Hindawi
Security and Communication Networks
Volume 2022, Article ID 2886795, 24 pages
https://doi.org/10.1155/2022/2886795

Review Article

Security and Privacy Threats to Federated Learning: Issues, Methods, and Challenges

Junpeng Zhang,[1,2] **Hui Zhu**,[1] **Fengwei Wang**,[1] **Jiaqi Zhao**,[1] **Qi Xu**,[1] **and Hui Li**[1]

[1]*School of Cyber Engineering, Xidian University, Xi'an, Shaanxi, China*
[2]*College of Computer and Cyber Security, Hebei Normal University, Shijiazhuang, Hebei, China*

Correspondence should be addressed to Hui Zhu; zhuhui@xidian.edu.cn

Received 6 January 2022; Accepted 6 September 2022; Published 28 September 2022

Academic Editor: Zhen Wang

Federated learning (FL) has nourished a promising method for data silos, which enables multiple participants to construct a joint model collaboratively without centralizing data. The security and privacy considerations of FL are focused on ensuring the robustness of the global model and the privacy of participants' information. However, the FL paradigm is under various security threats from the adversary aggregator and participants. Therefore, it is necessary to comprehensively identify and classify potential threats to provide a theoretical basis for FL with security guarantees. In this paper, a unique classification of attacks, which reviews state-of-the-art research on security and privacy issues for FL, is constructed from the perspective of malicious threats based on different computing parties. Specifically, we categorize attacks with respect to performed by aggregator and participant, highlighting the Deep Gradients Leakage attacks and Generative Adversarial Networks attacks. Following an overview of attack methods, we discuss the primary mitigation techniques against security risks and privacy breaches, especially the application of blockchain and Trusted Execution Environments. Finally, several promising directions for future research are discussed.

1. Introduction

The rapid development of cloud-centric machine learning has witnessed a new round of artificial intelligence outbreaks that rely on continuous breakthroughs in algorithms and computing capabilities [1]. Simultaneously, modern mobile devices empower to generate massive data suitable for machine learning in the big data era, which significantly improves the training model accuracy [2]. Obviously, such an unprecedented amount of data collection in a cloud server for model training is expensive and time-consuming [3]. In addition, the direct access of training datasets also raises public concerns about privacy and confidentiality [4]. As a result, increasingly stringent regulations are ensuring the protection of users' sensitive data against general violation of privacy [5]. However, such invisibility property of training data would induce data silos to impede the advancement of machine learning. Federated learning (FL) has been introduced to get rid of the aforementioned constraints by avoiding direct access to the original datasets while collaboratively building high-quality global models.

The FL paradigm has a collaborative training process that performs model training by distributing datasets over massive participants. In the FL setting, each participant benefits from the datasets of other participants only through the shared global model in the federation without explicitly accessing their sensitive data. With its unique feature of collaborative training that distributes models and makes predictions by participants, FL has various substantial advantages as follows:

1.1. Enhanced Data Protection. The overall learning process only transmits model updates calculated on local datasets instead of raw datasets to the aggregator. Therefore, the training data do not leave the location where it is generated, which provides a degree of data protection [6].

1.2. Decreased Energy Consumption. The distributed data architecture always keeps training data locally rather than transferring it to the cloud for centralized storage and processing [7], which significantly reduces network bandwidth and energy consumption, especially in tasks involving unstructured data.

1.3. Reduced Latency. In realistic scenarios, the latency of inferring directly from participants is much lower communication than predicting in the cloud and then transferring to participants [8, 9]. The implementation of FL in mobile edge networks accelerates content delivery and improves mobile service quality by reducing unnecessary system communication load [10]. The model inference is completed locally without a cloud round-trip that avoids propagation delay caused by transferring data, and thus latency-sensitive applications can benefit from such a solution.

Because of the aforementioned advantages, FL has been widely used in many areas such as finance, healthcare, smart city, edge computing, IoT, etc. For example, FL is helpful to provide the assisted diagnosis service of intelligent healthcare by collaborative modeling across medical institutions [11–13]. Apparently, more patient records would be required to support high-quality predictive models for precision diagnosis. By deploying FL, hospitals can train predictive models locally instead of exchanging their original healthcare data. The approach also alleviates concerns about the absence of standardized electronic medical records, which only requires transmitting calculated local updates to the aggregator rather than considering the local data structure of medical records. Moreover, in smart city services, FL provides an opportunity for ride-hailing applications to reduce latency and mitigate data leakage [14]. Each ride-hailing acts as a participant to perform FL that makes intelligent decisions instantly for resource allocation in urban vehicular networks. It indicates that many industries and areas can benefit from FL deployments to facilitate the rapid development of artificial intelligence applications.

To make the FL a better application prospect, it is significant to be at the forefront of this research field to explore all potential security threats. The existing FL protocols do not provide sufficient security and privacy guarantees in the presence of adversary participants. Recent researches demonstrate that an adversarial attacker may raise privacy concerns by launching inference attacks contra other participants, even when the iterations are performed only a few times [15]. Specifically, the adversary participant steals other participants' sensitive information from shared parameters since the transmitted gradient value is transformed from the original training data. Recently, security and privacy issues have become critical concerns due to potential security attacks and internal theft [16].

Currently, existing FL surveys tend to classify attacks according to security and privacy without considering different computing parties. This work is motivated by the lack of a comprehensive survey of the performed attacks from different computing parties. Compared with the existing surveys on FL-based topics, our work starts with an introduction to recent advances in FL and offers a state-of-the-art overview of security and privacy issues with the following contributions:

(1) We introduce the fundamentals of FL and present in detail various categorizations and their corresponding natures from different perspectives.

(2) We provide a novel threat classification of the effective attacks arising from the untrusted aggregator and malicious participants from the perspective of different computing parties.

(3) We discuss the most advanced mitigation techniques for security vulnerabilities to analyze their performance and limitations in the FL environment and provide taxonomy tables to help to have an insight into defense strategies. In particular, we focus on the utilized effects and prospects of blockchain and TEEs for FL security defense.

(4) We put forward current trends and challenges to facilitate FL that can be implemented on a large scale in realistic scenarios.

This survey structure is shown in Figure 1. The rest of this survey is outlined as follows. Section 2 provides a concise description of the FL concept and categorizations. We subsequently present research issues about privacy risks and security breaches of FL in Section 3, including various attacks from different computing parties. The mitigation techniques for security vulnerabilities in FL are shown in Section 4. Section 5 sketches some potential research directions for security and privacy in the FL field. Finally, Section 6 concludes this survey.

2. Preliminaries and Overview

Motivated by minimizing user privacy leakage, we assume that the central entity can accomplish model learning without collecting original training data from data owners. It naturally triggers the idea of distributed learning solutions in which all training data is kept in the place where it is generated. In this context, McMahan et al. [17] proposed the concept of FL. FL is a collaborative machine learning paradigm that carries out data utilization and model learning by asking participants to share local training model parameters. Consequently, each participant benefits from the datasets of other participants only through the global shared model in the federation, without explicitly accessing their privacy-sensitive information. The background and features of FL are presented in this section. The taxonomies with different perspectives are also discussed.

2.1. Fundamentals of FL. The FL architecture has salient advantages over conventional cloud-centric training approaches, including distributed storage and computing, reduced latency, suitable for massive datasets, etc. Given the learning process, the comparison between FL and cloud-centric training is as follows:

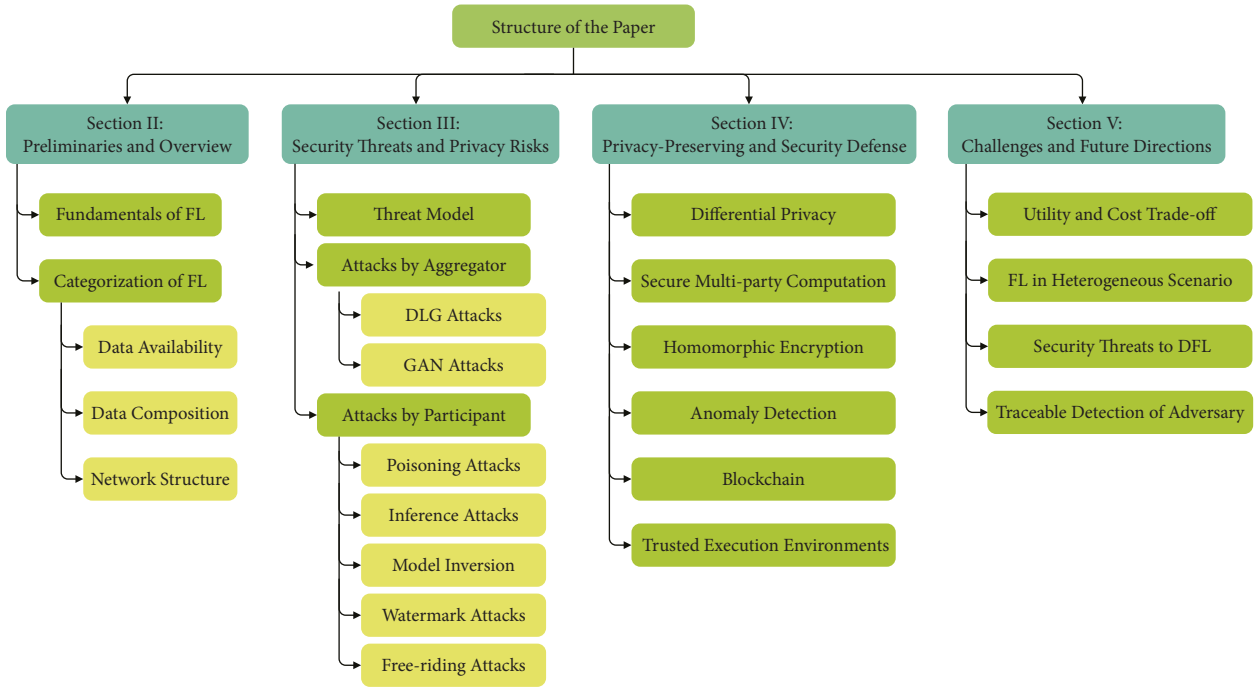

Figure 1: Organization of this survey.

Consider a group of $n \in \mathbb{N}$ participants $\mathcal{N} = \{1, 2, \ldots, n\}$ such that each participant has its data, respectively. Traditional machine learning collects these data to the cloud server, $D = D_1 \cup D_2 \cup \cdots \cup D_n$ with $d = |D|$ training samples. Then, the training dataset D is used as input data to execute machine learning algorithms, e.g., Deep Neural Networks (DNN). Formally, a typical supervised machine learning objective function can be expressed as:

$$\arg\min_\theta \mathscr{L}(\theta) = \frac{1}{d}\sum_{i \in d} \mathscr{L}\left(f\left(x_i, \theta\right), y_i\right), \tag{1}$$

where a learning algorithm is stated as f and its corresponding parameters θ are estimated from the dataset D with a total sample size of d by minimizing the loss function $\mathscr{L}$ between the predictions on input x_i and the actual label y_i in the training set.

While traditional cloud-centric training relies on the assumption of collecting participants' data, this is not always feasible in practice because increased awareness of privacy-preserving leads to the inaccessibility of private data. In addition, massive communication loading is required to transfer the original training dataset to the cloud server. It is thus particularly appealing for FL to use the participants' local data to learn a model M_{fed} collaboratively rather than putting all training data together.

In the FL, data are assumed to be distributed over a set of N participants, and the training data of each participant can be considered as P with $m = |P|$. The objective in this setting evolves to minimize the aggregated loss

$$g(\theta) = \sum_{n \in N} \frac{m_n}{d} \mathscr{F}\left(P_n, \theta\right), \tag{2}$$

where $\mathscr{F}$ is the local loss defined by

$$\mathscr{F}\left(P_n, \theta\right) = \frac{1}{m_n}\sum_{j \in P_n} \mathscr{L}\Big(f\left(x_j, \theta\right), y_j\Big), \tag{3}$$

Take the parallel gradient descent algorithm as an example; the FL training process usually contains the following steps:

2.1.1. Initialization. Participants receive the initial global model parameter θ from the aggregator;

2.1.2. Distributed Learning. Participants train the model with global model parameter θ and their training data, respectively;

2.1.3. Update. The ith participant computes training gradient g_i and transfers g_i to the aggregator;

2.1.4. Aggregation. The server aggregates all gradients $g = g_1 + \cdots + g_n$ without learning any information about participants, updating global model parameters $\theta \leftarrow \theta - \eta \cdot g$ (where η represents the learning rate) to transfer it for participants.

Steps 2–4 are repeated multiple iterations until the loss function converges to achieve a learning objective. Thus, a desired set of model M_{fed} is obtained. Figure 2 visualizes the typical FL training process in the above steps. In summary, the FL learning framework eliminates the need to have learning data centralized in a server. It learns a global model over distributed participants, transfers the computed update parameters to the aggregator, and enhances training quality by aggregating updates with the aggregator.

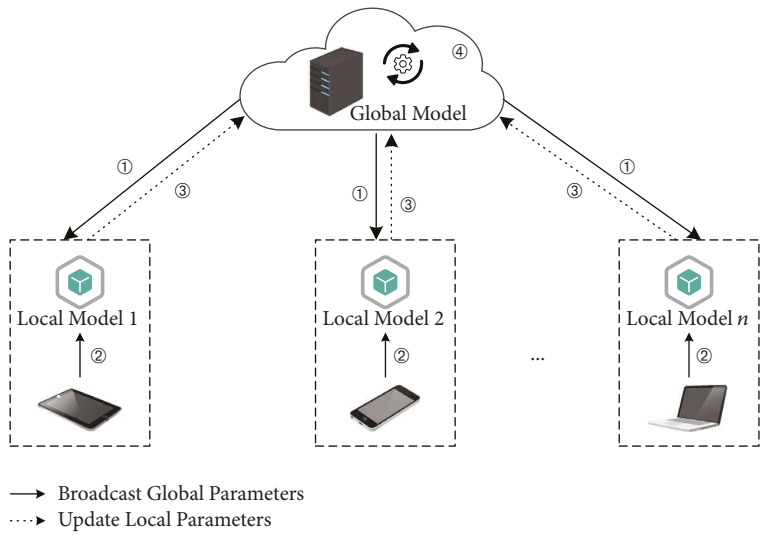

Figure 2: The representative FL training process. Each participant downloads incipient parameters from the server and trains the model locally. The server receives participants' model updates to aggregate and subsequently feedback new global model parameters to the participants for their use.

2.2. Categorization of FL. The introduction of FL would be a better choice for traditional centralized learning to advance the further widespread application of Artificial Intelligence (AI). For an emerging distributed AI framework, the precise classification of FL is the key initial stage to help investigate and understand it. Next, we gather and summarize internal works from different perspectives to understand privacy and security concerns in the following sections.

2.2.1. Data Availability. FL was initially introduced for large-scale distributed clients with a focus on mobile edge computing and then gradually extended to other scenarios, such as containing only a few stable clients. FL can be either cross-device FL or cross-silo FL based on data availability and different participant entities.

(1) Cross-Device FL (CDFL). CDFL is a large-scale distributed system that allows mobile terminals or IoT devices to freely join or leave the training process with absolute control over one's data. In this form of learning, communication costs are higher than computation costs because such huge numbers of devices are usually joined to the learning process over Wi-Fi. Therefore, it necessitates choosing a fixed number of participants to join in each training round. Moreover, participants involved in the training are unstable due to battery power or wireless network connection [23]. It is worth noting that the local training data possessed by participants are nonidentically and independently distributed (non-IID) and are severely imbalanced, which directly affects training quality for the global model.

(2) Cross-Silo FL (CSFL). In contrast to CDFL, the clients of CSFL are geographically distributed data centers with limited numbers but high computing power and network speed. CSFL expands the method of dividing training data based on features, which is more meaningful for solving practical problems such as supporting VFL. In addition, participants who work in this FL setting may have competitive relationships with others, so it is vital to establish incentive mechanisms to motivate participants to join learning.

Intuitively, CDFL refers to the fact that data owners consist of a large number of edge devices with limited computing resources and offline issues. In contrast, CSFL means that the data owners involve some relatively stable data centers with rich computing resources and limited quantity. Moreover, in CDFL, the aggregator obtains the final global model through FL to provide prediction services for users, while participants only collect data (e.g., mobile crowdsensing). Therefore, there is no requirement to protect the global model parameters against the aggregator. In CSFL, participants (i.e., data centers) possess the final global model and provide prediction services, while the aggregator only assists in this process. As a result, in this case, it is thus necessary to ensure the global model cannot be perceived by the aggregator. As shown in Table 1, both cross-device FL and cross-silo FL have salient features different from datacenter distributed learning.

2.2.2. Data Composition. According to how training data are distributed among the features and samples of various participants, FL can be split into three subcategories: horizontal federated learning, vertical federated learning, and federated transfer learning [24]. Table 2 summarizes these three FL frameworks and their associated characteristics.

(1) Horizontal Federated Learning (HFL). HFL is suitable for collaborative learning with similar features but different samples that refer to sample-partitioned FL. Therefore, all participants are allowed to use the same learning model for their local training due to the same data features. Mobile keyboard prediction is an instance of HFL, which demonstrates the benefits of training language models on mobile

Table 1: Comparative analysis among CDFL, CSFL, and datacenter distributed learning.

Category	Datacenter distributed learning	Cross-device FL [8, 18, 19]	Cross-silo FL [20–22]
Server role	Model training	Parameter aggregation	Parameter aggregation
Client composition	In a single cluster	Mobile or IoT devices	Geo-distributed data nodes
Client scale	General quantity	Huge quantity	General quantity
Data distribution	Centrally stored and IID	Imbalance and non-IID	Imbalance and non-IID
Data availability	Almost always available	Client unstable	Almost always available
Data partition	Arbitrary partition	Example-partition	Example-partition or feature-partition
Primary bottleneck	Computation	Communication	Computation or communication

Table 2: Taxonomy for FL frameworks and salient features.

Category	Sample overlap	Feature overlap	Dataset partition	Applicable scenario
HFL	Less	More	Sample dimension	Same features but different samples
VFL	More	Less	Feature dimension	Same samples but different features
FTL	Less	Less	Without dividing	Different features and samples

devices without exporting sensitive user data [25]. In this case, different users (sample space) enter input texts (feature space) on their smartphones, and the cloud server aggregates local updates to create a federated language model for next-word predictions. Note that HFL can be realized with various machine learning algorithms without changing the central structure of the algorithm.

(2) Vertical Federated Learning (VFL). Contrary to HFL, VFL enables collaborative learning over participants with the same sample space and different data features [26]. It can be called feature-partitioned FL since each participant shares a common set of samples to divide data in feature dimension [6]. For instance, banks and retail companies may have the same users in a city. More specifically, they keep records of payments and purchases with the same user, respectively. Therefore, VFL can leverage purchase records from retail companies to train a prediction model for credit rating. In VFL, overlapping data samples of multiple participants conduct sample alignment through an encryption approach to build a model.

(3) Federated Transfer Learning (FTL). FTL considers scenarios where participants do not have many overlapping samples and features to overcome the data scarcity problem [27]. Inspired by transfer learning, FTL allows complementary data to communicate across fields in a federated manner that uses the source domain with rich labels to build a practical model for the target domain [28, 29]. In realistic scenarios, the server can indirectly take advantage of more information from multiple participants by using FTL to get a more generalized global model. On the other hand, participants can leverage the global model to get a more personalized local model.

2.2.3. Network Structure. FL framework can be deployed in collaborative learning environments with distinct network topologies. According to the underlying architecture, it can be divided into centralized FL and decentralized FL. Figure 3 visualizes the FL model of two network structures.

(1) Centralized Federated Learning (CFL). CFL is a general architecture that takes massive decentralized participants to connect with a centralized server (i.e., aggregator). As shown in Figure 3(a), the participants download the incipient parameters and learn a model with their training data, respectively, e.g., via Federated Averaging (FedAvg) [30], and transfer local model updates to the aggregator. Finally, each participant obtains a shared model as well as a personalized model. In CFL, the aggregator is ordinarily regarded as a crucial component for planning and coordinating participants to accomplish learning tasks, which is prone to the problem of single-point-of-failure.

(2) Decentralized Federated Learning (DFL). DFL does not require a centralized server to collect all training data, nor does it need an aggregator that maintains the global model on the network by aggregating the model updates of all participants, as shown in Figure 3(b). DFL is designed to replace CFL fully or partially when the aggregator is unavailable, or the topology is highly scalable [31, 32]. In DFL, each participant distributed on a network that only communicates with neighbor nodes can be selected as an aggregator in a polling manner. However, given the system security, this is more relaxed and inserts a backdoor to increase the risks of potential threats accordingly. Some peer-to-peer schemes are presented based on verified strategies such as belief [33], consensus mechanisms [34] to perform model learning.

Regardless of the FL type, they have a common feature to push the model training to the device by requiring participants to share local model parameters instead of raw data. Unfortunately, FL protocol does not always provide sufficient security and privacy guarantees that malicious attacks and data leakage yet exist. Next, we present security and privacy issues in the following section.

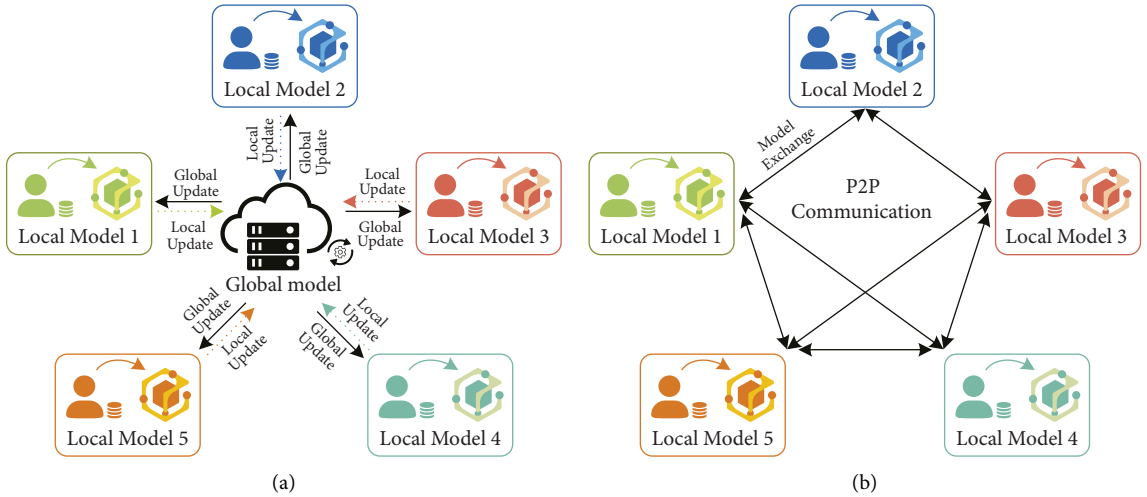

FIGURE 3: FL model for taxonomy based on network structure. (a) CFL involves a server (i.e., aggregator), which trains the global model through the iterative exchange of global aggregation updates and local model updates between the aggregator and participants. (b) DFL carries out collaborative learning by establishing a model parameter exchange between two participants in point-to-point communication.

3. Security Threats and Privacy Risks

FL gives a privacy-aware learning framework that does not require sharing the original training data and allows each participant to enter or leave the training process freely. However, the distributed strategy with frequent data interaction for collaborative learning is vulnerable to malicious attacks due to the open training environment and exposed model parameters [35]. After introducing the FL concept and its distinct categories, we focus on security and privacy issues in this section.

The attack types in the FL system can be commonly divided into model threats for robustness and data leakage for privacy. Specifically, model threat attacks that violate FL system integrity lead to incorrect predictions. In addition, attacks on model availability induce misclassification of the FL system, which is broader and more damaging than integrity breaches. Data leakage attacks that violate training data confidentiality give rise to the disclosure of privacy data, which is a major privacy concern for illegally obtaining sensitive information.

3.1. Threat Model. In the FL scenario, various attacks can be carried out by the untrusted aggregator and malicious participants. At present, research works on the threat model involve the semi-honest adversary and malicious adversary.

3.1.1. Semi-honest Adversary Model. In the semi-honest adversary setting (resp. honest-but-curious), an attacker is considered to passively attempt to infer sensitive information about a specific participant through exposed parameters while not deviating from the protocol [36]. It is worth noting that passive attacks are performed by observing results calculated from the aggregator instead of obtaining training data or gradients directly from participants. For security solutions of FL, how the aggregator provides secure aggregation upon received model updates is always the focus of researchers.

3.1.2. Malicious Adversary Model. In the malicious adversary setting, an attacker can actively perform arbitrary attacks in an attempt to steal sensitive information from global model parameters shared during the training process. Moreover, the malicious attacker can also conduct devastating attacks on the global model by deviating from the protocol or tampering with data.

Attacks on the FL system can occur at any stage that involves global model training and inference. In general, the goal of attacks in the training phase is to compromise the global model, reveal training samples, or both. The typical attack methods principally consist of a series of poisoning attacks, inference attacks, free-riding attacks, generative adversarial networks (GAN) attacks, etc. Intuitively, data poisoning can undermine the integrity of training data, while model poisoning can violate the target model availability. In practice, the ultimate goal of both poisoning attacks is to reduce the global model quality. Inference attacks serve as a major privacy leakage concern of FL, e.g., user-level training samples can be revealed through inferring gradients. Free-riding attacks can benefit from the global model without providing any actual samples. In the model inference stage, commonly used attack methods include evasive attacks that produce wrong output and model reversal attacks that violate data privacy. Note that the effectiveness of these attacks depends on the available model information to the attacker. In this survey, we focus on attacks during the training phase specifically.

Next, various attacks on FL are summarized from the perspective of different computing parties. According to the FL system category in Section 2, computing parties in the learning process consist of the aggregator and

participant, both of whom can adopt certain strategies to perform effective attacks.

3.2. Attacks by Aggregator. The central intuition is that a malicious (or at least honest-but-curious) aggregator aims to reveal record-level training samples of a specific target from received model updates while pretending to be a benign server to provide computing services for aggregation. It is worth mentioning that the aggregator potentially has power attack capabilities with a global view of all participants' model updates, even directly determining the participant involved in each training iteration. In fact, the adversary aggregator attempts to completely steal record-level samples by updating the victim's view from the aggregated model. Furthermore, an adversary aggregator can recover the original training data without any prior knowledge. More precisely, the shared local model is updated according to those private data, whose pattern is encoded into the model parameters. Therefore, if a corresponding decoder could be constructed, private data or statistics will be recovered inversely. In addition, such privacy violations may occur in the aggregator for CFL or any neighbors in DFL, which challenges the fundamental privacy assumption.

The adversary aggregator can passively analyze periodic updates of participants or actively isolate the shared model trained by the victim to perform more powerful attacks. Active attacks by the aggregator can be divided into three categories based on attack strength.

(i) Gradient Ascent: The first is to apply the gradient ascent strategy to the local model of the attack target, which triggers the loss minimization of the target model. In contrast, for nonattack targets, the model does not change the gradients to keep the quality of the main task. This approach also applies to attacks performed by participants.

(ii) Isolating: The second is to isolate the attack target by creating a local view of the training process rather than providing the attack target with an aggregated model that induces the target's local model to store more of one's own training data.

(iii) Isolating Gradient Ascent: The third attack serves as the gradient ascent algorithm combined with the isolating strategy.

Unlike participants who perform attacks with good stealth, the malicious aggregator cannot perform poisoning attacks, given that updating the aggregation model through mislabeled training samples would significantly influence the global model accuracy. Due to the powerful attack capabilities of the aggregator, our discussion of aggregator attacks primarily focuses on the ways of stealing record-level training data, which can occur in the following manner.

3.2.1. Deep Leakage from Gradients. Gradient Leakage attacks, e.g., deep leakage from gradients (DLG), refer to one of the critical privacy threats in both CFL and DFL systems. The DLG attacks serve as gradient-based feature attacks that can extract ground-truth samples from shared gradients. By using DLG attacks, the aggregator of CFL can steal the private data of all participants, while any participant in DFL can obtain the training samples of its neighboring nodes.

Given the model architecture and weights that are usually shared in most FL settings, dummy samples can be used to calculate dummy gradients on intermediate local models. Therefore, as shown in Figure 4, the adversarial aggregator starts from an initial random sample and iteratively updates its dummy inputs and corresponding labels to minimize the distance between the dummy gradient and the victim. When the gradient distance loss optimization is completed, the constructed dummy data also converge to the victim's training samples with high confidence.

DLG attacks have shown powerful attack performance over both natural language processing and image classification tasks [37–42]. However, it requires the model structure to be twice differentiable, which is a challenge for neural networks of ReLU units with discontinuous high-order derivatives.

3.2.2. Generative Adversarial Networks Attacks. The state-of-the-art generative adversarial networks (GAN) method is utilized for adversary aggregator attacks, which can invisibly reconstruct the victim's participant-level samples through discrimination of the participant's identity. The GAN attacks from the aggregator are primarily inspired by the decoder. Model updates calculated based on participants' local private data are encoded into the model parameters. In addition, the aggregator can obtain model updates directly from the victim. Therefore, once the GAN generator for a victim is constructed, the corresponding private data would be recovered in reverse. More specifically, the adversary aggregator aims to reconstruct the training samples of a specific target, which pretends to be a benign server to provide computing services. After obtaining local updates of all participants, the adversary aggregator adds local updates of the attack target to the current global model while generating a copy of the local updates. Finally, local updates of the attack target are used to reconstruct the corresponding data sample through GAN. It is worth noting that the adversary aggregator trains GAN to achieve stealth attacks without modifying the shared model.

To be effectively resilient against GAN attacks, participants can transmit model updates via an anonymity network (e.g., Tor) to safeguard their identities in the FL setting, which increases the difficulty for an adversary aggregator to recognize participant-level privacy. However, the adversary aggregator can correlate data representatives of different participants to re-identify anonymized model updates. More specifically, trained GAN combines identification and verification models to measure the similarity of two representatives. In this way, FL learns a discriminative embedded class representative, which helps to match more accurately. A novel GAN framework is proposed to explore an adversary aggregator, which utilizes a multi-task discriminator

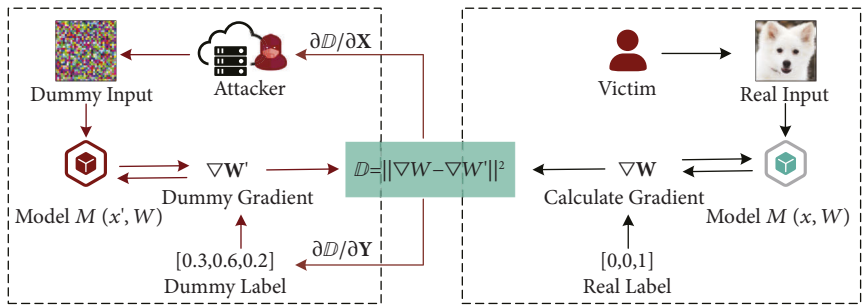

Figure 4: Overview of DLG attacks against FL based on an image classification task. The adversary aggregator constantly updates its dummy inputs and labels to minimize gradient distance from the calculated gradient of a victim.

to distinguish input samples to calculate local updates of the attack target [43, 44]. The adversary aggregator measures the similarity of associated client representatives from periodic model updates to identify anonymized updates. Figure 5 illustrates the GAN-based attack framework from the aggregator. GAN attacks are considered high impact and priority threats since the GAN-based generative model cannot be easily identified.

It is inherently a stronger threat that the adversary aggregator precisely recovers raw training samples from a specific participant. However, when an adversary aggregator trains a GAN, the participants' identity requires periodic updates. Therefore, the dynamic participation of participants with frequent drop-in and drop-out in FL would violate the attack performance, especially under the non-IID assumption of data distribution. In addition, an auxiliary dataset needed for training GAN is also a limitation.

3.3. Attacks by Participant. Considering that the aggregator in real scenarios is commonly credible, more challenging attacks in FL come from an attacker who pretends to be a benign participant to carry out various attacks. Obviously, an adversary participant can only access the global model parameters instead of the victim's local model when performing attacks. Unlike adversary aggregator attacks that solely attempt to steal victim samples, adversary participants can compromise global model availability and training data confidentiality due to their learning process without supervision. Next, we discuss various threats introduced by participants.

3.3.1. Poisoning Attacks. Poisoning attacks are considered severe security threats that increase the risk of the hypothesis produced to induce the FL system more likely to fail on a particular instance. Attackers can implicitly influence local training samples to mislead learning model output by embedding crafted samples, submitting specific gradient updates, or both. Depending on the goal of attacks, poisoning attacks can be divided into objective-driven data poisoning and model-targeted model poisoning. Specifically, data poisoning occurs in the data gathering phase, while model poisoning occurs in the model learning phase. Both poisoning methods attempt to tamper with the global model

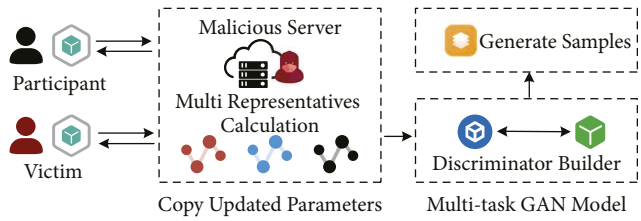

Figure 5: Overview of GAN attacks from the aggregator. On the adversary aggregator, a discriminator and generator are trained based on local updates from the victim.

to reduce global model accuracy. Figure 6 illustrates the two poisoning attacks at different stages.

(1) Data Poisoning. Objective-driven data poisoning has a specific attack objective, which enforces the model to perform well overall while degrading the victim model's accuracy. Through injecting specially crafted training samples, data poisoning manipulates the distribution of training data to affect learning model results, which compromises the availability and integrity of the global model. The local optimization strategy based on gradients is generally used to construct poisoning points [45, 46]. By defining test loss related to attacker objective, gradient-based attacks can modify candidate poisoning points iteratively in the dataset.

Several researchers present the clean label approach to replace the particular local datasets during model training, which involves correctly labeled training samples that do not reduce the performance of nontarget samples [47, 48]. label-flipping [49] is a typical approach of data poisoning attacks, which changes the labels of honest training samples to another class while retaining data features unchanged. It is more accessible to perform label-flipping attacks because no technical experience is required. An attack method of gradient ascent strategy is proposed, in which the gradient calculation is based on the properties of SVM's optimal solution, and the core processing can significantly increase the test error of the classifier [50].

Another type of attack objective increases the probability of bad property for the final attack hypothesis, while the hypothesis has a negligible chance of happening without attacks [51]. These attacks are not feasible in model-oriented attacks since there are no known means to build a target

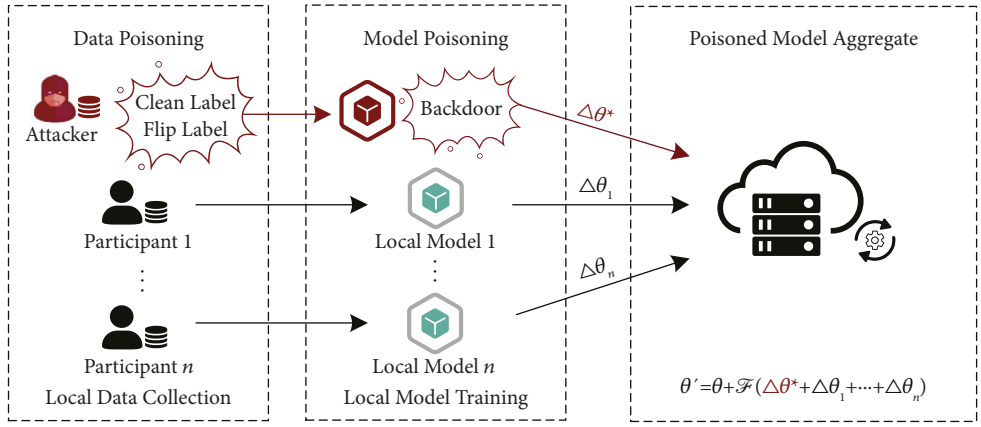

FIGURE 6: Overview of the poisoning attacks against FL. The attacker pretends to be a benign participant, and shares crafted training data or deliberately tainted model updates to the aggregator.

model with the required properties. In addition, the attacker manipulates the training dataset to perform the data poisoning under its control, which is ineffective against the Byzantine-robust FL that applies the Krum [52], Bulyan [53], trimmed mean, and median aggregation rules.

(2) Model Poisoning. Model-targeted attacks result in a victim model as close as possible to a specific model in mind that satisfies some attacker objectives [54]. Model poisoning attacks are carried out by transferring incorrect update parameters to the aggregator or inserting a concealed backdoor into the final model, which disrupts the secure aggregation process more effectively than data poisoning [55]. By sending corrupted model updates, the attacker affects the convergence direction of global model parameters and slows down the convergence speed of the loss function.

In the semi-honest threat model, the deviated model due to incorrect update parameters can help identify model poisoning. The alternate minimization strategy can be used to improve stealth and evasion detection for the attacker [56]. However, this strategy performs model poisoning based on the assumption of the IID dataset, which is inconsistent with realistic scenarios. To provide robustness against Byzantine failures of certain participants, Byzantine-robust FL provides multiple aggregation rules (e.g., Krum, Bulyan), which can asymptotically limit the error rates of the global model under certain assumptions of the loss function. Nevertheless, they are vulnerable to new local model poisoning attacks formulated as an optimization problem. Aiming at the Byzantine-robust FL attacks, a local model parameter optimization approach was introduced in Reference [57]. The attacker manipulates the local model parameters of the controlled terminal and crafts the local model by optimizing directed deviation in each iteration, which significantly reduces the accuracy of the target model. Based on this work, another general framework suitable for the FL model poisoning was also considered in Reference [58]. The attacker calculates the malicious disturbance along with participant update parameter information and server aggregation algorithm. The method generates a unit vector in contrast to the optimal aggregation direction and disturbs the optimal aggregation in the malicious direction to calculate its model update.

This poisoning update can be generated by injecting a concealed backdoor, and even a single attacker can introduce the backdoor into the final model, which corrupts the trained model performance [59]. Moreover, the performance of model poisoning primarily depends on the fraction of attackers and the task complication. Multiple attackers can collude to carry out distributed poisoning attacks to improve the accuracy and effectiveness of backdoor attacks, which may intuitively lead to enormous disasters. Obviously, the attack success rate is linearly proportional to the number of poisoned samples. When the number of poisoned samples does not change, the attack success rate increases with the increase in the number of attackers [60].

In summary, data poisoning is a subset of model attacks since data poisoning ultimately changes model parameters that are sent incorrectly in any given iteration. Generally, in order not to be identified by participants with the shared model, poisoning attacks do not significantly alter the predicted results of other categories. In addition, distributed poisoning attacks colluded by multiple attackers have become a direction of interest to researchers because of their greater destructiveness.

3.3.2. Inference Attacks. While FL significantly reduces users' concerns about uncontrollable data cloud processing, private training data can still be disclosed through adversary inference attacks. Inference attacks refer to attackers inferring training data from revealed model updates during the FL learning process. In fact, both the honest-but-curious aggregator and the participant can perform inference attacks on the training data of other participants. In this section, we focus on effective inferences initiated by adversary participants. A brief illustration of inference attacks against FL is provided in Figure 7. Inference attacks can be divided into membership inference attacks (MIA) and property inference attacks (PIA) based on the correlation between extracted information and training tasks.

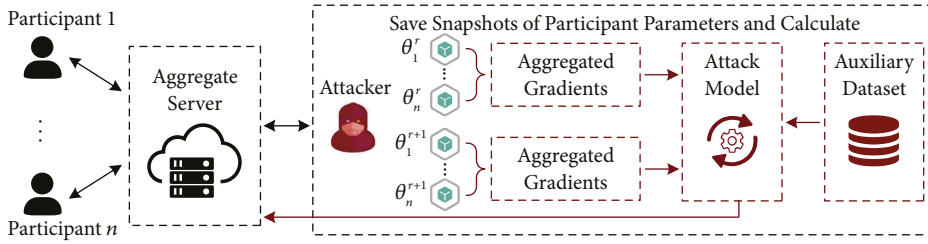

Figure 7: Overview of inference attacks in FL. The attacker saves the snapshots of the aggregated model parameters in each round and performs inference attacks by employing the difference between the continuous snapshots.

(1) Membership Inference Attacks (MIA). As a privacy violation, the MIA judges whether a specific data point is used to model training while maintaining the high quality of main tasks in FL [61]. The aggregator can directly receive model updates based on local data training in any given iteration. Therefore, an adversary aggregator can easily infer whether a particular data point is involved in the training process through server-based MIA. Most researchers pay attention to client-based MIA because it complies with stronger security assumptions (i.e., trusted aggregator) and higher complexity of privacy attacks. For client-based MIA, given that each training sample can affect the gradients of the loss function recognizably, i.e., an attacker takes advantage of the Stochastic Gradient Descent algorithm (SGD) to judge whether a targeted sample exists from other participants.

MIA can be divided into passive and active ones [62]. In passive MIA, the attacker only observes the aggregated model updates and extracts information about the union of the training dataset of all other participants without influencing the FL training process, such as the shadow model that mimics the behavior of the target model. The active MIA injects adversarial model updates to perform more powerful inference attacks on target training data. In this case, the attacker can implement a gradient ascent attack to update the model parameters on the contrary direction of the loss gradient, which forces the target model to minimize the loss by descending in the direction of its local model's gradient.

(2) Property Inference Attacks (PIA). In the context of FL, PIA tries to extract valuable properties that other participants do not intend to share, and are even uncorrelated to the main task. Those properties hold in specific subsets of the participants rather than global properties of the training data. For instance, when Alice's photos are used to train a gender classifier, the attacker may infer whether a person in Alice's photos wears glasses through the PIA. In particular, it can reveal changes about when properties emerge and disappear in training data during learning, e.g., when a person first appears in photos during the training of a gender classifier [63].

The active and passive attackers can perform PIA, both of which require auxiliary data generation and attack model training. The passive attacker with auxiliary data similar to the training data distribution of participants trains the specific property classifier to infer information [64]. The auxiliary data are correctly labeled for the attacker desired properties and main task labels. The active attacker need not strictly follow the main task's procedure, which aims to identify a set of participants with a specific property by modifying the model updates to separate data with (or without) properties.

Recent research shows that even with the employment of secure aggregation schemes, PIA can still infer a subset of participants with desired properties through aggregated global model while maintaining the high quality of the main task [65]. However, the attack relies on the assumption that the attacker has white-box knowledge of the learning model.

In general, both MIA and PIA reveal some information about the victim but have certain limitations, respectively. Essentially, member inference requires an existing data sample for attacks, which is challenging to get when the input data are not text, e.g., image or voice. While property inference is relaxed that requires only a label, attack results reduce the scope and do not provide record-level identification.

3.3.3. Model Inversion Attacks. Model inversion attacks exploit the real-time nature of the training process to reconstruct the victim's prototype samples in which access to the model is abused to infer information about the training data [66]. Generative Adversarial Networks (GAN) commonly learn private training data for model inversion based on the gradient and confidence of predicted results [67]. Note that the predictive power of the model and its robustness to inversion attacks are negatively correlated, as highly predictive models are able to establish strong correlations between features and labels, which is perfectly consistent with exploitation by malicious attackers. Therefore, model inversion attacks are also called generative model inversion attacks, which can invert complex models with high success rates, e.g., DNN [68]. As aforementioned, poisoning attacks and inference attacks combined with GAN attacks can construct more powerful attacks to threaten federated learning security. By using GAN, the mode inversion can simulate the training samples almost identical to the original data of the victim as the samples reconstructed expect to meet the distribution of the victim's data. Figure 8 illustrates the framework of GAN-based model inversion attacks.

The user-side GAN attacker exploits the shared model as the discriminator to train a GAN, which induces the victim to release more sensitive information [69, 70]. With the continuous injection of mislabeled training samples, the victim has to provide more training data to distinguish

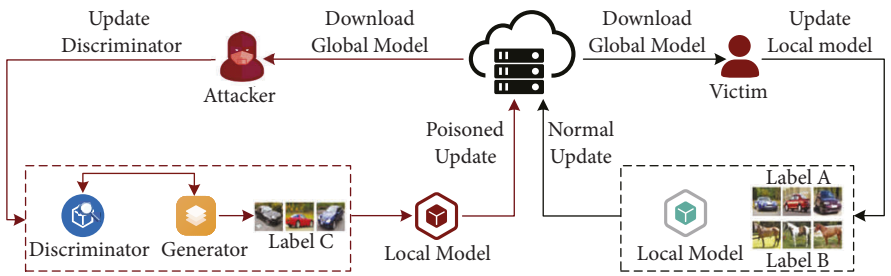

Figure 8: By using the client-side GAN attacks, the attacker can reconstruct sensitive information from the victim.

the correct and wrong about it, which has significant benefits for the iterative improvement of the discriminator. The power of user-side GAN attacks lies in that an attacker can invisibly complete all malicious acts and simulate as a normal participant to follow the FL protocol successfully without violating the victim directly. However, the simulated samples generated by model inversion attacks using GAN are general samples with characterized class properties rather than accurate samples from the victim. More specifically, GAN-based attacks require beforehand class labels to be known and can only perform well on simple samples, such as MNIST with clean backgrounds. It is worth noting that the performance of the GAN-based model inversion introduced by the adversary participant would decrease when FL is attacked since the adversarial influence becomes weaker after aggregation.

3.3.4. Watermark Attacks. By exploiting the perplexity of the language model, watermark attacks extract text records of training data even without access to the victim's local model in federated natural language processing (NLP) tasks [71, 72]. With a character sequence as input, the FL model that aggregates the local model from the participants' private records can predict the next word. Unlike image classification tasks, the training data of NLP tasks are independent records composed of sequential text without class representatives. Note that most attacks performed by injecting data (e.g., data poisoning, backdoor attacks) compromise the global model robustness, while watermark attacks result in privacy leakage. Specifically, the attacker injects the watermark into the victim's dataset rather than accessing it and then extracts the interest records by comparing the exposure rates' changes between the watermark and potential records. The distinctive watermark attacks performed in an invisible fashion do not affect the global model's overall performance.

To explore the privacy leakage of the NLP task, a unified framework for record-level privacy attacks was introduced in [73]. By tracking the victim's training footprints exposed in the asynchronous aggregation process, the victim's text records are accurately extracted with the calculated exposure rate. The watermark is injected as random data records to avoid detection. Therefore, in this way, watermark attacks with perfect invisibility and poses a challenge to intrusion detection identification.

3.3.5. Free-Riding Attacks. The free-riding attacks serve as dissimulated participation in learning to benefit from the global model but do not contribute to the training process [76]. It is critical for sensitive applications of FL to identify this attack type when there is training data scarcity. Given data privacy concerns and computational overhead in the local training process, the adversary participant, i.e., free-rider, refuses to utilize its training dataset for honest local model training. It is more serious that the adversary free-rider does not even have local training data on learning tasks, and its purpose is only to benefit from the global model for free. To avoid being identified, typical free-riding attacks send generated false weight reports (e.g., additive Gaussian noise with zero mean and one standard deviation based on stochastic updates) to the aggregator. Besides, the free-riding attacks also need to ensure that the whole training process converges to the desired goals represented by honest participants. To bring the result closer to the aggregated model that only considers honest participants, the total number of samples declared by free-rider will be smaller. Moreover, to synchronize the evolution of more reasonable parameters in FL, the disturbance of free-riding should follow time-varying asymptotic behavior.

In summary, FL faces the challenges of unique security threats and privacy issues compared with traditional machine learning due to its distributed nature. The various attacks show that FL has weaknesses in both robustness and privacy. Depending on different computing parties, we list the various attack approaches of security threats in taxonomy Table 3 to summarize each attack approach of severity, strategy, and description. Compared with various targets of attacks initiated by participants, the adversary aggregator exploits DLG, GAN, etc., to perform attacks solely for the goal of stealing private data. However, adversary aggregator attacks may produce serious record-level data leakage. Adversary participants are allowed to perform attacks from both dimensions of security and privacy. The goal of poisoning attacks is to decrease the availability of the shared model, which leads to system classification errors or prediction failures. Inference attacks focus on the confidentiality violation of training data to induce the disclosure of sensitive information. Model inversion attacks based on GAN are more invisible and pose a more significant threat to FL systems. In the following section, we focus on defense approaches for privacy and security issues in FL.

Table 3: Taxonomy of adversary attacks in the FL system.

Attack types	Attacker	Severity	Target	Approach	Strategy	Description	Ref.
DLG	Aggregator	High	Record-level privacy	Calculate partial derivatives based on gradient distance	Isolating	The attacker creates a local view of the victim's training process, dummy inputs, and labels to minimize the gradient's distance.	[38, 42]
GAN	Aggregator	High	Record-level privacy	Incorporate GAN with a multi-task discriminator	GAN	The attacker correlates data representatives of different participants to identify anonymous model updates, and trains the GAN by fusing the recognition and verification models.	[43, 44]
Data poisoning	Participant	Medium	Specific objective	Gradient-based local optimization	Clean-label, label-flipping	Clean-label attacks can only perturb features of the training sample. Label-flipping attacks are allowed to change labels.	[46, 74]
Model poisoning	Participant	High	Desired model	Hand-crafted heuristic, bi-level optimization	Embed backdoor	The inner level training on a poisoned dataset and the outer level corresponds to updating these poisons to achieve the desired model.	[54, 74, 75]
MIA	Both	Medium	Participant privacy	Determine whether a specific data point is used to model training	Gradient ascent attack	The attacker performs a gradient ascent attack to update model parameters in the opposite direction of the loss gradient.	[61, 62]
PIA	Both	Medium	Participant privacy	Extract valuable properties uncorrelated with the main task	Separate data	The attacker identifies a set of participants with a specific property by modifying model updates to separate data with (or without) property.	[63, 65]
Model inversion	Participant	High	Record-level privacy	Exploit the shared model as the discriminator	GAN	The attacker utilizes the real-time nature of the training process to reconstruct the victim's prototype samples.	[69, 70]
Watermark attacks	Participant	High	Record-level privacy	Extract the interest records by comparing the exposure rates' changes	Watermark	The attacker tracks the victim's training footprint exposed during the asynchronous aggregation process.	[72, 73]
Free-riding	Participant	Low	Shared model	Send generated false weight reports	Time-varying noise	The attacker utilizes additive Gaussian noise with zero mean and one standard deviation based on stochastic updates.	[76, 77]

4. Security Defense and Privacy-Preserving

FL system has some weak phases about the security aspect, e.g., model distribution, interactive training, and parameter aggregation. In this section, we discuss a series of security defense approaches of FL with different techniques, including Differential Privacy, Secure Multi-Party Computation, Homomorphic Encryption, Trusted Execution Environments, blockchain, etc. Table 4 summarizes the defense approaches and associated features in this section.

4.1. Differential Privacy. Differential Privacy (DP) adopts obfuscated mechanisms with perturbed noise without revealing privacy-sensitive data at individual levels to realize lightweight privacy-preserving. It can be quantified by privacy loss parameters (ϵ, δ), where ϵ is the privacy budget, smaller ϵ tightens the standard for privacy protection, and lower δ signifies greater confidence. More formally, given a randomized algorithm $\mathscr{A}: \mathscr{D} \longrightarrow \mathscr{R}$ with domain $\mathscr{D}$ and range $\mathscr{R}$, for any two adjacent datasets $\mathscr{D}, \mathscr{D}'$ and any set of outcomes $\mathscr{S} \subseteq \mathscr{R}$, the $\mathscr{A}$ is said to be (ϵ, δ)-differential privacy if the following equation is satisfied:

$$\mathbb{P}(\mathscr{A}(\mathscr{D}) \in \mathscr{S}) \leq e^{\epsilon}\mathbb{P}(\mathscr{A}(\mathscr{D}') \in \mathscr{S}) + \delta. \tag{4}$$

Intuitively, it is difficult for a potential attacker to infer whether a specific data point has been added to the input $\mathscr{D}$ based on the change in the output distribution. Therefore, the information of any single data point is protected. By using the randomized algorithm output, DP provides statistical guarantees to defend against the attacker from stealing data during the learning process.

Table 4: Taxonomy of defense techniques to enhance security in FL system.

Threat model	Technique	Description	DGL	GAN	MIA	PIA	Data poisoning	Model poisoning	Model inversion	Free-riding	Cost	Ref.
Semi-honest	DP	Add random noise to uploaded parameters	√	√	√	×	√	√	√	×	Accuracy loss due to added noise	[37], [39], [78–81]
	SMC	Add a security aggregation protocol to the FL system	√	×	√	√	×	×	√	×	Increase communication overhead	[82–85]
	HE	Encrypt uploaded local model updates	√	√	√	√	×	×	√	×	Efficiency loss due to encryption	[86–89]
	TEEs	Put parameter aggregation and local training into TEEs units	×	×	√	√	×	×	√	×	Enhance energy consumption in participant side	[90–92]
Malicious	AD	Exploit specific judgment algorithms to identify outliers	×	×	×	×	√	√	×	√	Training time for recognition algorithm	[52, 57, 93, 94]
	Blockchain	Offer data confidentiality, computation auditability, and incentive mechanism	×	×	×	×	√	√	×	√	Induce communication delays with data exchange	[95–100]

4.1.1. Noise Property. There are several factors that affect DP noise property, such as data types and processing capabilities. The continuous noise for numerical data is particularly appealing, while discrete noise for nominal data is generally adopted. Perturbed noise is less related to the noise types (e.g., Gaussian noise, Laplace noise) on the premise that acceptable utility loss can produce the indistinguishability of protected objects. In FL, DP adds noise to the gradient to provide participant-level or record-level training data resilience, thereby effectively avoiding inverse data retrieval to defend against *DLG attacks* and *MIA*. Based on the location and whether the noise disturbance is performed on per-participant gradient updates after completing local training or per-example gradient updates during local training, the noise added to FL can be categorized into three types. (1) Server-side participant-level gradient: per-participant shared model updates intercepted by the adversary aggregator (e.g., *DLG attacks*). (2) Client-side participant gradient: gradient updates after participants complete local training. (3) Client-side sample gradient: the gradient of each training example during local training. Moreover, the results obtained from simulations show that Laplace noise tends to be a slightly better defense than Gaussian noise with the same distribution variance [37].

4.1.2. Central and Local Differential Privacy. DP can be classified into two categories based on the location of perturbed noise added that consist of central differential privacy (CDP) [101, 102] and local differential privacy (LDP) [78]. Figure 9 illustrates that CDP restricts the information learned about a specific participant, while LDP does so for local sample-level training data in a participant's dataset. Specifically, CDP implements noise perturbation on centrally collected results to provide privacy guarantees for the participant-level data, while LDP performs a random perturbed algorithm to add noise locally before sending results to the server by each participant. Interestingly, LDP is implemented at the sample-level to hide the contribution of specific samples in a participant dataset, which allows participants to customize their privacy budget [79]. The noise of LDP can be injected over local update parameters or calculated gradients, e.g., differentially private stochastic gradient descent (DP-SGD) [103]. It is worth noting that gradient noise can refine the privacy budget cost through moments accountant to help researchers evaluate the performance of DP. Compared with LDP, CDP is generally insufficient to protect the participant training data from sample-level gradient leakage [39]. In addition, a limitation of CDP in practice is needed for a trusted server because of noise disturbance after receiving the participant's model update.

4.1.3. Defense against Attacks. DP can effectively mitigate against *poisoning attacks* to increase system robustness. Researches show that CDP is more effective than LDP in reducing the accuracy of *backdoor attacks* while providing better utility.

To prevent inferring uncorrelated properties, Naseri et al. [80] evaluated the DP scheme to mitigate PIA for gender classification on the LFW face dataset. Experimental results demonstrate that LDP is ineffective against PIA since it only provides privacy guarantees at the sample-level but is not directly applicable to defend the leakage of population-level properties. Similarly, CDP is also not feasible in defense against PIA because of significant utility loss. Therefore, both CDP and LDP fail to mitigate PIA effectively.

To defend against *GAN attacks*, Xu et al. [104] presented a gradient-pruning approach to enhance the scalability and stability of data training, which allowed that users could use GAN to generate large amounts of synthetic data without disclosing private data. A new method of FL based on LDP and Paillier homomorphic encryption in a fog computing environment was proposed to defend against GAN attacks [81].

DP commonly introduces noise into the data, which inevitably leads to the loss of model accuracy, and the trade-off between availability and privacy is complicated. In addition, under the assumption of a large number of participants with poor amounts of training data, DP can also give rise to lower accuracy.

4.2. Secure Multi-Party Computation. Secure Multi-Party Computation (SMC) refers to the collaborative calculation of a model or function by multiple participants without revealing their private inputs to other participants for privacy-preserving. Therefore, a secure aggregation protocol constructed by SMC in the FL framework protects participants' security updates [82]. In this protocol, each participant masks local updates through Secret Sharing and pairwise masking, for example, using the weighted average of update vectors from a random subset. In this way, additional random factors would be canceled out when the masked updates are aggregated to the aggregator. Therefore, the aggregator can only learn about the aggregation of participant models rather than any information about an individual model since they are covered by random factors unknown.

4.2.1. Secret Sharing. Secret Sharing refers to an encryption method in which a secret divided by multiple shares is distributed to various parties. When the parties put the shares together, the secret can be recovered. Secret Sharing applies consistency checks to verify correctness to ensure each participant follows protocol. Therefore, the privacy-preserving based on the Secret Sharing mechanism realizes secure aggregation of gradient updates by each participant securely sharing its local update with others [83]. Figure 10 illustrates the model aggregation process by calculating the gradient's average value through secret sharing in DFL. The Secret Sharing protocol used in FL includes additive Secret Sharing and Shamir ones. Specifically, additive Secret Sharing utilizes random numbers from each party to generate corresponding secrets, while Shamir Secret Sharing applies polynomial interpolation with safety in the finite field. Note that although the security level and computation

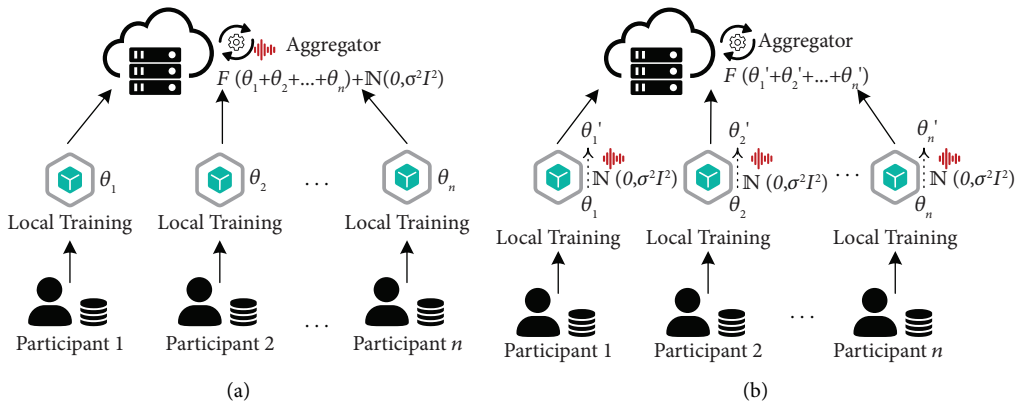

FIGURE 9: Overview of the CDP and LDP in FL. CDP collects model updates to add noise to provide participant-level protection, while LDP performs perturbation locally to restrict learning to samples in the participant dataset. (a) CDP Approach in FL. (b) LDP Approach in FL.

cost of the Shamir Secret Sharing protocol is much higher than additive Secret Sharing, the number of messages exchanged remains the same. Moreover, the dropped participant can be recovered through Secret Sharing to solve offline issues when a participant leaves learning. However, although Secret Sharing can protect model update parameters, it cannot defend against poisoning attacks.

4.2.2. Pairwise Masking. Pairwise Masking is an approach in which mutually distrustful participants utilize Diffie–Hellman key exchange to generate pairwise masks. It has a unique property in that paired masks would cancel out when the masking models of whole participants are aggregated [84]. In fact, when the aggregator attacks a victim for applying Secret Sharing, other honest participants will supply all secret shares to the aggregator to eliminate the interference of model updates masked for the victim, which leads to victim data leakage. As a result, Pairwise Masking can thwart model updates from being revealed when the adversary aggregator reconstructs the participant's perturbation [85]. Unfortunately, with a massive number of participants, the aggregated results are generally insufficient to provide reasonable privacy-preserving for an individual participant. Therefore, Pairwise Masking is generally not suitable for CDFL scenarios.

Generally, the MPC protocol inevitably leads to higher communication overhead. For example, in Secret Sharing, all shares generated by one participant require to interact with other participants. This overhead increases exponentially with the number of participants joining the FL. Therefore, how to design an optimized communication overhead SMC solution for FL is a hot topic for researchers.

4.3. Homomorphic Encryption. Homomorphic Encryption (HE) is a security defense technique based on cryptology, which can perfectly prevent data breaches in theory. This technique does not require access to the plaintext directly since the decrypted operation results of the ciphertext are equal to the plaintext operation results. The homomorphic operation can be expressed as $\mathrm{Enc}(m_1)\oplus\mathrm{Enc}(m_2) = \mathrm{Enc}(m_1 \otimes m_2)$, where an encryption algorithm is stated as *Enc* and its corresponding plaintexts m_1, m_2, $\oplus$, and $\otimes$ represent operators. The encryption operation can be an addition (e.g., Paillier algorithm) or multiplication (e.g., RSA, ElGamal algorithm) operation that belongs to semi-HE, while encryption algorithms that satisfy both addition and multiplication are fully homomorphic encryptions (FHEs). The industrial FL frameworks allow participants to use additive HE to mask local gradient updates to ensure that local updates are not revealed during aggregation. Therefore, the FL process based on HE can directly calculate the encrypted model parameters to ensure the security of model updates. Moreover, the training model accuracy is almost not impaired since HE has no obfuscation and distortion operation. As shown in Figure 11, HE is used in FL to provide a security guarantee that encrypted model updates are transmitted between the aggregator and participants.

In the FL setting, it is essential to consider the high communication cost between the aggregator and participants due to the network bandwidth limitation. In fact, since the data transfer amount of HE operation would be significantly inflated, the introduction of HE into FL would be bound to increase the communication overhead substantially. Moreover, HE may bring large computational overhead to FL and even dominate the training time. To overcome these challenges, it is necessary to optimize the operation of HE and aggregation strategy.

4.3.1. Batch Encryption. To improve the encryption efficiency, a batch of quantized gradients encoded into a long integer was encrypted at one time instead of individual gradient encryption [20]. The batch encryption scheme allows parallel encryption, decryption, and homomorphic operations on multiple plaintexts. Therefore, it can carry out a substantial training acceleration, while the accuracy loss

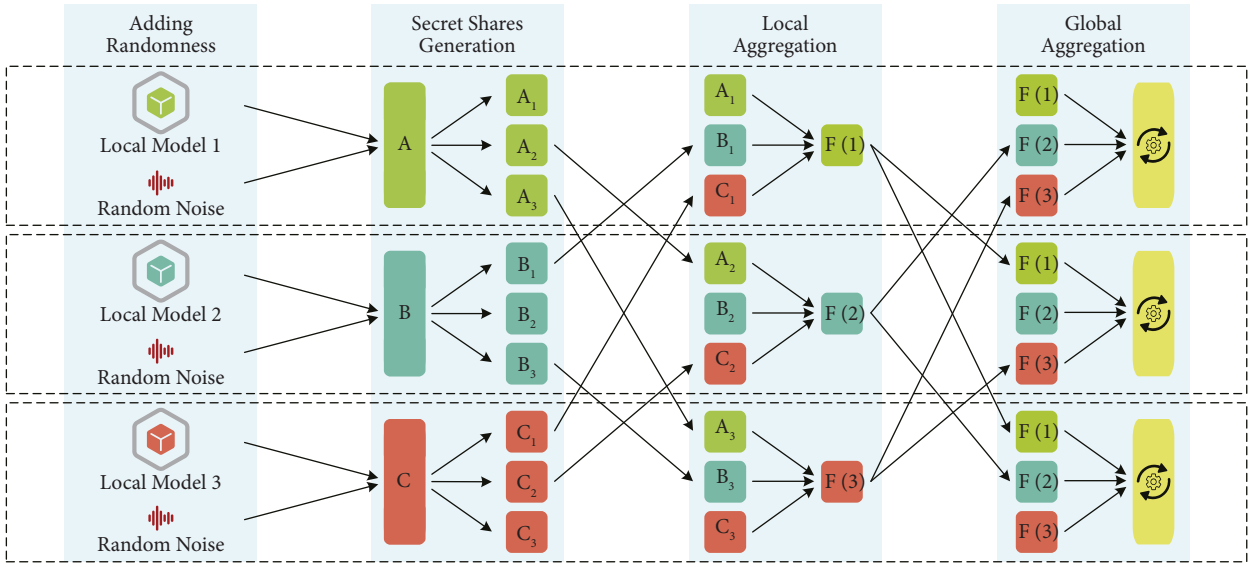

Figure 10: The local model with added randomness is split into multiple parts for other participants. Each participant aggregates all the model shares obtained locally and then cancels out the randomness through global aggregation.

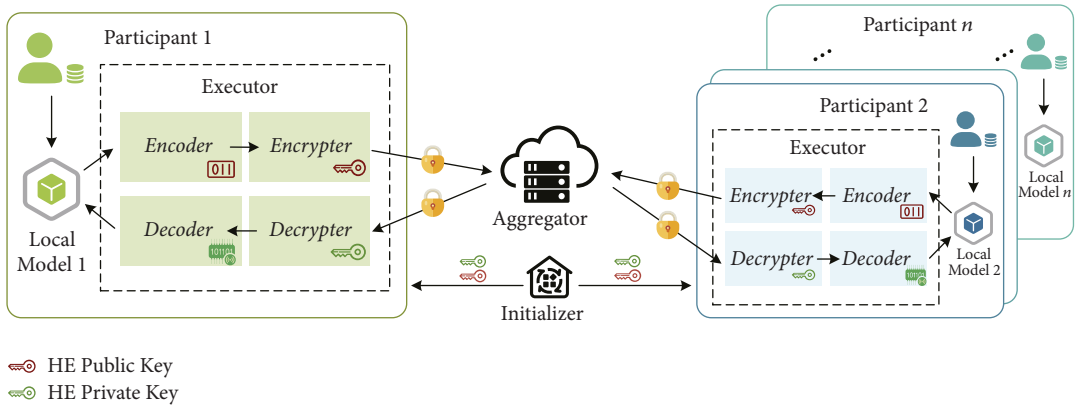

Figure 11: Overview of the HE in FL. The participant who initializes the key pair sends the encrypted local gradient update to the aggregator for ciphertext aggregation. Then, the decrypted aggregation result can be used to update the local model.

caused by quantization errors is acceptable. At present, batch encryption is the most advanced optimization direction for general HE schemes in CSFL. However, batch encryption is not compatible with sparsification techniques. Furthermore, even with the aid of batch encryption, the large communication overhead caused by message inflation remains a concern.

4.3.2. Secure Aggregation Strategy. Additive HE (such as Paillier mask technology) is more utilized to protect the privacy of training datasets and the global model in FL aggregation due to relatively better efficiency than other alternatives. A scheme to defend against model inversion caused by the adversary aggregator and colluding parties was considered in Reference [86], where ElGamal encryption protocol was used to provide provable privacy protection. However, although this scheme effectively mitigates the communication overhead, the lightweight Paillier encryption can be used instead of the ElGamal encryption to enhance learning efficiency. A regression model training method based on Paillier encryption was proposed in Reference [87], which realizes the security training of the noninteractive regression model with gradient descent. During the training process, there was no required interaction between the aggregator and participants. Similarly, based on the idea of modules, a noninteractive FL framework was designed in Reference [88]. The scheme reframes the Paillier by involving the modular addition of random numbers to avoid multiple rounds of data interaction between the aggregator and participants. A privacy-preserving FL scheme based on HE was proposed in Reference [89] while applying asynchronous SGD to neural networks and combining with additive HE to prevent the aggregator from contacting the participant's model gradient plaintext to ensure the global model security.

In summary, most defense schemes based on HE have certain limitations due to the encryption and decryption process. Cloud servers for outsourced computation generally

support limited operations without meeting the HE calculation requirement. In addition, it is not compatible with most CNNs directly that the secure aggregation strategy based on the HE requires the gradient values to be an integer. Finally, given that the HE generally involves a large number of complex operations, this is less suitable for resource-constrained CDFL scenarios.

4.4. Anomaly Detection. In the context of FL, anomaly detection is a defense approach primarily meant to exploit specific judgment algorithms to identify outliers. In this approach, the outliers provide attackers identification associated with attack strategies to defend against *poisoning attacks*. Furthermore, anomaly detection can mitigate the *free-riding attacks* that obtain the final global model without actually providing any data. By using model parameters far away from other local models, an individual Byzantine attacker can prevent the convergence of any traditional average-based aggregation. However, such attacks can be detected by employing simple countermeasures at the server level. Indeed, anomaly detection cannot just identify abnormal poisoning points that exceed a predetermined threshold, as an attacker can ensure that every update is returned within that threshold. Therefore, some anomaly detection schemes have been proposed to deal with Byzantine attacks.

4.4.1. K-Means. The K-means clustering method utilizes Euclidean distance to group parameter updates to distinguish between malicious and benign models. K-means rule selects the local models closest to the barycenter among the proposed local models for grouping. For example, take a local model, which is the minimum sum of the squared distances to other local models. In short, larger clusters are considered benign, while smaller clusters are judged to be malicious [93]. However, this aggregation rule tolerates only an individual Byzantine attacker without defense against collusion attacks.

4.4.2. Krum. The Byzantine resilience scheme, namely Krum, essentially combines the majority-based and distance-square-based outlier removal mechanism to remove the influence of potential attackers [52]. Specifically, when f Byzantine attackers appeared among n participants participating in FL training, the majority-based approach selects the subset with the smallest diameter by looking at the subset of $n - f$ local models. Although this approach is more robust to Byzantine attackers, the exponential computation cost is unacceptable. Combining the majority-based method with the distance-square-based method can achieve a lower time complexity $(O(n^2 \cdot d))$, where d is the dimension of local model due to the linear gradient dimension. In addition, the Krum method to resist collusive Byzantine attackers requires $2f + 2 < n$.

4.4.3. Bulyan. A high-dimensional malicious model may severely affect the Euclidean distance between two local models, which can be maintained close to benign parameters. To address this issue, Bulyan uses a variant of trimmed mean to aggregate local models to optimize the Krum method [94]. Specifically, Bulyan first iteratively selects $n - 2f$ local models in the same way as Krum. Then, Bulyan sorts the i th parameter of the selected local models to find the parameters closest to the median. Finally, the parameters' mean can be computed as ith parameter of the global model. Moreover, Bulyan requires $n \geq 4f + 3$ for robustness guarantee and the convergence of the objective function. In addition, the Bulyan Based on Krum has a nonscalable constraint since Krum calculates the pairwise distance between local models, which would be performed many times in each iteration.

Overall, data poisoning attacks can be mitigated strongly by the Byzantine resilient aggregation. Currently, optimal data poisoning has not been found to circumvent an individual Byzantine resilient aggregation scheme to the best of our knowledge, which is an interesting concern for researchers. Moreover, model poisoning may break the Byzantine resilient aggregation achieved by anomaly detection [57].

4.5. Blockchain. Blockchain has the advantage of data confidentiality and computing auditability to provide secure solutions for input verifiability problems in decentralized FL scenarios. The blockchain network can replace the aggregator to exchange and verify local updates in the FL system to filter out unreliable participants while providing corresponding rewards. Moreover, model updates stored on a tamper-proof distributed ledger allow authorized participants to retrieve it to improve training efficiency.

4.5.1. Coupled Architecture. FL architecture combines the blockchain network to indicate the integration of two networks. As shown in Figure 12, according to the coupling degree and miner role, integrated networks can be categorized into fully coupling, standard coupling, and loosely coupling.

(1) Full Coupling. By deploying blockchain and FL in the same network, the full coupling architecture allows FL participants to play the role of miners in the blockchain and generate new blocks through local model update verification. In other words, every miner on the blockchain has the opportunity to participate in local training and global aggregation, while the aggregator in CFL is entirely replaced by the blockchain, as shown in Figure 12(a). In the full coupling architecture setting, each miner avoids single-point-of-failure through its distributed ledger copy without transferring data to the aggregator, minimizing the risk of privacy breaches [95, 96]. However, blockchain and FL work on the same network, which requires more computational resources to satisfy participants' operations.

(2) Standard Coupling. When blockchain and FL are deployed on separate networks, the architecture in which blockchain miners replace the central aggregator belongs to the standard coupling architecture suitable for DFL, as

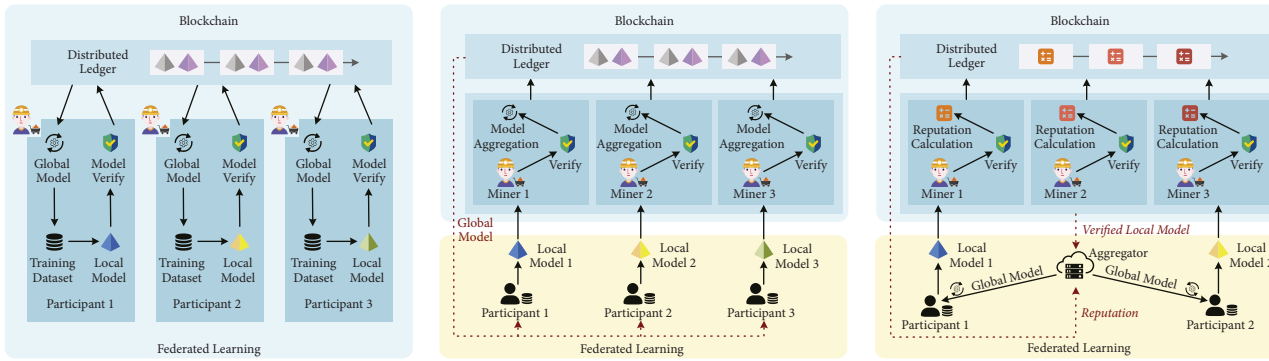

FIGURE 12: Types of coupling architectures with FL and blockchain network (a) Full coupling. (b) Standard coupling. (c) Loose coupling.

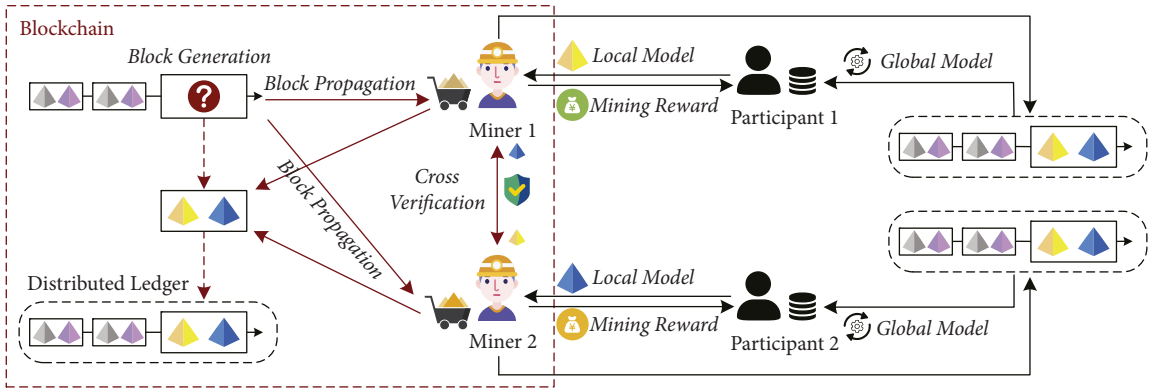

FIGURE 13: Take standard coupling architecture as an example that a workflow of FL is combined with blockchain. Miners exchange and verify local updates of participants while providing corresponding rewards. Validated local model updates and aggregated models are added as new blocks to the distributed ledger.

shown in Figure 12(b). Specifically, participants who perform local training calculate model updates, which are verified and aggregated by blockchain miners [34, 99]. It is worth noting that standard coupling architecture deploys FL and blockchain on different networks to reduce certain communication delays, but coordinated management between networks increases potential risks.

(3) Loose Coupling. The loose coupling architecture of blockchain utilizes verified local model updates to manage the reputation of participants without aggregation operations, as shown in Figure 12(c). Traceable distributed ledger records reputation-related data of participants as a critical criterion to measure the reliability and credibility of participants [97, 98]. A loose coupling architecture compatible with CFL that includes a central aggregator allows participants to retain their data better. Moreover, the reputation management mechanism ensures submitted data quality during model training to improve the global model's accuracy. Since blockchain solely performs model validation and reputation management, FL still relies on a central aggregator, which may lead to a single-point-of-failure.

At present, it is a challenge to point out which coupling architecture is the most reliable. The system designer should evaluate specific application scenarios to select the appropriate coupling architecture.

4.5.2. Consensus Mechanism. The consensus mechanism deals with scenarios in which participants' model updates are exchanged and verified without centralized data. Recently, a series of solutions based on consensus mechanisms have been proposed to find untrusted and unreliable local model updates from the adversary participant, e.g., proof-of-work (PoW) consensus mechanism [34, 100], reputation evaluation mechanism [98], committee consensus mechanism [96], etc. Figure 13 illustrates the consensus mechanism between miners, which introduces the blockchain into the FL training process with the standard coupling architecture. The verification process of local model updates can prevent *poisoning attacks* and *free-riding attacks*, and the results can also be used for reward allocation guidance. In this way, validated model updates and aggregated models would be added as new blocks to the immutable distributed ledger, which provides an accessible platform for qualified participants to download data. Note that each participant calculates shared model updates from the new block to overcome the single-point-of-failure problem. More precisely, the malfunction of each miner only affects the shared model update of its associated

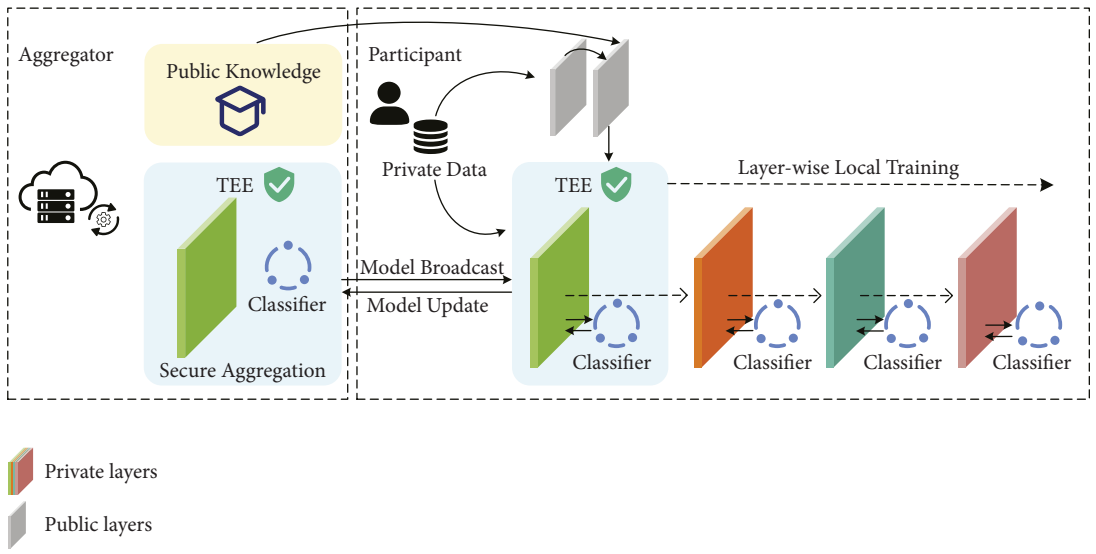

Figure 14: Overview of the architecture of FL with TEEs. The aggregator broadcasts model parameters initialized within its TEEs to participants. Participants utilize decrypted model parameters to calculate local model updates on each layer in TEEs, which are sent to TEEs of aggregator through secure exchange channels.

participants, which can be recovered by other participants associated with the miner in normal operation.

4.5.3. Incentive Mechanism. Blockchain network provides mining rewards proportional to the training sample size, encourages participants to provide reliable local updates, and facilitates more participants' federation [105]. In fact, a participant with a tremendous number of data samples contributes more to global training and is less desirable to federate with other participants without rewards. Moreover, besides data sample size, the sample quality also affects the global model accuracy that adversary participants may use arbitrary model updates to inflate the number of samples and even deploy *free-riding attacks*. The application compares the sample size with its corresponding calculation time to verify truthful local updates to overcome this problem. Therefore, the incentive mechanism encourages participants in distributed learning to provide more effective samples to support the model training process.

Overall, it is costly to realize and maintain miners' operations in the blockchain network, which is unsuitable for resource-constrained devices. Moreover, consensus protocols (e.g., PoW) on the blockchain network also induce a certain degree of information exchange delay and even cause errors. For example, when a certain miner generates its block within the propagation delay of a generated block, other miners may add a second generated block to their local ledger to form an incorrect global model update. In summary, resource allocation and communication delays arising from blockchain networks impede the efficient operation of blockchain in FL, which future research needs to address.

4.6. Trusted Execution Environments. The hardware-based trusted execution environments (TEEs) have become a promising way to prevent attacks on private information by allowing secure data storage and code execution on untrusted devices. Intuitively, TEEs present an isolated environment parallel to the operating system to ensure the confidentiality of loaded data and code. Through the privacy strength based on hardware-backed cryptographic keys stored in TEEs, FL can securely aggregate and iteratively train models to hide model parameters and gradient updates against the attacker [90]. Moreover, the TEEs currently deployed on almost all mobile phones and tablets execute arbitrary code at nearly native speed through the secure memory component, which makes TEEs a suitable candidate for model training with complete privacy protection [91]. However, limited memory due to keeping the Trusted Computing Base (TCB) small has become a bottleneck for TEEs applications.

In general, to overcome constraints posed by the limited memory size of TEEs, the idea of greedy layer-wise training in DNN is considered for training in the trusted area until convergence [92]. Figure 14 illustrates the system architecture of FL with the introduction of TEEs, which allows the aggregator and participants to collaboratively train the DNN model layer by layer within TEEs during learning. Specifically, by using greedy layer training, each DNN layer is trained in TEEs of each participant. In addition, the greedy added layer constructs a new classifier to output predictions that can be personalized by participants and calculates the training loss. Finally, all layers are securely trained in TEEs one by one to protect sensitive data from MIA and PIA. However, TEEs rely on the assumption that the aggregator is trusted, while the adversary aggregator is able to perform Denial-of-Service (DoS) attacks by refusing to forward data to its TEEs to affect system availability.

Because the model training method and network structure of FL are different from traditional machine learning, some threat mitigation techniques in machine

learning may not be applicable to FL, e.g., whether Adversarial Training (AT) with IID dataset in machine learning is also suitable for the FL framework. In addition, as another common defense strategy for machine learning, Data Sanitization filters abnormal training samples before training. However, due to participants' private data inaccessibility, Data Sanitization cannot be performed in FL.

5. Challenges and Future Directions

FL serves as an emerging distributed ML paradigm that allows cross-domain and cross-platform training models without sharing actual datasets. Therefore, FL has obvious advantages over traditional machine learning. However, research on FL is still in its incipient stage, and some key challenges in practice affect the availability and robustness of FL. Some challenges and future research directions are discussed next.

5.1. Utility and Cost Trade-Off. It is inevitable with increased cost to deploy a defense mechanism for checking whether the system is under attack. All defense schemes for improving FL security have a common feature to increase overhead or reduce data utility and model accuracy. Moreover, even with the same defense strategy, various protection levels have different overheads. Therefore, the trade-off between protected levels and related costs has been a hot research topic. The game theory method is a promising solution, which has emerged in the resource management of FL to provide practical ideas. The resource management in the FL system is formulated into a Nash equilibrium problem to minimize the time taken for a calculation under the constraints of energy consumption, communication resources, and training model performance [106]. Moreover, the Stackelberg game model can formulate a market-oriented architecture to analyze and solve the optimal behavior of all participants in the system. To balance the payoff and energy consumption to reveal the optimal utility of all participants, the trade-off between satisfaction and cost of participants is measured through a predetermined pricing scheme to find a unique Stackelberg equilibrium point [107].

5.2. FL in Heterogeneous Scenarios. Currently, FL research on shared model updates and security protection algorithms is usually limited to a homogeneous architecture that shares the same model with all participants. Unfortunately, most training data and model parameters that require collaborative training in realistic scenarios are multi-source and heterogeneous. It is thus essential for exploring learning solutions that efficiently extend FL to collaboratively train models with heterogeneous architecture [108–110]. In addition, whether current defense attacks and privacy-preserving techniques adapt to this paradigm is another interesting research topic.

5.3. Security Threats to DFL. As aforementioned, the DFL is a potential peer-to-peer learning framework suitable for participants to collaborate with learning without an aggregator. In this paradigm, each participant can be selected as an aggregator in a polling manner. It is natural to consider whether the existing threats in centralized FL still apply to this scenario. More importantly, a decentralized FL system may add new attack surfaces. For example, an attacker can effectively compromise the training model by easily inserting a backdoor if an attacker is selected as an aggregator.

5.4. Traceable Detection of Adversary. FL allows participants to join and leave a federation freely during the learning process without participating from the beginning to end, which leads to the complexity of tracking adversary participants. This problem is expected to be solved by introducing a comprehensive application of credit evaluation, smart contracts, and threshold normalization. For example, through multiple rounds of dynamic measurement to determine the reputation of all participants, credit evaluation can follow up and identify traceable violations in FL.

6. Conclusion

Security and privacy considerations of FL mainly focus on two seemingly independent directions, i.e., ensuring the global model's robustness and individual participants' privacy. Therefore, in a secure FL system, the participants' local model updates need to be securely masked without being identified by the aggregator, and the aggregator should identify different model updates and eliminate outliers. In this paper, we firstly conduct a comprehensive study of state-of-the-art potential threats and attack approaches from the perspective of different computing parties. Then, we elaborate on a detailed summary of the defense mechanisms against the above attack approaches. It is worth noting that a combination of these defense mechanisms will achieve a more satisfactory result. Finally, several open challenges in FL which require further in-depth research in this direction are discussed.

Data Availability

The data that support the findings of this study can be obtained from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by National Natural Science Foundation of China (61972304 and 61932015), Science Foundation of the Ministry of Education (MCM20200101), Technical Research Program of Public Security Ministry (2019JSYJA01), Central Government Guides Local Science and Technology Development Found Projects, under grant 216Z0701G, and Shaanxi Innovation Team Project (2018TD-007), and the Fundamental Research Funds for the Central Universities (YJS2212).

References

[1] A. V. Joshi, *Machine Learning and Artificial Intelligence*, Springer, Berlin, Germany, 2020.

[2] H. Fourati, R. Maaloul, and L. Chaari, "A survey of 5g network systems: challenges and machine learning approaches," *International Journal of Machine Learning and Cybernetics*, vol. 12, no. 2, pp. 385–431, 2021.

[3] F. Samie, L. Bauer, and J. Henkel, "From cloud down to things: an overview of machine learning in internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4921–4934, 2019.

[4] E. De Cristofaro, "A critical overview of privacy in machine learning," *IEEE Security & Privacy*, vol. 19, no. 4, p. 27, 2021.

[5] M. Goddard, "The EU General Data Protection Regulation (GDPR): European regulation that has a global impact," *International Journal of Market Research*, vol. 59, no. 6, pp. 703–705, 2017.

[6] K. Cheng, T. Fan, and Y. Jin, "Secureboost: a lossless federated learning framework," *IEEE Intelligent Systems*, vol. 436, p. 34, 2021.

[7] Y. Qian, L. Hu, J. Chen, X. Guan, M. M. Hassan, and A. Alelaiwi, "Privacy-aware service placement for mobile edge computing via federated learning," *Information Sciences*, vol. 505, pp. 562–570, 2019.

[8] W. Shi, S. Zhou, and Z. Niu, "Device scheduling with fast convergence for wireless federated learning," in *Proceedings of the ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, Dublin, Ireland, June 2020.

[9] F. Ang, Li Chen, N. Zhao, Y. Chen, W. Wang, and F. R. Yu, "Robust federated learning with noisy communication," *IEEE Transactions on Communications*, vol. 68, no. 6, pp. 3452–3464, 2020.

[10] 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 Network*, vol. 33, no. 5, pp. 156–165, 2019.

[11] D. Zhu, H. Zhu, X. Liu et al., "CREDO: efficient and privacy-preserving multi-level medical pre-diagnosis based on ml," *Information Sciences*, vol. 514, pp. 244–262, 2020.

[12] G. H. Lee and S. Y. Shin, "Federated learning on clinical benchmark data: performance assessment," *Journal of Medical Internet Research*, vol. 22, no. 10, Article ID e20891, 2020.

[13] G. A. Kaissis, M. R. Makowski, D. Rückert, and R. F. Braren, "Secure, privacy-preserving and federated machine learning in medical imaging," *Nature Machine Intelligence*, vol. 2, no. 6, pp. 305–311, 2020.

[14] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Federated learning for data privacy preservation in vehicular cyber-physical systems," *IEEE Network*, vol. 34, no. 3, pp. 50–56, 2020.

[15] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: passive and active white-box inference attacks against centralized and federated learning," in *Science Progress*, pp. 739–753, IEEE, 2019.

[16] J. Zhang, M. Li, S. Zeng, B. Xie, and D. Zhao, "A survey on security and privacy threats to federated learning," in *Proceedings of the 2021 International Conference on Networking and Network Applications (NaNA)*, pp. 319–326, IEEE, Lijiang, China, October 2021.

[17] B. McMahan, E. Moore, D. Ramage, S. Hampson, and y A. Blaise Agüera, "Communication-efficient learning of deep networks from decentralized data," *Data*, PMLR, vol. 54, pp. 1273–1282, 2017.

[18] N. H. Tran, W. Bao, Y. Albert, and M. N. H. N. Zomaya, "Federated learning over wireless networks: optimization model design and analysis," *INFOCOM*, pp. 1387–1395, IEEE, 2019.

[19] H. H. Yang, Z. Liu, T. Q. S. Quek, and H. V. Poor, "Scheduling policies for federated learning in wireless networks," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 317–333, 2020.

[20] C. Zhang, S. Li, and J. Xia, "Batchcrypt: efficient homomorphic encryption for cross-silo federated learning," *USENIX Annual Technical Conference*, pp. 493–506, USENIX Association, 2020.

[21] Y. Huang, L. Chu, and Z. Zhou, "Personalized cross-silo federated learning on non-iid data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 7865–7873, AAAI Press, Vancouver, Canada, February 2021.

[22] H. Diddee and B. Kansra, "Crosspriv: user privacy preservation model for cross-silo federated software," in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pp. 1370–1372, IEEE, Melbourne, Australia, September 2020.

[23] M. H. U. Rehman, A. M. Dirir, K. Salah, E. Damiani, and D. Svetinovic, "Trustfed: a framework for fair and trustworthy cross-device federated learning in iiot," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 12, pp. 8485–8494, 2021.

[24] 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.

[25] A. Hard, K. Rao, and R. Mathews, "Federated learning for mobile keyboard prediction," *CoRR*, vol. 1811, Article ID 03604, 2018.

[26] R. Nock, S. Hardy, and W. Henecka, "The impact of record linkage on learning from feature partitioned data," in *ICML,* vol. 139, pp. 8216–8226, PMLR, 2021.

[27] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, "A secure federated transfer learning framework," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 70–82, 2020.

[28] D. Gao, Y. Liu, and A. Huang, "Privacy-preserving heterogeneous federated transfer learning," *IEEE BigData*, pp. 2552–2559, IEEE, 2019.

[29] S. Sharma, C. Xing, Y. Liu, and Y. Kang, "Secure and efficient federated transfer learning," *IEEE BigData*, pp. 2569–2576, IEEE, 2019.

[30] A. Reisizadeh, A. Mokhtari, H. Hassani, J. Ali, and R. Pedarsani, "Fedpaq: a communication-efficient federated learning method with periodic averaging and quantization," in *AISTATS,* vol. 108, pp. 2021–2031, PMLR, 2020.

[31] L. Lyu, J. Yu, K. Nandakumar et al., "Towards fair and privacy-preserving federated deep models," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 11, pp. 2524–2541, 2020.

[32] L. Lyu, Y. Li, J. Yu, and X. Ma, "How to democratise and protect AI: fair and differentially private decentralised deep learning," *CoRR*, vol. 2007, Article ID 09370, 2020.

[33] A. Lalitha, T. Javidi, and F. Koushanfar, "Peer-to-peer federated learning on graphs," *CoRR*, vol. 1901, Article ID 11173, 2019.

[34] Y. Qu, L. Gao, T. H. Luan et al., "Decentralized privacy using blockchain-enabled federated learning in fog computing," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5171–5183, 2020.

[35] G. Xu, H. Li, H. Ren, K. Yang, and R. H. Deng, "Data security issues in deep learning: attacks, countermeasures, and opportunities," *IEEE Communications Magazine*, vol. 57, no. 11, pp. 116–122, 2019.

[36] H. Fereidooni, S. Marchal, and M. Miettinen, "Safelearn: secure aggregation for private federated learning," *IEEE Security and Privacy Workshops*, pp. 56–62, IEEE, 2021.

[37] L. Zhu and S. Han, "Deep leakage from gradients," in *Federated Learning - Privacy and Incentive*, pp. 17–31, Springer, Berlin, Germany, 2020.

[38] Bo Zhao, K. R. Mopuri, and H. Bilen, "idlg: improved deep leakage from gradients," *CoRR*, vol. 2001, Article ID 02610, 2020.

[39] W. Wei, L. Liu, Y. Wu, Su Gong, and Arun Iyengar, "Gradient-leakage resilient federated learning," *ICDCS*, pp. 797–807, IEEE, 2021.

[40] G. Jonas, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients - how easy is it to break privacy in federated learning?" *NeurIPS*, vol. 33, pp. 16937–16947, 2020.

[41] W. Wei, L. Liu, and M. Loper, "A framework for evaluating client privacy leakages in federated learning," in *ESORICS,* vol. 12308, pp. 545–566, Springer, 2020.

[42] H. Yin, A. Mallya, and A. Vahdat, "See through gradients: image batch recovery via gradinversion," *CVPR*, pp. 16337–16346, 2021.

[43] M. Song, Z. Wang, Z. Zhang et al., "Analyzing user-level privacy attack against federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2430–2444, 2020.

[44] Z. Wang, M. Song, and Z. Zhang, "Beyond inferring class representatives: user-level privacy leakage from federated learning," in *INFOCOM,* vol. 2019, pp. 2512–2520, IEEE, 2019.

[45] C. Zhu, W. R. Huang, and H. Li, "Transferable clean-label poisoning attacks on deep neural nets," in *ICML,* vol. 97, pp. 7614–7623, PMLR, 2019.

[46] T. Vale, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *ESORICS,* vol. 12308, pp. 480–501, Springer, 2020.

[47] L. Lyu, Y. Han, and X. Ma, "Privacy and robustness in federated learning: attacks and defenses," *CoRR*, vol. 2012, Article ID 06337, 2020.

[48] S. Ali, W. R. Huang, and M. Najibi, "Poison frogs! targeted clean-label poisoning attacks on neural networks," *NeurIPS*, pp. 6106–6116, 2018.

[49] C. Fung, J. Chris, M. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses, RAID 2020*, pp. 301–316, USENIX Association, 2020.

[50] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *ICML. icml.Cc/Omnipress*, 2012.

[51] S. Mahloujifar, M. Mahmoody, and A. Mohammed, "Data poisoning attacks in multi-party learning," in *ICML,* vol. 97, pp. 4274–4283, PMLR, 2019.

[52] P. Blanchard, E. Mahdi El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: byzantine tolerant gradient descent," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 119–129, Long Beach, CA, USA, December 2017.

[53] Y. Dong, Y. Chen, R. Kannan, and P. L. Bartlett, "Byzantine-robust distributed learning: towards optimal statistical rates," in *ICML,* vol. 80, pp. 5636–5645, PMLR, 2018.

[54] F. Suya, S. Mahloujifar, A. Suri, D. Evans, and T. Yuan, "Model-targeted poisoning attacks with provable convergence," in *ICML,* vol. 139, , pp. 10000–10010, PMLR, 2021.

[55] Z. Sun, K. Peter, A. Theertha Suresh, and H. Brendan McMahan, "Can you really backdoor federated learning?" *CoRR*, vol. 1911, p. 07963, 2019.

[56] S. Chakraborty, P. Mittal, and B. Seraphin, "Analyzing federated learning through an adversarial lens," in *ICML,* vol. 97, pp. 634–643, PMLR, 2019.

[57] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to byzantine-robust federated learning," *USENIX*, pp. 1605–1622, USENIX Association, 2020.

[58] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: optimizing model poisoning attacks and defenses for federated learning," *NDSS*, The Internet Society, 2021.

[59] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *AISTATS,* vol. 108, pp. 2938–2948, PMLR, 2020.

[60] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun, "Understanding distributed poisoning attack in federated learning," *ICPADS*, pp. 233–239, IEEE, 2019.

[61] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021.

[62] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: passive and active white-box inference attacks against centralized and federated learning," *Science Progress*, pp. 739–753, IEEE, 2019.

[63] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," *Science Progress*, pp. 691–706, IEEE, 2019.

[64] K. Ganju, Qi Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," *CCS*, pp. 619–633, ACM, 2018.

[65] M. Shen, H. Wang, B. Zhang et al., "Exploiting unintended property leakage in blockchain-assisted federated learning for intelligent edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2265–2275, 2021.

[66] M. Fredrikson, E. Lantz, and S. Jha, "Privacy in pharmacogenetics: an end-to-end case study of personalized warfarin dosing," *USENIX Security Symposium (USENIX Security 14)*, pp. 17–32, 2014.

[67] D. Usynin, A. Ziller, M. R. Makowski et al., "Adversarial interference and its mitigations in privacy-preserving collaborative machine learning," *Nature Machine Intelligence*, vol. 3, no. 9, pp. 749–758, 2021.

[68] Y. Zhang, R. Jia, and H. Pei, "The secret revealer: generative model-inversion attacks against deep neural networks," in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, pp. 250–258, Seattle, WA, USA, June 2020.

[69] J. Chen, J. Zhang, and Y. Zhao, "Beyond model-level membership privacy leakage: an adversarial approach in federated learning," in *The International Conference on Computer Communications and Networks*, pp. 1–9, IEEE, 2020.

[70] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep models under the GAN: information leakage from collaborative deep learning," in *CCS*, pp. 603–618, ACM, 2017.

[71] N. Carlini, L. Chang, Ú. Erlingsson, J. Kos, and D. Song, "The secret sharer: evaluating and testing unintended

memorization in neural networks," in *USENIX Security Symposium*, pp. 267–284, USENIX Association, 2019.

[72] N. Carlini, F. Tramèr, and E. Wallace, "Extracting training data from large language models," in *USENIX Security Symposium*, pp. 2633–2650, USENIX Association, 2021.

[73] X. Yuan, X. Ma, L. Zhang, Y. Fang, and D. Wu, "Beyond class-level privacy leakage: breaking record-level privacy in federated learning," *IEEE Internet Things*, vol. 7, p. 372, 2021.

[74] W. Ronny Huang, G. Jonas, L. Fowl, G. Taylor, and T. Goldstein, "Metapoison: practical general-purpose clean-label data poisoning," *Classmate*, vol. 457, p. 8969, 2020.

[75] A. Manna, H. Kasyap, and S. Tripathy, "Moat: model agnostic defense against targeted poisoning attacks in federated learning," in *International Conference on Information and Communications Security,* vol. 12918, pp. 38–55, Springer, 2021.

[76] Y. Fraboni, R. Vidal, and M. Lorenzi, "Free-rider attacks on model aggregation in federated learning," *AISTATS*, PMLR, vol. 130, , pp. 1846–1854, 2021.

[77] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1279–1283, 2020.

[78] M. Kim, O. Günlü, and F. Rafael, "Federated learning with local differential privacy: trade-offs between privacy, utility, and communication," *ICASSP*, pp. 2650–2654, IEEE, 2021.

[79] S. Truex, L. Liu, K. H. Chow, M. E. Gursoy, and W. Wei, "Ldp-fed: federated learning with local differential privacy," in *Proceedings of the 3rd International Workshop on Edge Systems, Analytics and Networking*, pp. 61–66, ACM, Melborn, Sep, 2020.

[80] M. Naseri, J. Hayes, and E. De Cristofaro, "Toward robustness and privacy in federated learning: experimenting with local and central differential privacy," *CoRR*, vol. 09, p. 03561, 2020.

[81] C. Zhou, A. Fu, S. Yu, W. Yang, H. Wang, and Y. Zhang, "Privacy-preserving federated learning in fog computing," *IEEE Internet of Things Journal*, vol. 7, no. 11, pp. 10782–10793, 2020.

[82] 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 of Things Journal*, vol. 8, no. 8, pp. 6178–6186, 2021.

[83] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2020.

[84] J. So, B. Güler, and A. S. Avestimehr, "Turbo-aggregate: breaking the quadratic aggregation barrier in secure federated learning," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 1, pp. 479–489, 2021.

[85] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2168–2181, 2021.

[86] C. Fang, Y. Guo, N. Wang, and A. Ju, "Highly efficient federated learning with strong privacy preservation in cloud computing," *Computers & Security*, vol. 96, Article ID 10188, 2020.

[87] F. Wang, H. Zhu, R. Lu, Y. Zheng, and H. Li, "A privacy-preserving and non-interactive federated learning scheme for regression training with gradient descent," *Information Sciences*, vol. 552, pp. 183–200, 2021.

[88] L. Tong, L. Jin, and X. Chen, "NPMML: a framework for non-interactive privacy-preserving multi-party machine learning," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 6, pp. 2969–2982, 2021.

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

[90] Y. Chen, F. Luo, T. Li, T. Xiang, Z. Liu, and J. Li, "A training-integrity privacy-preserving federated learning scheme with trusted execution environment," *Information Sciences*, vol. 522, pp. 69–79, 2020.

[91] X. Zhang, F. Li, and Z. Zhang, "Enabling execution assurance of federated learning at untrusted participants," *INFOCOM*, pp. 1877–1886, IEEE, 2020.

[92] M. Fan, H. Haddadi, and K. Katevas, "PPFL: privacy-preserving federated learning with trusted execution environments," in *Proceedings of the 19th Annual International Conference on Mobile Systems*, pp. 94–108, ACM, Wisconsin, USA, June 2021.

[93] P. Rieger, M. Miettinen, and A.-R. Sadeghi, "Poisoning attacks on federated learning-based iot intrusion detection system," in *Workshop on Decentralized IoT Systems and Security (DISS)*, pp. 1–7, United States, February 2020.

[94] E. Mahdi El Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in byzantium," in *ICML,* vol. 80, pp. 3518–3527, PMLR, 2018.

[95] H. Chai, S. Leng, Y. Chen, and K. Zhang, "A hierarchical blockchain-enabled federated learning algorithm for knowledge sharing in internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 3975–3986, 2021.

[96] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Network*, vol. 35, no. 1, pp. 234–241, 2021.

[97] K. Salah, E. Damiani, and D. Svetinovic, "Towards blockchain-based reputation-aware federated learning," in *INFOCOM*, pp. 183–188, IEEE, 2020.

[98] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 72–80, 2020.

[99] S. R. Pokhrel and J. Choi, "Federated learning with blockchain for autonomous vehicles: analysis and design challenges," *IEEE Transactions on Communications*, vol. 68, no. 8, pp. 4734–4746, 2020.

[100] X. Qu, S. Wang, Q. Hu, and X. Cheng, "Proof of federated learning: a novel energy-recycling consensus algorithm," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 8, pp. 2074–2085, 2021.

[101] Y. Lei, L. Liu, C. Pu, M. E. Gursoy, and S. Truex, "Differentially private model publishing for deep learning," in *Science Progress*, pp. 332–349, IEEE, 2019.

[102] H. Brendan McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *ICLR*OpenReview.net, 2018.

[103] M. Abadi, A. Chu, and J. Ian, "Deep learning with differential privacy," *CCS*, pp. 308–318, ACM, 2016.

[104] C. Xu, J. Ren, D. Zhang, Y. Zhang, Z. Qin, and K. Ren, "Ganobfuscator: mitigating information leakage under GAN via differential privacy," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 9, pp. 2358–2371, 2019.

[105] Y. Zhao, J. Zhao, L. Jiang et al., "Privacy-preserving blockchain-based federated learning for iot devices," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1817–1829, 2021.

[106] C. Wutyee Zaw and C. Seon Hong, "A decentralized game theoretic approach for energy-aware resource management in federated learning," in *Proceedings of the IEEE International Conference on Big Data and Smart Computing*, pp. 133–136, IEEE, Jeju Island, Korea, January 2021.

[107] J. Lee, D. J. Kim, and D. Niyato, "Market analysis of distributed learning resource management for internet of things: a game-theoretic approach," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8430–8439, 2020.

[108] C. Li, G. Li, and K. Varshney, "Federated Learning with Soft Clustering," *IEEE Internet Things J.*, vol. 9, 2021.

[109] J. Pang, Y. Huang, Z. Xie, Q. Han, and Z. Cai, "Realizing the heterogeneity: a self-organized federated learning framework for IoT," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3088–3098, 2021.

[110] P. Zhang, C. Wang, C. Jiang, and Z. Han, "Deep reinforcement learning assisted federated learning algorithm for data management of iiot," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 12, pp. 8475–8484, 2021.