Lam, G. Wei, D. Brooks, V. J. Reddi, and M. Mitzenmacher. 2021. Gradient Disaggregation: Breaking Privacy in Federated Learning by Reconstructing the User Participant Matrix. In *Proceedings of the 38th International Conference on Machine Learning*, Vol. 139. 5959–5968.

[39] Qiongxiu Li and Mads Graesbøll Christensen. 2019. A Privacy-Preserving Asynchronous Averaging Algorithm based on Shamir's Secret Sharing. In *2019 27th European Signal Processing Conference (EUSIPCO)*. 1–5.
[40] Y. Liu, X. Hou, J. Chen, C. Yang, G. Su, and W. Dou. 2014. Facial expression recognition and generation using sparse autoencoder. In *2014 International Conference on Smart Computing*. 125–130.
[41] S. Lu, Y. Zhang, and Y. Wang. 2020. Decentralized Federated Learning for Electronic Health Records. In *2020 54th Annual Conference on Information Sciences and Systems (CISS)*. 1–5.
[42] Y. Lu and M. Zhu. 2018. Privacy preserving distributed optimization using homomorphic encryption. *Automatica* 96, 10 (October 2018), 314–325.
[43] Y. Lu and M. Zhu. 2019. A control-theoretic perspective on cyber-physical privacy: Where data privacy meets dynamic systems. *Annual Reviews in Control* 47 (2019), 423–440.
[44] N. E. Manitara and C. N. Hadjicostis. 2013. Privacy-preserving asymptotic average consensus. In *2013 European Control Conference*. 760–765.
[45] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Vol. 54. 1273–1282.
[46] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.
[47] Y. Mo and R. M. Murray. 2017. Privacy Preserving Average Consensus. *IEEE Trans. Automat. Control* 62, 2 (2017), 753–765.
[48] C. Mouchet, E. Bertrand, and J. Hubaux. 2023. An Efficient Threshold Access-Structure for RLWE-Based Multiparty Homomorphic Encryption. *Journal of Cryptology* 36, 10 (2023), 1–20.
[49] C. Mouchet, J. R. Troncoso-Pastoriza, J. Bossuat, and J. Hubaux. 2021. Multiparty Homomorphic Encryption from Ring-Learning-with-Errors. In *Proceedings on Privacy Enhancing Technologies*, Vol. 2021. 291–311.
[50] M. Nasr, R. Shokri, and A. Houmansadr. 2019. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *2019 IEEE Symposium on Security and Privacy*. 739–753.
[51] S. Niknam, H. S. Dhillon, and J. H. Reed. 2019. Federated Learning for Wireless Communications: Motivation, Opportunities and Challenges. *arXiv preprint* (2019). [Online] https://arxiv.org/pdf/1908.06847.pdf.
[52] E. Nozari, P. Tallapragada, and J. Cortés. 2017. Differentially private average consensus: Obstructions, trade-offs, and optimal algorithm design. *Automatica* 81 (2017), 221–231.
[53] A. Olshevsky and J. N. Tsitsiklis. 2009. Convergence Speed in Distributed Consensus and Averaging. *SIAM Journal on Control and Optimization* 48, 1 (2009), 33–55.
[54] D. Pasquini, D. Francati, and G. Ateniese. 2022. Eluding Secure Aggregation in Federated Learning via Model Inconsistency. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2429–2443.
[55] D. Pasquini, M. Raynal, and C. Troncoso. 2022. On the Privacy of Decentralized Machine Learning. *arXiv preprint* (2022). [Online] https://arxiv.org/abs/2205.08443.
[56] A. Guha Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger. 2019. BrainTorrent: A Peer-to-Peer Environment for Decentralized Federated Learning. *arXiv preprint* (2019). [Online] https://arxiv.org/pdf/1905.06731.pdf.
[57] M Ruan, H. Gao, and Y. Wang. 2019. Secure and Privacy-Preserving Consensus. *IEEE Trans. Automat. Control* 64, 10 (2019), 4035–4049.
[58] S. Sav, A. Pyrgelis, J. R. Troncoso-Pastoriza, D. Froelicher, J. Bossuat, J. Sousa, and J. Hubaux. 2021. POSEIDON: Privacy-Preserving Federated Neural Network Learning. *Network and Distributed Systems Security Symposium (NDSS)*, 1–18.
[59] S. Savazzi, M. Nicoli, and V. Rampa. 2020. Federated Learning With Cooperating Devices: A Consensus Approach for Massive IoT Networks. *IEEE Internet of Things Journal* 7, 5 (2020), 4641–4654.
[60] V. Schwarz, G. Hannak, and G. Matz. 2014. On the convergence of average consensus with generalized Metropolis-Hasting weights. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 5442–5446.
[61] A. Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (November 1979), 612–613.
[62] C. E. Shannon. 1949. Communication Theory of Secrecy Systems. *Bell System Technical Journal* 28, 4 (October 1949), 656–715.
[63] Y. Shoukry, K. Gatsis, A. Alanwar, G. J. Pappas, S. A. Seshia, M. Srivastava, and P. Tabuada. 2016. Privacy-Aware Quadratic Optimization Using Partially Homomorphic Encryption. In *Proceedings of the 2016 IEEE 55th Conference on Decision and Control*. 5053–5058.
[64] P. Su, X. Tian, Y. Wang, S. Deng, J. Zhao, Q. An, and Y. Wang. 2017. Recent Trends in Load Forecasting Technology for the Operation Optimization of Distributed Energy System. *Energies* 10, 9 (2017), 1–13.
[65] Y. Wang. 2019. Privacy-Preserving Average Consensus via State Decomposition. *IEEE Trans. Automat. Control* 64, 11 (2019), 4711–4716.
[66] Y. Wang, H. Yao, and S. Zhao. 2016. Auto-encoder based dimensionality reduction. *Neurocomputing* 184 (2016), 232–242.
[67] Y. Wen, J. A. Geiping, L. Fowl, M. Goldblum, and T. Goldstein. 2022. Fishing for User Data in Large-Batch Federated Learning via Gradient Magnification. In *Proceedings of the 39th International Conference on Machine Learning*, Vol. 162. 23668–23684.
[68] L. Xiao and S. Boyd. 2003. Fast linear iterations for distributed averaging. In *42nd IEEE International Conference on Decision and Control*, Vol. 5. 4997–5002.

[69] L. Xiao, S. Boyd, and S. Kim. 2007. Distributed average consensus with least-mean-square deviation. *J. Parallel and Distrib. Comput.* 67, 1 (2007), 33–46.
[70] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang. 2021. Federated Learning for Healthcare Informatics. *Journal of Healthcare Informatics Research* 5 (2021), 1–19.
[71] B. Yang, M Liang, and R. Urtasun. 2018. HDNET: Exploiting HD Maps for 3D Object Detection. In *Proceedings of The 2nd Conference on Robot Learning*, Vol. 87. 146–155.
[72] X. Yi, R. Paulet, and E. Bertino. 2014. *Homomorphic Encryption and Applications*. Springer.
[73] Z. Yu, J. Hu, G. Min, H. Xu, and J. Mills. 2020. Proactive Content Caching for Internet-of-Vehicles based on Peer-to-Peer Federated Learning. In *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*. 601–608.
[74] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao, and M. S. Hossain. 2021. Mobility-Aware Proactive Edge Caching for Connected Vehicles Using Federated Learning. *IEEE Transactions on Intelligent Transportation Systems* 22, 8 (2021), 5341–5351.
[75] C. Zhao, J. Chen, J. He, and P. Cheng. 2018. Privacy-Preserving Consensus-Based Energy Management in Smart Grids. *IEEE Transactions on Signal Processing* 66, 23 (2018), 6162–6176.
[76] C. Zhou and R. C. Paffenroth. 2017. Anomaly Detection with Robust Deep Autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 665–674.