IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 30 March 2023, accepted 20 April 2023, date of publication 24 April 2023, date of current version 3 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3269980

SURVEY

A Survey on Securing Federated Learning: Analysis of Applications, Attacks, Challenges, and Trends

HELIO N. CUNHA NETO[1], JERNEJ HRIBAR[2], IVANA DUSPARIC[3], DIOGO MENEZES FERRAZANI MATTOS[1], AND NATALIA C. FERNANDES[1]

[1]MídiaCom, PPGEET, Universidade Federal Fluminense (UFF), Niterói 24210-240, Brazil
[2]Department for Communication Systems, Jožef Stefan Institute, 1000 Ljubljana, Slovenia
[3]School of Computer Science, Trinity College Dublin, Dublin 2, D02 PN40 Ireland

Corresponding author: Helio N. Cunha Neto (heliocunha@midiacom.uff.br)

This work was supported in part by the Science Foundation Ireland (SFI)-NSFC Partnership Program under Grant 17/NSFC/5224, in part by the European Regional Development Fund through the SFI Research Centers Program under Grant 13/RC/2077 P2 SFI CONNECT, in part by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), in part by the Rede Nacional de Ensino e Pesquisa (RNP), and in part by the Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ).

ABSTRACT The growth of data generation capabilities, facilitated by advancements in communication and computation technologies, as well as the rise of the Internet of Things (IoT), results in vast amounts of data that significantly enhance the performance of machine learning models. However, collecting all necessary data to train accurate models is often unfeasible due to privacy laws. Federated Learning (FL) evolved as a collaborative machine learning approach for training models without sharing private data. Unfortunately, several in-design vulnerabilities have been exposed, allowing attackers to infer private data of participants and negatively impacting the performance of the federated model. In light of these challenges and to encourage the development of FL solutions, this paper provides a comprehensive analysis of secure FL proposals that both protect user privacy and enhance the performance of the model. We performed a systematic review using predefined criteria to screen and extract data from multiple electronic databases, resulting in a final set of studies for analysis. Through the systematic review methodology, the paper groups the security vulnerabilities of FL into model performance and data privacy attacks. It also presents an analysis and comparison of potential mitigation strategies against these attacks. Additionally, the paper conducts a security analysis of state-of-the-art FL applications and proposals based on the vulnerabilities addressed. Finally, the paper outlines the main applications of secure FL and lists future research challenges. The survey highlights the crucial role of security strategies in ensuring the protection of user privacy and model performance in the context of future FL applications.

INDEX TERMS Federated learning, machine learning, collaborative learning, information security, multi-access edge computing.

I. INTRODUCTION

Machine learning techniques have shown excellent performance in solving complex problems in the last few years [1], [2]. However, machine learning models require a large dataset for training, especially for deep learning [1]. Deep learning performance considerably increases when exposed to a large amount of data [3]. Collecting data from diverse sources provides a solution to data acquisition [4]. The conventional cloud computing or cloud-centric method involves mobile devices acting as data collection points that transmit the collected data to centralized cloud servers. Subsequently, the cloud servers process the data by performing various analytical and computational tasks [5]. The cloud-centric approach is widely used in several scenarios where the data is generated by local devices and processed by a system

The associate editor coordinating the review of this manuscript and approving it for publication was Pedro R. M. Inácio.

in the cloud. It is important to highlight that the processing system must not be machine learning-based [6], [7], [8]. A potential implementation for healthcare monitoring using a cloud-centric approach involves equipping a patient with Internet of Things (IoT) monitoring devices that generate sensor data. The data is subsequently transmitted to a cloud-based system that utilizes predictive modeling to anticipate potential diseases [6]. Unfortunately, centralized data merging for training a machine learning model harms data privacy protected by personal data protection laws in several countries worldwide.

Increasingly strict private data protection policies limit cloud-centric approaches to extracting knowledge from remote data. Personal data protection laws stipulate rights to data owners and obligations to institutions that hold data. A prominent law is the *General Data Protection Regulation* (GDPR),[1] in force throughout the European Union (EU), which establishes guidelines on storing and processing personal data in the EU [9]. The GDPR emphasizes the importance of protecting the fundamental rights and freedoms of individuals in the handling of their data [10]. It has inspired other countries[2] to adopt similar guidelines, such as the *Lei Geral de Proteo de Dados* (LGPD) in Brazil, the California Consumer Privacy Act (CCPA) in the United States, and the Personal Information Protection and Electronic Documents Act (PIPEDA) in Canada. In Brazil, the LGPD identifies the entities, either a public or private organization, that carries out any processing operation on the personal data [11]. Among the duties established for these entities is collecting explicit consent from the data owner and providing reports that identify the processing operations applied to the data, including the specification of its storage location, data masking, and data protection measures.

On top of privacy concerns, the cost of uploading raw data to the cloud also presents a significant challenge for the cloud-centric approach. Uploading data from a mobile device in an area with a poor network connection, for example, causes long delays to the training due to low throughput. The cloud-centric approach results in propagation delays that can cause unacceptable latency for real-time decision-making applications, such as anomaly detection [12]. Transferring data to the cloud for processing burdens both core and access networks. Overload is even more relevant when considering unstructured data, such as text, voice, or video. The restrictions of sending data to the cloud (*e.g.*, band and delay) are critical when cloud-centric training depends on wireless access networks [4]. Thus, current proposals consider developing mobile applications at the Multiaccess Edge Computing (MEC) [13], in which training becomes a task performed by three distinct actors: devices, edge, and cloud. The training anchored in the MEC model incurs high communication costs and is unsuitable for constant retraining applications [14], such as smart keyboards. Furthermore, outsourcing computing and data processing on edge servers involve transmitting potentially sensitive personal data, exposing privacy-sensitive data.

The aforementioned limitations of the cloud-centric approach have led to the development of Federated Learning (FL). This collaborative learning solution addresses issues such as privacy preservation and communication efficiency [15]. By allowing training with real data from mobile devices and preserving privacy-sensitive information, FL enables machine learning algorithms to run collaboratively without transferring private data to a cloud server. As a result, FL is a critical component in ensuring data privacy in distributed environments.

FL has two main entities, the participants — often referred to as clients — that train the machine learning models with their personal data, and the aggregation server that aggregates the local models, generating the global model. The global model aggregates knowledge of participants' local data in a single model. Participants perform the following tasks: i) every participant must retrieve parameters from the global model, ii) selected participants must update their local models with their data, and then iii) send the updated local parameters of the models to the server. The aggregation round is the process of updating the global model with data stored on participants, uploading the parameters, and performing aggregation. The aggregation server is responsible for controlling the aggregation rounds, selecting a subset of participants, and for aggregating the updates provided by the selected participants to improve the global model. The server randomly selects a subset of participants for the model update at each aggregation round. The federated approaches introduce the concept of using local computational resources, such as Central Processing Unit (CPU) or Graphics Processing Unit (GPU), for model training while participants can keep their data secure and private. Thereby, FL presents itself as a powerful approach to preserving privacy. Since data is always processed locally, the global operation aggregates the models without accessing the data stored on participants.

As FL enables the use of large amounts of data while preserving user privacy, it has become a popular solution in many areas, such as cyberattack detection, vehicle networks, smart healthcare, and IoT in general [16], [17], [18], [19], [20]. Unfortunately, the FL approach presents new vulnerabilities and security challenges. For example, a malicious entity may infer the honest data stored on participants despite FL sharing only the parameters of the model. The malicious entity may be a participant or an aggregation server willing to know the data stored on honest participants [21]. In addition, a malicious participant may contaminate the global model with poisoned models and data. The malicious participant may intentionally compromise the global model to mispredict a specific class or degrade the performance of the model [22]. Hence, achieving FL premises requires mechanisms to protect participants' private data and the global model performance.

[1] Available at https://gdpr-info.eu/. Accessed on 07/11/2022.

[2] Available at https://www.dlapiperdataprotection.com/. Accessed on 07/11/2022.

Numerous efforts have been made in the literature to conduct a comprehensive survey on FL and its applications [1], [14], [23], [24]. Although previous works [25], [26] have proposed a taxonomy of current FL attacks, they lack a thorough analysis of the literature on FL proposals from a security perspective. Additionally, none of them provided an analysis of the current applications from a security point of view. The contributions of our paper are as follows:

1) We provide an in-depth analysis of the fundamental concepts underlying FL, including its background and the role of Secure Multiparty Computation (SMC) and MEC in FL;
2) We present the main threats and vulnerabilities of FL, and possible solutions, including the main proposals to address the threats;
3) We provide insights into the security and privacy of current FL applications by analyzing potential security threats and vulnerabilities. The analysis contributes to developing more robust and secure FL systems.

Our contributions provide a comprehensive roadmap for researchers, developers, and practitioners to navigate the complexities of FL and design more privacy-preserving and secure solutions for various real-world applications.

The remainder of this paper is structured as follows. In Section II, we present the methodology used in this survey. In Section III, we review related work. We present a comprehensive overview of collaborative machine learning in Section IV. Section V provides background information on FL. Section VI describes the role of MEC and SMC in FL. We identify the current vulnerabilities of FL in Section VII. Section VIII analyzes the primary research challenges and opportunities in FL and current proposals to address these challenges. In addition, we provide a security analysis of these proposals. In Section IX, we examine the main applications of FL and perform a security analysis of these applications. Finally, in Section X, we provide concluding remarks.

II. METHODOLOGY

In this paper, we conducted a systematic review to identify and retrieve relevant studies for our survey. We follow the Preferred Reporting Items for Systematic reviews and Meta-Analyses (PRISMA) method, which is a systematic review in a rigorous and transparent methodology that comprehensively reviews the literature to identify, assess, and synthesize all relevant studies on a particular research question [27]. Our research questions are:

1) What is the level of security provided by current FL proposals and applications in light of known threats and vulnerabilities?
2) What are the primary threats to FL security, and what are the most effective solutions to mitigate them?

Our primary aim for this review is to conduct a comprehensive security analysis of current FL applications and solutions by evaluating their associated threats and vulnerabilities. To achieve this objective, we will systematically identify and categorize the threats to FL that are currently present.

The eligibility criteria for paper selection in this survey include peer-reviewed conference papers, journals, and magazines, as well as high-cited arXiv papers relevant to the survey. Although arXiv is not explicitly included in the databases, we include those relevant arXiv papers that are discovered by analyzing the reference citations of the papers we select in our primary database. By utilizing this method, we ensure the inclusion of relevant, high-quality literature that might not have been captured through our initial databases. The database we use are IEEE Xplorer, ACM Digital Library, and Scopus.

Our search is executed by conducting keyword searches on the databases. Our search keywords are: ''federated learning'', combined with (AND) ''threats'' and (OR) ''vulnerabilities''. Following this, we eliminate duplicate search results and limit our search to documents published in conference papers, journals, and magazines. Additionally, to ensure consistency, we exclude non-English papers and conduct a preliminary review of all papers' abstracts to eliminate those that do not align with our research objective. The primary basis for excluding papers from the review was their lack of relevance to the study of FL threats and vulnerabilities, including papers that focused on using FL for threat detection or described threats in other machine learning approaches. Thus, any paper that did not align with these criteria was excluded from the review process.

In addition to our primary search, we conducted another search for related surveys of FL application and concepts in the literature, utilizing the same designated database. Our survey-specific search utilized the keywords ''federated learning'' combined with the Boolean operator (AND) and the word ''survey'', which returned hundreds of papers. Our inclusion criteria were designed to identify surveys that addressed FL concepts, primary FL applications, and FL threats. At the same time, we excluded surveys that focused mainly on new FL techniques or hypothetical scenarios.

A. THREATS TO VALIDITY

Our systematic literature review may be susceptible to specific threats to validity, which could affect the reliability and robustness of our findings. If not properly addressed, these threats can potentially distort the interpretation and synthesis of the available evidence, and may result in inaccurate or incomplete conclusions.

Search bias is a significant threat to the validity of our study, if our search strategy is incomplete or biased toward certain types of studies or sources. This could lead to the exclusion of relevant studies or the inclusion of studies that do not accurately represent the population of interest, which may affect the generalizability of our findings. For example, if we limit our search to high-impact journals and conferences, we may inadvertently exclude relevant studies that have been published in lower-tier venues. This can result in an

incomplete or biased representation of the available evidence, and may lead to inaccurate or incomplete conclusions.

Another threat to validity is quality assessment bias. The quality assessment process may introduce bias if the criteria used to evaluate study quality are unclear, inconsistent, or subjective. This could affect the inclusion or exclusion of studies, as well as the weighting of studies in the analysis, which may affect the robustness and reliability of our findings.

In order to mitigate potential threats to validity, we employ a search strategy that includes multiple sources of information, such as traditional bibliographic databases, grey literature, and hand-searching of relevant journals and conference proceedings. We will also employ a snowballing technique to identify additional relevant studies by reviewing the reference lists of papers that have been identified through our search strategy. We recognize that grey literature can be particularly challenging to identify and assess, and to ensure that we capture the most relevant studies by focusing only on highly cited grey literature.

To mitigate the potential threat of quality assessment bias in this systematic review, we employed some strategies to ensure that the included studies are assessed for quality and risk of bias. In our revision, we conducted a blind quality assessment to minimize bias due to preconceptions or expectations. Then, we compare the results of the included studies with other relevant papers in the field. This can help to evaluate the quality and relevance of the contributions, avoiding inclusion of low-quality papers.

III. RELATED WORK

Previous works have provided a review of threats present in the FL environment. Lyu et al. [28] provided a brief and comprehensive overview of FL, along with a taxonomy that includes threat models and two major types of attacks, *i.e.*, poisoning attacks and inference attacks. Similarly, Liu et al. [25] provided a taxonomy of the threat of FL, followed by an analysis of the threats and possible defenses, along with a summary of their issues. The paper emphasized the importance of adequate measures to mitigate security and privacy threats at each phase to establish a trusted FL. Finally, Mothukuri et al. [29] aimed to bridge the security gap in FL and provided a study of FL security and privacy aspects, including diverse implementation styles, challenges, and potential risks.

In summary, while previous papers provided valuable insights into the threats and vulnerabilities of FL, they neglected to provide a security analysis of the current state of the art of FL applications and proposals. To fill this gap, we present an analysis of the state-of-the-art FL applications and assess their security and privacy implications based on the threats in the literature. In addition, our work provides a background of FL, and we present current FL challenges and applications that are missing in the mentioned works.

In addition to surveys addressing FL threats, surveys encompassing the concepts underlying FL are also considered related work, as we also discuss the FL background. Yang et al. [1] described the concepts of FL. The survey also proposed a categorization for FL, which includes horizontal, vertical, and transfer learning. Furthermore, the authors provided practical examples of how FL can be utilized by organizations. Zhang et al. [30] presented a review of the development of FL and introduced the existing work of FL from five different aspects: Data Partitioning, Privacy Mechanism, Machine Learning Model, Communication Architecture, and Systems Heterogeneity. Lim et al. [14] provided a survey of issues related to the implementation of FL for collaborative model training at MEC. It reviewed and analyzed the approaches to deal with emerging implementation challenges, such as communication cost, resource allocation, data privacy, and data security.

While the surveys mentioned above provided a brief overview of privacy and security concerns associated with FL, they do not delve into current attacks and defenses. ‘‘This constitutes a notable limitation of previous surveys, as several types of attacks can compromise the integrity and confidentiality of FL systems, including model poisoning attacks, inference attacks, and backdoor attacks. Additionally, the surveys do not thoroughly analyze the security and privacy aspects of current FL proposals and applications.

Our paper presents a comprehensive analysis of existing proposals on FL by focusing on current applications and challenges from a security perspective. We conduct a review of threats and vulnerabilities associated with FL and provide potential solutions, including proposals to address the discussed threats. We use the reviewed threats and vulnerabilities to analyze the current FL applications and proposals and emphasize the importance of addressing FL security aspects. This paper also provides a detailed overview of the fundamental concepts required to understand the covered topics, including FL background and the role of SMC and MEC in FL.

IV. COLLABORATIVE MACHINE LEARNING

Collaborative machine learning has emerged as an approach that enables distributed processing of a vast amount of available data. This data is necessary to train the ever more sophisticated machine learning models. However, in order to understand collaborative learning, we must first define centralized learning, as it serves as a basis for comparison and contrast to collaborative learning.

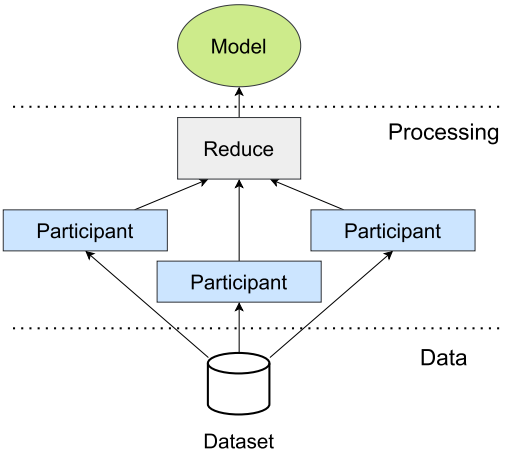
Centralized machine learning training is defined as a training approach for machine learning models where all data is collected and stored in a central location or server. Typically, the owner or an authorized entity collects the data and performs the model training. Furthermore, in a centralized approach, the model is trained on the entire dataset at once on a central server or cluster of servers. Such an approach is most useful when the training entity is the owner of the data or has permission to access it.

Often, the training in the centralized approach is performed by a cluster of servers in a data center to decrease the training

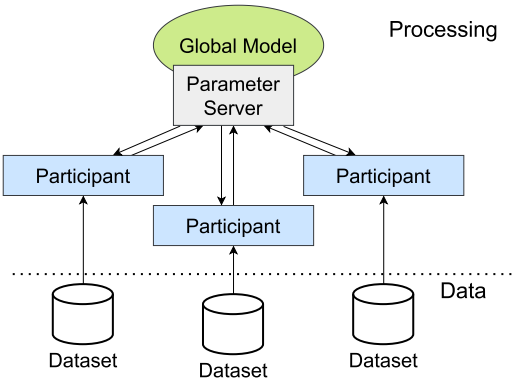
time. Each server in the cluster can be called by participants. The participant is a node within the cluster that contributes to the processing of the machine learning model. For training a machine learning model in a cluster, every participant has access to the entire dataset and is responsible for computing a portion of the model. After every participant computes their part of the machine learning model, a reduce function creates the model [4], [31], [32]. MapReduce, Hadoop [33], and Apache Spark [34] are well-known implementations of centralized machine learning with distributed processing [4]. Figure 1(b) shows centralized training on a cluster server architecture, in which the participants have full access to a single dataset shared between nodes. Unfortunately, this approach has a few downsides, including privacy concerns around sharing sensitive data with a central cluster of servers and the potential for the training process to become a bottleneck as the dataset grows. The centralized approach may be unfeasible in a scenarios where the central entity is now the data owner.

Decentralized training is a machine learning training paradigm in which each participant trains its model to contribute to developing a global model. Decentralized training has at least two entities, the participants and the parameter server [35]. The parameter server is responsible for managing tasks among participants. On the other hand, the participants are responsible for performing the tasks demanded by the parameter server using their local data [35]. The parameter server is fundamental to speeding up the training process and allocating computational resources through an interface to train the model efficiently. The parameter server is also responsible for combining the participant models into a single global model. Figure 1 shows that each participant has access to a local dataset, and the parameter server coordinates the participants. It is worth highlighting that the parameter server cannot access the participant data.

Training in the FL approach is decentralized. The parameter server is called the aggregator server, which has no control or access over the data stored on participants. Its function is to select participants and aggregate the updated parameters received by selected participants. The participant can refuse to participate or even lose connection during the training.



(a) In centralized training in a cluster, each participant computes a piece of the model in a data center. The reduce function can be performed by one of the participants.



(b) In decentralized training, each participant has a local dataset, but all nodes compute a model alongside the parameter server to create a global model.

FIGURE 1. Comparison of Centralized and Decentralized Training Architectures. (a) Centralized training in a cluster of servers. (b) Decentralized training architecture, where the training is performed on the participants devices. The blue rectangles represent the tasks performed on the participant side and the grey box on the server side. The green ellipse represents the machine learning model.

V. FEDERATED LEARNING: BACKGROUND

The FL system consists of two main entities: the participants, who own the data, and the aggregation server, which owns the global model. Let $N = \{1, \ldots, n\}$ be the set of participants. Each participant n has its own private dataset $D_n, n \in N$, and uses their dataset D_n to train a local model w_n^t at every aggregation round t. In each aggregate round, the aggregation server randomly selects a subset of the participants $S^t, S^t \in N$. Each selected participant sends only the local model parameters to the aggregation server. Then, the aggregation server aggregates all parameters from the selected participants to generate an updated global model w_G^t, where t is the current aggregation round. The participants update the Local models for τ local updates before sending the parameters to the server for global aggregation. After the global aggregation, the aggregation server sends the global model w_G^t to all federated participants. The participants update the global model w_G^t with their local dataset in the aggregation round $t + 1$. The aggregation round refers to a specific stage in the training process of the machine learning model where the participating devices send their locally computed model updates to a central server for aggregation. An underlying assumption is that participants are honest, i.e., they use their actual private data to train and send the proper parameters to the server, and they are not attempting to threaten the training.

The training consists of at least three basic steps in each aggregation round. The first step is Local Update, the second step is Participant Selection, and the last one is Global Aggregation. Before these three basic steps, the aggregation server first creates the initialized global model template w_G^0. This

initial model is set with random parameter values, usually, from a normal distribution, this process is called Initialization. The server also specifies the training hyperparameters, such as learning rate (η), local updates number (τ), and mini-batch size (B). The three basic steps of the training are described as follows:

1) **Local Update.** Based on the global model w_G^t received from the server, the selected participants use their local data and processing power to update the model parameters for τ local iterations, generating w_n^t. The goal of the participant n in the aggregation round t is to find the optimized parameters w_n^t that minimizes the local loss function $F_n(w_n^t)$.
2) **Participant Selection.** The server randomly selects the subset S^t,$S^t \in N$ of participants for training. Only the selected participants will send their parameters to the aggregation server.
3) **Global Aggregation.** The server aggregates the parameters of the selected participants and updates the global model w_G^{t+1} on every participant. The goal of the server is to minimize the global loss function $F(w_G^t)$. A global aggregation occurs each aggregation round.

The three steps are repeated until the convergence of the global loss function or the algorithm meets a stop condition.

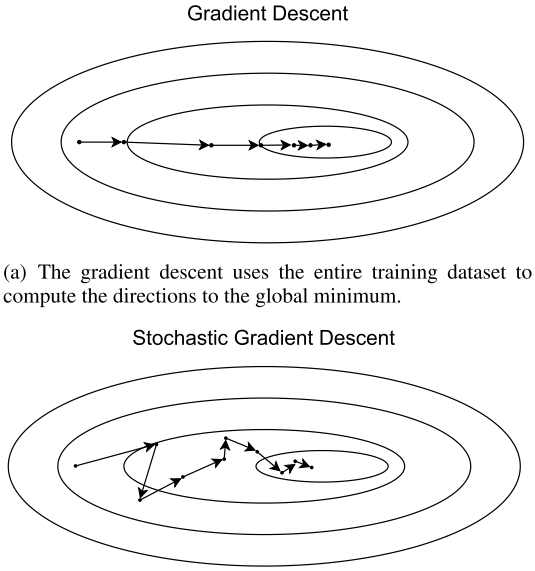
Machine learning models have a set of updated parameters based on training data. A training data sample j has two parts. The first part is the vector x_j, which are the features of the sample j that are the input to the machine learning model; the other part is y_j, which is the desired output of the model. Each model has a loss function for training the model. The loss function calculates the error of the predicted value based on each sample's desired value y_j. The training process consists of minimizing the loss function based on a training dataset. The loss function is different according to the problem. For example, the loss function of a regression problem usually is $F_n(w_n^t) = \frac{1}{2}\left(x_j^T w_n^t - y_j\right)^2$. Besides the local loss function $F_n(w_n^t)$, in FL, we have the global loss function, which is defined by Equation 1. The global loss function measures the loss of the global model considering all the selected participants, as follows:

$$F(w_G^t) = \sum_{n=1}^{N} \frac{|D_n|}{|D|} F_n(w_n^t), \tag{1}$$

where $|\cdot|$ denotes the cardinality of a set. Assuming that $D = \bigcup_{n=1}^{N} D_n$ and $D_n \cap D_{n'} = \emptyset \; \forall \; n \neq n'$.

Note that $F(w_G^t)$ cannot be calculated directly without sharing information among participants [36]. The pieces of information shared by the participants are the dataset size $|D_n|$ and the local loss $F_n(w_n^t)$. The optimization problem is then to minimize $F(w_G)$, *i.e.*, to find $w_G^* = \arg\min F(w_G)$.

It is important to highlight that the local and global loss functions are almost identical. It is because the global loss function $F(w_G)$ is the weighted sum of the local loss function of the participants $F_n(w)$, as demonstrated in Equation 1. FL



(a) The gradient descent uses the entire training dataset to compute the directions to the global minimum.

(b) SGD updates a small data shard, called mini-batch. SGD takes more steps toward the global minimum but is computationally cheap.

FIGURE 2. Comparison of Gradient Descent and Stochastic Gradient Descent. (a) Gradient Descent is a deterministic optimization algorithm that takes large steps to converge to a minimum. (b) Stochastic Gradient Descent is a variant of Gradient Descent that takes smaller steps but is less computationally expensive.

works with models based on Stochastic Gradient Descent (SGD) methods [15], such as neural networks, linear regression, and support vector machines. Gradient, in simple terms, means a surface slope direction. Therefore, the gradient descent is the direction to reach a surface minimum point. Hence, the main goal of gradient descent algorithm is to find the best parameters to minimize the loss function. Thereupon, the model parameters are updated using the partial derivative of the loss function with respect to the parameters of the model w_n^t for each sample in the entire dataset. However, this process causes overhead if the dataset has a large number of samples. SGD is a variant of the gradient descent that aims to minimize overhead. Figure 2 shows a difference between gradient descent and SGD. In SGD, the parameters are updated in mini-batches at each epoch instead of using the entire dataset samples. Mini-batch denotes the total number of data used to calculate the gradient at each iteration [37]. That is the reasoning for choosing a subset of participants at each aggregation round in FL instead of asking the local model of every participant.

In FL, the aggregation server cannot preprocess the dataset, as in centralized models. Besides, federated optimization properties are different from a typical distributed optimization problem, as follows:

1) **Non Independent and Identically Distributed (Non-IID) data.** Local datasets do not have the same probability distribution, and the samples are dependent.

The dependency is due to the context of the use of each participant;

2) **Unbalanced Data.** Some users have larger datasets with more samples than others. The local datasets can also be an imbalanced dataset that stands for a dataset with distinct class proportions;
3) **Large amount of participants.** The number of participants in a federated optimization is expected to be large. The FL algorithm must handle the massive amount of participants. For example, the smart keyboard of Google uses FL for next words prediction with millions of clients [38];
4) **Limited communication.** Mobile and IoT devices, typical of a FL environment, are often disconnected or have low throughput connections.

A. THE FEDERATED AVERAGING ALGORITHM

The first and most used aggregation algorithm for FL is Federated Averaging (FedAvg) [15]. FedAvg convergence has been empirically proven, particularly for problems where the loss function is non-convex [14]. However, FedAvg does not have convergence guarantees and may diverge in practical scenarios when data is heterogeneous [24].

Google researchers implemented FedAvg on Gboard [38]. The Gboard is a smart keyboard implemented by Google for next-word prediction. Since then, other studies have explored FL in a range of scenarios where data is sensitive, for example, developing predictive models for health diagnosis [19] to promote collaboration between hospitals [39] and Government agencies [40].

The FedAvg algorithm is SGD-based because SGD optimizes the parameter of the model based on a gradient vector $\nabla F_n(w_n^t)$ pointing to the best direction in which the model should evolve. It is simple to perform operations in the gradients of multiple participants. Another point is that deep learning models lean on SGD and variants method to compute parameter optimization [15].

For each aggregation round t, FedAvg algorithm randomly selects a subset of participants, $S^t \in N$, which performs the local update. Each participant computes $\nabla F_n(w_n^t)$, which are the gradients of their local data for the current model w_n^t, and the server aggregates these gradients by applying the update:

$$w_G^{t+1} \leftarrow w_G^t - \eta \sum_{n=1}^{N} \frac{D_n}{D} \nabla F_n(w_n^t). \tag{2}$$

The hyperparameter η is the learning rate and directly influences the convergence speed. A small learning rate implies a smooth trajectory and small weight changes at each iteration. A very high learning rate implies a more significant weight change, increasing convergence speed. However, it can also lead to fluctuations around a local minimum. An equivalent and commonly used aggregation type is:

$$w_n^{t+1} \leftarrow w_n^t - \eta \nabla F_n(w_n^t), \quad \forall n; \tag{3}$$

Algorithm 1 Federated Averaging pseudo-code [15].

Input: Local mini-batch size B, number of local updates τ, number of participants per aggregation round μ, learning rate η, number of aggregation rounds T
Output: Global model w_G

1 [*participant n - Update the local model*]
2 **Function** *LocalUpdate(n, w)*
3 Split the local dataset D_n into mini-batches of size B creating the set B_n
4 **for** *each local_epoch from 1 to* τ **do**
5 **for** *each* $b \in B_n$ **do**
6 $w_n^{t+1} \leftarrow w_n^t - \eta \nabla F_n(w_n^t; b)$
7 **end**
8 **end**
9 **return** w_n^t
10 *[Server side - Performs a global weighted average using the selected local parameters of the models]*
11 **INIT** w_G^0
12 **for** *each iteration t from 1 to* T **do**
13 Randomly selects a subset $S_n \in N$ of size m
14 **for** *each participant* $n \in S_n$ **do**
15 $w_n^{t+1} \leftarrow$ **LocalUpdate**(n, w_G^t)
16 **end**
17 $w_G^t = \sum_{n=1}^{N} \frac{D_n}{D} w_n^{t+1}$
18 **end**

and updated as:

$$w_G^{t+1} \leftarrow \sum_{n=1}^{N} \frac{D_n}{D} w_n^{t+1}. \tag{4}$$

Each client locally performs τ gradient descent steps on the current model using its local data, and the server then computes a weighted average of the resulting models. τ controls the local train epochs amount. Hence, three main parameters control the computation amount: i) S^t, the portion of participants that perform computation in each aggregation round (parallelization); ii) τ, the number of training iterations each participant performs on their local dataset; iii) and B, the local mini-batch size used for local updates. The pseudo-code for FedAvg is presented in Algorithm 1 [15].

The algorithm follows the FL three basic steps aforementioned. In step 1, the server starts the training (lines 11 - 16). Then, in step 2, participant n performs the local training and optimizes its loss function on the local dataset mini-batches (lines 2 - 9). In iteration t (line 17), the server minimizes the overall loss by aggregating the average gradients received from the participants. The FL training process will continue until the global loss function achieves a desirable loss, or reaches a maximum aggregation round number.

B. FEDERATED LEARNING REFERENCE ARCHITECTURES

There are three general architectures for a FL system: horizontal FL, vertical FL, and federated transfer learning [1]. We describe each architecture based on a matrix. The rows represent the sample space, and the columns the feature space.

1) HORIZONTAL FEDERATED LEARNING

Horizontal FL is the most commonly used FL architecture. The main characteristic of this architecture is that the n participants have the same data structure, *i.e.*, the dataset of participants have the same feature space, with a different sample space. An example of such a case is shown in [1], where the authors use horizontal FL to train on the datasets of customers from two regional supermarket chains, with a very small overlapping set of customers in between. In the horizontal architecture, participants collaboratively train a machine learning model with the guidance of a server. A conventional assumption is that the participants are honest, while the server is honest-but-curious, which means that the server maintains the functionality of the FL environment but is willing to discover participant data. Therefore, no information leakage from any participant to the server is allowed [41].

A well-known challenge in the horizontal FL architecture is to protect sensitive data from an honest-but-curious server [1]. Even sharing only the parameters of the model, it is still possible to infer data stored on participants. The most used techniques to protect the parameters of the model are homomorphic cryptography [41] and SMC [42].

2) VERTICAL FEDERATED LEARNING

Vertical FL or feature-based FL, applies to cases where two datasets share the same sample space but differ in the feature space. For example, two companies operating in the same city may have similar customers but have different information about those customers. Vertical FL is the process of aggregating these different features and computing the loss and gradients with privacy-preserving to build a model with data from both parties collaboratively [1].

Some privacy-preserving machine learning algorithms for vertically partitioned data have been proposed in the literature, including cooperative statistical analysis [43], association rule mining [44], secure linear regression [45], classification [46], and gradient descent [47].

Some recent works propose a federated vertical learning scheme to train a privacy-preserving logistic regression model [48], [49]. These works apply Taylor expansion to the loss function and adopt homomorphic cryptography for privacy-preservation gradient descent calculations.

Suppose companies A and B want to collaboratively train a machine learning model in their business systems, each with its data. Furthermore, Company B also has labeled data that the model needs to predict. For privacy and data security reasons, A and B cannot exchange data directly. A third entity C is involved to ensure data confidentiality during the training process [1]. Entity C is assumed to be honest and not collude with A or B, whereas parties A and B are honest-but-curious about each other. Trusting in entity C is a reasonable assumption, as part C can be performed by authorities such as governments or replaced by a secure compute node [1]. Vertical FL has two parts, as shown in Figure 3. In part 1, an alignment between entities is done using cryptography. Since the two companies have different customers, the system uses cryptography-based user Identification (ID) alignment techniques to confirm mutual users in A and B [50], [51]. The system does not use samples of users that do not overlap among the entities during the alignment of entities. Part 2 consists of the encrypted model training. After determining similar users, the model uses the data from those samples to train the machine learning model. We divide the training process into the following four steps [1].

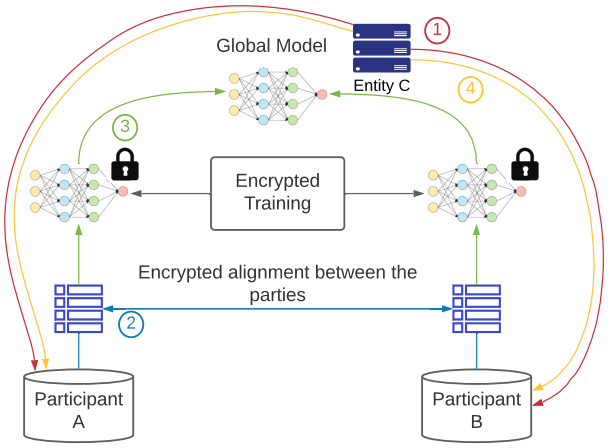


FIGURE 3. The vertical FL system architecture. The entity creates the key pairs in the first step and sends the public key to the participants. In step 2, A and B conduct the security verification to find intersections in their samples, *i.e.*, mutual customers. Participants send their parameters encrypted and masked for aggregation by entity C in step 3. Finally, in step 4, entity C returns the result to the participants.

1) Entity C creates homomorphic additive cryptographic pairs of keys and sends the public key to A and B, and encrypts masks for A and B to apply;
2) A and B encrypt and exchange their intermediate results using C's mask for gradient and loss calculations;
3) A and B calculate the encrypted gradients and add an extra mask. B also calculates the encrypted loss. A and B send encrypted values to C;
4) C decrypts and sends the decrypted gradients and loss back to A and B. A and B unmask the gradients and update the model parameters.

In short, the vertical FL system helps participants establish a "commonwealth" strategy without affecting data privacy.

A vertical FL system typically assumes honest-but-curious participants. For example, in a two-party case, both parties are not malicious; however, one of the parties may be committed to be an adversary. An adversary can only learn information

from the dishonest participant since the adversary is not allowed to participate. The third participant C, is introduced to facilitate safe processing between the two parties. In this case, it has an assumption that this third party is not in collusion with either party. Secure multiparty computing provides formal proof of privacy for these protocols [52]. At the end of learning, each party has only the model parameters associated with its features. Therefore, the two parties must collaborate to generate the output model at the inference time.

3) FEDERATED TRANSFER LEARNING

Federated transfer learning applies to scenarios where datasets differ in the samples and the feature space. An example is the case of two different institutions operating in different countries [1]. In this case, transfer learning techniques [53] provide solutions for the entire sample and feature space in a FL environment.

Suppose parties A and B have only a small set of overlapping samples. The main interest is to train a model to predict the labels for the participant A dataset (destination domain) using the participant B model (source domain) knowledge. The architecture described in vertical FL works only for the overlaid dataset samples. Federated transfer learning does not change the general architecture. However, it changes the intermediate results exchanged between parts A and B. Transfer learning usually involves exchanging common representation from one model to another to minimize the error of the destination domain party, using the knowledge of the source domain party (B, in this case). Therefore, the gradient calculations for parties A and B differ from the vertical FL scenario. Federated transfer learning is an extension of existing FL systems as it deals with problems that exceed the scope of existing FL algorithms.

VI. DATA SHARING AND PROCESSING IN FEDERATED LEARNING

Current FL applications use MEC and SMC techniques to ensure low latency and privacy. MEC brings the aggregation server near to the user, while SMC encrypts the parameters of the participants enabling arithmetic operations.

A. MULTIACCESS EDGE COMPUTING

Multiaccess Edge Computing (MEC) is a technology that enables computing resources to be deployed closer to end devices, such as mobile devices and IoT devices. By deploying computing resources closer to end devices, MEC can reduce the data transmission latency. It is particularly important for applications that require real-time data processing or low-latency communications, such as virtual reality, augmented reality, autonomous vehicles, etc. MEC also can support reducing the amount of data that needs to be transmitted over the Internet, which can help to lower the network traffic and reduce congestion. Therefore, MEC allows the model training close to the data sources [14], that is, in devices at the access network. Figure 4 shows the MEC traditional architecture.

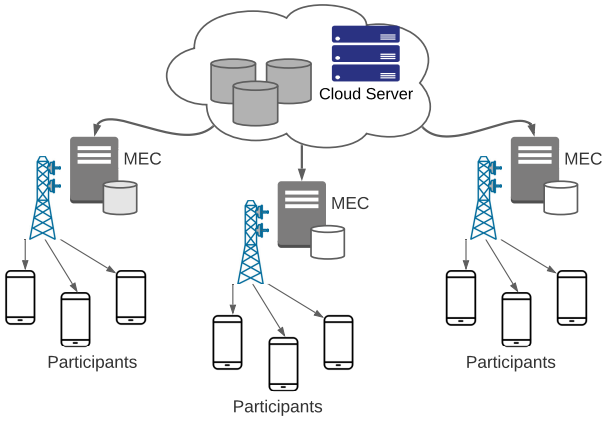


FIGURE 4. MEC architecture, in which servers are located in the structure of the carrier, thus offering high throughput and low latency.

A collaborative paradigm is widely used for training machine learning models in conventional MEC approaches [14]. The users send their data to the edge servers for the model training instead of sending it directly to the cloud servers, decreasing the communication cost. However, the paradigm still incurs high communication costs for applications that require constant training [54]. Furthermore, data processing on edge servers still involves transmitting potentially sensitive personal data to network edge servers [14]. The leakage possibility may discourage users concerned about their data privacy from participating. Data storage or usage may violate increasingly strict privacy-enforcement laws, such as GDPR. MEC applications are increasingly adopting FL to ensure that training data remains local on the device of participants. FL enables complex model training collaboratively among distributed devices without data sharing [54].

B. SECURE MULTIPARTY COMPUTATION

FL is closely related to SMC. SMC is a cryptographic protocol that distributes computations among multiple parties, with neither party able to access the data of the other. Secure multiparty computation protocols allow compatible, secure, and private distributed computation [55]. With SMC, multiple parties can collaboratively compute a common function without revealing their private inputs to other parties. For a secure SMC protocol, the parties may learn no other information but the final result. Before FL, previous works proposed algorithms for secure multiparty decision trees for vertically partitioned data [56], [57]. Vaidya and Clifton proposed secure association mining rules [44], secure k-means [58], and a naive Bayes classifier [59] for vertically partitioned data. Du et al. proposed secure protocols for linear regression and multiparty classification [46]. Wan et al. proposed secure multiparty gradient descent methods [47].

Several SMC approaches were proposed for FL, such as homomorphic encryption [60], pairwise masking [61], and

secret sharing [62]. In Homomorphic Encryption, mathematical operations can be performed on encrypted data. Therefore, the aggregation server cannot access the raw values of local models but can aggregate the models, generating an unencrypted global model [60]. Although homomorphic encryption ensures privacy, the scheme is computationally expensive. In pairwise masking, the participants agree with a pairwise mask. The pairwise mask can be exchanged via Diffie-Helman key exchange [61]. However, the approach is insufficient to provide reasonable privacy for the participants. In secret sharing, a secret is broken into multiple parts. Only when the parts are together can the secret be revealed [62]. The secret can be a mask, a key, or parameters. The main goal of using SMC is to protect the local models against a possible honest-but-curious aggregation server.

VII. VULNERABILITIES IN FEDERATED LEARNING

FL is susceptible to attacks against collaborative training. This section categorizes the main attacks on FL as model performance attacks and privacy attacks. The section discusses FL attacks and proposed countermeasures to address these attacks.

A. MODEL PERFORMANCE ATTACKS

This kind of attack aims to directly or indirectly affect the performance of the global model. Malicious participants can send incorrect or corrupted parameters to bias the global model during global aggregation. Attacks on the model performance can be targeted attacks (backdoor attacks), and untargeted attacks (byzantine attacks) [25]. The malicious participant can backdoor subtasks by sending poisoned models. A backdoor subtask makes the global model fail to predict a particular class, leading to mispredictions [22]. The Byzantine attack goal is to cause the collapse of the global model. Unlike backdoor attacks, byzantine attacks do not intend the misprediction on a specific task. Consequently, when attackers try to poison the training, the aggregation server will update the global model incorrectly, and the entire collaborative training will be compromised [26]. Besides, a participant may want to get the global model but is unwilling to contribute, sending random parameters to aggregation (free-riding). Even unintentionally, free-riding can be harmful to collaborative training.

1) DATA POISONING ATTACKS

The objective of the FL is to preserve the privacy of data stored on participants by training machine learning models locally and transmitting only the model parameters to a central server for aggregation. However, the server cannot guarantee that the participants have used actual data during their training process, rendering the system vulnerable to attacks by malicious participants [63]. By poisoning the global model, an attacker can introduce mislabeled data, which biases the training of the global model and produces falsified parameters [64]. Mislabeled data refers to instances in a dataset where the label assigned to a data point is incorrect, *i.e.*, the data is labeled with a category or class that does not accurately reflect its true identity or characteristics. One potential method of carrying out such an attack is to generate a series of counterfeit samples and incorporate them into the local model update, thereby impeding or sabotaging the convergence of the global model.

This attack only affects collaborative training if a group of participants colludes to poison the global model. The server randomly selects the participants for aggregation; then, a malicious participant has a $\frac{1}{N}$ chance of being selected. Even being selected once or twice, a single malicious participant cannot harm the training by poisoning its dataset [14]. The malicious participant may collude, infect other participant machines, or use a Sybil attack. The Sybil attack involves a malicious participant attempting to amplify the impact of data poisoning by generating multiple false identities of legitimate participants, all containing poisoned data. Only two false participants can collapse the entire training [64].

Similar to the Sybil attack, Distributed Backdoor Attack (DBA) is a threat assessment framework that exploits the distributed nature of FL to manipulate a subset of training data by injecting adversarial triggers in a distributed manner [65]. In DBA, a global trigger pattern is decomposed into different local patterns and embedded into the training set of different adversarial parties. A trigger pattern can be a specific image or a sequence of pixels that can manipulate the machine learning model's behavior when added to the training data. For example, a trigger pattern in an image classification task could be a small, specific shape or pattern (such as a red square) added to a corner of an image. When this trigger pattern is present in the image, the model will misclassify the image to a targeted class. DBA aims to create backdoors in the FL model that will make arbitrarily incorrect predictions on the test set with the same trigger embedded. Compared to standard centralized backdoors, DBA is more persistent and stealthy against FL on diverse datasets such as finance and image data [65].

Data poisoning attacks in FL can be executed through Generative Adversarial Networks (GANs) [66]. The attacker initially trains the GAN to replicate the training samples of other participants and subsequently leverages these replicated samples to generate poisoned updates. Such a poisoning attack is characterized by increased generality and efficacy, making it challenging to identify and mitigate [66].

Clean-label and dirty-label are the two main categories of data poisoning attacks [28]. In clean-label attacks, the adversary is assumed to be unable to modify the labels of training data due to a certification process that verifies the correct class of the data and requires imperceptible modifications to any data samples. Conversely, in dirty-label attacks, the adversary can deliberately mislabel a set of data samples with a desired target label, leading to the misclassification of future data when these samples are introduced into the training set.

Possible Solution: Robust aggregation and differential privacy are the most common defense against data poisoning attacks. Robust aggregation in FL refers to aggregating model

updates from multiple participants in a way resistant to various types of attacks or noise. Robust aggregation techniques are used to aggregate the updates robustly to such noise and attacks. Another solution to mitigate model poisoning attacks involves incorporating differential privacy mechanisms. This approach entails the introduction of noise to the model updates of each participant, thereby ensuring that the updates do not disclose any information about the participant private data. The differential privacy prevents attackers from generating poisoned updates, similar to the GAN poisoned dataset. Fung et al. introduced a robust aggregation method to identify and mitigate Sybil attacks [64]. The proposed defense strategy is called FoolsGold. The authors have discovered the differentiation of honest participants from malicious ones by analyzing their updated gradients. In FL, the training data of each participant has a unique distribution and is not shared. Sybil attackers share a common goal and will contribute updates that are similar to each other, unlike honest participants. FoolsGold uses this assumption to modify each local learning rate in each aggregation round to minimize the impact of malicious participants. The proposal is to maintain the learning rate of participants who provide unique updates while reducing the learning rate of customers who repeatedly contribute similar gradient updates.

Fang et al. proposed a defense strategy for detecting and mitigating poisoning datasets in FL [67]. The proposed defense is based on identifying malicious participants by analyzing their parameter updates, which have unique characteristics compared to honest participants. The defense strategy is designed to differentiate between malicious and honest participants using Principal Component Analysis (PCA) for dimensionality reduction and pattern visualization. After computing the gradients in a model update and compared to the global model, only the subset of models corresponding to the participants suspected to be the source of a poisoning attack is extracted. This subset is added to a global list. Then, the standardized list is fed into PCA, generating a two-dimensional data visualization. The results show that the defense can effectively identify malicious participants, even in scenarios with a few participants, and remains robust to gradient drift. Similarly, Tolpegin et al. propose a method for identifying malicious participants by comparing their updates and either blacklisting them or disregarding their updates in future aggregation rounds [68]. Finally, Sun et al. proposed to add Gaussian noise with small standard deviations to the aggregation to mitigate the backdoor attacks [69].

2) MODEL POISONING ATTACKS

In model poisoning attacks, the malicious participant attempts to manipulate the local model updates before sending them to the aggregation server to poison the global model directly [63]. The adversary aims to cause the global model to misclassify specific inputs with high confidence, which is achieved by manipulating the training process. The attacker can even scale up their parameters to prevail over the averaging, increasing their influence on the global model. Previous works [22], [70], [71], [72] have demonstrated that model poisoning attacks are significantly more effective than data poisoning attacks. Furthermore, these attacks can be executed with just a single attempt, making them a serious threat to the security and integrity of the global model [70].

Bagdasaryan et al. introduced a highly effective model poisoning attack for FL [71]. By sharing their poisoned model that contains a backdoor to bias the global model to misclassify, a malicious participant can compromise one or more participants using the proposed constrain-and-scale backdoor. This algorithm allows attackers to create a model with high accuracy in both the main task and the backdoor, enabling the global model to be manipulated without modifying the local dataset of the malicious participant. For instance, in a sentiment analysis task, a backdoor attack could bias the model to classify all reviews containing a particular keyword as positive, regardless of their actual sentiment. Bagdasaryan et al. showed that their approach is more effective than dataset poisoning attacks. Only eight participants are sufficient to compromise an entire FL environment with high accuracy in the malicious classification. However, the aggregation server can detect malicious participants by comparing the received models, as poisoned models will have large parameter values compared to other participants.

Zhou et al. proposed a deep model poisoning that can be stealthy among benign models [72]. The proposed attack trains a mini-batch for the main task and backdoor subtask, respectively. In this way, the poisoned models will have similar parameters to benign ones, making detection by model comparison complex. The authors reported that some neurons are more important for the main task, and others are more important for the backdoor sub-task. The authors found that calculating the second-order derivative makes it possible to find the neurons that considerably affect the loss function. Then, capturing the second-order derivative, the Hessian matrix can measure the distance and direction of the update [72]. Hence, the authors proposed to find the neurons important to the main task and use a regularization term to penalize SGD, avoiding updating those neurons.

Wang et al. proposed a new class of backdoor attack called edge-case backdoor [22]. The edge-case backdoor target underrepresented input data for misclassification. For example, in an image classification task, the malicious participants can label samples of people using a kilt[3] as ''airplane''. In an image dataset, it is relatively common to have ordinary people, but people wearing kilts are rare. The edge-case backdoor train the poisoned model using Projected Gradient Descent (PGD) to reduce the detection probability. Using PGD, model of the malicious participant does not differ much from the global one at every aggregation round. Finally, before sending the model to the aggregation server, the malicious participant scales its parameter by a scalar to cancel the contribution of the honest participants [71].

[3]Kilt is a traditional Scottish garment.

Possible Solution: The main defense for model poisoning attacks is robust aggregation [25]. There are two possible ways to achieve robust aggregation [25]. One approach involves evaluating the performance of local models using a validation dataset and avoiding those with significantly poor performance. The other approach involves comparing local models from each participant to detect any statistical differences with the updates made by other local workers. Typically, malicious participants have different goals than honest participants, leading to differences in their local models. Therefore, identifying statistically different local models can help prevent model poisoning attacks.

Andreina et al. proposed a robust aggregation called Backdoor detection via Feedback-based FL (BAFFLE) [73]. BAFFLE introduced a validation phase in the collaborative training. In this phase, the aggregation server sends the global model to a random participant set for validation using their local dataset. The participants in this set will vote if the aggregation is genuine. Based on the feedback of the participants, the aggregation server accepts or rejects the global model.

Shen et al. proposed a mechanism called AUROR for robust aggregation [74]. The authors observed that the parameters of most honest participants have a similar distribution. On the other hand, malicious participants present an anomalous distribution. AUROR uses k-means to cluster the updates of the participants at each aggregation round and discard outliers. Xie et al. proposed a general framework called Certifiably Robust Federated Learning (CRFL) [75]. CRFL clips and smooths the local parameters using parameter smoothing [76] before aggregation.

Pillutla et al. introduced a new approach called Robust Federated Aggregation (RFA) to make FL more robust in settings where some participants may send corrupted updates to the aggregation server [77]. RFA relies on a robust aggregation based on the geometric median of the parameters and preserves the privacy of participating devices through secure multi-party computation. The paper establishes RFA's convergence for least squares estimation of the global model. It provided experimental results with linear models and deep networks for three tasks in computer vision and natural language processing. RFA outperforms classical aggregation approaches in terms of robustness when the level of malicious participants is high and competitive in low corruption regimes. Similarly, Yin et al. presented a robust aggregation algorithm against model poisoning [78]. The focus is on achieving optimal statistical performance, and the authors analyzed two robust distributed gradient descent algorithms based on median and trimmed mean operations. Their median-based aggregation algorithms also improved communication because they required fewer aggregation rounds for convergence.

Mhamdi et al. introduced a new approach called Bulyan, which reduces the space for adversarial attacks and achieves convergence as if only non-Byzantine gradients had been used to update the model [79]. Bulyan combined Krum and a variant of the trimmed mean. Krum is a distributed algorithm that stands for ''K-reverse nearest neighbor Using Minimization''. The Krum algorithm operates by collecting model updates from a subset of participants and then selecting the updates that are closest to the median using the Euclidean distance as a metric for comparison. Specifically, Krum chooses the K updates farthest away from the other updates, then averages them to produce the final aggregated update. Bulyan is shown to avoid convergence to ineffectual models and achieves comparable performance to a non-attacked averaging scheme. However, the authors acknowledge that finding the best direction for non-convex loss functions remains a challenging problem.

3) FREE-RIDING ATTACKS

In FL, free-riding is a deceptive attack where a participant tries to exploit the benefits of the global model without investing sufficient resources in the training process. Essentially, a free-rider selects a smaller subset of their actual dataset for training or uses random noise instead, conserving their computational resources. This behavior results in the honest participants having to contribute more resources to the global model training process. Consequently, the overall model performance gets compromised by the poor quality of data provided by the free-rider.

Possible Solution: Various solutions have been proposed to mitigate the issue of free-riding in FL. One commonly suggested solution is to utilize blockchain technology to track participant updates and ensure their contributions. However, this approach can lead to potential data privacy attacks. An alternative approach is incentivizing participants to contribute by implementing reward mechanisms that benefit participants with contributions and penalizing those who do not.

Kim et al. proposed BlockFL, a FL architecture that leverages blockchain technology to exchange and verify local model updates, thereby addressing the problem of free-riding in FL. [80]. Each participant trains and sends the trained local model to its associated miner in the blockchain and then receives a reward proportional to the number of samples of trained data. Then, the proposed framework avoids free-riding participants and encourages all participants to contribute to the learning process. A similar model, also based on blockchain, is introduced by Weng et al., aiming to provide data confidentiality, computational auditability, and incentives for participants in FL [81]. However, using blockchain technology implies the implementation and maintenance of miners to operate the blockchain. Furthermore, consensus protocols used in blockchain networks, such as Proof-of-Work (PoW), tend to cause long delays in information exchange and, thus, are not appropriate for implementing dynamic FL models.

Another approach to avoid free-riding is using incentive mechanisms for participants contributing to the collaborative training [82]. The incentive mechanism compensates for the effort of a contributing participant. Richardson et al. define the influence of the participant as the effect of its contribution

TABLE 1. Proposals in literature against model performance attacks. None of the proposals protect against poisoning and free-riding simultaneously.

Defense proposal	Targeted (Backdoor)	Untargeted (Byzantine)	Free-riding	Main goal	Weaknesses
FoolsGold [64]	✓	×	×	Detect malicious participants by comparing the local parameters of the model.	Rely on malicious participants using the learning rate provided by the aggregator.
Fang *et al.* [67]	✓	×	×	Analyzed the participants parameter and block participants with divergent updates.	Increases computation on server side and is prone to false and positive and negative.
Tolpegin *et al.* [68]	✓	×	×	Analyzed the participants parameter and block participants with divergent updates.	Increases computation on server side and is prone to false and positive and negative.
Sun *et al.* [69]	✓	✓	×	Uses norm-bounding and differential privacy to avoid backdoor.	Affect the model performance.
BAFFLE [68]	✓	×	×	Introduce a validation phase where participants use their private data to vote if an aggregation is genuine.	Increases computation and communication.
AUROR [74]	×	✓	×	Clusters participants by the distribution of their gradients.	Affect the model performance, and introduce server side computation.
RFA [77]	×	✓	×	Aggregation based on geometric median. The proposal also support privacy protection with SMC.	Affect the model performance, and introduce server side computation.
Yin *et al.* [78]	×	✓	×	Using median and trimmed mean operations for improved communication and optimal statistical performance.	Affect the model performance, and fail to detect backdoor attacks.
Bulyan [79]	×	✓	×	Combining Krum and a variant of the trimmed mean to reduce adversarial attacks.	Affect the model performance, and fail to detect backdoor attacks.
CRFL [75]	✓	×	×	Uses clipping and smoothing techniques to avoid backdoor.	Affect the model performance, and fail to detect byzantine attacks.
BlockFL [80]	×	×	✓	Uses blockchain technology to store the local models for posterior free-riders detection.	Affect privacy of the participants.
Weng *et al.* [81]	×	×	✓	Uses blockchain technology to store local models for posterior free-riders detection.	Affect privacy of the participants.
Richardson *et al.* [82]	×	×	✓	Measures the contribution by the loss reduction and monetizes the the participants according to their contribution.	Depends on sharing profits with participants.
Huang *et al.* [83]	×	×	✓	Incentive framework based on game theory, which monetizes the global model and allocates profits to each participants.	Depends on sharing profits with participants.

on the loss function of the FL model. The participants receive incentives according to their influence. The total reduction in the loss function bounds the expenses. The aggregation server has to obtain the required budget for the rewards. Huang et al. model their incentive framework as a game theory [83]. The idea is to monetize the global model and allocate the profits to each participant according to their contributions.

Table 1 summarizes the state-of-the-art defenses against model performance attacks. It is important to highlight that none of the proposals simultaneously protect collaborative training against poisoning and free-riding.

B. DATA PRIVACY ATTACKS

One of the main goals of FL is to protect the privacy of the participant in collaborative training. However, data privacy attacks can infer the data stored on participants. Any entity possessing local models can infer their data [21]. The aggregation server is the most likely entity to perform such an attack. In particular, this threat is against the FL privacy assumption because the data stored on participants may be leaked.

1) MODEL INVERSION AND GRADIENT INFERENCE

Model inversion is the attack in which an adversary possessing a trained model uses its parameters to predict the dataset used as input to train that model, thus characterizing an attack on the privacy of a participant [21]. The attacker seeks to take advantage of the correlation between the target, which would be the unknown features, and the result predicted by the model. This attack can be performed by the aggregation server that has the updated local models of the participants. The model inversion is harmful to blockchain-based proposals because the models are stored in clear text on the chain. Every blockchain client has a copy of the chain and can be able to perform model inversion to reveal data stored on participants. Fredrikson et al. proposed the model inversion

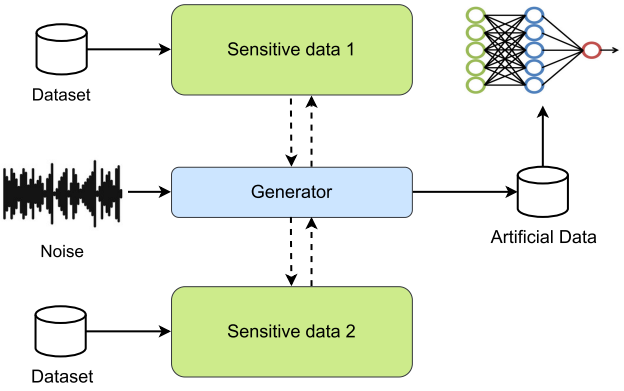


FIGURE 5. FedGP architecture for two participants. Sensitive data is used to train a GAN, producing an artificial private dataset. The green rectangle represents the data of each participant, and the blue one represents the generator model. Adapted from [85].

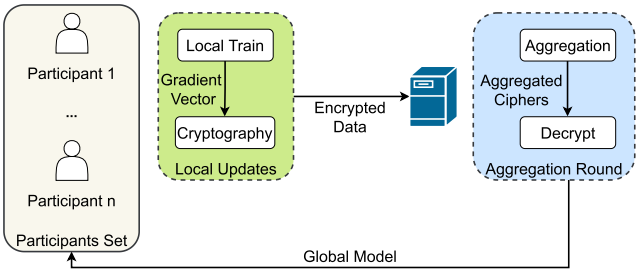


FIGURE 6. In the PEFL Architecture, the participants send the data encrypted using homomorphic encryption, and the server performs the aggregation returning the aggregated weights to all the participants. Adapted from [60].

attack to retrieve images from a facial recognition model [84]. The authors developed a new class of model inversion attacks that exploits the training parameters revealed with the predictions.

Possible Solution: The main solution for model inversion is using cryptography, differential privacy, and the generation of artificial data with Generative Adversarial Networks (GAN). Triastcyn and Faltings proposed a framework called Federated Generative Privacy (FedGP) [85]. The main idea of this approach is to train GANs on data stored on participants to produce an artificial dataset to replace the actual participants dataset. As some participants may have insufficient data to train a GAN locally, the authors proposed a federated GAN model. Thus, user data always remains on their devices. In addition, federated GAN generates a dataset following all data stored on participants distribution rather than a single one, which increases privacy. Figure 5 shows the architecture of the framework. The authors evaluated the protection by running the model inversion attack and showed that federated GAN reduces information leakage.

Zhang et al. proposed a homomorphic encryption scheme to preserve the local model parameters of FL [60]. The authors proposed Privacy-Enhanced Federated Learning (PEFL) to protect gradients from an untrusted server. PEFL enhances privacy by encrypting the gradients of local models with Paillier homomorphic cryptosystem. The proposal uses Distributed Selective Stochastic Gradient Descent (DSSGD) [86] algorithm in the local update to reduce the computational costs of the cryptographic system. In addition, encrypted gradients are used for server-side secure sum aggregation, as shown in Figure 6. In this way, the untrusted server only learns the aggregated statistics of the updates, keeping local data protected. The authors theoretically prove that the scheme is secure. Evaluations demonstrate that PEFL has low computational costs and, at the same time, achieves a high-accuracy global model. Zhang et al. also analyze the time to encrypt the parameters of a fragment of the weights, using DSSGD, and conclude that it is short, sometimes less than one second [60]. However, the authors use state-of-the-art equipment in the evaluations, which does not correspond to the reality of the IoT and mobile environments.

Titcombe et al. proposed a mechanism called NoPeekNN to protect the participants against model inversion by adding noise to the intermediate data representation [87]. The authors used additive Laplacian noise to obscure the private data stored on participants. The noise increases privacy but decreases the performance of the model. On the other hand, Qi et al. proposed a privacy-preserving method for FL training using differential privacy [88]. The authors used Local Differential Privacy (LDP) for the model gradients before uploading them to the server to protect privacy. LDP is a privacy-preserving technique that adds random noise to individual data points before releasing them or computing aggregate statistics. This way, it can protect the privacy of individuals in a dataset by making it difficult to link their specific data points to their identities.

2) GAN RECONSTRUCTION ATTACK

The GAN reconstruction attack is a class of FL privacy attacks that is even more effective than model inversion attacks [91]. Model inversion attacks struggle to infer data stored on participants when the deep learning structure is more complex. Hitaj et al. introduced the GAN reconstruction attack and showed that a malicious participant could reconstruct the data of the participants [91]. In this attack, the malicious participant creates a replica of the global model to be the discriminator and then trains a generator to create replicas of the data stored on participants. The malicious participant inputs data produced by the generator into the discriminator, calculate the loss of the discriminator outputs and then updates the generator. The malicious participant could infer the data stored in the participants even by applying moderate differential privacy. It is essential to highlight that the more differential privacy is used, the lower the accuracy is.

Possible Solution: Secure multiparty computation or mechanisms for malicious participant detection are the primary defense for FL from GAN reconstruction. Chen et al.

TABLE 2. Proposals against data privacy attacks. Currently, there is no definitive solution for data privacy in FL. The more effective solution requires high computation and communication costs.

Defense proposal	Honest-but-curious	GAN Reconstruction	Main goal	Weaknesses
FedGP [85]	✓	×	Use GAN to generate artificial data for training	Possibility of GAN reconstruction by both server and participants.
PEFL [60]	✓	×	Use homomorphic cryptography to protect the parameters of local models.	Increase of computational cost.
NoPeekNN [87]	✓	×	Use of additive Laplacian noise to obscure the participants data.	Decrease of the global model accuracy.
Qi *et al.* [88]	✓	×	Use of differential privacy to improve privacy.	Decrease of the global model accuracy.
Chen *et al.* [89]	✓	✓	Use of multiparty computation and change on the aggregation protocol providing no access to the global model among participants while training.	Increase of computational and communication costs.
Yan *et al.* [90]	×	✓	Use of additional hidden layer to identify honest participants	Fragile structure, the malicious participant can discover the buried point layer.

proposed a mechanism to protect collaborative training against GAN reconstruction attacks using secure multiparty computation [89]. For this sake, the authors used an improved Du-Atllah scheme, a method for multiple parties to perform calculations without knowledge of the raw data [92]. To achieve privacy, each party adds a mask to their data in a way that the other party mask can cancel the mask during the calculation. The proposed mechanism needs a Trusted Third Party to generate the masks. Since participants do not have full access to the global model during training, the aggregation server must communicate with participants several times at each local update to assist the local training. Thus, neither the participants nor the aggregation server can access the data [89]. The advantage is that the data is protected against the honest-but-curious server and the other malicious participants. The downside is the increased communication cost and the need for a new trusted party.

Yan et al. protected the collaborative training against GAN attacks by changing the parameter exchanging protocol [90]. In the proposal, every honest participant embeds an additional layer in the local model called the buried point layer. During the training, when a malicious participant sends its model for aggregation, the aggregation server can identify it due to the absence of the buried point layer. When a malicious participant is detected, communication with the participant is blocked.

Table 2 summarizes the main defenses for data privacy attacks. Data privacy is one of the main concerns in FL and is still an open issue. The current solutions do not defend against all known privacy attacks and increase computational and communication costs.

VIII. RESEARCH CHALLENGES AND OPPORTUNITIES IN FEDERATED LEARNING

The FL challenges differ from other classical problems, such as distributed learning in data center or traditional private data analytics. The main three challenges in FL are:

1) Expensive communication: The FL environment comprises a large number of devices, *e.g.*, millions of smartphones, and network communication can be slower than local computing due to limited resources such as bandwidth and energy [1]. Two main aspects must be considered to reduce communication in the federated environment: i) reduce the total number of communication rounds and ii) decrease the size of data transmitted in each aggregation round;
2) Device heterogeneity: Storage, computing, and communication capabilities of each device in the federated environment may differ due to variability in hardware (CPU and memory), network connectivity (2G, 3G, 4G, 5G, and Wi-Fi), and power (battery level) [1]. Furthermore, each network size and system-related restrictions typically result in only a small fraction of devices being active at the same time [93]. It is common for an active device to fail in a given aggregation round due to connectivity or power constraints [93]. Device heterogeneity intensifies challenges such as stragglers mitigation and fault tolerance. The FL methods must, therefore, predict participant failure, tolerate heterogeneous hardware, and be robust enough to enable that participant failure does not affect aggregation;
3) Federated Optimization: Devices often generate and collect data in a Non-IID manner in the FL environment. Smartphone users, for example, have varied language usage in the context of a next-word prediction task. Also, the number of data points (feature space — x) between devices can vary significantly, and there may be an underlying statistical structure that captures the relationship between devices and their associated distribution [1]. This data generation paradigm violates the Independent and Identically Distributed (IID) data assumptions often used in distributed optimization and can add complexity to problem modeling, theoretical analysis, and empirical evaluation of solutions.

TABLE 3. Security and privacy summary of recently FL proposals to main challenges.

	Proposal	Poisoning attack	Model inversion	GAN Reconstruction	Privacy-preserving
Expensive Communication	Liu *et al.* [94]	×	×	×	Not guaranteed
	Yao *et al.* [95]	×	×	×	Not guaranteed
	Wang *et al.* [36]	×	×	×	Not guaranteed
	Konevcny *et al.* [96]	×	×	×	Not guaranteed
	Caldas *et al.* [97]	×	×	×	Not guaranteed
	Tao *et al.* [98]	×	×	×	Not guaranteed
	Wang *et al.* [99]	×	×	×	Not guaranteed
Device Heterogeneity	Nishio *et al.* [13]	×	×	×	Not guaranteed
	Kang *et al.* [100]	×	×	×	Not guaranteed
	Sprague *et al.* [101]	×	×	×	Not guaranteed
	Xie *et al.* [102]	×	×	×	Not guaranteed
Federated Optimization	Kim *et al.* [103]	×	×	×	Not guaranteed
	Smith *et al.* [104]	×	×	×	Not guaranteed
	Corinzia *et al.* [105]	×	×	×	Not guaranteed
	Li *et al.* [106]	×	×	×	Not guaranteed
	Huang *et al.* [107]	×	×	×	Not guaranteed

A. EXPENSIVE COMMUNICATION

Effective communication is crucial to achieving the desired accuracy in FL. Complex deep learning model training, such as in Convolutional Neural Network (CNN), can comprise millions of parameters in each update [108], resulting in expensive communication and potentially impacting training. The bottleneck is aggravated by network conditions of participating devices and asymmetries in Internet connections, where the transmitting rate is lower than the receiving rate, leading to delays [96], [99]. Therefore, works in the literature [36], [95], [96], [97], [98], [109] aim to improve the communication of FL in three ways: i) increasing the local computation, thus reducing the need for communication rounds; ii) performing the compression of the local model, reducing the size of the data sent to the server; and iii) performing updates based on importance, in which only the parameters that have relevant changes during the local training are sent. Liu et al. proposed the aggregation algorithm Federated Stochastic Block Coordinate Descent (FedBCD), in which each participating device performs several local updates before communicating for global aggregation [94]. Convergence assurance is provided with an approximate calibration of the optimal number of local updates calculated at each aggregation round. Similarly, Yao et al. proposed augmented computing on each participating device, adopting a two-stream model [95]. The two-stream model is commonly used in domain transfer and adaptation learning [110]. During each aggregation round, the global model is received by the participants and set as a reference in the local training process. During the local update, the participant learns from local data and other participants by using the global model as a reference. The authors incorporate the Maximum Mean Discrepancy (MMD) into the local loss function to use the global model as a reference during local updates. MMD is a measure of the difference between two probability distributions. The authors used MMD to measure the difference between the local model from the participants and the global model. Then, the proposal aims to minimize the MMD. The authors used the Canadian Institute For Advanced Research - 10 Classes (CIFAR-10) and MNIST datasets and deep learning models such as CNN in the simulations. The results show that the proposed two-stream FL can converge in fewer aggregation rounds, even when the data is Non-IID [95]. In both papers [94], [95], no mechanism ensures that the participants will follow the protocols. Malicious participants can add backdoors or try to affect the performance of global model. In addition, free-riders can also fake their effort in training. A possible solution is to compare the discrepancy in the parameters of the local models.

Wang et al. proposed an algorithm to determine the global aggregation frequency to use the available resources more efficiently [36]. For this, the authors analyze the convergence limit of FL based on gradient descent from a theoretical perspective. A new convergence limit is proposed, incorporating the distribution of Non-IID data between nodes and an arbitrary number of local updates between two global aggregations. The authors proposed a control algorithm that learns data distribution, system dynamics, and model characteristics using this theoretical convergence limit. Based on the proposed algorithm, the system dynamically adapts the frequency of global aggregation in real time to minimize loss. Finally, the authors evaluate the performance of the proposed control algorithm through experiments using real datasets, both in a scenario with a hardware prototype and in a simulated environment. The results confirm that the proposed approach provides near-optimal performance for different data distributions, various machine learning models, and system configurations with different numbers of edge nodes. The system obtains a desirable trade-off between local update and global aggregation to minimize the loss function. From a security perspective, the participants can lie about their data distribution, forcing more aggregation rounds on fake or mislabeled data.

While local update methods can reduce the total number of communication rounds, model compression schemes can also be used to reduce the volume of data transmission in

FL. Examples of compression schemes include ''sparsing'', subsampling, and quantization, which significantly reduce the size of the messages communicated in each aggregation round. These methods have been extensively studied in the literature for distributed training in data center environments, both empirically and theoretically. Konevcny et al. [96] introduced the structured update and sketched update methods for updating the local model. The purpose of these models is to decrease the amount of information sent from participants to the aggregation server during each aggregation round. The structured update method imposes a predefined structure on participant updates: a low-rank matrix and random mask. Each update must be the product of two matrices in the low-rank matrix structure in this approach. One of the matrices remains constant during each round, while the other is optimized. The random array is compressed as a seed, and only the optimized array needs to be sent to the server. Alternatively, the random mask structure requires each local update to be a sparse matrix following a random, predefined pattern generated independently during each round. Participants must send only non-zero entries to the server. Sketch updates encode the update in a compressed form before communicating with the server, which decodes the updates before aggregation. The server then averages the sub-sampled updates to obtain an unbiased estimate of the true average. A structured random rotation is applied before quantization to reduce quantization error, which is the product of a Walsh-Hadamard matrix and a binary diagonal matrix [111]. Caldas et al. extended Konevcny et al. approach by proposing a loss compression to reduce communication costs [97]. From a security perspective, compressing the updates can make it difficult for the aggregation server to find an infected model because compressing mechanism force the models to have a similar structure. Then, compressing the model may facilitate malicious participants to deceive a backdoor detection mechanism.

Another way to reduce the number of bytes transmitted in FL is called importance-based updating. This technique assumes that most parameters of a deep neural network model are sparsely distributed and their values near to zero [112]. Thus, Tao et al. proposed the edge Stochastic Gradient Descent (eSGD) algorithm, which sends only a small fraction of important gradient to the aggregation server [98]. The eSGD algorithm tracks the loss values in two consecutive aggregation rounds. If the loss value of the current aggregation round is less than the previous one, it implies that the current training gradients and parameters are important for minimizing the training loss. Thus their respective hidden weights are assigned a positive value.

Thus, Wang et al. introduced the Communication-Mitigated Federated Learning (CMFL) algorithm, which selectively transmits relevant local model updates to the aggregation server to reduce communication costs and ensure global convergence [99]. The local model update is first compared with the global model in each aggregation round to determine its relevance. The simulation results showed that CMFL requires fewer aggregation rounds to achieve 80% accuracy for image classification using the MNIST dataset and for next-word prediction, both compared to the FedAvg algorithm, which is usually used as a benchmark. In this case, free-riders can forge their parameter in such a way that the aggregation server will never select them. On the other hand, malicious participants can forge slight contributions on both main and backdoor tasks.

B. DEVICE HETEROGENEITY

In the FL environment, there is significant variation in the device characteristics of network participants, as these devices may have different hardware, network connectivity, and battery levels. These system characteristics make issues such as delays significantly more prevalent than in typical data center environments. To solve this problem, several solutions have been proposed.

Typically, the participant selection in FL is random. FL training progress is limited by the training time of the slowest participating devices [14] *i.e.* stragglers. Therefore, participant selection protocols are investigated to solve the training bottleneck in FL. Kang et al. considers system overheads incurred by each device when designing incentive mechanisms to encourage devices with high-quality data to participate in the training process [100]. While these methods primarily focus on the variability of systems to perform live sampling, it is advantageous to consider live sampling a set of small but sufficiently representative devices based on the statistical data structure. From a security perspective, the malicious participant can forge the results to receive the incentive and be selected using their poisoned or mislabeled data. Likewise, Nishio et al. proposed a new FL protocol called Federated Learning with Client Selection (FedCS) [13]. The proposed framework is a MEC-based solution where the aggregation server coordinates the training process across a cellular network comprising mobile devices with varying resources. The server initiates the process by requesting resource-related information from a randomly selected subset of participants, such as CPU information and wireless connection. With the gathered data, the aggregation server selects the maximum number of participants to complete the training within a pre-determined period for the upcoming aggregation round. The simulation results showed that, compared to the FedAvg protocol, the FedCS is more accurate, as it involves more participants in each aggregation round instead of a fixed number. In this case, the malicious participants can lie about their pieces of information to be always selected or never be selected in the case of a free-rider.

Traditional data center configurations are based on synchronous and asynchronous schemes. In the synchronous scheme, participants wait for each other to sync — in the asynchronous scheme, participants run independently without synchronization [24]. Synchronous schemes are simple and guarantee a trivial equivalent serial computational model, but they are also more susceptible to delays due to device variability. Asynchronous schemes are approaches used to

mitigate stragglers in heterogeneous environments, particularly in shared memory systems. However, they rely on bounded delay assumptions to control the degree of staleness. The FedAvg [15] algorithm synchronously aggregates parameters. It is, therefore, susceptible to the lag effect as each training round progresses at the speed of the slowest device as the server waits for all selected devices to complete their local training before global aggregation. The training will be affected if various participants face network problems or a Distributed Denial of Service (DDoS) attack.

Sprague et al. have empirically found that an asynchronous approach is robust for participants joining during the aggregation round in progress, as well as when the federation involves participating devices with heterogeneous processing capabilities [101]. Nevertheless, training models on Non-IID data often results in a significant slowdown in the global model convergence. Similarly, Xie et al. proposed the FedAsync algorithm, where each newly received local update is adaptively weighted according to its degree of obsolescence, which is defined as the difference between the current aggregation round and the aggregation round to which the received update belongs [102]. Thus, an obsolete update from a straggler is still considered but receives less importance. Furthermore, the authors prove the convergence guarantee for a restricted group of non-convex problems. However, it FedAsync accepts every update from honest and malicious participants. FedAsync has no mechanism to identify a malicious participant, and every participant is considered honest. Despite the potential benefits of asynchronous FL, synchronous methods remain more prevalent due to their relative immaturity and lack of reliability.

C. FEDERATED OPTIMIZATION

Distributed machine learning and FL are both approaches for performing machine learning on data that is distributed across multiple devices. However, there is a key difference between these two approaches. In distributed machine learning, a central server can access the entire training dataset and can subsample it into smaller subsets with similar distributions. The central server can then forward these subsets to participating nodes for distributed training. There are frameworks such as Apache Spark and Apache Hadoop that are used for such a purpose. On the other hand, FL is a method where the central server does not have access to the data. In the latter, the training data remains on the devices that generated it and only the model parameters are shared among the devices.

Previous works on machine learning aim to model statistical heterogeneity using meta-learning and multitasking learning methods. Meta-learning consists of machine learning algorithms applied to metadata [113]. Multitasking learning is a transfer of learning approach that improves generalization by using the information in the training parameters of related tasks as an inductive bias [114]. These ideas were only recently extended to the FL environment [24].

Kim et al. proposed FL Acceleration with Momentum (FedAGM) to use global momentum to maintain knowledge from previous aggregations [103]. The global momentum prevents performance instabilities during collaborative training, reducing the gap between the local and global objective functions. The FedAGM algorithm aims to send the global momentum to the selected participants, enabling the incorporation of global momentum into the local updates at each participant.

Smith et al. proposed MOCHA, an optimization framework for FL that enables the development of customized models for each device while utilizing a shared representation via multitasking learning [104]. MOCHA is calibrated according to the resources of participating devices, such as network conditions and processing load. Although multitasking learning effectively captures internal relationships in local models to improve performance and increase the effective sample size for each node, it has limitations in scalability to massive networks, and is restricted to convex problems [104].

Corinzia et al. proposed an algorithm for federated multitask learning, which extends the FL paradigm to handle real-world federated datasets that show statistical heterogeneity among devices. The algorithm is designed to work with general non-convex models. It uses approximated variational inference to perform learning on the federated network, treating it as a star-shaped Bayesian network. The aggregation server aggregates the model parameters received from each participant and uses them to compute a posterior distribution over the shared model parameters. Variational inference is used to approximate this posterior distribution.

When modeling data in the FL environment, it is important to consider metrics beyond accuracy. In particular, naively solving a global loss function may imply taking advantage and/or disadvantage of some devices, as the learned model may become biased towards devices with larger amounts of data [24]. Li et al. introduced FedProx, a framework for addressing heterogeneity in FL. FedProx is a generalization and re-parametrization of FedAvg [106]. FedProx provides convergence guarantees for learning over non-identically distributed data while allowing each participating device to perform variable amounts of work. Additionally, it modifies the global loss function to ensure convergence [106]. Huang et al. proposed a federated machine learning method called Loss-based AdaBoost (LoAdaBoost) for Non-IID scenarios [107]. The method adapts to different data distributions in diverse sources and achieves higher predictive accuracy with lower computational complexity than the baseline method. When the training reaches a predefined number of aggregation rounds, the cross-entropy loss for each client is calculated and compared with the median loss from the previous aggregation round. Participants with a loss greater than the median are considered ''poorly fitted'', those with a loss less than the median are considered ''well-fitted'', and those with a loss equal to the median are considered ''neutral''. Poorly-fitted clients are trained for more aggregation rounds, well-fitted clients are trained for fewer, and neutral clients are trained

for the same number of aggregation rounds as in the previous round.

It is out of the scope of the aforementioned works to deal with malicious participants. However, dealing with model optimization can harden the malicious participant work since the entire update is not aggregated. It is important to highlight that mechanism to detect malicious participants is crucial on FL environments. Similarly to the corporative environment that must require security mechanisms to keep their business secure, the FL environments require security to ensure the safety of the training. As shown in Table 3, none of the proposals guarantee privacy nor mechanism against poisoning.

IX. MAIN APPLICATIONS

FL is a technology that is used for various applications such as vehicular networks [18], cyberattack detection [115], Edge Caching and Computation Offloading [36], base station association [17], vehicular networks [116], and smart healthcare [117]. This section describes FL applications in IoT and summarizes their vulnerabilities. Table 4 presents a security and privacy summary of the FL applications. We observe that most applications use the aggregation algorithm FedAvg, which is prone to poisoning and data leakage [25]. In such a case, considering the security and privacy issues in designing FL applications is crucial.

A. CYBERATTACK DETECTION

Detecting cyberattacks is essential in preventing and mitigating the consequences of attacks on networks. Several works have proposed using machine learning for Intrusion Detection Systems (Intrusion Detection System (IDS)) [16], [115], [118]. Deep learning is considered one of the most efficacious approaches in the cyberattack detection domain. Deep learning can accurately identify various attacks, thus establishing it as the preeminent tool for network attack detection. [115]. However, the IDS model accuracy depends on the dataset used. Deep learning models demand a relatively large dataset for training. Therefore, current cyberattack detection proposals use FL regarding the privacy issue [118]. In this scenario, we have n IDSes protecting their network. Every intrusion detection system is a participant in FL. The dataset consists of network traffic generated by the network monitored by the IDS.

In previous work, Cunha Neto et al. proposed federated hyperparameter optimization of the FedAvg and participant selection [118]. The authors claim that the optimized selection of participants tends to improve accuracy and decrease the loss of the federated global model. Hence, the authors proposed Federated Simulated Annealing (FedSA), an extension of the Simulated Annealing (SA) meta-heuristic for a distributed scenario where the solutions tend to change at new rounds. The results show that the proposal can achieve slightly better accuracy than FedAvg [15], but with fewer aggregation rounds and participants. In addition, using FedSA in participant selection helps mitigate the risk of selecting malicious participants.

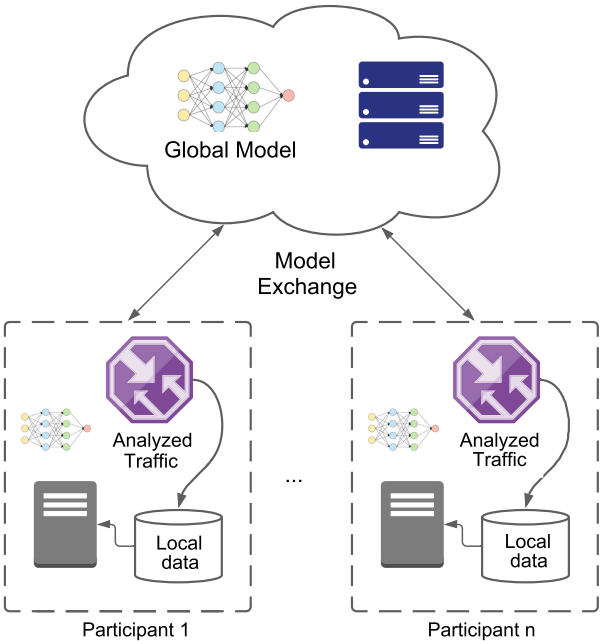


FIGURE 7. Reference architecture of the federated learning-based IDS proposed by [16]. The participants in this scenario are intermediary intrusion detection systems.

Nguyen et al. proposed an autonomous self-learning distributed system for detecting compromised IoT devices that utilize federated learning [115]. The authors systematically and extensively evaluated over 30 off-the-shelf IoT devices and show that the proposal can detect devices compromised by Mirai malware without false alarms in a real-world smart home deployment setting. In the paper, the authors assume that the participants are honest and willing to contribute to training using the parameters of their updated models. In the event of a group of participants being malicious, they can compromise the entire system. In contrast, Preuveneers et al. proposed using blockchain technology to manage data shared by participants [16]. The authors aim to use blockchain to store updates generated during training and thereby facilitate the identification of malicious activities. The authors identified that using blockchain generates training latency compared to training using the FedAvg algorithm without blockchain. The work, therefore, focuses only on using FL for intrusion detection, but it is not the scope of the work to use efficient models. It is important to highlight that it is not possible to ensure the participants will register their actual model in the blockchain. The use of blockchain in FL has the potential to introduce new security threats, including model inversion and GAN attacks. This is because the transparent nature of blockchain transactions could allow attackers to access the parameters of the trained model, which can then be used to reconstruct the original training data or generate fake data. Figure 7 shows a reference architecture for attack detection systems using FL.

TABLE 4. Security and privacy summary of recently federated learning applications.

	Proposal	Model	Poisoning attack	Model inversion	GAN Reconstruction	Privacy-preserving
Cyberattack Detection	Cunha Neto *et al.* [118]	MLP	✓	×	×	Not guaranteed
	Nguyen *et al.* [115]	GRU	×	×	×	Not guaranteed
	Preuveneers *et al.* [16]	Autoencoder	✓	×	×	Not guaranteed
	Chen *et al.* [119]	GRU	×	×	×	Not guaranteed
	Li *et al.* [120]	CNN	×	×	×	Not guaranteed
	Zhao *et al.* [121]	LSTM	×	×	×	Not guaranteed
	Mothukuri *et al.* [23]	LSTM and GRU	×	×	×	Not guaranteed
	Rahman *et al.* [122]	MLP	×	×	×	Not guaranteed
Optimizing Multiaccess Edge Computing Performance	Wang *et al.* [123]	DRL	×	×	×	Not guaranteed
	Ren *et al.* [124]	DRL	×	×	×	Not guaranteed
	Tam *et al.* [125]	CNN and DRL	×	×	×	Not guaranteed
	Chen *et al.* [17]	ENS Alg.	×	×	×	Not guaranteed
Vehicular Networks	Samarakoon *et al.* [18]	MLE	×	×	×	Not guaranteed
	Ye *et al.* [126]	CNN	×	×	×	Not guaranteed
	Elbir *et al.* [127]	CNN	×	×	×	Not guaranteed
	Otoum *et al.* [128]	MLP	×	×	×	Not guaranteed
	Saputra *et al.* [116]	MLP	×	×	×	Not guaranteed
Smart Healthcare	Brinimi *et al.* [19]	SVM	×	×	×	Not guaranteed
	Hao *et al.* [129]	CNN	×	✓	✓	Homomorphic encryption and perturbation
	Wu *et al.* [20]	CNN	×	×	×	Not guaranteed
	Chen *et al.* [130]	CNN	×	×	×	Not guaranteed
	Yan *et al.* [131]	GAN	×	✓	✓	Use synthetic data
	Wenqi *et al.* [39]	CNN	×	✓	✓	Differential privacy

Chen et al. proposed the Federated Learning-based Attention Gated Recurrent Unit (FedAGRU) algorithm [119] for IDS. The authors deploy a *Gated Recurrent Unit* (GRU) Neural Network but replace the output layer with a Support Vector Machine (SVM). The Global aggregation is asynchronous, *i.e.*, the server does not wait for all selected participants to send their parameters. The participants compare their model updates with the current global model at each global iteration by correlating the parameters. The participants only send their parameters if their update is relevant to the training.

Li et al. proposed a distributed IDS using FL in the Satellite-Terrestrial Integrated Network (STIN) scenario [120]. STIN is a valuable supplement to the wireless network allowing large-capacity information transmission service to space access networks and terrestrial networks. Besides using FL for distributed IDS, the authors also developed a dataset applied to the STIN challenges and limitations. The authors adapted the FL algorithm for the STIN scenario, proposing an efficient processing time synchronization, considering the network limitations of the satellites. The datasets were crafted in a prototype containing approximately forty nodes and eleven simulated attack types. The authors used CNN as the deep learning model.

Zhao et al. proposed an intrusion detection based on commands in Command Line Interface (CLI) [121]. The author used the Long Short-Term Memory (LSTM) model to provide richer semantic information in feature space combined with context. The author compared their proposal with centralized learning. The proposal achieve better performance since the model is built collaboratively. However, according to the simulation results, the performance of Federated Learning-based Long Short-Term Memory (FL-LSTM) and Centralized-LSTM is very close.

Mothukuri et al. proposed a FL approach using an ensemble to enable anomaly detection on the IoT networks [23]. The authors used GRU, neural network models, to train the machine learning model on a Modbus network dataset. The authors experimented with both GRU and LSTM, and GRU models outperformed LSTM to achieve a higher accuracy rate and be computationally inexpensive. In this paper, the authors used a Random Forest classifier to ensemble seven global model outputs.

Rey et al. proposed a federated learning-based IDS using Multilayer Perceptron (MLP) and autoencoder models [132]. The authors used the N-BaIoT, a dataset for modeling network traffic of several real IoT devices while affected by malware. The authors compare centralized, distributed, and FL architectures.

B. OPTIMIZING MULTIACCESS EDGE COMPUTING PERFORMANCE

In order to tackle the dynamic conditions and temporal variables inherent in a MEC system, Wang et al. proposed a novel methodology that employs both Deep Reinforcement Learning (DRL) and FL to optimize memory transient and compute offload decisions within said system [123]. The DRL agent is responsible for determining whether to store a downloaded file in temporary memory and, if so, which local file to replace in the temporary storage. At the same time, the reward function is defined as the Quality of Experience (QoE) of the user equipment. QoE refers to the overall quality of the user experience when accessing services and

applications that are offloaded to the edge of the network. Given the extensive number of states and actions within the MEC environment, the authors utilized a Double Deep Q-Network (DDQN) approach to manage communication and computing resources jointly.

The use of DDQN was motivated by demonstrated improved performance in [133] and [134]. Wang et al. used a Federated Learning approach to protect participant privacy [123] but did not implement parameter protection, leaving the possibility of data leakage. Additionally, Ren et al. have also suggested the use of DRL to optimize computation offloading decisions in IoT systems [124].

Chen et al. proposed a framework called Deep Echo State Networks (DeepESNs) to minimize Breaks In Presence (BIP) for wireless Virtual Reality (VR) users by considering the VR application type, transmission delay, VR video quality, and user awareness. BIP refers to a phenomenon in VR systems where the user becomes disconnected from the virtual world due to various factors such as transmission delays, video quality, and awareness of the virtual environment. BIPs can negatively impact the user's experience and immersion in the VR system. The proposed approach uses directional transmission links from Base Stations (BS) to users to reduce BIP occurrences. The BIP minimization problem is formulated as an optimization problem that considers user location, orientation, and BS association and is solved using a distributed learning algorithm based on federated Echo State Networks (ESN). The proposed algorithm allows multiple BS to locally train their DeepESNs and build a global learning model to predict users' locations and orientations. Simulation results show that the proposed algorithm reduces BIP occurrences by up to 26% compared to centralized ESN and deep learning algorithms.

Tam et al. proposed a reliable model communication scheme for securing real-time multiple spatial-resolution image sensing classifications in multi-platform IoT applications, considering both FL model computation and network communication efficiency [125]. The proposed system model is based on the Software-Defined Network (SDN)/Network Function Virtualization (NFV)-enabled architecture and uses an NFV-enabled MEC environment to ensure high-precision real-time classifications for mission-critical IoT image sensing in peak hour intervals and various congestion conditions. The proposed algorithms use Deep Q-Learning (DQL)-based models to sample allocation rules and target weak episodes with low aggregation rewards to obtain deficient policies. An agent controller configures long-term self-organizing IoT resource management, while an NFV-orchestrator maps virtual MEC resource pools based on selected model actions of particular network congestion states. The proposed FL classification model is validated using six different bottleneck scenarios in SDN/NFV-based IoT architecture. The simulation results show high precision and efficient Quality of Service (QoS) metrics for handling future congestion environments.

C. VEHICULAR NETWORKS

Vehicular networks enable data collection from multiple vehicles to improve traffic management and safety. Federated learning can also improve communication efficiency and reduce privacy concerns by training models locally on user devices and sharing only aggregated model updates.

Samarakoon et al. discusses the use of Extreme Value Theory (EVT) [135] for modeling rare events in radio resource management of Ultra-Reliable Low Latency Communications (URLLC) vehicular networks [18]. EVT can model the distribution of extreme events with low probability using Maximum Likelihood Estimation (MLE) and can be used to analyze network traffic, delays, peak rates, and Vehicle-to-Vehicle (V2V) communication in wireless systems. The proposed FL approach trains the learning model with locally held data and loads only the updated model parameters to the Roadside Units (RSU), which averages the model parameters and returns an updated global model for the Vehicular Users (VUEs). In contrast, in an asynchronous approach, each VUE evaluates and loads its model parameters after collecting a predefined number of Queue State Information (QSI) samples. The simulation results show that the proposed framework reduces the number of vehicles with long queue lengths while minimizing data exchange compared to a centralized approach.

Ye et al. proposed a selective model aggregation approach for federated learning in Vehicular Edge Computing (VEC) to overcome the potential impact of diverse image quality and computation capability of vehicular clients on the accuracy and efficiency of image classification [126]. The approach evaluates the image quality in terms of motion blur level. It selects 'fine' local Deep Neural Networks (DNN) models with satisfactory image quality and computation capability by formulating a two-dimensional contract problem, which is transformed into a tractable problem solved by a greedy algorithm. A two-dimensional contract problem is an optimization problem where the objective is to design a contract that incentivizes two parties to take actions that jointly maximize their utility. The contract specifies the payments each party will receive based on their actions and the outcome of the joint action. The two dimensions of this problem refer to the different types of actions or decisions each party can take. In this context, the two dimensions are the quality of the image data and the computation capability of the vehicular clients participating in the federated learning process. The central server designs a contract to incentivize the clients to select the best quality images and use their computation capabilities efficiently, resulting in a better global model for the federated learning system. The proposed approach outperformed the original federated averaging approach regarding accuracy and efficiency while achieving higher utility at the central server.

Elbir et al. proposed the FL approach in vehicular network applications to build intelligent transportation systems [127]. Unlike centralized learning, FL can reduce transmission

overhead by transmitting only model updates rather than the entire dataset. The paper analyzes the feasibility of FL in ML-based vehicular applications and identifies significant challenges from both learning and communication perspectives. However, the paper does not address privacy and security concerns, potentially rendering the application infeasible.

Otoum et al. proposed an integrated FL and Blockchain solution to ensure data privacy and network security in critical infrastructures [128]. The framework decentralizes the mutual machine learning models on end devices, enabling on-end device machine learning without centralized data training. The proposed model utilizes a consensus method in the blockchain for trustworthy shared training on the fog, while a practical Byzantine Fault Tolerance (pBFT) protocol overcomes many faulty issues. pBFT is a characteristic of a distributed system that allows it to continue functioning correctly and reliably even if some of its nodes or components fail or behave faultily. The results show that the proposed Blockchain-Federated Learning model outperforms other solutions in accuracy, latency, throughput, lifetime reduction, and energy consumption, with an accuracy rate of around 97%. While the proposed model utilizes pBFT for blockchain consensus, the proposal is not protected against a byzantine attack on FL. The paper does not discuss the specific mechanisms to detect and prevent targeted or untargeted attacks on the model.

Saputra et al. proposed machine learning-based approaches for predicting energy demand in Electric Vehicle (EV) networks. The first approach is an Energy Demand Learning (EDL) algorithm that uses data from all Charging Stations (CS) to predict energy demand for the entire network. A Federated Energy Demand Learning (FEDL) approach is introduced to address privacy concerns and reduce communication overhead. The CSs share only their trained models with the Charging Station Provider (CSP). The paper further proposes a clustering-based EDL approach to improve prediction accuracy by grouping CSs into clusters before applying the FEDL algorithm. Experimental results demonstrate that the proposed approaches improve prediction accuracy by up to 24.63% and reduce communication overhead by 83.4% compared to baseline machine learning algorithms. The proposed methods leverage deep learning techniques to improve energy demand prediction accuracy and reduce communication overhead in EV networks. Although the paper proposed using FL to enhance privacy, this objective was not achieved due to a lack of parameter protection. As a result, the privacy benefits expected from FL are not achieved.

D. SMART HEALTHCARE

Deep learning algorithms have been extensively used in the healthcare field to predict diseases, remote health monitoring, and medical imaging [117]. Training an accurate deep learning model requires a large amount of data. However, medical data is sensitive and private, and transferring the data to a data center for training a deep learning model is prohibited by current privacy laws. Then, the main challenge of training this model is privacy leakage. For reliable and private collaborative training, FL is increasingly used in the smart healthcare field.

Brisimi et al. proposed a FL approach to predict forthcoming hospitalizations for patients with heart diseases using Electronic Health Record (EHR) data on various hospitals [19]. The aggregation server aggregates the local model parameters to build a global Support Vector Machine (SVM) classifier. Furthermore, Hao et al. proposed a privacy-aware and resource-saving collaborative training based on FL for an EHR management system with the collaboration of multiple institutions [129]. In the proposal, each institution (hospital) is a participant in the FL training. The participant collaboratively trains a Convolutional Neural Network (CNN) in medical images to identify diseases. The authors proposed a lightweight data perturbation method and utilized packed partially homomorphic encryption to protect the training against data inferring. The authors also proposed the split of the CNN into three parts. The first part is the shallow layers, which consist of the input layer and the first two hidden layers. The second part consists of all the remaining hidden layers except the last one. The last part consists of the last hidden layer and the output layers. Only the second part of the local model is sent to aggregation, decreasing the communication cost.

Besides using FL for EHR management, works on literature proposed federated learning-based remote health monitoring. Wu et al. proposed a FL framework for in-home health monitoring using CNN model [20]. In this framework, smartphones of the participants at each house learn personal data and train a local CNN. The authors used the Generative Convolutional Autoencoder (GCAE) alongside the CNN to generate synthetical data samples of minority classes. By using GCAE to balance the dataset, the framework deals with imbalanced data and Non-IID, increasing the CNN performance. Chen et al. proposed a federated transfer learning scheme for wearable health monitoring using CNN models [130]. In the proposed scheme, the aggregation server initializes the global model by training it with a public dataset instead of starting the global model with random parameters. Subsequently, the participants update this model with their private data and send it to the global model for aggregation.

Yan et al. proposed a Variation-Aware Federated Learning (VAFL) framework, where each participant trains a GAN to generate synthetic image data [131]. The generated images are shared among the participants to minimize the variation in the datasets of the participants. After the participants generating a synthetic dataset with a common image space, a cloud server can use it to train a classifier. The authors validated VAFL on prostate cancer-related images, achieving 97,22% accuracy. Li et al. proposed using FL to support brain tumor segmentation [39]. Each hospital is a participant in this environment, and the local data consists of Magnetic Resonance Images (MRI). The authors used differential privacy techniques to protect the data stored in the participants against leakage.

X. CONCLUSION

FL is an emerging area of research that takes advantage of the increased processing capacity of edge devices to ensure the privacy of data stored on participants. The FL technique is based on developing a collaborative learning model in which participants, often mobile nodes, perform part of the learning task locally and contribute to a global model. However, the development of collaborative learning models presents several challenges, including the heterogeneity of participants' devices and the statistical heterogeneity of the data. This paper analyzed the main FL proposals and highlighted the difficulties these architectures face when dealing with privacy, heterogeneous data, and devices. We enumerated the main applications in FL and compared them in a privacy optic. We conclude that most FL applications do not concern about security and privacy. It is important to highlight that an application must not propose a new privacy and security-maintaining mechanism but must ensure them. Finally, we listed the main research challenges in FL and analyzed the current proposals to address each issue. The solutions for these issues are in the initial stage and require improvement.

The paper also analyzed the main FL attacks. We categorized the attacks into model performance and data privacy attacks. Model performance attacks aim to degrade or backdoor the global model during the training. On the other hand, privacy attacks aim to infer the participants' private data. The main defenses for the attacks consume essential resources, such as computation and network. Also, less resource-intensive defenses add noises during the updates, affecting the performance of the model. The increased computational or network resources can make most FL scenarios unfeasible. The lack of maturity in the security and privacy proposals is reflected in the security problems of FL applications, as pointed out in this paper. Consequently, the security and privacy of FL is a critical open issue.

REFERENCES

[1] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[2] H. N. C. Neto, M. A. Lopez, N. C. Fernandes, and D. M. Mattos, "Minecap: Super incremental learning for detecting and blocking cryptocurrency mining on software-defined networking," *Ann. Telecommun.*, vol. 75, pp. 1–11, Jan. 2020.

[3] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. S. Awwal, and V. K. Asari, "A state-of-the-art survey on deep learning theory and architectures," *Electronics*, vol. 8, no. 3, p. 292, Mar. 2019.

[4] D. S. V. Medeiros, H. N. C. Neto, M. A. Lopez, L. C. S. Magalhães, N. C. Fernandes, A. B. Vieira, E. F. Silva, and D. M. F. Mattos, "A survey on data analysis on large-scale wireless networks: Online stream processing, trends, and challenges," *J. Internet Services Appl.*, vol. 11, no. 1, pp. 1–48, Dec. 2020.

[5] P. Li, J. Li, Z. Huang, T. Li, C.-Z. Gao, S.-M. Yiu, and K. Chen, "Multi-key privacy-preserving deep learning in cloud computing," *Future Generat. Comput. Syst.* vol. 74, pp. 76–85, Sep. 2017.

[6] P. Verma, S. Sood, and S. Kalra, "Cloud-centric IoT based student healthcare monitoring framework," *J. Ambient Intell. Humanized Comput.*, vol. 9, no. 5, pp. 1293–1309, 2018.

[7] U. A. Butt, M. Mehmood, S. B. H. Shah, R. Amin, M. W. Shaukat, S. M. Raza, D. Y. Suh, and M. J. Piran, "A review of machine learning algorithms for cloud computing security," *Electronics*, vol. 9, no. 9, p. 1379, Aug. 2020.

[8] P. Arulanthu and E. Perumal, "An intelligent IoT with cloud centric medical decision support system for chronic kidney disease prediction," *Int. J. Imag. Syst. Technol.*, vol. 30, no. 3, pp. 815–827, Sep. 2020.

[9] (2016). *Parlamento Europeu e Conselho da União Européia, Regulamento (ue) 2016/679*. [Online]. Available: https://eur-lex.europa.eu/legal-content/PT/TXT/PDF/?uri=CELEX:32016R0679&from=PT

[10] M. T. De Oliveira, L. H. A. Reis, Y. Verginadis, D. M. F. Mattos, and S. D. Olabarriaga, "SmartAccess: Attribute-based access control system for medical records based on smart contracts," *IEEE Access*, vol. 10, pp. 117836–117854, 2022.

[11] *Lei n 13.709, de 14 de Agosto de 2018. Institui a Lei Geral de Proteção de Dados Pessoais (LGPD)*, Governo Brasileiro, Brasil, 2018.

[12] A. Gupta and R. K. Jha, "A survey of 5G network: Architecture and emerging technologies," *IEEE Access*, vol. 3, pp. 1206–1232, 2015.

[13] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[14] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart., 2020.

[15] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist. (AISTATS)*, vol. 54, 2017, pp. 1273–1282.

[16] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, "Chained anomaly detection models for federated learning: An intrusion detection case study," *Appl. Sci.*, vol. 8, no. 12, p. 2663, 2018.

[17] M. Chen, O. Semiari, W. Saad, X. Liu, and C. Yin, "Federated echo state learning for minimizing breaks in presence in wireless virtual reality networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 177–191, Jan. 2020.

[18] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Federated learning for ultra-reliable low-latency V2V communications," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–7.

[19] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *Int. J. Med. Inform.*, vol. 112, pp. 59–67, 2018.

[20] Q. Wu, X. Chen, Z. Zhou, and J. Zhang, "FedHome: Cloud-edge based personalized federated learning for in-home health monitoring," *IEEE Trans. Mobile Comput.*, vol. 21, no. 8, pp. 2818–2832, Aug. 2022.

[21] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *Proc. 23rd USENIX Secur. Symp. (USENIX Secur.)*, 2014, pp. 17–32.

[22] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-Y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Red Hook, NY, USA: Curran Associates, 2020, pp. 16070–16084.

[23] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, "Federated learning-based anomaly detection for IoT security attacks," *IEEE Internet Things J.*, vol. 9, no. 4, pp. 2545–2554, Feb. 2022.

[24] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[25] P. Liu, X. Xu, and W. Wang, "Threats, attacks and defenses to federated learning: Issues, taxonomy and perspectives," *Cybersecurity*, vol. 5, no. 1, pp. 1–19, Dec. 2022.

[26] M. S. Jere, T. Farnan, and F. Koushanfar, "A taxonomy of attacks on federated learning," *IEEE Secur. Privacy*, vol. 19, no. 2, pp. 20–28, Mar. 2021.

[27] M. J. Page et al., "The PRISMA 2020 statement: An updated guideline for reporting systematic reviews," *Systematic Rev.*, vol. 10, no. 1, Dec. 2021, Art. no. 105906.

[28] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," 2020, *arXiv:2003.02133*.

[29] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Gener. Comput. Syst.*, vol. 115, pp. 619–640, Feb. 2021.

[30] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowl.-Based Syst.*, vol. 216, Mar. 2021, Art. no. 106775.

[31] M. A. Lopez, D. M. F. Mattos, O. C. M. B. Duarte, and G. Pujolle, "Toward a monitoring and threat detection system based on stream processing as a virtual network function for big data," *Concurrency Comput., Pract. Exper.*, vol. 31, no. 20, p. e5344, 2019.

[32] M. Assefi, E. Behravesh, G. Liu, and A. P. Tafti, "Big data machine learning using apache spark MLlib," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 3492–3498.

[33] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[34] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica, "Apache spark: A unified engine for big data processing," *Commun. ACM*, vol. 59, no. 11, pp. 56–65, Nov. 2016.

[35] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. A. Gibson, G. Ganger, and E. P. Xing, "More effective distributed ML via a stale synchronous parallel parameter server," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates, 2013, pp. 1223–1231.

[36] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 1205–1221, Jun. 2019.

[37] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT*. Heidelberg, Germany: Physica-Verlag HD, 2010, pp. 177–186.

[38] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," 2018, *arXiv:1811.03604*.

[39] W. Li, F. Milletarì, D. Xu, N. Rieke, J. Hancox, W. Zhu, M. Baust, Y. Cheng, S. Ourselin, M. J. Cardoso, and A. Feng, "Privacy-preserving federated brain tumour segmentation," in *Machine Learning in Medical Imaging*. Cham, Switzerland: Springer, 2019, pp. 133–141.

[40] D. Verma, S. Julier, and G. Cirincione, "Federated AI for building AI solutions across multiple agencies," 2018, *arXiv:1809.10036*.

[41] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1333–1345, May 2018.

[42] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*. New York, NY, USA: Association for Computing Machinery, 2017, pp. 1175–1191.

[43] W. Du and M. J. Atallah, "Privacy-preserving cooperative statistical analysis," in *Proc. 17th Annu. Comput. Secur. Appl. Conf.*, 2001, pp. 102–110.

[44] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *Proc. KDD*. New York, NY, USA: Association for Computing Machinery, 2002, pp. 639–644.

[45] A. F. Karr, X. Lin, A. P. Sanil, and J. P. Reiter, "Privacy-preserving analysis of vertically partitioned data using secure matrix products," *J. Off. Statist.*, vol. 25, no. 1, p. 125, Jan. 2009.

[46] W. Du, Y. S. Han, and S. Chen, "Privacy-preserving multivariate statistical analysis: Linear regression and classification," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2004, pp. 222–233.

[47] L. Wan, W. K. Ng, S. Han, and V. C. S. Lee, "Privacy-preservation for gradient descent methods," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2007, pp. 775–783.

[48] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," 2017, *arXiv:1711.10677*.

[49] R. Nock, S. Hardy, W. Henecka, H. Ivey-Law, G. Patrini, G. Smith, and B. Thorne, "Entity resolution and federated learning get a federated resolution," 2018, *arXiv:1803.04035*.

[50] G. Liang and S. S. Chawathe, "Privacy-preserving inter-database operations," in *Proc. Int. Conf. Intell. Secur. Inform.* Berlin, Germany: Springer, 2004, pp. 66–82.

[51] M. Scannapieco, I. Figotin, E. Bertino, and A. K. Elmagarmid, "Privacy preserving schema and data matching," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2007, pp. 653–664.

[52] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game, or a completeness theorem for protocols with honest majority," in *Proc. Providing Sound Found. Cryptogr., Work Shafi Goldwasser Silvio Micali*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 307–328, doi: 10.1145/3335741.3335755.

[53] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[54] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020.

[55] P. Bogetoft, D. L. Christensen, and I. Damg, "Multiparty computation Goes live," *Rev. Literature Arts Americas*, vol. 5628, pp. 1–13, 2009.

[56] J. Vaidya, C. Clifton, M. Kantarcioglu, and A. S. Patterson, "Privacy-preserving decision trees over vertically partitioned data," *ACM Trans. Knowl. Discovery From Data*, vol. 2, no. 3, pp. 1–27, Oct. 2008.

[57] W. Du and Z. Zhan, "Building decision tree classifier on private data," Dept. Elect. Eng. Comput. Sci., Syracuse Univ., Syracuse, NY, USA, Tech. Rep. 8, 2002.

[58] J. Vaidya and C. Clifton, "Privacy-preserving *k*-means clustering over vertically partitioned data," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*. New York, NY, USA: Association for Computing Machinery, 2003, pp. 206–215, doi: 10.1145/956750.956776.

[59] J. Vaidya and C. Clifton, "Privacy preserving Naïve Bayes classifier for vertically partitioned data," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2004, pp. 522–526.

[60] J. Zhang, B. Chen, S. Yu, and H. Deng, "PEFL: A privacy-enhanced federated learning scheme for big data analytics," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[61] S. Goryczka and L. Xiong, "A comprehensive comparison of multiparty secure additions with differential privacy," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 5, pp. 463–477, Oct. 2015.

[62] Y. Li, Y. Zhou, A. Jolfaei, D. Yu, G. Xu, and X. Zheng, "Privacy-preserving federated learning framework based on chained secure multiparty computing," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6178–6186, Apr. 2021.

[63] M. Goldblum, D. Tsipras, C. Xie, X. Chen, A. Schwarzschild, D. Song, A. Madry, B. Li, and T. Goldstein, "Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 1563–1580, Feb. 2023.

[64] C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," 2018, *arXiv:1808.04866*.

[65] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "DBA: Distributed backdoor attacks against federated learning," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–19.

[66] J. Zhang, J. Chen, D. Wu, B. Chen, and S. Yu, "Poisoning attack in federated learning using generative adversarial nets," in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./13th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2019, pp. 374–380.

[67] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *Proc. 29th USENIX Conf. Secur. Symp.*, 2020, pp. 1623–1640.

[68] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Computer Security—ESORICS*, L. Chen, N. Li, K. Liang, and S. Schneider, Eds. Cham, Switzerland: Springer, 2020, pp. 480–501.

[69] Z. Sun, P. Kairouz, A. T. Suresh, and H. Brendan McMahan, "Can you really backdoor federated learning?" 2019, *arXiv:1911.07963*.

[70] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 634–643.

[71] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Mach. Learn. Res.*, vol. 108, Aug. 2020, pp. 2938–2948.

[72] X. Zhou, M. Xu, Y. Wu, and N. Zheng, "Deep model poisoning attack on federated learning," *Future Internet*, vol. 13, no. 3, p. 73, 2021. [Online]. Available: https://www.mdpi.com/1999-5903/13/3/73

[73] S. Andreina, G. A. Marson, H. Möllering, and G. Karame, "BaFFLe: Backdoor detection via feedback-based federated learning," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2021, pp. 852–863.

[74] S. Shen, S. Tople, and P. Saxena, "AUROR: Defending against poisoning attacks in collaborative deep learning systems," in *Proc. 32nd Annu. Conf. Comput. Secur. Appl. (ACSAC)*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 508–519.

[75] C. Xie, M. Chen, P.-Y. Chen, and B. Li, "CRFL: Certifiably robust federated learning against backdoor attacks," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 11372–11382.

[76] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1310–1320.

[77] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Trans. Signal Process.*, vol. 70, pp. 1142–1154, 2022.

[78] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. 35th Int. Conf. Mach. Learn.*, in Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80, Jul. 2018, pp. 5650–5659.

[79] E. M. El Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in Byzantium," in *Proc. 35th Int. Conf. Mach. Learn.*, in Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80, Jul. 2018, pp. 3521–3530.

[80] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "On-device federated learning via blockchain and its latency analysis," 2018, *arXiv:1808.03949*.

[81] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2438–2455, Sep./Oct. 2019.

[82] A. Richardson, A. Filos-Ratsikas, and B. Faltings, *Budget-Bounded Incentives for Federated Learning*. Cham, Switzerland: Springer, 2020, pp. 176–188, doi: 10.1007/978-3-030-63076-8_13.

[83] C. Huang, S. Ke, C. Kamhoua, P. Mohapatra, and X. Liu, "Incentivizing data contribution in cross-silo federated learning," 2022, *arXiv:2203.03885*.

[84] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. CCS*. New York, NY, USA: Association for Computing Machinery, 2015, pp. 1322–1333.

[85] A. Triastcyn and B. Faltings, "Federated generative privacy," *IEEE Intell. Syst.*, vol. 35, no. 4, pp. 50–57, Jul. 2020.

[86] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, Q. Le, and A. Ng, "Large scale distributed deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1223–1231.

[87] T. Titcombe, A. J. Hall, P. Papadopoulos, and D. Romanini, "Practical defences against model inversion attacks for split neural networks," 2021, *arXiv:2104.05743*.

[88] T. Qi, F. Wu, C. Wu, Y. Huang, and X. Xie, "Privacy-preserving news recommendation model learning," 2020, *arXiv:2003.09592*.

[89] Z. Chen, A. Fu, Y. Zhang, Z. Liu, F. Zeng, and R. H. Deng, "Secure collaborative deep learning against GAN attacks in the Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5839–5849, Apr. 2021.

[90] X. Yan, B. Cui, Y. Xu, P. Shi, and Z. Wang, "A method of information protection for collaborative deep learning under GAN model attack," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 18, no. 3, pp. 871–881, May 2021.

[91] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*. New York, NY, USA: Association for Computing Machinery, 2017, pp. 603–618, doi: 10.1145/3133956.3134012.

[92] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," in *Proc. Asia Conf. Comput. Commun. Secur. (ASIACCS)*. New York, NY, USA: Association for Computing Machinery, 2018, pp. 707–721.

[93] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. Brendan McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," 2019, *arXiv:1902.01046*.

[94] Y. Liu, X. Zhang, Y. Kang, L. Li, T. Chen, M. Hong, and Q. Yang, "FedBCD: A communication-efficient collaborative learning framework for distributed features," *IEEE Trans. Signal Process.*, vol. 70, pp. 4277–4290, 2022.

[95] X. Yao, C. Huang, and L. Sun, "Two-stream federated learning: Reduce the communication costs," in *Proc. IEEE Vis. Commun. Image Process. (VCIP)*, Dec. 2018, pp. 1–4.

[96] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.

[97] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," 2018, *arXiv:1812.07210*.

[98] Z. Tao and Q. Li, "eSGD: Communication efficient distributed deep learning on the edge," in *Proc. USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, 2018, pp. 1–6.

[99] L. Wang, W. Wang, and B. Li, "CMFL: Mitigating communication overhead for federated learning," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 954–964.

[100] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10700–10714, Dec. 2019.

[101] M. R. Sprague, A. Jalalirad, M. Scavuzzo, C. Capota, M. Neun, L. Do, and M. Kopp, "Asynchronous federated learning for geospatial applications," in *Proc. ECML PKDD Workshops*. Cham, Switzerland: Springer, 2019, pp. 21–28.

[102] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," 2019, *arXiv:1903.03934*.

[103] G. Kim, J. Kim, and B. Han, "Communication-efficient federated learning with acceleration of global momentum," 2022, *arXiv:2201.03172*.

[104] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4424–4434.

[105] L. Corinzia, A. Beuret, and J. M. Buhmann, "Variational federated multi-task learning," 2019, *arXiv:1906.06268*.

[106] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, 2020, pp. 429–450.

[107] L. Huang, Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu, "LoAdaBoost: Loss-based AdaBoost federated machine learning with reduced computational complexity on IID and non-IID intensive care data," *PLoS ONE*, vol. 15, no. 4, Apr. 2020, Art. no. e0230706.

[108] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[109] Y. Liu, Y. Kang, X. Zhang, L. Li, Y. Cheng, T. Chen, M. Hong, and Q. Yang, "A communication efficient collaborative learning framework for distributed features," 2019, *arXiv:1912.11187*.

[110] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 97–105.

[111] A. T. Suresh, X. Y. Felix, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3329–3337.

[112] N. Strom, "Scalable distributed DNN training using commodity GPU cloud computing," in *Proc. 16th Annu. Conf. Int. Speech Commun. Assoc.*, 2015, pp. 1–5.

[113] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artif. Intell. Rev.*, vol. 18, no. 2, pp. 77–95, 2002.

[114] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, Jul. 1997.

[115] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DÏoT: A federated self-learning anomaly detection system for IoT," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 756–767.

[116] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, E. Dutkiewicz, M. D. Mueck, and S. Srikanteswara, "Energy demand prediction with federated learning for electric vehicle networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[117] D. C. Nguyen, Q.-V. Pham, P. N. Pathirana, M. Ding, A. Seneviratne, Z. Lin, O. Dobre, and W.-J. Hwang, "Federated learning for smart healthcare: A survey," *ACM Comput. Surv.*, vol. 55, no. 3, pp. 1–37, Feb. 2022, doi: 10.1145/3501296.

[118] H. N. C. Neto, I. Dusparic, D. M. F. Mattos, and N. C. Fernande, "FedSA: Accelerating intrusion detection in collaborative environments with federated simulated annealing," in *Proc. IEEE 8th Int. Conf. Netw. Softwarization (NetSoft)*, Jun. 2022, pp. 1–5.
[119] Z. Chen, N. Lv, P. Liu, Y. Fang, K. Chen, and W. Pan, "Intrusion detection for wireless edge networks based on federated learning," *IEEE Access*, vol. 8, pp. 217463–217472, 2020.
[120] K. Li, H. Zhou, Z. Tu, W. Wang, and H. Zhang, "Distributed network intrusion detection system in satellite-terrestrial integrated networks using federated learning," *IEEE Access*, vol. 8, pp. 214852–214865, 2020.
[121] R. Zhao, Y. Yin, Y. Shi, and Z. Xue, "Intelligent intrusion detection based on federated learning aided long short-term memory," *Phys. Commun.*, vol. 42, Oct. 2020, Art. no. 101157. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1874490720302342
[122] S. A. Rahman, H. Tout, C. Talhi, and A. Mourad, "Internet of Things intrusion detection: Centralized, on-device, or federated learning?" *IEEE Netw.*, vol. 34, no. 6, pp. 310–317, Nov. 2020.
[123] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Netw.*, vol. 33, no. 5, pp. 156–165, Sep. 2019.
[124] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang, "Federated learning-based computation offloading optimization in edge computing-supported Internet of Things," *IEEE Access*, vol. 7, pp. 69194–69201, 2019.
[125] P. Tam, S. Math, C. Nam, and S. Kim, "Adaptive resource optimized edge federated learning in real-time image sensing classifications," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 10929–10940, 2021.
[126] D. Ye, R. Yu, M. Pan, Z. Han, and H. Ma, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23920–23935, 2020.
[127] A. M. Elbir, B. Soner, S. Çöleri, D. Gündüz, and M. Bennis, "Federated learning in vehicular networks," in *Proc. IEEE Int. Medit. Conf. Commun. Netw. (MeditCom)*, Sep. 2022, pp. 72–77.
[128] S. Otoum, I. Al Ridhawi, and H. T. Mouftah, "Blockchain-supported federated learning for trustworthy vehicular networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–6.
[129] M. Hao, H. Li, G. Xu, Z. Liu, and Z. Chen, "Privacy-aware and resource-saving collaborative learning for healthcare in cloud computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.
[130] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, "FedHealth: A federated transfer learning framework for wearable healthcare," *IEEE Intell. Syst.*, vol. 35, no. 4, pp. 83–93, Jul. 2020.
[131] Z. Yan, J. Wicaksana, Z. Wang, X. Yang, and K.-T. Cheng, "Variation-aware federated learning with multi-source decentralized medical image data," *IEEE J. Biomed. Health Informat.*, vol. 25, no. 7, pp. 2615–2628, Jul. 2021.
[132] V. Rey, P. M. S. Sánchez, A. H. Celdrán, and G. Bovet, "Federated learning for malware detection in IoT devices," *Comput. Netw.*, vol. 204, Feb. 2022, Art. no. 108693.
[133] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and scalable caching for 5G using reinforcement learning of space-time popularities," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 180–190, Feb. 2018.
[134] T. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 44–55, Jan. 2018.
[135] L. De Haan and A. Ferreira, *Extreme Value Theory: An Introduction*. New York, NY, USA: Springer-Verlag, 2006.

HELIO N. CUNHA NETO received the Graduate degree in computer networks from the University of Grande Rio, in 2016, and the master's degree, in 2019. He is currently pursuing the Ph.D. degree in the electrical and telecommunications engineering postgraduate program with Fluminense Federal University (UFF). He has experience in computer networking, software defined networking, machine learning, and federated learning. In 2020, he received the PPGEET Best Master Thesis Award (second place), and the Brazillian Army Medal for the UFF Telessaude Project, in 2018.

JERNEJ HRIBAR received the bachelor's degree in electrical engineering from the University of Ljubljana, Slovenia, in 2014, and the Ph.D. degree from Trinity College Dublin, Ireland, in 2020. He is currently a Marie Skłodowska-Curie Action Fellow with the Department of Communication Systems, Jožef Stefan Institute, Slovenia. His MSCA action—TimeSmart—investigates the applicability of the novel age of information metric in smart grids. From 2019 to 2022, he was a Postdoctoral Researcher with the CONNECT Centre for Future Networks and Communications, Trinity College Dublin. His research interests include the age of information, deep reinforcement learning, and federated learning.

IVANA DUSPARIC received the B.Sc. degree from the La Roche College, Pittsburgh, USA, in 2001, and the M.Sc. and Ph.D. degrees from TCD, in 2005 and 2010, respectively. She has been an Ussher Associate Professor with the School of Computer Science and Statistics, Trinity College Dublin, since 2016. Her research expertise is the development of new artificial intelligence algorithms, and specifically reinforcement learning, for optimization of large-scale infrastructures. She has authored numerous peer-reviewed articles in the areas of reinforcement learning agents, multi-agent systems, intelligent mobility, and future communication networks.

DIOGO MENEZES FERRAZANI MATTOS (Member, IEEE) received the Ph.D. degree in electrical engineering from Universidade Federal do Rio de Janeiro (COPPE/UFRJ), in 2017. From 2015 to 2016, he was a Sandwich Doctoral Fellow with Laboratoire d'Informatique de Paris 6 (LIP6), Université Sorbonne, Campus Pierre et Marie Curie, Paris, France. He is currently a Professor with the Department of Telecommunications Engineering, Universidade Federal Fluminense (UFF), Niterói, Brazil. He received research grants from the Brazilian National Council for Scientific and Technological Development (CNPq), the Rio de Janeiro State's Research Support Foundation (FAPERJ), and the Municipality of Niterói. His research interests include network security, next-generation networks, and the future internet applications.

NATALIA C. FERNANDES received the degree in electronics and computer engineering, and the M.Sc. and D.Sc. degrees in electrical engineering from Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil, in 2006, 2008, and 2011, respectively. She is currently an Associate Professor with the Telecommunications Engineering Department, Universidade Federal Fluminense (UFF), Niterói, Rio de Janeiro. She has received several research grants from the Brazilian National Council for Scientific and Technological Development (CNPq), the Rio de Janeiro State's Research Support Foundation (FAPERJ), and other funding agencies. Her primary research interests include the IoT applications, network security, and wireless networks.