FedSecure: A Privacy-Preserving Federated Learning Algorithm

Mojtaba Kaheni[1], Martina Lippi[2], Andrea Gasparri[2], and Alessandro V. Papadopoulos[1]

***Abstract*— The federated learning (FL) paradigm effectively distributes the training burden among several units, each possessing local datasets. This paper presents a novel privacy-preserving FL algorithm, FedSecure. Our approach is distinct from the state of the art as it redefines the conventional distributed optimization problem inherent in FL. Unlike traditional frameworks that assume a common weight vector as the global decision variable, our method introduces an equivalent constrained problem. Each agent maintains its own weight vector as the local decision variable under the constraint that these local weight vectors must be equal. This enables the dual decomposition method to solve the distributed optimization problem. A key advantage of FedSecure is its ability to eliminate the necessity of sharing both the initial weights and the updated network weight values of each agent with the server. This feature ensures that the information related to agents' training samples remains impervious to potential state-of-the-art cyber espionage attempts, underscoring the robust security measures of our algorithm. We validate FedSecure on the MNIST and CIFAR-10 datasets and compare it to a differential privacy algorithm, in which artificial noise is added to parameters at the clients' side before aggregating, namely, noising before model aggregation FL (NbAFL).**

I. INTRODUCTION

The fundamental concept of FL [1] involves distributing the training process among various computation units, denoted as "agents", i.e., computers, smartphones, etc., each possessing local training datasets [2], [3]. While agents typically aim to keep their local training datasets private to safeguard their privacy [4], they share a common interest in collaboratively learning a globally optimal model.

The standard architecture employed in FL follows a server-worker structure, as illustrated in Fig. 1. Within this architecture, each agent i initiates from a global model and refines its local model for its specific training dataset, $\mathcal{D}_i$, using a Stochastic Gradient Descent (SGD) algorithm. Subsequently, the agent communicates the resulting parameters $\mathbf{w}_i$ to a central server. In the subsequent step, the server updates the global model by aggregating the received parameters $\mathbf{w}_i$. The updated global parameters $\mathbf{w}$ are then sent back to the respective agents.

Despite the promising advantages of FL [5], the fact that this distributes the learning process to numerous processing units at the edge level makes it vulnerable to various types of adversarial attacks. These attacks can be classified under two general umbrellas:

- *Attacks on performance*: Adversaries manipulate the training data of agents (data poisoning attacks [6], [7]) or the model communicated from agents to the server (model poisoning attacks [8], [9]) to reduce the accuracy of the FL process.
- *Attacks on privacy*: Adversaries try to infer information about the agents' private data [10], [11].

In recent years, several promising aggregation rules e.g., trimmed average or median [12], Krum [13], Bulyan [14], Byrd-SAGA [15], Zeno [16], RSA [17], SETA [18] and [19] have been introduced to cope with the undesired effects of data and model poisoning attacks in FL. However, the privacy defenses in FL still require considerable steps to be taken. In this paper, we focus on privacy and confidentiality preserving.

Privacy-preserving was the driving force behind the inception of FL. While FL facilitates privacy preservation by alleviating the necessity to transfer agents' datasets to an external server, it does not entirely eradicate the risk of information leakage. Recent studies show that adversaries can extract

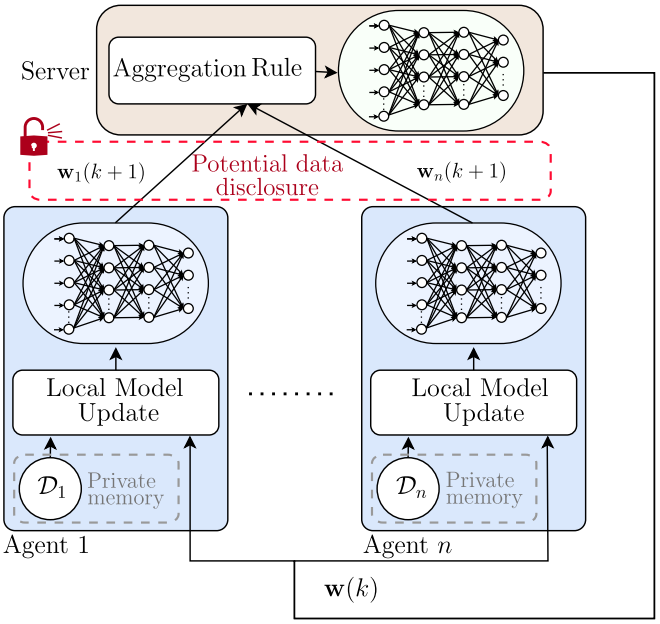

Fig. 1: Server-worker architecture wherein n agents individually update their models and transmit the calculated weight vector $\mathbf{w}_i$ to a server responsible for aggregating them into a global weight vector $\mathbf{w}$.

This work was supported by the Swedish Research Council (VR), with grant "PSI: Pervasive Self-Optimizing Computing Infrastructures", n. 2020-05094, and by the Knowledge Foundation (KKS) with grant "Mälardalen University Automation Research Center (MARC)", n. 20240011.

[1] M. Kaheni and A. V. Papadopoulos are with Mälardalen University, Västerås, Sweden. e-mail: mojtaba.kaheni@mdu.se, alessandro.papadopoulos@mdu.se.

[2] Martina Lippi and Andrea Gasparri are with the Department of Civil, Computer Science and Aeronautical Technologies Engineering, Roma Tre University, Italy. e-mail: martina.lippi@uniroma3.it, andrea.gasparri@uniroma3.it.

information about the training data through interaction and analysis of the weight vectors, $\mathbf{w}_i$, e.g., [20], [21]. Consequently, various privacy attacks have been demonstrated, revealing the ability to extract meaningful insights directly from the model parameters that store information about the training data. These attacks fall into categories such as model inversion [20], [21] in which an adversary, who has access to $\mathbf{w}_i$ in the training phase, attempts to reverse-engineer or invert the model to glean information about the individual data points used during training, and membership inference [22], [23], where an adversary attempts to determine whether a specific sample was used in the training set of an agent.

The most popular approach to mitigate attacks on privacy is *differential privacy*. Differential privacy aims to protect privacy by adding noise to sensitive attributes. In FL, differential privacy is used to add noise to the weight vectors of agents communicated within the network [24], [25], [26]. Nevertheless, differential privacy methods typically diminish the accuracy of FL [26]. As detailed below, this paper proposes Fedsecure, a novel FL algorithm that improves typical accuracy levels compared to differential privacy while preserving privacy.

A. Statement of Contributions

In summary, this paper's main contributions include:

- Restructuring the conventional FL optimization problem to an equivalent distributed optimization problem with constraints.
- Introduction of a novel FL algorithm, FedSecure, which eliminates the need to share information required to craft state-of-the-art privacy attacks.
- Evaluation of FedSecure's accuracy against noising before model aggregation FL (NbAFL) [26] and to the standard average aggregation method (FedAvg) with no privacy guarantees.

B. Organization

The remainder of the paper is organized as follows. We present the conventional distributed optimization representation of FL in Section II. In Section III, we introduce FedSecure. Simulation results over MNIST and CIFAR-10 are presented in Section IV. Finally, we conclude this paper by summarizing remarks and introducing possible future research directions in Section V.

C. Notation

Throughout this paper, we adopt the following notation. The variable $\mathbb{R}$ represents the sets of real numbers, scalars are denoted with light typeface, e.g., x, while (column) vectors and matrices are denoted as bold lower-case and capital letters, such as $\mathbf{x}$ and $\mathbf{X}$, respectively. Furthermore, we mean a coordinate-wise comparison when a comparison operator compares two vectors. In other words, let us denote the ith coordinate of $\mathbf{x}$ as $x(i)$, the notations $\mathbf{a} < \mathbf{b}$, $\mathbf{a} \leq \mathbf{b}$ and $\mathbf{a} = \mathbf{b}$ mean that $|\mathbf{a}| = |\mathbf{b}|$ and $a(i) < b(i)$, $a(i) \leq b(i)$ and $a(i) = b(i)$ for all $i \in \{1, \ldots, |\mathbf{a}|\}$, respectively. The variables $\mathbf{0}$ and $\mathbf{1}$ denote the vectors composed of all zeros and ones, respectively, with proper dimension. Given a vector $\mathbf{x}$, the notation $[\mathbf{x}]_+$ denotes the element-wise maximum function with respect to 0, i.e., $\max\{\mathbf{x}, \mathbf{0}\}$.

II. FEDERATED LEARNING PROBLEM STATEMENT

Traditional centralized learning algorithms necessitate the availability of all training samples to a central processing unit, which computes the optimal model. However, these algorithms may not be suitable for certain scenarios. For example, in cases where:

- The owners of the training samples prefer not to disclose private information with a central processing unit or
- The number of samples is excessively large, making it impractical or even impossible to process them with a single processing unit.

FL addresses these limitations by distributing the learning process among multiple agents holding private local datasets. This approach enhances privacy preservation and facilitates the processing of large datasets in a distributed manner. Consider a set of n agents, represented as $\mathcal{V}$, communicating with a central server unit and collectively aiming to learn the weight vector $\mathbf{w}$ of a global model. Each agent, denoted as i, has access to a local training dataset $\mathcal{D}_i$. The goal for each agent is to determine the optimal weight vector $\mathbf{w} \in \mathbb{R}^m$, solving the following optimization problem:

$$\min_{\mathbf{w}} \sum_{i=1}^{n} f\left(\mathbf{w}, \mathcal{D}_i\right) = \min_{\mathbf{w}} \sum_{i=1}^{n} f_i\left(\mathbf{w}\right), \tag{1}$$

where f_i, generally referred to as loss function, depends on the local dataset $\mathcal{D}_i$. For instance, the cross-entropy loss or the Mean Square Error functions can be selected as f_i.

At each time step $k+1$ in conventional FL algorithms, each agent i begins with a global model received from the server at time k, denoted as $\mathbf{w}(k)$, and iteratively refines it based on its individual training dataset $\mathcal{D}_i$ using a Stochastic Gradient Descent (SGD) algorithm,

$$\mathbf{w}_i\left(k+1\right) = \mathbf{w}(k) - \alpha \cdot \nabla f_i\left(\mathbf{w}(k)\right), \tag{2}$$

where $\alpha > 0$ is usually called the *learning rate*. In the subsequent step, the updated values of the weight vectors at time step $k+1$, $\mathbf{w}_i\left(k+1\right)$, are communicated to the server. Given that the server possesses knowledge of $\mathbf{w}(k)$, it can determine $\nabla f_i\left(\mathbf{w}(k)\right)$ if it is aware of α. Alternatively, even if α is kept private from the server, it can still discern the direction of $\nabla f_i\left(\mathbf{w}(k)\right)$.

The core concept in state-of-the-art privacy attacks revolves around how an adversary can glean information about the training dataset, $\mathcal{D}_i$, through insights derived from the disclosure of $\nabla f_i\left(\mathbf{w}(k)\right)$.

Remark 1: As depicted in Fig. 1, in this paper, we consider the general scenario where there is a vulnerability in disclosing all individual weight vectors, $\mathbf{w}_i(k)$. The scenario in which the aggregated vector $\mathbf{w}(k)$ (i.e., the link from a server to agents) is not secured can be considered a special case of our setting. Since $\mathbf{w}(k)$ is the outcome of the

aggregation function that the server unit applies to $\mathbf{w}_i(k)$, $\mathbf{w}(k)$ can therefore be easily determined if the aggregation function and all $\mathbf{w}_i(k)$ are known.

III. PRIVACY PRESERVING FEDERATED LEARNING

Following the discussion regarding the potential disclosure of information about the local training dataset through eavesdropping of $\mathbf{w}_i(k)$ and $\nabla f_i(\mathbf{w}(k))$, this section introduces our proposed privacy-preserving FL algorithm, FedSecure.

A. Problem formulation

In FedSecure, we reformulate the general optimization problem in FL, as introduced in (1), into an equivalent constrained optimization problem that mathematically shares a common optimal value with (1), i.e., each agent i aims to solve the following problem

$$\min_{\mathbf{w}_i} \sum_{i=0}^{n} f_i(\mathbf{w}_i), \tag{3a}$$

$$\text{subject to} \quad \mathbf{w}_i = \mathbf{w}_j, \quad \forall i,j \in \mathcal{V}. \tag{3b}$$

In the above formulation, each agent i holds a copy of the weight vector $\mathbf{w}_i$, and all these copies are required to agree in the constraint (3b), $\forall i,j \in \mathcal{V}$, while minimizing the same cost function as in (1). The total number of constraints in (3b) is equal to the number of unique unordered pairs from the set $\mathcal{V}$, which is:

$$\binom{n}{2} = \frac{n(n-1)}{2}. \tag{4}$$

However, many of these pairs are redundant because of the transitive property of equality. For instance, if $n = 3$, the relations $\mathbf{w}_1 = \mathbf{w}_2$, $\mathbf{w}_1 = \mathbf{w}_3$ and $\mathbf{w}_2 = \mathbf{w}_3$ are redundant; any two of these relations are sufficient to imply the third. In the following proposition, we prove that $n-1$ constraints are sufficient to represent (3b), and there are infinity many sets of $n-1$ constraints that represent (3b).

Proposition 1: There are infinitely many sets of $n-1$ constraints that equivalently represent the full set of pairwise constraints $\mathbf{w}_i = \mathbf{w}_j$ for all $i,j \in \mathcal{V}$, where $\mathcal{V}$ contains n agents.

Proof: Let us consider $\mathcal{V} = \{1,2,\dots,n\}$, and the full set of constraints $\mathbf{w}_i = \mathbf{w}_j$ for all $i,j \in \mathcal{V}$. The complete system of constraints requires that all elements $\mathbf{w}_i$ for $i \in \mathcal{V}$ are equal, i.e.,

$$\mathbf{w}_1 = \mathbf{w}_2 = \mathbf{w}_3 = \cdots = \mathbf{w}_n.$$

This can be viewed as a fully connected graph, where each node corresponds to an element $\mathbf{w}_i$, and each edge corresponds to a constraint $\mathbf{w}_i = \mathbf{w}_j$ between nodes i and j. The number of edges (i.e., constraints) in this graph is $\frac{n(n-1)}{2}$, representing the total number of pairwise constraints.

To represent the same equality constraints with fewer equations, we can construct a minimal set of constraints that ensures that all elements are equal by using the transitive property of equality. This is equivalent to finding a *spanning tree* in the graph of constraints.

A spanning tree of n vertices has exactly $n-1$ edges (constraints), and once these edges are specified, the equality of all elements follows transitively. Thus, any spanning tree of the fully connected graph corresponds to a valid set of $n-1$ constraints that fully describe the system.

According to Cayley's formula, a complete graph with n vertices has n^{n-2} spanning trees. Furthermore, any linear combination of these set of constraints weighted by $\alpha_i \in \mathbb{R}$, $i = 1,\dots,n^{n-2}$ leads to another valid set of $n-1$ constraints that represents (3b). As a result, there exists an infinite number of sets of $n-1$ constraints that represent (3b). ■

The equality constraints in (3) can be replaced with pairs of double-sided inequality constraints, i.e.,

$$\min_{\mathbf{w}_i} \sum_{i=0}^{n} f_i(\mathbf{w}_i), \tag{5a}$$

$$\text{subject to} \quad \mathbf{w}_i \le \mathbf{w}_j, \quad \text{and} \quad \mathbf{w}_j \le \mathbf{w}_i, \quad \forall i,j \in \mathcal{V}. \tag{5b}$$

Therefore, the formulation in (3) can be rewritten as a standard constrained distributed optimization problem with inequality constraints as follows

$$\min_{\mathbf{w}_i} \sum_{i=0}^{n} f_i(\mathbf{w}_i), \tag{6a}$$

$$\text{subject to} \quad \sum_{i=0}^{n} \mathbf{g}_i(\mathbf{w}_i) \le \mathbf{0}, \tag{6b}$$

where according to the results we obtained in Proposition 1, $\mathbf{g}_i(\cdot) \in \mathbb{R}^m \to \mathbb{R}^{2\cdot m\cdot(n-1)}$ in (6b) and can be designed in infinitely possible ways to represent the equality constraint in (3b). For instance, the following matrix

$$\mathbf{G} = \left[\mathbf{g}_1(\mathbf{w}_1), \mathbf{g}_2(\mathbf{w}_2), \dots, \mathbf{g}_n(\mathbf{w}_n)\right]_{2\cdot m\cdot(n-1)\times n} = \begin{bmatrix} \mathbf{w}_1 & -\mathbf{w}_2 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ -\mathbf{w}_1 & \mathbf{w}_2 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{w}_2 & -\mathbf{w}_3 & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{w}_2 & \mathbf{w}_3 & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{w}_{(n-1)} & -\mathbf{w}_n \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & -\mathbf{w}_{(n-1)} & \mathbf{w}_n \end{bmatrix}, \tag{7}$$

represents one of infinitely many realization that $\mathbf{G}$ can take to make (6b) equal to (3b). The matrix (7), represents the case when we select

$$\mathbf{w}_1 = \mathbf{w}_2,\ \mathbf{w}_2 = \mathbf{w}_3,\ \dots,\ \mathbf{w}_{n-1} = \mathbf{w}_n.$$

as the set of $n-1$ constraints to represent (3b), and the equality constraint is replaced with pairs of double-sided inequality constraints as in (6).

To achieve the desired results, we assume that the following assumptions hold for the functions $f_i(\cdot)$ and $\mathbf{g}_i(\cdot)$.

Assumption 1: For all agents $i \in \mathcal{V}$, we assume that the respective gradients $f_i(\cdot)$ and $\mathbf{g}_i(\cdot)$ are convex and uniformly bounded. In other words, there exists a positive constant C such that $\|\nabla f_i(\mathbf{w}_i)\| \le C$ and $\|\nabla \mathbf{g}_i(\mathbf{w}_i)\| \le C$, where $\|\nabla \mathbf{g}_i(\mathbf{w}_i)\|$ represents a Jacobian matrix at $\mathbf{w}_i$.

Assumption 2: For all $i \in \mathcal{V}$, we assume that the functions

Algorithm 1 FedSecure Protocol

Phase 1 - Initialization

for each $i \in \mathcal{V}$ **do**

Randomly initialize $\mathbf{w}_i(1) \in \mathbb{R}^m$.

Define learning rate $\alpha > 0$.

Define $\sigma > 0$.

Initialize $\boldsymbol{\lambda}_i(1) \in \mathbb{R}^{2 \cdot m \cdot (n-1)} \leftarrow \mathbf{0}$.

Initialize $\mathbf{y}_i(1) \in \mathbb{R}^{2 \cdot m \cdot (n-1)} \leftarrow m \cdot \mathbf{g}_i\left(\mathbf{w}_i(1)\right)$.

end for

while the stop criterion is not satisfied, do:

Phase 2 - Aggregation in Server Unit

Collect $\boldsymbol{\lambda}_i(k)$ and $\mathbf{y}_i(k)$, $i \in \mathcal{V}$.

Update $\bar{\boldsymbol{\lambda}}(k) \leftarrow \frac{1}{n} \cdot \left(\sum_{i=1}^{n} \boldsymbol{\lambda}_i(k) \right)$.

Update $\overline{\mathbf{y}}(k) \leftarrow \frac{1}{n} \cdot \left(\sum_{i=1}^{n} \mathbf{y}_i(k) \right)$.

Send $\bar{\boldsymbol{\lambda}}(k)$ and $\overline{\mathbf{y}}(k)$ to all agents.

Phase 3 - Agents Update

for each $i \in \mathcal{V}$ **do**

$\mathbf{u}_i(k) \leftarrow \nabla f_i\left(\mathbf{w}_i(k)\right) + \nabla \mathbf{g}_i\left(\mathbf{w}_i(k)\right)^{\intercal} \bar{\boldsymbol{\lambda}}(k)$.

$\mathbf{v}_i(k) \leftarrow \overline{\mathbf{y}}(k) - \sigma \cdot \alpha \cdot \bar{\boldsymbol{\lambda}}(k)$.

$\mathbf{w}_i(k+1) \leftarrow \mathbf{w}_i(k) - \alpha \cdot \mathbf{u}_i(k)$.

$\boldsymbol{\lambda}_i(k+1) \leftarrow \left[\bar{\boldsymbol{\lambda}}(k) + \alpha \cdot \mathbf{v}_i(k) \right]_+$.

$\mathbf{y}_i(k+1) \leftarrow \overline{\mathbf{y}}(k) + m\left(\mathbf{g}_i\left(\mathbf{w}_i(k+1)\right) - \mathbf{g}_i\left(\mathbf{w}_i(k)\right)\right)$.

Send $\boldsymbol{\lambda}_i(k+1)$, $\mathbf{y}_i(k+1)$ to the server.

end for

end while

$\mathbf{g}_i(\cdot)$ are private and non-disclosed to the server unit. In other words, the server unit is unaware of the exact realization of $\mathbf{g}_i(\cdot)$ for all $i \in \mathcal{V}$. For instance, these functions can be provided by a trusted third party that assigns proper vectors, $\mathbf{g}_i(\cdot)$, to all agents in such a way that the constraint (6b) represents (3b).

B. FedSecure algorithm

The basic idea of FedSecure is that, by reshaping the FL problem as in the standard constrained distributed optimization form in (6), we can apply a dual decomposition method and avoid sharing primal information, such as $\mathbf{w}_i$ and $\nabla f_i\left(\mathbf{w}(k)\right)$, with the server, thus boosting privacy preservation features. More in detail, in dual decomposition approaches, constraints are added to the objective function considering Lagrange multipliers, and then a $minmax$ problem is solved. Several algorithms have been published in the literature to solve this type of problem [27], [28], [29], [30], [31]. Algorithm 1 represents our proposed privacy-preserving FL, which indeed adopts the results in [30] to this case study.

Phase 1 of Algorithm 1 represents the first (offline) phase to initialize the system. More specifically, agents randomly initialize their estimate of the optimal network weight vector, $\mathbf{w}_i(1)$, and define a common learning rate, $\alpha > 0$. The parameter $\sigma > 0$ in Phase 1 is intended to bring consensus among agents. Furthermore, the Lagrange multipliers vector

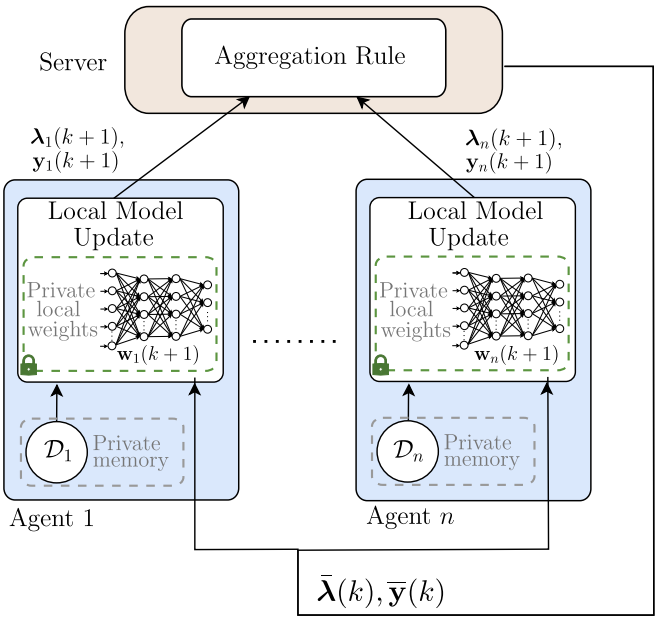

Fig. 2: FedSecure architecture where each agent i (blue boxes) holds a private dataset $\mathcal{D}_i$ and a local model $\mathbf{w}_i$ sends the variables $\boldsymbol{\lambda}_i$ and $\mathbf{y}_i$ to a server unit (on the top). The latter aggregates the received variables and sends the resulting vectors back to the agents.

denoted as $\boldsymbol{\lambda}_i(k)$ is set to the zero vector, and the output vector $\mathbf{y}_i(k) = m\,\mathbf{g}_i(\mathbf{w}_i(1))$ is initialized. These represent the vectors that will be transmitted to the server.

Next, Phases 2 and 3 are iteratively executed at each time step until a certain criterion is satisfied, such as a certain number of training steps is reached, or the local training loss is below a predefined threshold. In Phase 2, the server collects the vectors $\boldsymbol{\lambda}_i(k)$ and $\mathbf{y}_i(k)$ from all agents, aggregates them by computing the respective average vectors, denoted as $\bar{\boldsymbol{\lambda}}(k)$ and $\overline{\mathbf{y}}(k)$, respectively, and then sends the averages back to the agents.

In Phase 3, the agents locally update their estimates of the optimal weight vector, along with the vectors $\boldsymbol{\lambda}_i(k+1)$ and $\mathbf{y}_i(k+1)$ by following [30]. More specifically, for each agent we introduce two additional auxiliary variables, i.e., $\mathbf{u}_i \in \mathbb{R}^m$ and $\mathbf{v}_i \in \mathbb{R}^{2 \cdot m \cdot (n-1)}$, which are updated based on the gradients of f_i and $\mathbf{g}_i$ as well as the aggregate variables $\bar{\boldsymbol{\lambda}}(k)$ and $\overline{\mathbf{y}}(k)$, as reported in the first lines of Phase 3. Then, the local weight vector $\mathbf{w}_i(k+1)$ is computed using the learning rate α and the Lagrange multiplier vector $\boldsymbol{\lambda}_i(k+1)$ and the output vector $\mathbf{y}_i(k+1)$ are updated. The last two vectors are sent to the server to perform aggregation in the next iteration.

A visual representation of FedSecure is provided in Fig. 2. In the figure, each agent i (represented as a blue box) holds a private dataset $\mathcal{D}_i$ and a local model with weights $\mathbf{w}_i$. At each time step k, it receives the aggregated vectors $\bar{\boldsymbol{\lambda}}(k)$ and $\overline{\mathbf{y}}(k)$ from the server unit (in the top), use them to update the local model and sends the updated variables $\boldsymbol{\lambda}_i(k+1)$ and $\mathbf{y}_i(k+1)$ to the server.

Remark 2: Assumption 2 ensures that, when execut-

ing Algorithm 1, the server unit is unable to reconstruct $\mathbf{w}_i(.)$, thus preventing it from accessing information about $\nabla f_i\left(\mathbf{w}_i(.)\right)$. Noticing that state-of-the-art privacy attacks, such as [20], [21], rely on this information to reveal details about training datasets, Algorithm 1 strengthens the privacy of FL by safeguarding the privacy of $\nabla f_i\left(\mathbf{w}_i(.)\right)$.

Let $\mathbf{w}^*$ denote the optimal solution of (3). We define the regret functions as follows:

$$\text{Reg}^f(T) \triangleq \sum_{i=1}^{n}\left(\sum_{k=1}^{T}\left(f_i\left(\mathbf{w}_i(k)\right)-f_i\left(\mathbf{w}^*\right)\right)\right), \quad (8)$$

and

$$\text{Reg}^g(T) \triangleq \left\|\left[\sum_{i=1}^{n}\sum_{k=1}^{T}\mathbf{g}_i\left(\mathbf{w}_i(k)\right)\right]_+\right\|. \quad (9)$$

where T represents the number of training steps. A successful FL algorithm must ensure that:

$$\lim_{T\to\infty}\frac{\text{Reg}^f(T)}{T}=0, \quad (10a)$$

$$\lim_{T\to\infty}\frac{\text{Reg}^g(T)}{T}=0. \quad (10b)$$

The success criteria, as expressed through (10), articulate the convergence of regrets over time, indicating the effectiveness and convergence properties of the FL algorithm. The following theorem, which relies on the results from [30] in our FL case study, guarantees the desired performance.

Theorem 1: Suppose Assumption 1 is satisfied. Consider that FedSecure Algorithm 1 is executed by all agents with $\alpha=\sigma^{-1}T^{-\beta}$, where $\sigma=2n(nC^2+1)$, C is the upper bound of $\|\nabla f_i(\mathbf{w}_i)\|$ and $\|\nabla \mathbf{g}_i(\mathbf{w}_i)\|$ as in Assumption 1, and $\beta\in(0,1)$. Then the limits introduced in (10) tend to 0.

Proof: The aggregation process in the server unit during Phase 2 of Algorithm 1 resembles the scenario where agents communicate in a complete graph, assigning $\frac{1}{n}$ to their incoming edges. In this context, the graph's adjacency matrix is doubly stochastic and strongly connected. Consequently, noticing the proposed values for α and σ, the fulfillment of Assumption 1 satisfies all the essential conditions outlined in Theorem 1 of [30]. As a result, it holds

$$\lim_{T\to\infty}\frac{\text{Reg}^f(T)}{T}=0,$$
$$\lim_{T\to\infty}\frac{\text{Reg}^g(T)}{T}=0,$$

concluding the proof. ■

IV. Simulation results

In this section, we perform a comparative analysis to assess the effectiveness of FedSecure compared to a well-established differential privacy algorithm, NbAFL, and to a standard FL aggregation method, i.e., the average aggregation, FedAvg.

Datasets: We consider the MNIST dataset [32] for digit classification and the CIFAR-10 dataset [33] for image classification with various classes, including, for example, birds, dogs, cars, and trucks. The former is composed of $|\mathcal{D}|_{\text{MNIST}}=60000$ binary images, with dimension 28×28 pixel, for training and 10000 for testing. The latter is composed of $|\mathcal{D}|_{\text{CIFAR-10}}=50000$ color images, with dimension 32×32 and associated with 10 classes, for training and 10000 for testing. The test sets of both datasets are reserved for performance evaluation.

Agents: In the MNIST case study, we analyze the performance obtained with different numbers of agents. Specifically, we consider $n\in\mathcal{N}$ with $\mathcal{N}=\{5,10,20,30,50\}$. Each agent's neural network is a multi-layer perceptron with three fully connected layers with 128, 64, and 10 neurons, respectively. Local training datasets $\mathcal{D}_i,\forall i\in\mathcal{V}$ are obtained with uniform distribution. In particular, we analyze both the case where the local dataset size varies according to the number of agents, i.e., $|\mathcal{D}_i|=\lfloor|\mathcal{D}|_{\text{MNIST}}/n\rfloor,\forall i\in\mathcal{V}$, and the case where the local dataset size of each agent is fixed regardless of the number of agents in the system and is equal to $|\mathcal{D}_i|=\lfloor|\mathcal{D}|_{\text{MNIST}}/\max\{\mathcal{N}\}\rfloor,\forall i\in\mathcal{V}$. We refer to these two dataset distributions as *varying local dataset size* and *fixed local dataset size*, respectively.

In the CIFAR-10 case study, we analyze a system composed of $n=10$ agents. Each agent's neural network has three convolutional layers (16, 32, 64 filters) with batch normalization, followed by a fully connected layer with 128 neurons and a 10-class output layer. We consider that the local training datasets are obtained by uniformly distributing the CIFAR-10 training set among the agents, i.e., $|\mathcal{D}_i|=\lfloor|\mathcal{D}|_{\text{CIFAR-10}}/n\rfloor,\ \forall i\in\mathcal{V}$.

In both case studies, we consider ReLU activation functions and cross-entropy loss functions for the networks. Models are trained for 100 steps for the MNIST case and 200 steps for the CIFAR-10 case with a learning rate $\alpha=0.01$, and all agents contribute to the aggregation step. Regarding FedSecure, the functions $\mathbf{g}_i(\mathbf{w}_i)$ are set as in (7), and gain $\sigma=1$ is used. It is important to note that the chosen neural network architectures do not aim to achieve maximum performance. Instead, this numerical validation only aims to assess FedSecure's effectiveness compared to well-established algorithms with a similar network structure.

Baselines: We consider the following baselines based on the server-worker architecture in Fig. 1:

- FedAvg [34], that represents the standard aggregation rule (with no privacy guarantees) computing the global weight vector $\mathbf{w}$ as the coordinate-wise average of the agents' local weight vectors $\mathbf{w}_i$, $\forall i\in\mathcal{V}$;
- NbAFL algorithm [26], that enhances privacy by adding noise $\mathcal{N}(0,\sigma^2)$ to the agents' local weight vectors $\mathbf{w}_i$, $\forall i\in\mathcal{V}$ before transmitting them to the server. Specifically, the variance σ^2 is directly proportional to a constant c, a clipping value C, and inversely proportional to the minimum dataset size $\min_{i\in\mathcal{V}}\{\mathcal{D}_i\}$ and a threshold ϵ, which encodes the protection level, meaning that the higher ϵ, the lower the protection level provided by the algorithm.

All the methods are implemented using the PyTorch library, with the weights initialized according to the default uniform

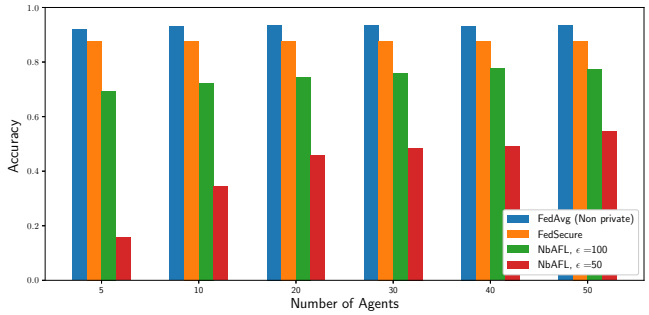

Fig. 3: Comparison of the accuracy on the MNIST test set achieved by the baselines and the proposed method when varying the number of agents and considering *fixed* local dataset size.

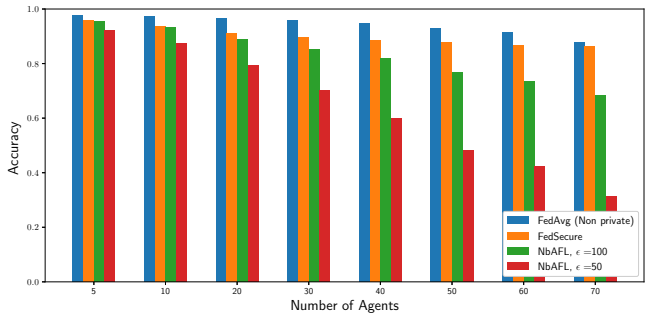

Fig. 4: Comparison of the accuracy on the MNIST test set achieved by the baselines and the proposed method when varying the number of agents and considering *varying* local dataset size.

distributions. Additionally, for NbAFL, all agents start with identical weight values, as specified by the algorithm. For the NbAFL variance, we set $c = 20$ and $C = 1$ and analyze the performance with $\epsilon = 50$ and $\epsilon = 100$.

MNIST results: Figure 3 shows the accuracy on the test set achieved by the baselines and the proposed method when varying the number of agents and considering fixed local dataset size. More in detail, FedAvg is reported in blue, the proposed FedSecure in orange, and NbAFL with $\epsilon = 100$ and $\epsilon = 50$ in green and red, respectively. As expected, FedAvg achieves the highest accuracy across all cases; however, this comes at the cost of providing no privacy guarantees. In comparison, FedSecure follows closely in all cases while providing privacy-preserving features. Furthermore, the plot shows that the accuracy remains stable for FedAvg and FedSecure by varying the set of agents involved. Regarding NbAFL results, we can observe that, coherently with the results in [26], as the protection level decreases (i.e., as ϵ increases), the accuracy improves. This indicates a trade-off between the accuracy and the level of privacy protection. Moreover, for NbAFL and a given ϵ value, the accuracy increases as the number of agents in the system grows. This is due to the reduction in the standard deviation of the additive noise and the expansion of the global dataset available for training as more agents participate.

Figure 4 presents a comparison similar to that in Figure 3, but with varying local dataset sizes taken into account. In this case, we can observe that as the number of agents

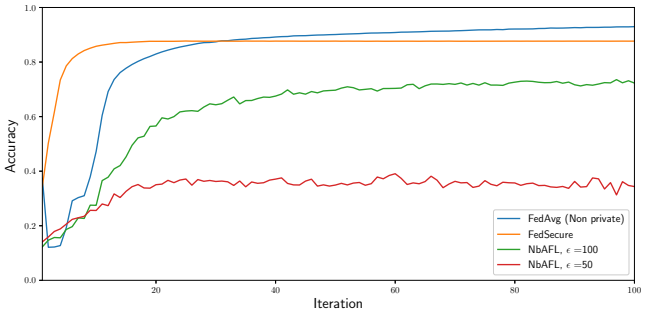

Fig. 5: Accuracy results on MNIST test set obtained during the training process with $n = 10$ and fixed local dataset size.

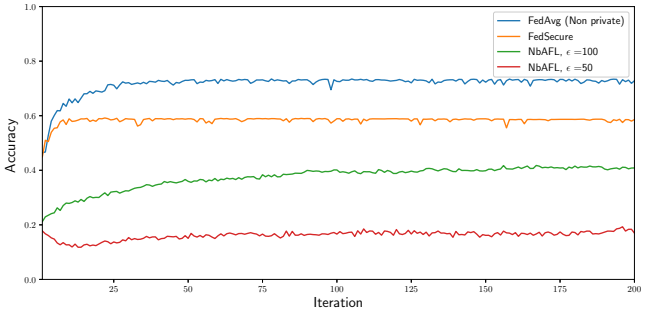

Fig. 6: Accuracy results on CIFAR10 test set obtained during the training process with $n = 10$.

increases, all methods exhibit a reduction in accuracy. This is motivated by the fact that as the number of agents increases, the local dataset size for each agent decreases, resulting in lower performance for each *individual* neural network, thus affecting the performance of the FL algorithm. However, the figure also shows that FedAvg and FedSecure's performance decline is more gradual than that of the NbAFL baseline.

Finally, Figure 5 shows the accuracy obtained by the baselines and the proposed FedSecure during the training process in the case of $n = 10$ and fixed local dataset size. In particular, at the end of the training process, FedAvg reaches accuracy equal to 0.93, followed by FedSecure achieving 0.88 and by NbAFL obtaining 0.72 and 0.34 for $\epsilon = 100$ and $\epsilon = 50$, respectively. Additionally, the figure highlights that FedSecure exhibits the fastest convergence rate compared to the other methods.

CIFAR-10 results: Figure 6 shows the accuracy of the CIFAR-10 test set obtained by the baselines and by the proposed FedSecure during the training process with $n = 10$. Specifically, FedAvg is reported in blue, the proposed FedSecure in orange, and NbAFL with $\epsilon = 100$ and $\epsilon = 50$ in green and red, respectively. Note that given that the classification task is inherently more challenging with the CIFAR-10 dataset than with the MNIST dataset, it is expected that the accuracy values are generally lower compared to MNIST. In particular, at the end of the training process, FedAvg reaches accuracy equal to 0.73, followed by FedSecure achieving 0.59 and by NbAFL obtaining 0.41 and 0.17 for $\epsilon = 100$ and $\epsilon = 50$, respectively. Also, in this case study, FedSecure outperforms other methods in terms

of convergence speed.

In summary, the results on MNIST and CIFAR-10 datasets show that FedSecure offers a good balance between accuracy and privacy protection, avoiding disclosing primal data, such as $\mathbf{w}_i$ or $\nabla f_i(\mathbf{w}_i)$.

V. Conclusion

In this paper, the FedSecure algorithm was proposed, representing an FL approach designed to preserve the privacy of the agents' datasets. Unlike state-of-the-art approaches, we did not assume any sharing of the weight vectors, which may disclose information regarding the local (private) datasets. This was achieved by reformulating the FL distributed optimization problem in a constrained form with inequality constraints and applying a dual decomposition method. We validated FedSecure on the MNIST and CIFAR-10 datasets and compared their performance with respect to state-of-the-art FL baselines, namely FedAvg and NbAFL. In future work, we aim to extend the algorithm to provide resiliency features against model and data poisoning attacks and extend FedSecure to a fully distributed setting. In this scenario, the server unit is omitted, and direct communication among neighboring agents is enabled, thereby enhancing the approach's scalability and circumventing the presence of a critical single point of failure in the server.

References

[1] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowl.-Based Syst.*, vol. 216, p. 106775, 2021.

[2] J. Le, X. Lei, N. Mu, H. Zhang, K. Zeng, and X. Liao, "Federated continuous learning with broad network architecture," *IEEE Trans. Cybern.*, vol. 51, no. 8, pp. 3874–3888, 2021.

[3] X. Li, Z. Qu, B. Tang, and Z. Lu, "FedLGA: Toward system-heterogeneity of federated learning via local gradient approximation," *IEEE Trans. Cybern.*, vol. 54, no. 1, pp. 401–414, 2024.

[4] L. Zhang, W. Cui, B. Li, Z. Chen, M. Wu, and T. S. Gee, "Privacy-preserving cross-environment human activity recognition," *IEEE Trans. Cybern.*, vol. 53, no. 3, pp. 1765–1775, 2023.

[5] T. Zhang, L. Gao, C. He, M. Zhang, B. Krishnamachari, and A. S. Avestimehr, "Federated learning for the Internet of Things: Applications, challenges, and opportunities," *IEEE Internet Things Mag.*, vol. 5, no. 1, pp. 24–29, 2022.

[6] N. M. Jebreel, J. Domingo-Ferrer, D. Sánchez, and A. Blanco-Justicia, "LFighter: Defending against the label-flipping attack in federated learning," *Neural Networks*, vol. 170, pp. 111–126, 2024.

[7] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *European Symp. Res. Comput. Secur. (ESORICS)*, Springer. Springer International Publishing, 2020, pp. 480–501.

[8] H. Yang, D. Gu, and J. He, "A robust and efficient federated learning algorithm against adaptive model poisoning attacks," *IEEE Internet Things J.*, vol. 11, no. 9, pp. 16 289–16 302, 2024.

[9] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," in *USENIX Secur. Symp.*, 2020, pp. 1605–1622.

[10] M. Chen, H. Liu, H. Chi, and P. Xiong, "Heterogeneous ensemble federated learning with GAN-based privacy preservation," *IEEE Trans. Sustain. Comput.*, vol. 9, no. 4, pp. 591–601, 2024.

[11] R. Xu, B. Li, C. Li, J. B. D. Joshi, S. Ma, and J. Li, "TAPFed: Threshold secure aggregation for privacy-preserving federated learning," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 5, pp. 4309–4323, 2024.

[12] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Int. Conf. Mach. Learn.*, J. Dy and A. Krause, Eds., vol. 80, 2018, pp. 5650–5659.

[13] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[14] E. M. El Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in Byzantium," in *Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 3521–3530.

[15] Z. Wu, Q. Ling, T. Chen, and G. B. Giannakis, "Federated variance-reduced stochastic gradient descent with robustness to Byzantine attacks," *IEEE Trans. Signal Process.*, vol. 68, pp. 4583–4596, 2020.

[16] C. Xie, S. Koyejo, and I. Gupta, "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance," in *Int. Conf. Mach. Learn.*, vol. 97, 2019, pp. 6893–6901.

[17] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *AAAI Conf. Artif. Intell.*, vol. 33, no. 01, 2019, pp. 1544–1551.

[18] M. Kaheni, M. Lippi, A. Gasparri, and M. Franceschelli, "Selective trimmed average: A resilient federated learning algorithm with deterministic guarantees on the optimality approximation," *IEEE Trans. Cybern.*, vol. 54, no. 8, pp. 4402–4415, 2024.

[19] C. Yin and Q. Zeng, "Defending against data poisoning attack in federated learning with non-IID data," *IEEE Trans. Comput. Soc. Syst.*, vol. 11, no. 2, pp. 2313–2325, 2024.

[20] S. Mohammadi, M. Mohammadi, S. Sinaei, A. Balador, E. Nowroozi, F. Flammini, and M. Conti, "Balancing privacy and accuracy in federated learning for speech emotion recognition," in *Conf. Comput. Sci. Intell. Syst. (FedCSIS)*, vol. 35, 2023, pp. 191–199.

[21] A. Hatamizadeh, H. Yin, P. Molchanov, A. Myronenko, W. Li, P. Dogra, A. Feng, M. G. Flores, J. Kautz, D. Xu, and H. R. Roth, "Do gradient inversion attacks make federated learning unsafe?" *IEEE Trans. Med. Imaging*, vol. 42, no. 7, pp. 2044–2056, 2023.

[22] X. He, Y. Xu, S. Zhang, W. Xu, and J. Yan, "Enhance membership inference attacks in federated learning," *Comput. Secur.*, vol. 136, p. 103535, 2024.

[23] Y. Gu and Y. Bai, "LDIA: Label distribution inference attack against federated learning in edge computing," *J. Inf. Secur. Appl.*, vol. 74, p. 103475, 2023.

[24] P. Wang, Y. Yang, W. Sun, Q. Wang, B. Guo, J. He, and Y. Bi, "Federated learning with privacy-preserving incentives for aerial computing networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 6, pp. 5336–5348, 2024, Early Access.

[25] M. Iqbal, A. Tariq, M. Adnan, I. Ud Din, and T. Qayyum, "FL-ODP: An optimized differential privacy enabled privacy preserving federated learning," *IEEE Access*, vol. 11, pp. 116 674–116 683, 2023.

[26] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. Vincent Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3454–3469, 2020.

[27] M. Kaheni, E. Usai, and M. Franceschelli, "Resilient and privacy-preserving multi-agent optimization and control of a network of battery energy storage systems under attack," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 4, pp. 5320–5332, 2024.

[28] B. Turan, C. A. Uribe, H.-T. Wai, and M. Alizadeh, "Resilient primal-dual optimization algorithms for distributed resource allocation," *IEEE Trans. Control Netw. Syst.*, vol. 8, no. 1, pp. 282–294, 2021.

[29] M. Kaheni, E. Usai, and M. Franceschelli, "Resilient constrained optimization in multi-agent systems with improved guarantee on approximation bounds," *IEEE Control Syst. Lett.*, vol. 6, pp. 2659–2664, 2022.

[30] J. Li, C. Gu, Z. Wu, and T. Huang, "Online learning algorithm for distributed convex optimization with time-varying coupled constraints and bandit feedback," *IEEE Trans. Cybern.*, vol. 52, no. 2, pp. 1009–1020, 2022.

[31] M. Kaheni, E. Usai, and M. Franceschelli, "A distributed optimization and control framework for a network of constraint coupled residential besss," in *IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, 2021, pp. 2202–2207.

[32] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, 2012.

[33] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Master's thesis, University of Toronto, 2009.

[34] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intell. Stat.* PMLR, 2017, pp. 1273–1282.