Received March 16, 2021, accepted April 14, 2021, date of publication April 23, 2021, date of current version May 3, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3075203

Vulnerabilities in Federated Learning

NADER BOUACIDA AND PRASANT MOHAPATRA, (Fellow, IEEE)
Department of Computer Science, University of California at Davis, Davis, CA 95616, USA

Corresponding author: Nader Bouacida (nbouacida@ucdavis.edu)

This work was supported by the U.S. Army Combat Capabilities Development Command Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA).

ABSTRACT With more regulations tackling the protection of users' privacy-sensitive data in recent years, access to such data has become increasingly restricted. A new decentralized training paradigm, known as Federated Learning (FL), enables multiple clients located at different geographical locations to learn a machine learning model collaboratively without sharing their data. While FL has recently emerged as a promising solution to preserve users' privacy, this new paradigm's potential security implications may hinder its widespread adoption. The existing FL protocols exhibit new unique vulnerabilities that adversaries can exploit to compromise the trained model. FL is often preferred in learning environments where security and privacy are the key concerns. Therefore, it is crucial to raise awareness of the consequences resulting from the new threats to FL systems. To date, the security of traditional machine learning systems has been widely examined. However, many open challenges and complex questions are still surrounding FL security. In this paper, we bridge the gap in FL literature by providing a comprehensive survey of the unique security vulnerabilities exposed by the FL ecosystem. We highlight the vulnerabilities sources, key attacks on FL, defenses, as well as their unique challenges, and discuss promising future research directions towards more robust FL.

INDEX TERMS Attacks, defenses, federated learning, security threats, vulnerabilities.

I. INTRODUCTION

The emerging Artificial Intelligence (AI) market is accompanied by an unprecedented growth of cloud-based AI solutions. This technological revolution was catalyzed by the rapid expansion of personal computing devices (such as smartphones and tablets) and internet usage in emerging and developing nations alike [1]. Most people are frequently carrying their smart personal devices equipped with multiple sensors (e.g., cameras, microphones, accelerometers, and GPS chips). As a result, personal computing devices offer access to a large amount of training data necessary to build reliable machine learning models.

Traditional machine learning requires gathering the training data in a single machine or in a data center. As a result, technology companies must go through the costly and lengthy process of harvesting their users' data, not mentioning the risks and responsibilities of storing data in a centralized location. As datasets proliferate in size and models are becoming more complex, it would eventually put the smaller startups at a disadvantage, leading to the market monopolization of a few influential players. Moreover, privacy dilemmas mount for technology giants. Recently, Facebook and Amazon have admitted that they have been listening to some users' conversations following a recent investigation [2], [3]. After an online protest campaign over their handling of users' personal information, they were forced to cease the practice or give their customers the option of turning off the sharing of any personal data with them or their affiliates.

Federated Learning (FL) [4]–[8] is conceived to address these issues by enabling end-users devices to collaboratively learn a shared global model using the locally-stored training data under the orchestration of a central server, decoupling training deep learning models from the need to collect and store the data in the cloud. With its decentralized data approach, FL is one of the fastest-growing research fields, as it comes with the privacy and security features that aim to comply with the requirements of the recent user data protection laws [9], [10]. However, FL is not immune to various kinds of attacks and failures that target each step of the system's training and deployment pipelines. Examples of attacks include data and model poisoning [11], [12], backdoor attacks [13], and inference attacks [14], [15].

By sharing the model parameters instead of data, FL introduces new attack surfaces at training time by enhancing the capabilities of the adversary [16]. Attackers have numerous ways to exploit vulnerabilities in the FL environment. For instance, once the attackers have full control over one or

The associate editor coordinating the review of this manuscript and approving it for publication was Li He.

multiple participating devices, they can maliciously modify training data, model parameters, and even training pipelines. Clearly defining these capabilities is essential to assess the efficacy of proposed defenses. Besides, training with thousands of personal devices makes it impossible to ensure that none of them are malicious or compromised. The distributed nature of FL, particularly when augmented with secure aggregation [17], causes many attacks to go unnoticed [18] because secure aggregation prevents the server from inspecting each user's model update. As FL is gaining more popularity in the AI community, many researchers are eagerly working to improve the existing algorithms and ensure the security of FL protocols.

To some extent, model parameters sharing combined with an increased number of communication rounds between the clients and the server exposes the FL environment to a new set of risks and opens new avenues for privacy leakage [19]. Besides, curious adverseries can trace vulnerabilities to manipulate the machine learning model outputs or gain unauthorized access to sensitive data. Therefore, we should question the unique security concerns occurring as a consequence of adopting FL technology? This paper aims to address the security aspects of FL and sheds light on possible unwanted vulnerabilities that we should be mindful of and prepare for when transitioning to a FL ecosystem.

To ensure that we leverage the benefits of FL and exploit its features properly while avoiding the associated risks, we need to investigate the potential security attacks on FL. Our work mainly touches on the issues related to FL security and not privacy, even though privacy vulnerabilities can play a major role in composing attacks against FL models [20]. Unless enough security guarantees are provided, FL may be pushed back without giving it a fair opportunity to leverage its full potential. In the following sections, we will take an in-depth look at the different attack vectors, identify and evaluate vulnerabilities unique to FL as well as mitigation strategies and areas that may be interesting to advance the cybersecurity research field.

The rest of the paper is structured as follows: Section II examines the shift of the threat models under FL paradigm. Section III identifies different vulnerabilities and threats related to FL. Section IV provides a classification and an overview of the attacks in FL. Section V gives insights into existing defenses against FL attacks and their limitations. Section VI extends the study of attacks and defenses to peer-to-peer FL. Section VII discusses the future directions towards secure and robust FL. Finally, we conclude the paper in Section VIII.

II. THREAT MODEL UNDER FEDERATED LEARNING

A standard FL training algorithm proceeds in rounds of training where a typical round consists of the following steps:

1) *Selecting clients:* The cloud server samples from a set of clients meeting eligibility requirements (e.g., mobile phone plugged in, device idle, Wi-Fi connection available).
2) *Broadcasting the global model:* The selected clients download the global model from the server.
3) *Local Training:* Each selected client computes an update to the model based on its local data following a training procedure (e.g., local Stochastic Gradient Descent (SGD)).
4) *Aggregation:* The clients send their model updates to the server. The latter aggregates these updates to construct an improved global model.
5) *Updating the global model:* The server updates the global model with the aggregate computed from the participating clients.

FL architecture introduces new threat models, resulting in unique vulnerabilities. To better understand what attack surfaces are exposed or widened, we analyze different actors' capabilities in FL and their susceptibility towards specific attacks. Unlike traditional machine learning, FL systems should withstand three potential adversaries: (1) Clients, (2) The aggregator, and (3) Outsiders or eavesdroppers. An adversary may hold a mixture of different capabilities when trying to impair the global model during training. It is important to note that defining these capabilities is necessary to understand how different attacks work. In Table 1, we try to summarize the capabilities that an adversary may possess, depending on its role or position in the FL process.

A. CAPABILITIES OF CLIENTS ADVERSARIES

FL allows the attackers to get full control over one or several participants (e.g., smartphones whose application software has been compromised by malware). Moreover, the adversary can stimulate multiple dummy client accounts to mount more successful attacks [13]. The capabilities of an attacker can take any combination of the following forms:

- The attacker controls the local training data of any compromised device.
- The attacker can modify or replace the resulting local model's parameters before submitting it to the server.
- The attacker controls the local training procedure, including the local loss optimization and the hyperparameters such as the learning rate, the local batch size, and the number of local epochs.
- The attacker can dynamically change the local training procedure from one training round to another.

The main difference between this setting and the traditional one is that the latter assumes that the adversary controls a significant fraction of the training data. By contrast, in FL, the attacker controls the entire training process for a few participants, which includes data processing, local training pipelines, and model updates. These potent adversary capabilities allow malicious clients to carry out particularly devastating attacks.

B. CAPABILITIES OF AGGREGATOR ADVERSARIES

Another possible attack vector is the aggregator itself. If an adversary can directly control the aggregator, he can

TABLE 1. Summary of adversaries' capabilities in federated settings.

Adversaries	Capabilities
Clients	• The adversary can observe the global model. • The adversary can corrupt or replace the local model update. • The adversary controls the local training procedure, including the loss function optimization and the hyperparameters such as the learning rate, the local batch size, and the number of local epochs. • Client adversaries can coordinate with current or future participants on attacks against current or future updates to the global model.
The aggregator	• The adversary does not have any access to the training or testing data. However, it has the ability to inspect and alter the parameters of the global model directly. • An aggregator adversary can examine all model updates of the clients in the absence of secure aggregation mechanisms. • The adversary can inject an attack throughout the aggregation algorithm.
Outsiders	• The adversary can intercept communications between the participating parties in the FL process. • The end-users of the deployed service are outsiders to the FL training process, and they can play the role of adversaries given they have access to the final trained model in the deployment phase. Thus, they have the ability to carry out inference-time attacks.

efficiently perform both targeted and untargeted attacks on the trained global model [21]. Methods designed to certify the integrity of the training process, such as Secure Multi-Party Computations (SMC) [22] or Zero-Knowledge Proofs (ZKP) [23], can detect these attacks on the aggregator. Nevertheless, honest-but-curious or semi-honest aggregator adversaries may attempt to infer private information using the model updates received throughout the protocol execution. This threat model aspect looks similar in both federated and distributed learning. Hence, we skip a detailed discussion of this line of attacks. We turn our attention to attacks that aim at corrupting the global model since the aggregator adversaries have direct access to it.

C. CAPABILITIES OF OUTSIDERS ADVERSARIES

FL threat models also embrace the potential attacks from adversaries existing outside of the system. Exchanging model updates during the training phase exposes them to theft by outsiders. Therefore, an adversary monitoring the communication channels can efficiently infer the private data of the participants [24]. Securing channels between clients and the server is a common practice to ensure the security of FL. However, the threat of collusion among participating parties and outsiders, including the aggregator authority, may add another dimension to the security challenge of FL. The aggregator has to authenticate incoming messages and prevent adversaries, whether they are outsiders or malicious data parties, from injecting their corrupted updates.

FL also considers the users of the final trained model as potential adversaries. The deployed model should remain resilient in inference time against adversaries who are users of the service.

III. VULNERABILITIES IN FEDERATED LEARNING

FL offers a new paradigm to protect user privacy while performing machine learning tasks on an extensive scale, but it is fraught with several vulnerabilities that must be addressed. In this section, we define the research questions for the survey, and we explore the various sources of vulnerabilities in FL.

A. RESEARCH QUESTIONS

The decentralized approach of FL presents new vulnerabilities. Insider attacks are usually carried out by clients but can also be launched by the server. In FL environments, clients are often selected randomly to participate in a training round. When training with thousands or millions of clients, there is no way to detect malicious ones by relying solely on their security guarantees. Malicious adversaries will try to learn other clients' private states and deviate from FL protocol by corrupting, replaying, or removing messages. Even benign clients cannot be trusted in a federated setting because they can be curious and attempt to infer sensitive information without necessarily corrupting the model updates. A malicious data provider is often limited to a static attack in conventional deep learning wherein the poisoned data is supplied before training begins. In contrast, a malicious client continuously participates in FL and could launch an attack throughout the model training process. Indeed, the attacker can adaptively alter training data or model updates as the training progresses.

FL protocol may also be affected by the server vulnerabilities. The server can observe client updates, tamper with the aggregation process, and control the view of the participants of the global model and compression parameters. FL gives rise to open and complex challenges surrounding the robustness of a collaboratively trained deep learning model. In this paper, we investigate the following research questions:

- *What are the various sources of vulnerabilities in FL?*
- *What are the security attacks in FL ecosystem?*
- *What makes the attacks unique to FL ecosystem?*
- *How can one protect FL against adversarial attacks?*
- *How FL topology affects the attack surfaces?*

We will also discuss broad questions concerning the relation between vulnerabilities and new attack surfaces in FL. As we will see later, FL allows for unprecedented adversarial

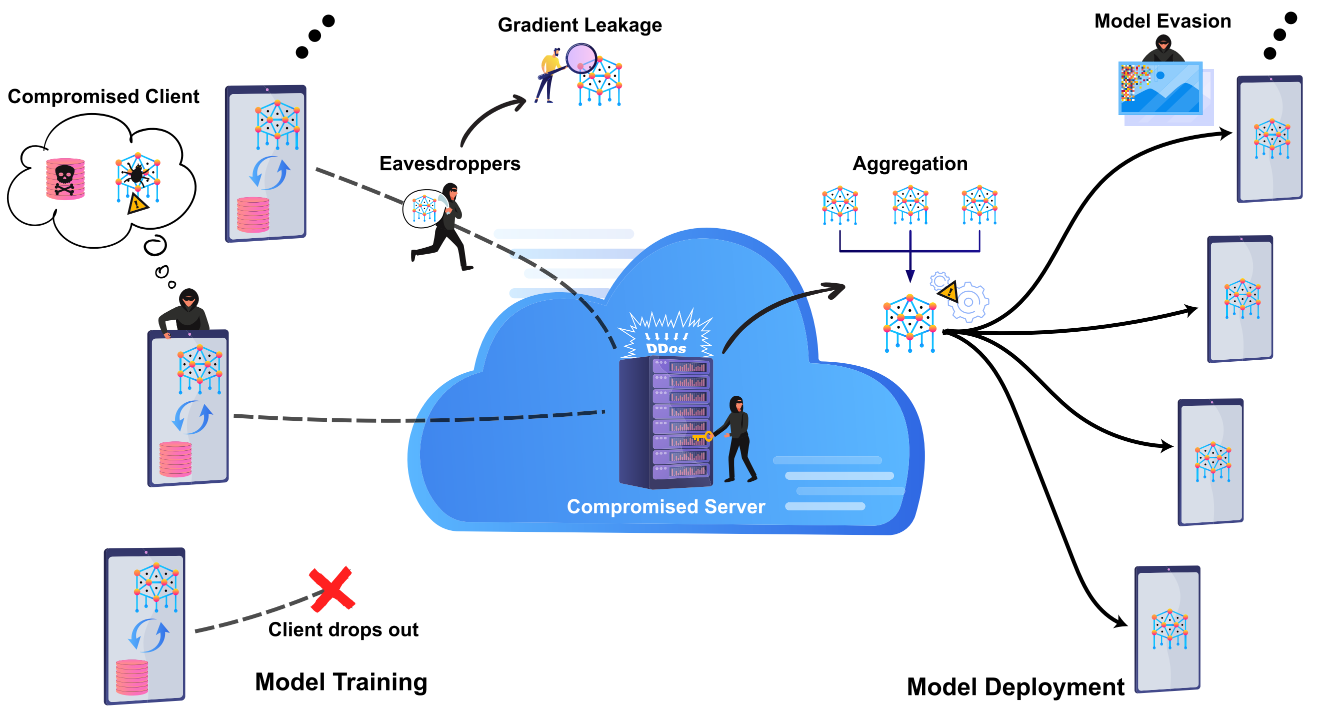

FIGURE 1. **The lifecycle of FL process and the various sources of vulnerabilities.**

attacks, enhances the potency of many existing attacks, and increases the hurdle of defending against these attacks.

B. SOURCES OF VULNERABILITIES

FL workflows involve many actors with distinct adversarial capabilities. In an ideal scenario, each actor is trusted and would learn nothing more than the information needed to fulfill his role. However, the standard FL environment is not naturally resilient against failures and attacks sourced from several vulnerabilities. A vulnerability is defined as a weakness that an attacker can exploit to cross privilege boundaries (i.e., perform unauthorized actions) [25]. Knowledge of FL vulnerabilities helps to manage and defend against possible attacks. Failure to identify FL vulnerabilities will result in weak defenses against attackers. As a first step, we examine the different sources of vulnerabilities in the FL process and illustrate them in Fig. 1.

1) COMMUNICATION

FL process usually reaches convergence after hundreds to thousands of communication rounds. A non-secure channel is an open vulnerability. Eavesdroppers can intercept the models exchanged between different participants, as well as the final FL model in the deployment phase, and replace them with malicious models. Homomorphic Encryption (HE) [26], [27] is usually employed to protect clients' data through model updates exchange between the clients and the server. Furthermore, it enables the server to perform computation on encrypted model updates without decrypting them. However, these protecting measures have been initially designed for privacy purposes and do not prevent FL from being susceptible to HE attacks [28], [29].

Besides, communication is considered the main bottleneck in FL since internet connections operate at lower rates than internal data centers links and can often be unreliable or expensive [30]. Communication bottlenecks increase the fraction of clients dropping out. Discarding clients based on their connectivity status introduces potential unwanted biases in the global model and weakens model updates aggregation.

2) GRADIENT LEAKAGE

FL offers a privacy-preserving solution for training with distributed data. Nevertheless, although the data is not explicitly shared in the training process, it is still feasible for the adversaries to reveal delicate information and even reconstruct the raw data approximately [31], [32]. Also, as demonstrated by [33], even a small portion of intermediate outcomes such as gradients updates can reveal sensitive information about local data. Thus, the gradients' transmission may actually leak private information in a phenomenon known as *gradient leakage*.

Adversaries can take advantage of leaked gradients to infer valuable information about benign participants. Such information is valuable in overcoming defenses [34]. The byzantine malicious clients can use information about the benign participants to tailor their updates to have similar distribution as the legitimate model updates, making them very difficult to detect [35]. Byzantine attacks are generally stronger than other types of attacks, as they strictly enhance the adversary's capabilities.

3) COMPROMISED CLIENTS

In traditional machine learning, the clients do not play any role in the training process other than providing training data. In FL, clients are considered a critical component of the architecture. They can observe intermediate global model states and contribute with updates as part of the decentralized training procedure. This creates opportunities for malicious clients to tamper with the training process freely. Compromised clients corrupt the FL training process by either exploiting model parameters or training data to craft an attack.

4) COMPROMISED SERVER

In centralized FL, the server is responsible for sharing the initial model parameters, aggregating model updates, and communicating the global model to the selected clients. Commonly, FL architecture comprises a server or a cluster of servers living in the cloud. A cloud-based or a physical server can be a target for cloud computing and hacking attacks [36], [37] or Distributed Denial of service (DDos) attacks [38]. Furthermore, a compromised server can inspect all gradient updates sent to the server in any training rounds and tamper with the aggregation process. The security of the server should be continuously accessed for any vulnerabilities that attackers could exploit.

5) AGGREGATION ALGORITHM

The aggregation algorithm orchestrates the learning of global model parameters. Since the clients' data and the training pipelines cannot be inspected for anomalies due to privacy guarantees, the aggregator represents the most critical defense line against attacks. It should incorporate appropriate mechanisms to detect abnormal client updates and discard them. Anomaly detection mechanisms incorporated within the aggregation algorithm ensure the convergence of the global model [39] and fairness with heterogeneous clients [40], [41]. Failure to correctly configure and maintain a robust aggregation algorithm will make the global model vulnerable and untrustworthy.

6) NON-MALICIOUS FAILURE

We recall that each training round in FL involves broadcasting the global model to the clients, local gradients computation, and client reports to the central aggregator. For any participating client, systems factors such as low bandwidth or limited computation power may cause failures at any of these steps. In some instances, selected clients will report failures and drop out of the training round as a result. Such failures may discard clients with valuable data from the training process, resulting in a low-quality biased model.

Additionally, any FL system architect must define how raw user training data is accessed and preprocessed. Bugs or unintended actions in the training pipeline can drastically alter the FL process. While standard data analysis tools can easily recognize data pipeline bugs in a data center setting, FL's data privacy restrictions make bug detection significantly more challenging.

An adversary can recreate the circumstances for a natural-occurring failure and exploit it to depreciate the quality of the model being trained. Even if no adversary is present, the model updates sent to the server may become distorted due to network and architectural factors [42], [43] (e.g., effects of model's compression, noisy features, or noisy labels in the clients' data).

7) DISTRIBUTED NATURE OF FEDERATED LEARNING

Distributed training opens the door to *colluding* and distributed attacks, where multiple parties collude to launch a coordinated attack. Past clients can coordinate with current or future participants to participate in attacks against current or future updates to the global model. For example, authors in [44] have recently proposed the Distributed Backdoor Attack (DBA), a novel colluding attack developed by fully exploiting the distributed nature of FL. DBA decomposes a global trigger pattern into separate local patterns and embeds them into the training set of different adversarial parties. They demonstrated that DBA is substantially more persistent and stealthy on diverse datasets such as finance and image data. Moreover, non-homogenous data distribution introduces a potential source of bias in FL. It also poses a challenge for defense strategies in terms of avoiding false positives.

8) FEDERATED LEARNING ENVIRONMENT SCOPE

By conducting model training at the network edge, FL spans multiple parties: clients, architects, developers, analysts, and deployers. FL provides an opportunity to capture more considerable data variability and analyze clients across different demographics and locations. Hence, its deployment in the real world requires agreements between different participating parties to define the scope, the aim, and the technologies used. Beyond fundamental and domain-specific constraints, designing and enforcing coordination protocols can be difficult to pin [45] and may result in situations where we cannot guarantee the robustness of collaborative training.

Besides, FL enables collaborative research for competing companies. In this context, one of the largest collaborative research initiatives is the Melloddy project [46]. It is a project aiming to deploy multi-task FL across the datasets of 10 pharmaceutical companies. The goal is to train a shared predictive model, which infers how chemical compounds bind to proteins. Project partners expect to optimize the drug discovery process without revealing their cherished in-house data. In order to warrant the security of an industrial FL platform, any party involved should establish an agreement on coordination protocols. Additionally, the amount of information sharing that will take place should be negotiated beforehand. Otherwise, it can turn out into a security threat, either due to

TABLE 2. Summary of attacks in federated settings.

Attacks	Description	Sources of Vulnerabilities
Data Poisoning	manipulate client data, perhaps by flipping labels or modifying specific features of the data	• Compromised Clients
Model Poisoning	manipulate model updates to bias the global model towards a certain objective	• Compromised Clients • Compromised Server
Backdoor Attacks	insert hidden backdoors into the global model while retaining the accuracy of the main task	• Compromised Clients
Evasion Attacks	circumvent a deployed model by carefully manipulating the data samples fed into it	• Compromised Clients • Model Deployment
Non-Robust Aggregation	aggregation algorithms with weak defense mechanisms or ill-disposed reweighting schemes that cause the global model to behave abnormally	• Aggregation Algorithm
Training Rules Manipulation	manipulating model training rules such training hyperparameters to prevent convergence or introduce bias in the trained model	• Compromised Clients • Distributed Nature of FL
Compromised FL Distributed Computation	diverging from executing the wanted behavior of distributed FL computation to generate unreliable intermediate results	• Compromised Clients • Compromised Server • Distributed Nature of FL • FL Environment Scope
Inference Attacks	analyze information leaked about participants in order to illegitimately gain knowledge about FL process (e.g, participants, data, features) and use this knowledge to craft an attack against FL	• Compromised Server • Communication
GAN Reconstruction Attacks	employ GANs to recover synthetic samples of the training data through inference and use these samples to poison the training data	• Compromised Clients • Communication
Malicious Server	manipulate the global model to utilize shared computational power in building malicious tasks	• Compromised Server
Communications Bottlenecks	communication bottlenecks can disrupt the FL process and techniques that aim to reduce the communication cost, such as compression, can be used in a harmful way to introduce noise in the model updates and degrade their quality	• Communication • Distributed Nature of FL
Free-Riding Attacks	dissimulate participation in the FL process with the goal of obtaining the final model without contributing to the training process	• Compromised Clients
Man-in-the-Middle Attacks	intercept the models exchanged between the clients and the server and replace them with malicious updates	• Communication
Dropout of Clients	clients drop out due to network issues, unexpected roadblocks, or otherwise becoming temporarily unavailable, resulting in fairness issues and depriving the model of potentially precious data	• Non-Malicious Failure • Communication

confusion or lack of understanding of user data's sensitivity level. Hence, today's large-scale initiatives can be the pioneers of tomorrow's standards for safe, fair, and innovative FL collaboration.

9) MODEL DEPLOYMENT

After assessing the quality of the trained model, eventually, the latter can be deployed to serve end-users. The model deployment process is far from being risk-free. An adversary could corrupt the training process in order to create or enhance inference-time vulnerabilities of the deployed model [47]. Moreover, white-box evasion attacks [48]–[50] commonly cause correctly trained models to attain low accuracy by crafting perturbed variants of the test inputs. These perturbations appear nearly indistinguishable from the initial testing inputs to a human eye but can trick the deployed model. Engineers typically only focus on FL robustness to the specific type of adversarial examples incorporated during training, potentially leaving the deployed model vulnerable to other forms of adversarial noise.

IV. ATTACKS IN FEDERATED LEARNING

An attack exploits vulnerabilities by a malicious attacker to manipulate the global model. Adversarial attacks can be generally classified into two categories based on their goals: targeted or untargeted attacks. The targeted attacks are often referred to as backdoor attacks because the adversary aims to change the model's behavior on specifically targeted sub-tasks while sustaining good overall accuracy on the main task. For instance, for an image classification application, the attacker may corrupt the model to misclassify "cars with stripes" as "birds" while ensuring other car instances are correctly classified. Under untargeted attacks, the adversary aims to reduce the model's global accuracy or "fully break" the global model. We classify attacks based on a broader perspective and describe them in the following sub-sections. Table 2 provides a comprehensive overview of attacks on FL.

A. ATTACKS FOCUSED ON DATA

The largest threat surface in FL ecosystem is the cluster of clients participating in the training phase by contributing with data and computation. Instead of solely exploiting the boundaries of a model deployment, client adversaries in FL have to power to shift the model's boundaries during training. Therefore, FL literature has primarily focused on defending against attacks centered on data [51].

1) POISONING ATTACKS

FL clients can observe intermediate global model states and contribute with model updates as part of the decentralized training procedure. An adversary may control one or more client devices to send compromised model updates to the server. Poisoning attacks [52], [53] seek to induce model corruption by maliciously manipulating training data samples and/or model updates. They can be separated into two types of attacks:

- Data poisoning attacks compromise the integrity of the training data to corrupt the global model.
- Model poisoning attempts to induce model corruption by manipulating the FL training procedure itself.

Independently from the poisoning sources, poisoning attacks try to modify the behavior of the target model in some undesirable way. The severity of this type of threat is very high, given that model updates are usually produced based on updates received from a large group of clients.

Data poisoning: Data poisoning attacks mainly fall into two categories: clean-label attacks [54] and dirty-label attacks [47], [55]. Clean-label attacks assume that the client adversary cannot permute the label of any training data instance. This assumption is due to the existence of a process by which the data is verified belonging to the correct class, and, hence, data poisoning has to be subtle. In contrast, to carry out dirty-label poisoning, all the adversary has to do is to introduce several copies of the data samples it wishes to misclassify with the desired target label into the training set. Indeed, there is no certification requirement in such a scenario to ensure a data sample belongs to the correct class. One typical example of a dirty-label poisoning attack is label-flipping. Fig. 2 highlights how an adversary might corrupt the trained model by flipping labels. The adversary flips the training data labels belonging to a particular class to another class while keeping the data features unchanged. For example, malicious clients can poison the MNIST handwritten digits dataset [56] by flipping all digits 3s into 5s and vice versa. A successful attack produces a model that cannot accurately classify 3s and incorrectly predicts them to be 5s and vice versa.

Another less common attack is backdoor poisoning [57]. Adversaries can manipulate the training data by modifying individual features or adding watermarks to small regions of the original training dataset. Backdoors embedded in the model will act on triggers in the inputs (e.g., watermarks, a stamp on images). Meanwhile, the accuracy of the poisoned model on clean data remains untouched.

Data poisoning attacks in FL focus on dirty-label poisoning for two reasons. First, FL operates under the assumption that data is never shared, only learned models. Thus, the adversary is not concerned with notions of imperceptibility for

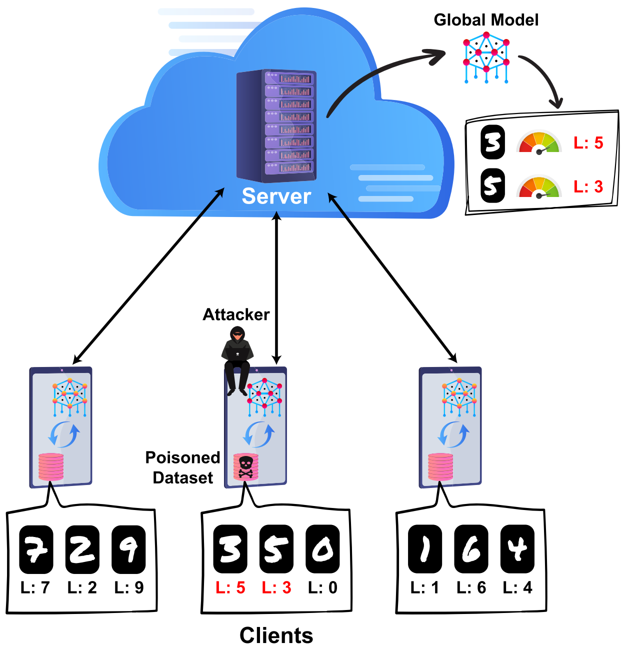

FIGURE 2. An example of data poisoning attack in FL.

data certification. Second, clean-label data poisoning assumes access at train time to the global parameter vector, which is absent in the FL setting. Dirty-label data poisoning has been shown to achieve high-confidence targeted misclassification for deep neural networks with just the addition of around 50 poisoned samples to the training data [58].

Model poisoning: These attacks [59], [60] aim to corrupt local model updates before sending them to the server. They involve a broad category of techniques to manipulate the FL local training procedure (e.g., direct gradient manipulation). Unlike the traditional setting, FL is by design vulnerable to model poisoning since the global model is exposed to the clients in all training stages and can be intercepted during communication. Common defenses against model poisoning enforce norm thresholding on client model updates (e.g., by ignoring the client updates whose norms exceed some threshold) [61] or add Gaussian noise to the aggregate [62]. However, to increase attack stealth and avoid detection, model poisoning attacks on FL exploit the fact that they directly control the local training procedure. They usually optimize for both training loss and adversarial objective and use parameter estimation for the benign clients' updates to avoid diverging from the current global model. When tuned properly, defense evasion techniques can conceal the majority of the poisoned updates as legitimate.

Generally speaking, model poisoning attacks are much more effective than data poisoning in FL settings. A single, non-colluding malicious participant can make the global model incorrectly classify a set of chosen inputs with high confidence. That occurs because the malicious client's updates are usually boosted and tailored to cause maximum damage to the global model's performance.

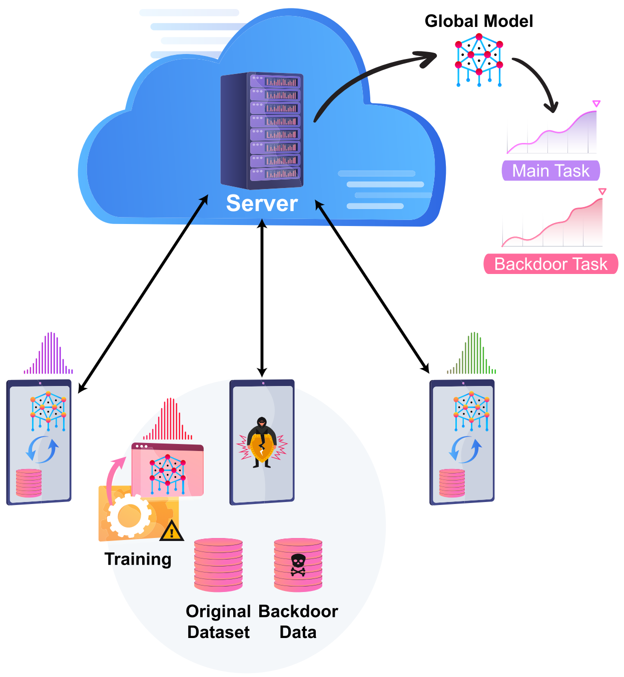

FIGURE 3. Backdoor attacks in FL.

2) BACKDOOR ATTACKS

Backdoor attacks can be classified under the category of model poisoning attacks. However, they are less transparent in comparison. They work by inserting hidden backdoors into the global model while retaining the accuracy of the main task [13], [63]–[65]. Compromised devices participating in FL can introduce a backdoor by training the model on chosen backdoor data as illustrated in Fig. 3. After incorporating detection evasion routines into the attacker's loss function, the poisoned model updates look and behave similarly to models trained without backdoors. The detection of a backdoor's presence becomes complicated when backdoored models do not diverge from other models. The evasion of anomaly detection is performed using an objective function that rewards the model for accuracy and penalizes it for deviating from what the aggregator defense considers within its acceptance threshold [66], [67]. The impact of backdoor attacks can be devastating, as they can predict the false positives with high confidence. Besides, sophisticated backdoor attacks in FL can effectively overturn the *catastrophic forgetting* problem [13], stopping the backdoor from being omitted while the training progress.

3) EVASION ATTACKS

In evasion attacks [48], [68]–[70], an adversary tries to evade a deployed model by carefully manipulating the data samples fed into it. One popular mode of evasion attacks is so-called "adversarial samples", which are modified versions of test samples that appear almost interchangeable with the original data samples to a human. However, in reality, they are carefully crafted to deceive a classifier. Although evasion attacks are not new or very popular in FL, they exhibit unique vulnerabilities in FL environments. In a white-box setting, perturbations are generated by maximizing the loss function subject to a norm constraint via constrained optimization techniques such as projected gradient ascent. In a black-box setting, adversaries can substitute models trained on similar data. FL facilitates these attacks by allowing the attacker to access the model and the local training loss function. Fig. 4 shows an adversarial sample crafting process for an image classification task (handwritten digits recognition). The attacker determines the sensitivity of a class change to each input feature by recognizing trends in the data manifold around the sample, in which the deployed global model is most sensitive and prone to result in a class change. In the example shown in Fig. 4, the model infers the digit 1 as 7 after adding the perturbation to the input.

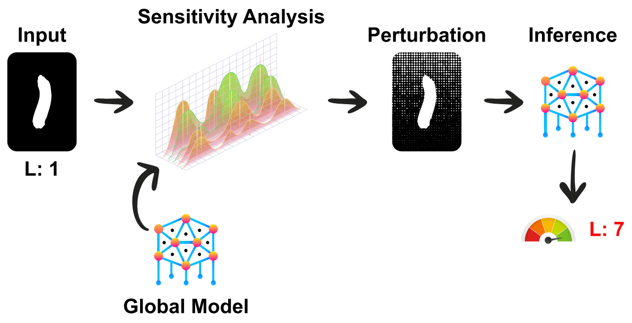

FIGURE 4. Adversarial samples crafting process for evasion attacks.

B. ATTACKS FOCUSED ON ALGORITHMS

It is considered a potentially more restrictive class of attacks than the previously cited threats. In this category, the adversary violates the integrity of either the aggregation algorithm (assuming he can access the server or the aggregator in general) or the local training pipeline, perhaps by changing the optimizer's healthy functioning. The attacker can also manage the local training scheme, including the hyperparameters. This attack vector is more natural when the adversary has full control over one or several participants at the edge of the FL system. It can indirectly corrupt derived quantities within the learning system (e.g., model updates).

1) NON-ROBUST AGGREGATION

The aggregation algorithm in FL is vulnerable to adversarial attacks [71], [72]. In the presence of label-flipping, backdoor attacks, and noisy updates, non-robust aggregation algorithms will produce unusable and compromised models. Current defenses may not be adequate to ensure the robustness of the aggregation algorithm. As a matter of fact, non-i.i.d data distribution and large variance exhibited by gradients among different clients disrupt their basic assumptions in practical FL environments. Besides, aggregation algorithms with ill-disposed reweighting schemes cause the global model to behave abnormally. To address these challenges, aggregation algorithms should be thoroughly inspected to evaluate their resiliency against adversaries.

2) TRAINING RULES MANIPULATION

Most, if not all, existing FL attacks focused on algorithms can be categorized as compromising either aggregation or computation. One way to compromise computation in FL involves manipulating model training rules. FL brings model training to the data sources (clients) instead of delivering the data to the model. Since the central FL authority has no control over the clients, adversaries controlling some of the participating devices may attempt to play with training hyperparameters such as the number of local epochs, the learning rate, and the batch size to prevent the model from being learned at all. Also, they might try to bias the global model to produce predictions in their favor [73]. Small changes to the training rules, such as increasing or decreasing the learning rate, may cause the optimizer to behave erratically and the global model convergence to fail.

3) COMPROMISED DISTRIBUTED COMPUTATION

Generally speaking, FL computation aims to evaluate a function on a distributed client dataset (usually a deep neural network training algorithm, but it can be something straightforward, such as a basic computation). The design of the information flow in the system can affect intermediate results and can make them susceptible to malicious actors. There is a problem of *verifiability*, which concerns a client or the server's ability to prove to other participants that they have genuinely executed the wanted behavior without revealing any of their private information upon which they were acting.

C. ATTACKS FOCUSED ON FEDERATION

FL can occur across a range of security settings, network topologies, data partitions, and objectives. Creating a federated system with ideal security properties is a daunting feat on its own due to multiple attacks that target the federation itself.

1) INFERENCE ATTACKS

Exchanging gradients during the FL training process can result in serious gradient leakage, as previously discussed. Model updates can leak information about the features of clients' training data to the adversarial participants. Inspection of deep learning models internally exposes many features of the data that are not associated with the main task [74]. The adversary can also take a snapshot of the FL model parameters and conduct property inference by calculating the difference between the consecutive snapshots, which is equivalent to the aggregated model updates from all participants minus the adversary.

The gradients are generated based on the clients' private data. In neural networks, gradients of a particular layer are calculated using this layer's features and the error backpropagated from the following layer. For instance, in sequential fully connected layers, the gradients are inner products of the activations and the error backpropagated from the following layers. Similarly, for a convolutional layer, the gradients are convolutions of the error from the layer above and the activations. As a result, examining the model updates can reveal a substantial amount of private information such as class membership [75], class representations [76] and properties of other participants' data [31]. Even worse, an adversary can infer labels from the gradients and recover the original training data without expecting any prior knowledge about the training set [14].

Membership inference attacks try to discover whether a data sample belongs to the training data. For example, the non-zero gradients of the embedding layer in a neural network trained on a natural-language task unveil which tokens emerge in the benign clients' training batches during FL training [77]. Class representations inference attacks seek to determine whether a data sample belongs to a class represented by the model. Besides, gradient exchange leakage can be used to infer when a property appears and disappears in the data during training (e.g., identifying when a person first appears in the photos used to train a face recognition classifier).

Inference attacks usually fall under the category of privacy threats. However, they also represent a major FL security menace. Adversarial access to the model is abused to learn sensitive genomic information about benign clients. Eavesdroppers and outsiders also use inference to craft a malicious model replica that looks legitimate and is supposed to replace the original model. With knowledge about class representations, data properties, memberships, and even original data samples, outsiders will have the necessary tools to produce a corrupted model update and perform a model replacement procedure. We depict examples of the discussed inference attacks in Fig. 5.

2) GAN RECONSTRUCTION ATTACKS

Generative Adversarial Networks (GAN) attacks resemble inference attacks. However, they can be more compelling given that they have demonstrated their ability to generate artificial samples that are statistically representative of the training data [76]. For instance, authors in [78] demonstrate how GAN can be used to get training samples through inference and use these recovered samples to poison the training data. An adversary can act as a benign participant and stealthily trains a GAN to simulate prototypical samples of the other clients' training set, which does not belong to the attacker. The latter will fully control these generated samples to create the poisoned updates. Then, it compromises the global model by uploading the scaled version of these poisoned updates to the server. The GAN-based threat can be a serious breach of FL security because it cannot be foreseen in advance.

3) MALICIOUS SERVER

In cross-device FL, most of the federation work is performed at the central server, from selecting the initial model

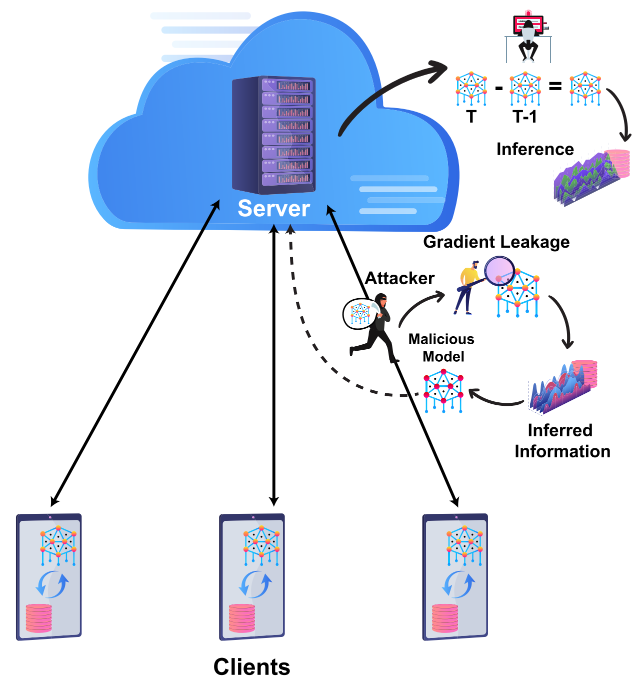

FIGURE 5. Examples of inference attacks in FL.

parameters and aggregating updates to deploying the global model. Compromised or malicious servers can have a significant impact by corrupting the global model. They can easily extract private client data or manipulate the global model to utilize shared computational power in building malicious tasks when training a machine learning model [79].

This is an interesting problem in the FL context. On the one hand, attackers can gain direct access to the global model from the server, which widens the attack surface. On the other hand, the server decides the clients view of the joint model, resulting into a substantial influence over the model being trained. Given the server's control over when each client can access and manipulate the model during the FL training process, there is a potential to design new tractable schemes for measuring a model's average-case or worst-case attack susceptibility.

4) COMMUNICATIONS BOTTLENECKS

A principal motivating example of FL arises when the training data comes from users' interaction with mobile applications. With increasingly improved hardware capabilities in personal computing devices, the computation overhead becomes marginal. The communication overhead is likely to be the main bottleneck of FL [30], [80], [81], especially with larger deep learning models. For instance, transferring a ResNet-50 model that contains 23.5 million parameters will incur an overhead of 94MB, assuming each weight is encoded using 4 bytes. This is a large amount of data to transfer over a wireless connection, while training a batch of data over a GPU will only take a few tens of milliseconds. With hundreds of communications rounds eventually needed for the model to converge, communication time poses a particular challenge to FL.

A naive implementation of the FL framework requires that each client submits a full model update to the server in each training round. The asymmetric property of internet connection speeds assumes the uplink is typically much slower than the downlink. To avoid violating the property that no individual client's updates can be inspected before aggregation in the server, an extra layer of security is usually added on top of the clients' raw updates, further increasing the amount of data to be uploaded [17]. Additionally, techniques that aim to reduce the communication cost, such as compression, can be applied in a negative way to degrade the quality of the model. Also, communication bottlenecks can disrupt the FL process significantly, as shown in Fig. 6.

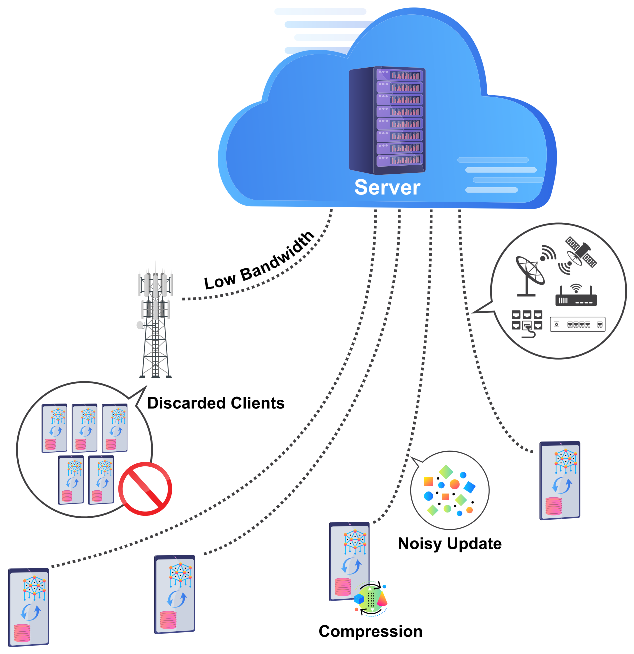

FIGURE 6. Overview of communications problems in FL.

5) FREE-RIDING ATTACKS

Free-riding attacks [82], [83] against FL consist of intentionally dissimulating participation in the FL process to obtain the final model without actually contributing to the training process. The free-riding clients play a passive role by either not updating the local model update parameters during the iterative federated optimization or inserting dummy updates without training the model with their local data. In this way, these clients benefits from the FL final product without contributing with their fair share of model training. This attack's impact can be particularly disastrous in a smaller landscape FL environment, where data is scarce and the model has high commercial value.

6) MAN-IN-THE-MIDDLE ATTACKS

Man-in-the-Middle attacks [84], [85] intercept the models exchanged between the clients and the server and replace them with malicious model updates. This attack is carried out through intervening with genuine networks or building fake networks that the attacker commands. Compromised traffic is often stripped of any encryption in order to steal, change or reroute these exchanged model updates to the attacker's destination of choice. Because adversaries may be quietly watching or re-encrypting hijacked traffic to its designed source once saved or modified, it can be a difficult attack to detect.

7) DROPOUT OF CLIENTS

FL assumes that participating clients, once selected, remain connected to the server during a training round. However, some clients will drop out when operating at a larger scale due to network issues, unexpected roadblocks, or otherwise becoming temporarily unavailable. Dropout of participating clients in training rounds may yield unproductive results and cause fairness issues when training the global model [86], [87]. Some of the dropped clients may have valuable or exclusive data entries needed to produce a high-quality deep learning model. FL requires the aggregation mechanism to be robust against the client dropouts' long-term effect. The problem of operating at scale when clients are anticipated to be intermittently available has a systematic consequence on FL analytics.

V. DEFENSES IN FEDERATED LEARNING

Defense methods help to protect FL against a diverse range of attacks and decrease the probability of risks. With data access being out of the question, most of these defenses prevent model corruption by ensuring that the trained model has learned to realize the actual training data's underlying statistical distribution. That does not mean we can necessarily prevent the model from being trained with some malicious data or updates. However, defenses often seek to ensure that the model does not overfit to these malicious updates, minimizing their overall influence on the global model.

There are two types of defenses, namely proactive and reactive. Proactive defense is a cost-effective way to figure out the threats and related risks prematurely. The reactive defense works by identifying the attack and taking precautionary measures. It is usually deployed as a patch-up in the production environment. FL exposes several new attack surfaces at training time, giving rise to complex and novel defenses. We review prominent types of FL defense techniques and analyze their effectiveness and limitations in this section. Table 3 presents an in-depth summary of the most popular defense methods in FL. Furthermore, Fig. 7 visualizes the discussed FL defenses used to ensure robustness against adversarial attacks.

A. ANOMALY DETECTION

Anomaly detection is considered a more proactive type of defense that explicitly detects malicious updates and prevents their impact on the system. In FL environments, attacks such as data poisoning and model poisoning can be discovered using anomaly detection techniques. One popular approach is measuring the test error rate of a given update and rejecting it if it does not improve the global model [88]. Although these types of anomaly detectors work well against untargeted adversarial attacks, they are most likely to fail when it comes to targeted backdoor attacks because training on backdoor data usually produces poisoned model updates that look and behave similarly to intact models [11].

The authors in [89] proposed AUROR, a defense mechanism that detects harmful client updates by using the K-Means clustering algorithm to separate all the clients into many groups based on their uploaded indicative features. For every indicative feature, it creates clusters of benign and malicious users. Another detection-based defense mechanism uses euclidean distance to expose any deviation in the model update parameters for each client [90]. A similar work [91] proposed identifying abnormal updates from clients in FL by generating low-dimensional surrogates of the model weight vectors.

One interesting type of anomaly detection is TRIM [92], which finds a subset of updates that reduces the global objective loss by removing outliers with large residuals. In a similar work, RONI (Reject On Negative Impact) [93] was proposed in the context of spam filters and tries to identify outliers by observing a model's performance trained with and without each point. RONI can be applied to FL by identifying outliers while observing a global model's performance aggregated with and without each model update. A combination of both previously cited defenses [60] identifies negatively impacting updates from clients. Furthermore, the research work in [94] extends spectral anomaly detection to FL. The low-dimensional embeddings of model updates generated using autoencoders [95] are expected to retain essential features that capture the data samples' inherent variability. After removing the noisy and repetitive features in the data instances, the embeddings of benign data instances and malicious data instances can be easily distinguished in low-dimensional latent space.

B. DIFFERENTIAL PRIVACY

Differential privacy is conceived initially to defend against privacy attacks, but it can be a practical defense against data poisoning [12], [96], [97]. It works by injecting statistical noise into the updates. The goal of differential privacy is to guarantee with high confidence that no single data record can be meaningfully distinguished from the rest. Intuitively, an attacker who can only modify a few training samples cannot cause a large change in the learned models' distribution. To prevent attacks, we add a small amount of noise that is empirically sufficient to restrict attacks' success.

The main weakness of differential privacy is that the noise injected into the learning procedure is added on top of the noise generated by the learning algorithm. The accumulated noise can contaminate the learned model.

TABLE 3. Summary of defense methods in federated settings.

Defenses	Description	Attacks Defended Against
Anomaly Detection	explicitly detect malicious updates and prevent their impact on the system	• Data Poisoning • Model Poisoning • Free-Riding Attacks
Differential Privacy	inject a small amount of statistical noise into the model updates that is empirically sufficient to restrict the success of attacks and minimizes the risk of gradient leakage from the model parameters	• Data Poisoning • Model Poisoning • Inference Attacks • Evasion Attacks
Trusted Execution Environment	a high-level trusted ecosystem for executing attested and verified code while maintaining confidentiality, authenticity, privacy, integrity, and data access rights	• Training Rules Manipulation • Compromised FL Distributed Computation • Malicious Server
Robust Aggregation	aggregation algorithms that can detect and discard faulty or malicious clients during training and sustain communications instabilities, clients dropout, erroneous model updates, and heterogeneous clients	• Non-Robust Aggregation • Data Poisoning • Model Poisoning • Backdoor Attacks • Dropout of Clients
Pruning	reduce the deep learning model's size by dropping neurons to decrease the complexity, improve the accuracy, and disable backdoors	• Backdoor Attacks • Communication Bottlenecks
Zero-Knowledge Proofs	cryptographic primitives used to verify that updates submitted by the clients are conforming to pre-specified properties without sharing or revealing underlying data	• Data Poisoning • Model Poisoning • Backdoor Attacks • Man-in-the-Middle Attacks
Adversarial Training	a minimax optimization problem, where the adversarial samples and the model parameters are alternatively updated	• Evasion Attacks
Federated Multi-Task Learning	learn models for multiple related tasks simultaneously to ensure fault tolerance	• Dropout of Clients • Communications Bottlenecks
Moving Target Defense	randomize FL system modules to reduce the likelihood of successful attacks and shorten attacks lifetime by adding new dynamics to the system	• Inference Attacks • Man-in-the-Middle Attacks • GAN Reconstruction Attacks
Recognizing Legitimate Clients	identify legitimate clients to dramatically reduce the success rate of poisoning attacks even when multiple adversaries are involved	• Data Poisoning • Model Poisoning • Backdoor Attacks
Federated Distillation	transfer knowledge from a fully trained teacher model to another student model. Knowledge sharing instead of weights sharing enhances the robustness of FL and saves communication and computation costs associated with FL	• Inference Attacks • Communication Bottlenecks • Man-in-the-Middle Attacks • GAN Reconstruction Attacks

Additionally, differential privacy offers robustness to poisoning attacks based on its group privacy property. Thus, this protection will exponentially diminish with a larger number of attackers. Most commonly, we enforce a norm constraint on the client model update on top of differential privacy (e.g., by clipping the client updates before adding noise) to exclude outliers, such as the aggregated model updates bound the variance in the global distribution.

Differential privacy has also been studied as a defense against inference and evasion attacks. By adding noise, we minimize the risk of gradient leakage from the model parameters. Furthermore, making the predictor itself differentially-private by adding noise and releasing the prediction with the highest probability limits the ability of the adversary to trick the learned model by manipulating test samples [98]. Further research is needed to determine the extent of differential privacy's usefulness as a defense technique beyond privacy preservation.

C. TRUSTED EXECUTION ENVIRONMENT

A Trusted Execution Environment (TEE) [99] is defined as a high-level trusted ecosystem for executing attested and verified code. TEE establishes digital trust by securing connected devices in FL. It protects them against injecting false training results by enabling an isolated, cryptographic electronic structure and permitting end-to-end security. This cybersecurity concept involves the execution of authenticated code, confidentiality, authenticity, privacy, integrity, and data access rights. TEE is applicable in FL [100], [101] and multi-party machine learning in general [102] to mainly defend against algorithmic attacks. TEE is a tamper-resistant isolated processing environment that provides the following amenities:

- Authentication: The legitimacy of a participating device should be verified by the associated service with which it is trying to enroll.
- Confidentiality: The state of code's execution remains concealed unless the corresponding party reveals a message.
- Integrity: The code's execution path cannot be altered unless it explicitly receives an input or a verified interruption.
- Data access rights: The data stored on and processed by clients is protected, and interactions between different

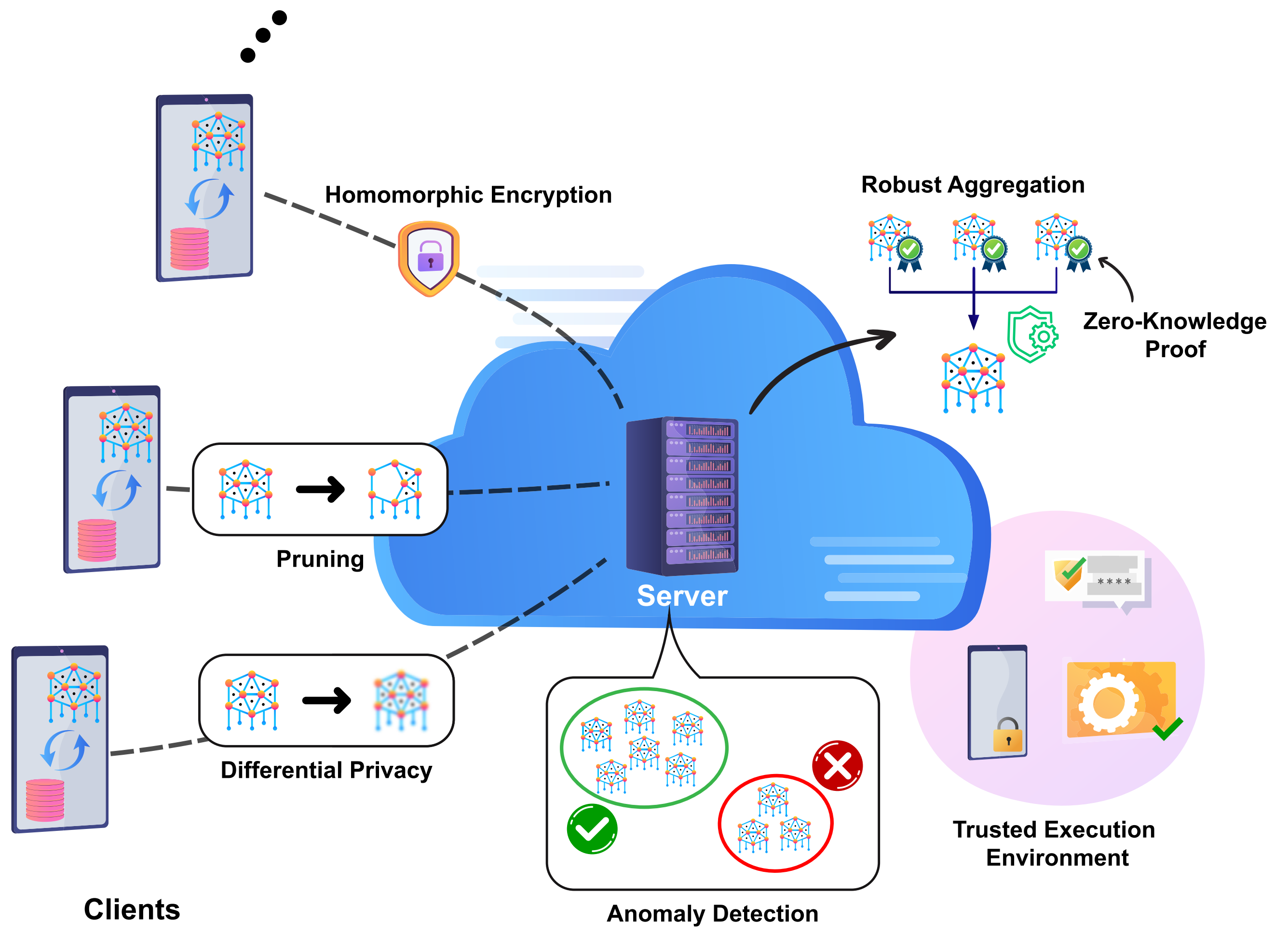

FIGURE 7. Most popular defense mechanisms to ensure FL security.

participating parties are executed securely. The TEE fully manages data access rights.

- Secure communication: Communications are secured using cryptographic methods. Private and public encryption keys are stored, maintained, and used only within the TEE secure environment.
- Attestation: The TEE can demonstrate to a remote party what code is currently executing and the initial state.

TEE can help solve a significant problem for FL security since it plays an increasingly central role in preventing hacking of the central server or clients, data breaches, and malware use.

D. ROBUST AGGREGATION

Aggregation algorithms security is fundamental to FL. There is a breadth of research on robust aggregation algorithms [103], [104] that can detect and discard faulty or malicious updates during training. Additionally, robust aggregation methods should be able to sustain communications instabilities, clients dropout, erroneous model updates on top of malicious actors [105]. Many of the state-of-the-art byzantine-robust aggregations rely on assumptions that are either unrealistic or do not hold in FL environments [18]. As a result, several novel aggregation methods have been proposed, such as adaptive aggregation. The latter combines repeated median regression with a reweighting scheme in iteratively reweighted least squares [71] and a novel robust aggregation oracle based on the classical geometric median [103]. This aggregation method is demonstrated to be robust against model update corruption in up to half the clients. Authors in [106] suggested using Gaussian distribution to measure clients' potential contributions. They also proposed layer-wise optimizing steps, so the aggregation works well on different functional units in the neural network. Experiments show that this aggregation mechanism outperforms the famous FedAvg in terms of convergence speed and global model robustness against attacks. Aggregation algorithms can also tackle the problem of heterogeneity of FL clients. FedProx [40] was introduced as a generalization and re-parametrization of FedAvg. In highly heterogeneous FL environments, FedProx demonstrates significantly more steady and accurate convergence behavior relative to FedAvg.

E. PRUNING

Pruning reduces the deep learning model's size by dropping neurons to decrease the complexity, improve the accuracy, and disable backdoors. In the FL ecosystem, clients are large in numbers and often connected to the server via unstable or expensive connections. Considering the limited computational power capacity on some edge devices, developers face a major problem when they have to train large-sized deep learning neural networks. Small local datasets in the clients do not need to be trained on the full global model to achieve good generalization. This is manifested by Federated Dropout (FD) [107] that allows clients to train their local datasets on smaller subsets of the global model. FD results in a decrease in both communication overhead and local computation. Furthermore, another pruning technique, known as PruneFL [67], performs adaptive pruning by maximizing the estimated empirical risk reduction divided by the time of a single FL round. Plus, pruning can be an effective defense against backdoor attacks [66]. It reduces the backdoored model's size by dropping neurons that are dormant on clean inputs, consequently disabling backdoor behavior. A stronger pruning-aware attack can evade the pruning defense by concentrating the clean and backdoor behavior into the same subset of activations. To defend against such a more potent attack variant, we can combine pruning with fine-tuning, which adds a small amount of local retraining on a clean training dataset. This combination was deemed the most effective in incapacitating backdoor attacks, in some cases reducing the backdoor task accuracy to zero.

F. ZERO-KNOWLEDGE PROOFS

Zero-Knowledge Proofs (ZKPs) [108] are cryptographic primitives used to verify statements by one party (known as the prover) to another party (known as the verifier) without sharing or revealing underlying data. MIT Researchers first started promoting the concept of zero-knowledge proofs in the mid 1980s [109]. Zero-knowledge protocols are probabilistic assessments, which implies they cannot prove something with the complete certainty that will uncover it. Instead, they provide small pieces of unlinkable knowledge that can accumulate to show that an assertion's validity is overwhelmingly presumable. Therefore, ZKPs provide an efficient solution for the verifiability problem on private data. In FL, ZKPs can ensure that clients are submitting model updates with prespecified properties to defend against model corruption and backdoor attacks, avoiding sharing private data. For instance, ZKPs can be used to validate that the model updates were trained correctly using data instances produced by a client's smartphone sensors. While ZKPs have plenty of compelling potentials to transform how FL systems privately and securely verify model updates' integrity, we should understand better how to effectively apply these techniques and spot flaws in how the components are composed and implemented. Nowadays, ZKPs protocols can attain proof sizes of hundreds of bytes and attestations of milliseconds' order regardless of the model updates' size.

G. ADVERSARIAL TRAINING

Adversarial training refers to a minimax optimization problem, where the adversarial samples and the model parameters are alternatively updated. Adversarial training [110] solves the minimax optimization problem. On the one hand, the inner maximization produces adversarial examples by maximizing the classification loss. On the other hand, the outer minimization obtains model parameters by minimizing the loss on adversarial samples generated from the inner maximization. This defense method provides robustness against evasion attacks to some extent [49], [111], [112]. However, there is no canonical form of adversarial training, and preferences such as the minimax problem formulation and training hyperparameters can significantly impair the trained model robustness. Moreover, adversarial training was originally developed for i.i.d data, and it is unclear how it performs for non-i.i.d data distribution [113]. In FL, it may require several epochs before reaching significant robustness. Worse than that, establishing proper bounds on the norm of perturbations to perform adversarial training is challenging in federated settings since training data cannot be inspected. Furthermore, adversarial training typically enhances robustness for the adversarial examples used during the training. Additionally, it often reuses adversarial examples that are expensive to generate [111], potentially stressing the limited computation resources of FL clients and leaving the trained model vulnerable to other types of adversarial noise [114], [115].

H. FEDERATED MULTI-TASK LEARNING

Federated multi-task learning [116]–[119] handles statistical and system challenges of FL like high communication costs, stragglers, and fault tolerance. In multi-task learning, the purpose is to learn models for multiple related tasks simultaneously. Federated multi-task learning can directly infer relationships amongst non-IID and unbalanced data, making it particularly well-suited for FL's statistical challenges. The most prominent example of this type of defense is MOCHA [116], a novel systems-aware optimization framework for federated multi-task learning. MOCHA speeds up convergence, is robust to statistical heterogeneity, and can handle clients periodically dropping out (fault tolerance).

I. MOVING TARGET DEFENSE

Moving Target Defense (MTD) [120]–[123] creates confusion for malicious adversaries by introducing a dynamic and continuously unfolding attack surface across various system dimensions. Its goal is to grow uncertainty and complicate attacks. That goal can be achieved by constantly moving the FL system's components in a randomized fashion. Moving components ensure that the adversary's information in the reconnaissance phase become stale during the attack phase, given we have moved to a new configuration within that time. A network-level moving target defense employs a variety of techniques such as DNS Redirect, IP hopping, Endpoint Information Shuffle, and Forwarding Path Migration to increase complexity and costs for attackers. As a result, this defense limits the vulnerabilities exposure and the possibilities for an attack. Also, it increases the system resiliency. MTD is designed to provide an end-to-end safeguard against the most damaging eavesdropping-based attacks. With this defense's dynamic protection, outsider adversaries will be unable to precisely identify the resources they require to target the FL training process.

J. RECOGNIZING LEGITIMATE CLIENTS

Different poisoning attacks, including data and model poisoning attacks, have been examined in a centralized FL setting for a long time. However, only a few works [44] investigate the poisoning attacks in a distributed environment where multiple malicious clients with the same attack strive to include poisoned training samples into the training procedures. Although distributed poisoning attacks are a more significant threat in FL, the effectiveness of distributed poisoning with multiple attackers is still unclear compared to traditional poisoning. This defense strategy operates by identifying genuine users and dramatically reducing the success rate of poisoning attacks even when multiple adversaries are involved.

K. FEDERATED DISTILLATION

Under restricted communication resources, exchanging model parameters becomes too costly, particularly for modern large deep neural networks. In this regard, federated distillation [124] is a compelling FL solution that only transfers the model outputs whose dimensions are commonly much smaller than the model sizes. One foundational algorithm of federated distillation is Knowledge Distillation (KD) [125]. It aims to transfer knowledge from a fully trained model (teacher model) to a smaller model (an empty student model). The teacher model's knowledge of KD can be constructed in several ways. Typically, the knowledge is a pre-trained teacher model's logits, transferred to a small student model for model compression. The knowledge can also be a collection of other student models' logits, in that the collection of forecasts is often more accurate than individual predictions. The notion of sharing knowledge instead of model weights can enhance the robustness of FL and saves communication and computations costs.

VI. PEER-TO-PEER FEDERATED LEARNING SECURITY

FL workflow can be realized with diverse topologies. The two most common ones are either the centralized approach via an aggregation server or peer-to-peer decentralized approaches [126]–[130]. As shown in Fig. 8, peer-to-peer FL does not need an aggregation server given that the training nodes exchange their partially trained models after each training round, and the aggregation happens on each node in parallel. Regardless of the topology, FL implicitly grants a certain degree of privacy, as FL participants cannot access other participants' private data and only obtain model parameters aggregated over several training nodes. In peer-to-peer FL, each client communicates directly with a subset of the other participants or all of them. In other words, there is no gatekeeper role. Thus, all training protocols must be agreed upon in advance between the different participants.

In the absence of a trusted third party who acts as the intermediary, coordination protocols' design necessitates significant agreement efforts and negotiations. Some participating parties may not have the best interests of other participants in mind, and establishing trust is not straightforward. However, in a trustless-based architecture such as peer-to-peer FL, the clients can be cryptographically locked into being honest by means of a secure protocol, but this may introduce high computational costs.

Peer-to-peer FL setup removes certain components that can create an active ground for vulnerabilities and threats. By eliminating the central server from FL architecture, there is no need to worry about attacks caused by a malicious server or hackers compromising the cloud security layers. Nevertheless, the overall FL system may require an independent entity to assume control and arbitration roles, which may not always be desirable due to additional expenses and procedural viscosity. Moreover, any changes to peer-to-peer communications and training protocols must be made in a synchronized fashion by all parties involved; otherwise, the system can fail in its entirety.

The unique characteristics of peer-to-peer FL also bring challenges such as preventing data leakage, ensuring data integrity when communicating models, and designing scheduling algorithms for distributed computation to reduce idle time. Indeed, data leakage poses a threat to peer-to-peer FL at a larger scale than centralized FL because clients receive model updates from their peers instead of receiving the global model from the server. Attackers can exploit the individual model updates to gain insights about the data and states of other participants. The inference attacks can be conducted without the need for eavesdropping on secure channels. Also, inspecting individual model updates reveals more information than the aggregated model. Hackers with expert knowledge of the communication network can launch attacks on the scheduling algorithm, causing increasing stragglers nodes or consensus issues.

Additionally, peer-to-peer FL systems have limited scalability due to the constraints in processing power and network bandwidth of participating hosts. Thus, it may not be suitable for cross-device FL, where we train a shared model with a huge number of clients (e.g., IoT devices, mobile applications). However, peer-to-peer topology is an excellent fit for cross-silo FL, where clients are typically of small-scale numbers, usually indexed, and almost always available for training. This scenario is usually applicable within organizations or groups of organizations to collaboratively train a machine learning model with their confidential data in the absence of a central authority.

In a trustless-based structure, the traditional defenses for centralized FL may not be compatible with peer-to-peer

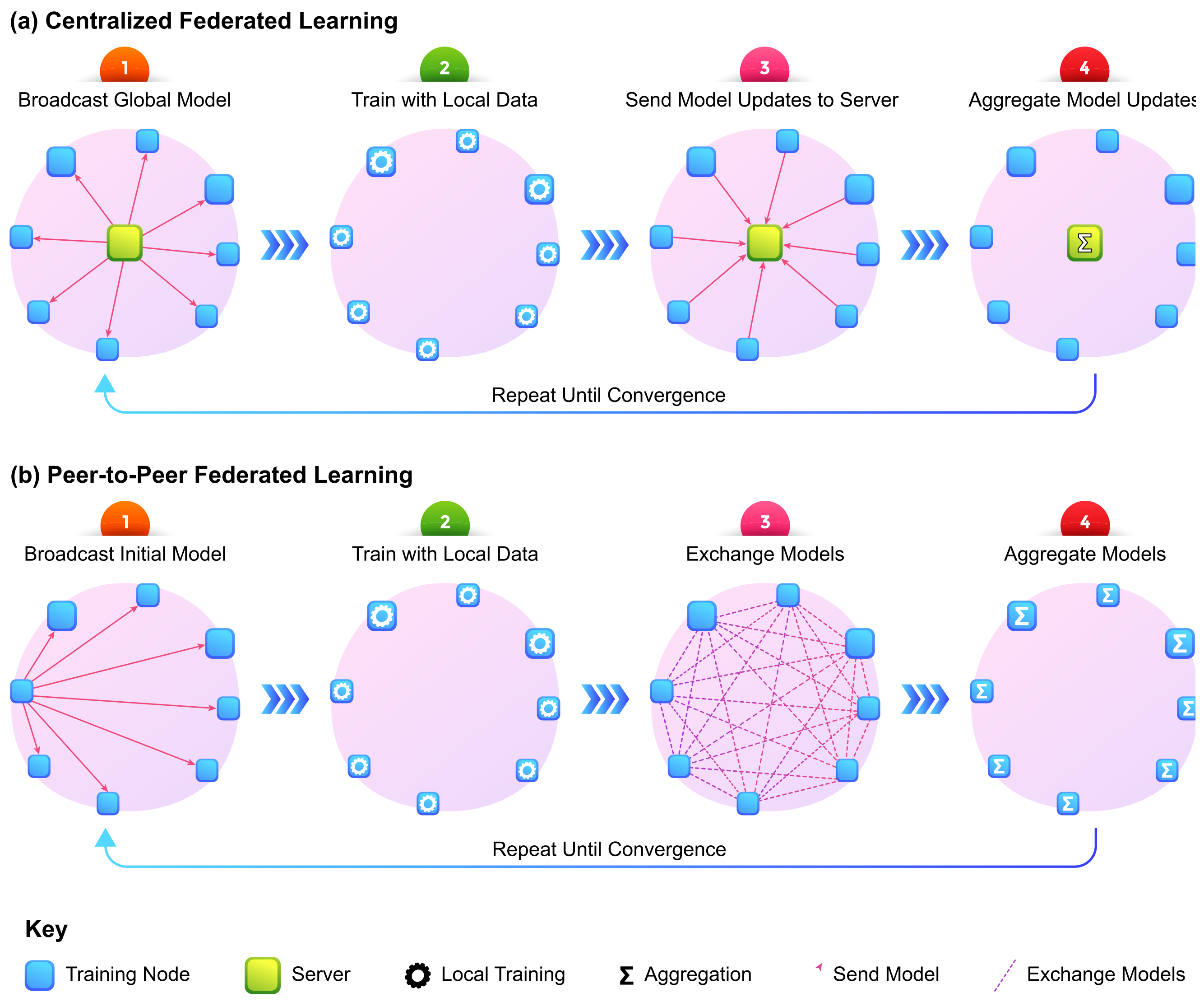

FIGURE 8. **(a) Centralized FL – Typical FL workflow in which a central server broadcasts the global model to a federation of training nodes, which perform local training, resubmit their trained model updates to the server intermittently for aggregation, and then restart training on the new global model that the server returns. (b) Peer-to-Peer FL – Decentralized FL workflow in which each training node exchanges its trained model updates with all or a fraction of its peers and aggregates received models on its own.**

topologies. There is no central authority that can be trusted to regulate and coordinate the machine learning efforts and ensure convergence. There is no longer a global state of the model as in standard FL. The peer-to-peer FL process is designed such that all local models in the training nodes converge to the desired global solution as the individual models gradually reach consensus. As previously mentioned, even in the decentralized FL setting, a central authority may still be in charge of establishing up the learning task. That poses many questions: Who decides the model architecture and training algorithms in this setting? What hyperparameters to use? Which party is responsible for monitoring and debugging when something goes wrong? How to resolve system conflicts and security breaches? A certain degree of trust of the participating parties should be established through a central authority that would still be needed to answer these questions. Alternatively, clients can collaboratively take decisions through a consensus scheme. In either way, each trust scheme comes with its own risks and vulnerabilities. All these factors take the security challenge of peer-to-peer FL to another dimension.

We discuss the key challenges in designing a peer-to-peer solution for secure multi-party machine learning and the critical pieces in a design that resolves each of these challenges. In a peer-to-peer FL setting, peering clients join and contribute to a federation system to train a global model, assuming that peers are willing to collaborate on building deep learning models but are unwilling to share their data. However, adversaries can collude or generate aliases to increase their impact in a sybil attack. We should ensure that an adversary cannot increase their influence over the system by creating multiple peers without increasing their total contribution.

In peer-to-peer settings, known baseline models are not available to participants, so we can validate a model update by evaluating it with respect to the models submitted by other clients. By observing a peer's model updates from each training round, an attacker can perform gradient leakage operations and retrieve details about his victim's training data. We can prevent such attacks using differentially-private updates and using consistent hashing based on proof of stake in combination with Verifiable Random Functions (VRFs). Selecting key roles helps arrange the privacy and security of peer model updates. We can prevent groups of colluding peers from overtaking the system if they do not have sufficient stake ownership after establishing a stake system (e.g., using blockchains). In this way, we can verify the updates of other peers without observing their un-noised versions.

The emergence of large-scale multi-party machine learning workloads and distributed ledgers for scalable consensus can offer practical solutions to peer-to-peer FL. With these strategies, peer-to-peer FL should be able to coordinate a collaborative learning process across more and more peers and produce a final model that is similar in utility to the state-of-the-art centralized FL. It should also have the ability to withstand poisoning and gradient leakage attacks, frequent failures, and an increased number of training nodes.

VII. FUTURE DIRECTIONS

The prospect of learning from private data while maintaining user privacy offers immense potentials in a variety of domains, such as education, healthcare, finances, insurance, and more. However, it leaves several open questions about how to reduce or safeguard the vulnerabilities in FL systems. Several vulnerabilities and attacks have been discussed in this paper. Model poisoning, backdoor attacks, and inference attacks are among the most common threats in FL. Continuous assessment of vulnerabilities is crucial to assure the security of FL systems. Ensuring and building trust among participating parties using cryptographic protocols, such as Zero-Knowledge Proofs and Trusted Execution Environments, may offer a vital safeguard against attacks and failures. Proactive forms of defenses such as anomaly detection and robust aggregation shield the global model from malicious updates. Differential privacy enhances the privacy and security guarantees of FL at a small cost. The trade-off between what can be shared and what cannot be shared should be weighted to balance the opportunities and the risks of crowd-sourcing the model. Trust between different participants in FL can be established through a quantitative measure. Additionally, we can filter information based on usability to reduce the uncertainties of compromised data existence. Traceability of the global model helps to expose hidden vulnerabilities through the FL lifecycle. Backward tracking capability can assist deployers in distinguishing which clients are behind any change in the aggregated model. Transparent tracing of the training process can be achieved through blockchain technologies or the notion of transactions [131], [132].

FL has a set of security challenges that need further review and research. Attacks against FL can be detected through anomaly detection or robust aggregation algorithms. However, it is unclear if all compromised participants have the same threat level when attacking the FL model. This direction is worth exploring to create a metric to understand different clients' attack capabilities. Furthermore, many previous works in FL do not consider the decomposition of neural networks in layers but treat them as black-box functions without internal structure. Apprehending the deep learning models at layer-level granularity can be the key to grasping new attacks and defenses. In particular, observations and conclusions made in [133] demonstrate that network layers behave differently when it comes to generalization. Authors set up a simple experiment where an eleven-layer MLP network composed of 10 identical layers is trained on a reduced MNIST dataset, such that training any of the ten layers in isolation results in 100% train accuracy. The experiment's purpose is to demonstrate that training with a single isolated layer will reveal its importance on the network. The first layers have shown a superior ability to promote generalization by achieving higher testing accuracy than the network's last layers. Therefore, layers are not made equal with respect to generalization. The above observation has triggered our attention to an important question: Can the future collaborative defenses operate on individual layers using *split learning* [134] rather than considering the whole model as a black-box?

FL is marketed based on a crowd-sourcing model. Despite this model being revolutionary and utilized through several domains, corporate collaboration project managers may remain skeptical regarding the adoption of FL to protect data privacy. Probably, it is not possible to firmly guarantee that all participating parties are considered trustworthy. However, we can eliminate many attacks by establishing the level of trust for each actor. Ensuring and building trust lessens the need for advanced counter-measures, falling back to standard collaborative projects' principles.

Major technology companies have already deployed FL in production, and a number of startups were founded on the basis of using FL to address privacy and data collection challenges in various industries. The breadth of works surveyed in this paper suggests that backdoor attacks are still representing a significant challenge for FL security. Motivated by the importance of defending against attacks that are hard to detect, we pose the question of whether there is a way to guarantee resilience against backdoors through intelligent model design. Besides, learning data distribution and checkpointing models can be explored as potential defense mechanisms for future FL systems. We argue that FL security is not binary and presents a range of threat models relevant under a variety of assumptions, each of which provides its own unique challenges. Federated systems should ensure that the data embodied in the model is protected against any potential future attacks.

It is noticeable that there are no simple and straightforward defenses to the attacks on the FL systems due to privacy restrictions. Model poisoning attacks are more likely to effectively contaminate the trained model. Server-based FL models are more vulnerable to backdoors insertion in later rounds of training nearing convergence. Thus, adaptive and proactive defense techniques should be explored to secure FL from adversaries. In FL systems that operate on larger scales, it might be impractical to establish an enforceable collaborative agreement. Some compromised clients may purposely try to deteriorate performance, bring the system down, or extract information from other parties. Hence, security strategies will be required to mitigate these risks, such as advanced encryption of model submissions, secure authentication of all parties, traceability of actions, differential privacy, verification systems, execution integrity, model confidentiality, and protections against suspect actions.

VIII. CONCLUSION

The ecosystem of federated networks is usually characterized by the decentralized learning of a shared model using a large number of personal devices. The adoption of FL may come to a halt if researchers fail to address the new security vulnerabilities resulting from shifting the training to personal devices and private organizations. In this paper, we conduct a comprehensive study on the security issues and defenses in the FL landscape. With the identification and classification of different threats, we hope to give new perspectives and bring the research community's attention towards building secure and robust FL environments suited for large-scale adoption. Also, we discussed the areas in which FL security can have promising horizons. FL is a fairly new concept in the machine learning community, which needs further research and advancements before deploying it in sensitive applications with confidence.

ACKNOWLEDGMENT

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Combat Capabilities Development Command Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

[1] J. Poushter. (Feb. 2016). *Smartphone Ownership and Internet Usage Continues to Climb in Emerging Economies, Pew Research Center's Global Attitudes Project*. [Online]. Available: https://www.pewresearch.org/global/2016/02/22/smartphone-ownership-and-internet-usage-continues-to-climb-in-emerging-economies/

[2] S. Frier. (Aug. 2019). *Facebook Paid Contractors to Transcribe Users' Audio Chats*. [Online]. Available: https://www.bloomberg.com/amp/news/articles/2019-08-13/facebook-paid-hundreds-of-contractors-to-transcribe-users-audio

[3] D. Phelan. (Apr. 2019). *Amazon Admits Listening to Alexa Conversations: Why it Matters*. [Online]. Available: https://www.forbes.com/sites/davidphelan/2019/04/12/amazon-confirms-staff-listen-to-alexa-conversations-heres-all-you-need-to-know

[4] J. Koneänẏ, H. Brendan McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," 2016, *arXiv:1610.02527*. [Online]. Available: http://arxiv.org/abs/1610.02527

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

[6] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Koneänẏ, S. Mazzocchi, H. Brendan McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," 2019, *arXiv:1902.01046*. [Online]. Available: http://arxiv.org/abs/1902.01046

[7] 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. 1–9, Jan. 2019.

[8] S. A. Rahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5476–5497, Apr. 2020.

[9] *Consumer Data Privacy in a Networked World: A Framework for Protecting Privacy and Promoting Innovation in the Global Digital Economy*, White House, Washington, DC, USA, 2013.

[10] P. Voigt and A. V. D. Bussche, *The EU General Data Protection Regulation (GDPR): A Practical Guide*, 1st ed. Cham, Switzerland: Springer, 2017.

[11] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, Jun. 2019, pp. 634–643.

[12] Y. Ma, X. Zhu, and J. Hsu, "Data poisoning against differentially-private learners: Attacks and defenses," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 4732–4738.

[13] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, vol. 108, Aug. 2020, pp. 2938–2948.

[14] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, Eds., 2019, pp. 14774–14784.

[15] Z. Ying, Y. Zhang, and X. Liu, "Privacy-preserving in defending against membership inference attacks," in *Proc. Workshop Privacy-Preserving Mach. Learn. Pract.*, 2020, p. 61–63.

[16] C. Ma, J. Li, M. Ding, H. H. Yang, F. Shu, T. Q. S. Quek, and H. V. Poor, "On safeguarding privacy and security in the framework of federated learning," *IEEE Netw.*, vol. 34, no. 4, pp. 242–248, Jul. 2020.

[17] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. Mcmahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, p. 1175.

[18] P. Kairouz, "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*. [Online]. Available: http://arxiv.org/abs/1912.04977

[19] H. Weng, J. Zhang, F. Xue, T. Wei, S. Ji, and Z. Zong, "Privacy leakage of real-world vertical federated learning," 2020, *arXiv:2011.09290*. [Online]. Available: http://arxiv.org/abs/2011.09290

[20] W. Wei, L. Liu, M. Loper, K.-H. Chow, M. Emre Gursoy, S. Truex, and Y. Wu, "A framework for evaluating gradient leakage attacks in federated learning," 2020, *arXiv:2004.10397*. [Online]. Available: http://arxiv.org/abs/2004.10397

[21] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–8.

[22] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Secur.*, 2019, p. 1–11.

[23] D. Froelicher, J. Ramón Troncoso-Pastoriza, J. Sa Sousa, and J.-P. Hubaux, "Drynx: Decentralized, secure, verifiable system for statistical queries and machine learning on distributed datasets," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3035–3050, 2020.

[24] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 19–38.

[25] *Information Technology Security Techniques Information Security Risk Management*, Standard ISO/IEC 27005, 2018.

[26] N. Papernot, M. Abadi, U. Erlingsson, I. J. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," in *5th Int. Conf. Learn. Represent.* 2017, pp. 1–8.

[27] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proc. Annu. Tech. Conf.*, Jul. 2020, pp. 493–506.

[28] Z. Zhang, T. Plantard, and W. Susilo, "Reaction attack on outsourced computing with fully homomorphic encryption schemes," in *Proc. Inf. Secur. Cryptol.*, H. Kim, Ed., 2012, pp. 419–436.

[29] A. Borovik and C. S. Yalçınkaya, "Homomorphic encryption and some black box attacks," in *Proc. Conf. Math. Softw.*, A. M. Bigatti, J. Carette, J. H. Davenport, M. Joswig, and T. de Wolff, Eds., 2020, pp. 115–124.

[30] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Proc. NIPS Workshop Private Multi-Party Mach. Learn.*, 2016, pp. 1–8.

[31] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 691–706.

[32] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, "Protection against reconstruction and its applications in private federated learning," 2018, *arXiv:1812.00984*. [Online]. Available: http://arxiv.org/abs/1812.00984

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

[34] D. Verma, S. Calo, and G. Cirincione, "Distributed AI and security issues in federated environments," in *Proc. Workshop Program 19th Int. Conf. Distrib. Comput. Netw.*, Jan. 2018, pp. 1–6.

[35] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-Robust distributed learning: Towards optimal statistical rates," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, vol. 80, Jul. 2018, pp. 5650–5659.

[36] J. Jang-Jaccard and S. Nepal, "A survey of emerging threats in cybersecurity," *J. Comput. Syst. Sci.*, vol. 80, no. 5, pp. 973–993, Aug. 2014.

[37] R. M. Jabir, S. I. R. Khanji, L. A. Ahmad, O. Alfandi, and H. Said, "Analysis of cloud computing attacks and countermeasures," in *Proc. 18th Int. Conf. Adv. Commun. Technol. (ICACT)*, Jan. 2016, pp. 117–123.

[38] K. N. Mallikarjunan, K. Muthupriya, and S. M. Shalinie, "A survey of distributed denial of service attack," in *Proc. 10th Int. Conf. Intell. Syst. Control (ISCO)*, Jan. 2016, pp. 1–6.

[39] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, Jul. 2020, pp. 5132–5143.

[40] T. Li, A. Kumar Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2018, *arXiv:1812.06127*. [Online]. Available: http://arxiv.org/abs/1812.06127

[41] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," 2020, *arXiv:2002.06440*. [Online]. Available: http://arxiv.org/abs/2002.06440

[42] A. Dutta, E. Bergou, A. Abdelmoniem, C.-y. Ho, A. Sahu, M. Canini, and P. Kalnis, "On the discrepancy between the theoretical analysis and practical implementations of compressed communication for distributed deep learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, Apr. 2020, pp. 3817–3824.

[43] H. Xu, C.-Y. Ho, A. M. Abdelmoniem, A. Dutta, E. H. Bergou, K. Karatsenidis, M. Canini, and P. Kalnis, "Compressed communication for distributed deep learning: Survey and quantitative evaluation," 2020. [Online]. Available: http://hdl.handle.net/10754/662495

[44] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "DBA: Distributed backdoor attacks against federated learning," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–9.

[45] N. Rieke, J. Hancox, W. Li, F. Milletarã, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, S. Ourselin, M. Sheller, R. M. Summers, A. Trask, D. Xu, M. Baust, and M. J. Cardoso, "The future of digital health with federated learning," *NPJ Digital Medicine*, vol. 3, no. 1, p. 119, Sep. 2020.

[46] L. David, J. Aràs-Pous, J. Karlsson, O. Engkvist, E. J. Bjerrum, T. Kogej, J. M. Kriegl, B. Beck, and H. Chen, "Applications of deep-learning in exploiting large-scale and heterogeneous compound data in industrial pharmaceutical research," *Frontiers Pharmacol.*, vol. 10, p. 1303, Nov. 2019.

[47] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017, *arXiv:1712.05526*. [Online]. Available: https://arxiv.org/abs/1712.05526

[48] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Årndiä, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Proc. Conf. Mach. Learn. Knowl. Discovery Databases*, H. Blockeel, K. Kersting, and S. Nijssen, Eds., 2013, pp. 387–402.

[49] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*. [Online]. Available: http://arxiv.org/abs/1706.06083

[50] D. Kang, Y. Sun, D. Hendrycks, T. Brown, and J. Steinhardt, "Testing robustness against unforeseen adversaries," 2019, *arXiv:1908.08016*. [Online]. Available: http://arxiv.org/abs/1908.08016

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

[52] L. Mu noz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, 2017, pp. 27–38.

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

[54] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 6106–6116.

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

[56] Y. LeCun, C. Cortes, and C. Burges, *MNIST Handwritten Digit Database*. Florham Park, NY, USA: ATT Labs, 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist

[57] T. Gu, B. Dolan-Gavitt, and S. Garg, "BadNets: Identifying vulnerabilities in the machine learning model supply chain," 2017, *arXiv:1708.06733*. [Online]. Available: http://arxiv.org/abs/1708.06733

[58] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," 2020, *arXiv:2003.02133*. [Online]. Available: http://arxiv.org/abs/2003.02133

[59] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Model poisoning attacks in federated learning," in *Proc. Workshop Secur. Mach. Learn.(SecML)*, 2018, pp. 1–9.

[60] 0 M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," in *Proc. 29th Secur. Symp.*, Aug. 2020, pp. 1605–1622.

[61] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 8635–8645.

[62] Z. Chuanxin, S. Yi, and W. Degang, "Federated learning with Gaussian differential privacy," in *Proc. 2nd Int. Conf. Robot., Intell. Control Artif. Intell.*, Oct. 2020, p. 296.

[63] Z. Sun, P. Kairouz, A. Theertha Suresh, and H. Brendan McMahan, "Can you really backdoor federated learning?" 2019, *arXiv:1911.07963*. [Online]. Available: http://arxiv.org/abs/1911.07963

[64] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-Y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," 2020, *arXiv:2007.05084*. [Online]. Available: http://arxiv.org/abs/2007.05084

[65] A. Huang, "Dynamic backdoor attacks against federated learning," 2020, *arXiv:2011.07429*. [Online]. Available: http://arxiv.org/abs/2011.07429

[66] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Proc. Res. Attacks Intrusions Defenses*, 2018, pp. 273–294.

[67] Y. Jiang, S. Wang, V. Valls, B. Jun Ko, W.-H. Lee, K. K. Leung, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," 2019, *arXiv:1909.12326*. [Online]. Available: http://arxiv.org/abs/1909.12326

[68] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*. [Online]. Available: http://arxiv.org/abs/1312.6199

[69] M. A. Ayub, W. A. Johnson, D. A. Talbert, and A. Siraj, "Model evasion attack on intrusion detection systems using adversarial machine learning," in *Proc. 54th Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2020, pp. 1–6.

[70] M. Carminati, L. Santini, M. Polino, and S. Zanero, "Evasion attacks against banking fraud detection systems," in *Proc. 23rd Int. Symp. Res. Attacks, Intrusions Defenses*, Oct. 2020, pp. 285–300.

[71] S. Fu, C. Xie, B. Li, and Q. Chen, "Attack-resistant federated learning with residual-based reweighting," 2019, *arXiv:1912.11464*. [Online]. Available: http://arxiv.org/abs/1912.11464

[72] L. Yu and L. Wu, "Towards byzantine-resilient federated learning via group-wise robust aggregation," in *Proc. Federated Learn. Privacy Incentive*, 2020, pp. 81–92.

[73] J. Tan, Y. C. Liang, N. C. Luong, and D. Niyato, "Toward smart security enhancement of federated learning networks," *IEEE Netw.*, vol. 35, no. 1, pp. 340–347, Jan./Feb. 2020.

[74] B. Zhao, K. Reddy Mopuri, and H. Bilen, "IDLG: Improved deep leakage from gradients," 2020, *arXiv:2001.02610*. [Online]. Available: http://arxiv.org/abs/2001.02610

[75] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 3–18.

[76] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, p. 603.

[77] 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 *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 739–753.

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

[79] M. Song, Z. Wang, Z. Zhang, Y. Song, Q. Wang, J. Ren, and H. Qi, "Analyzing user-level privacy attack against federated learning," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2430–2444, 2020.

[80] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, "Don't decay the learning rate, increase the batch size," 2017, *arXiv:1711.00489*. [Online]. Available: http://arxiv.org/abs/1711.00489

[81] S. U. Stich, "Local SGD converges fast and communicates little," 2018, *arXiv:1805.09767*. [Online]. Available: http://arxiv.org/abs/1805.09767

[82] J. Lin, M. Du, and J. Liu, "Free-riders in federated learning: Attacks and defenses," 2019, *arXiv:1911.12560*. [Online]. Available: http://arxiv.org/abs/1911.12560

[83] Y. Fraboni, R. Vidal, and M. Lorenzi, "Free-rider attacks on model aggregation in federated learning," 2020, *arXiv:2006.11901*. [Online]. Available: http://arxiv.org/abs/2006.11901

[84] B. Bhushan, G. Sahoo, and A. K. Rai, "Man-in-the-middle attack in wireless and computer networking—A review," in *Proc. 3rd Int. Conf. Adv. Comput., Commun. Autom. (ICACCA) (Fall)*, Sep. 2017, pp. 1–6.

[85] D. Wang, C. Li, S. Wen, S. Nepal, and Y. Xiang, "Man-in-the-middle attacks against machine learning classifiers via malicious generative models," *IEEE Trans. Dependable Secure Comput.*, early access, Sep. 1, 2018, doi: 10.1109/TDSC.2020.3021008.

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

[87] W. Huang, T. Li, D. Wang, S. Du, and J. Zhang, "Fairness and accuracy in federated learning," 2020, *arXiv:2012.10069*. [Online]. Available: http://arxiv.org/abs/2012.10069

[88] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Mach. Learn.*, vol. 81, no. 2, pp. 121–148, 2010.

[89] S. Shen, S. Tople, and P. Saxena, "A uror: Defending against poisoning attacks in collaborative deep learning systems," in *Proc. 32nd Annu. Conf. Comput. Secur. Appl.*, Dec. 2016, p. 508.

[90] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 119–129.

[91] S. Li, Y. Cheng, Y. Liu, W. Wang, and T. Chen, "Abnormal client behavior detection in federated learning," 2019, *arXiv:1910.09933*. [Online]. Available: http://arxiv.org/abs/1910.09933

[92] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 19–35.

[93] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia, "Exploiting machine learning to subvert your spam filter," in *Proc. 1st Usenix Workshop Large-Scale Exploits Emergent Threats*, 2008, pp. 1–8.

[94] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, "Learning to detect malicious clients for robust federated learning," 2020, *arXiv:2002.00211*. [Online]. Available: http://arxiv.org/abs/2002.00211

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

[96] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proc. 2016 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, p. 308–318.

[97] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2017, *arXiv:1712.07557*. [Online]. Available: http://arxiv.org/abs/1712.07557

[98] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 656–672.

[99] P. Subramanyan, R. Sinha, I. Lebedev, S. Devadas, and S. A. Seshia, "A formal foundation for secure remote execution of enclaves," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, p. 2435.

[100] F. Mo and H. Haddadi, "Efficient and private federated learning using TEE," in *Proc. EuroSys Conf.*, 2019. [Online]. Available: https://eurosys2019.org/wp-content/uploads/2019/03/eurosys19posters-abstract66.pdf

[101] 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," *Inf. Sci.*, vol. 522, pp. 69–79, Jun. 2020.

[102] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, "Oblivious multi-party machine learning on trusted processors," in *Proc. 25th Secur. Symp.*, Aug. 2016, pp. 619–636.

[103] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," 2019, *arXiv:1912.13445*. [Online]. Available: http://arxiv.org/abs/1912.13445

[104] M. Grama, M. Musat, L. Muñoz-González, J. Passerat-Palmbach, D. Rueckert, and A. Alansary, "Robust aggregation for adaptive privacy preserving federated learning in healthcare," 2020, *arXiv:2009.08294*. [Online]. Available: http://arxiv.org/abs/2009.08294

[105] F. Ang, L. Chen, N. Zhao, Y. Chen, W. Wang, and F. R. Yu, "Robust federated learning with noisy communication," *IEEE Trans. Commun.*, vol. 68, no. 6, pp. 3452–3464, Jun. 2020.

[106] Y. Lu and L. Fan, "An efficient and robust aggregation algorithm for learning federated CNN," in *Proc. 3rd Int. Conf. Signal Process. Mach. Learn.*, Oct. 2020, pp. 1–7.

[107] S. Caldas, J. Koneäny, H. Brendan McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," 2018, *arXiv:1812.07210*. [Online]. Available: http://arxiv.org/abs/1812.07210

[108] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *Proc. IEEE Symp. Secur. Privacy*, May 2013, pp. 238–252.

[109] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Proc. 17th Annu. ACM Symp. Theory Comput.*, 1985, p. 291.

[110] Y. Wang, X. Ma, J. Bailey, J. Yi, B. Zhou, and Q. Gu, "On the convergence and robustness of adversarial training," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, Jun. 2019, pp. 6586–6595.

[111] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–8.

[112] C. Xie, Y. Wu, L. V. D. Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 501–509.

[113] J.-H. Jacobsen, J. Behrmann, N. Carlini, F. Tramèr, and N. Papernot, "Exploiting excessive invariance caused by norm-bounded adversarial robustness," 2019, *arXiv:1903.10484*. [Online]. Available: http://arxiv.org/abs/1903.10484

[114] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1802–1811.
[115] F. Tramer and D. Boneh, "Adversarial training and robustness for multiple perturbations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–6.
[116] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4427–4437.
[117] S. Caldas, V. Smith, and A. Talwalkar, "Federated kernelized multi-task learning," in *Proc. SysML Conf.*, 2018. [Online]. Available: https://mlsys.org/Conferences/doc/2018/30.pdf
[118] R. Li, F. Ma, W. Jiang, and J. Gao, "Online federated multitask learning," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 215–220.
[119] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," 2020, *arXiv:2012.04221*. [Online]. Available: http://arxiv.org/abs/2012.04221
[120] R. Colbaugh and K. Glass, "Moving target defense for adaptive adversaries," in *Proc. IEEE Int. Conf. Intell. Secur. Informat.*, Jun. 2013, pp. 50–55.
[121] J. Xu, P. Guo, M. Zhao, R. F. Erbacher, M. Zhu, and P. Liu, "Comparing different moving target defense techniques," in *Proc. 1st ACM Workshop Moving Target Defense*, 2014, pp. 97–107.
[122] C. Lei, H.-Q. Zhang, J.-L. Tan, Y.-C. Zhang, and X.-H. Liu, "Moving target defense techniques: A survey," *Secur. Commun. Netw.*, vol. 2018, pp. 1–25, Jul. 2018.
[123] S. Sengupta, A. Chowdhary, A. Sabur, A. Alshamrani, D. Huang, and S. Kambhampati, "A survey of moving target defenses for network security," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1909–1941, 3rd Quart., 2020.
[124] H. Seo, J. Park, S. Oh, M. Bennis, and S.-L. Kim, "Federated knowledge distillation," 2020, *arXiv:2011.02367*. [Online]. Available: http://arxiv.org/abs/2011.02367
[125] D. Li and J. Wang, "FedMD: Heterogenous federated learning via model distillation," 2019, *arXiv:1910.03581*. [Online]. Available: http://arxiv.org/abs/1910.03581
[126] A. Guha Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "BrainTorrent: A Peer-to-Peer environment for decentralized federated learning," 2019, *arXiv:1905.06731*. [Online]. Available: http://arxiv.org/abs/1905.06731
[127] A. Lalitha, O. Cihan Kilinc, T. Javidi, and F. Koushanfar, "Peer-to-peer federated learning on graphs," 2019, *arXiv:1901.11173*. [Online]. Available: http://arxiv.org/abs/1901.11173
[128] Y. Liu, Z. Ai, S. Sun, S. Zhang, Z. Liu, and H. Yu, "Fedcoin: A peer-to-peer payment system for federated learning," in *Proc. Federated Learn.*, 2020, pp. 125–138.
[129] X. Liu, H. Li, G. Xu, R. Lu, and M. He, "Adaptive privacy-preserving federated learning," *Peer-Peer Netw. Appl.*, vol. 13, no. 6, pp. 2356–2366, Nov. 2020.
[130] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *J. Healthcare Informat. Res.*, vol. 5, no. 1, pp. 1–19, Mar. 2021.
[131] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020.
[132] M. Shayan, C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Biscotti: A blockchain system for private and secure federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1513–1525, Jul. 2021.
[133] S. Carbonnelle and C. De Vleeschouwer, "Layer rotation: A surprisingly powerful indicator of generalization in deep networks?" 2018, *arXiv:1806.01603*. [Online]. Available: http://arxiv.org/abs/1806.01603
[134] Y. Gao, M. Kim, S. Abuadbba, Y. Kim, C. Thapa, K. Kim, S. A. Camtep, H. Kim, and S. Nepal, "End-to-End evaluation of federated learning and split learning for Internet of Things," in *Proc. Int. Symp. Reliable Distrib. Syst. (SRDS)*, Sep. 2020, pp. 91–100.

NADER BOUACIDA received the B.Eng. degree in telecommunications from the Higher School of Communication of Tunis, in 2015, and the M.S. degree in computer science from the King Abdullah University of Science and Technology, in 2017. He is currently pursuing the Ph.D. degree in computer science with the University of California at Davis. His research interests include data-driven approaches in networking, federated learning optimization and security, and developing machine learning solutions for healthcare applications.

Mr. Bouacida awards and honors include the King Abdullah University of Science and Technology Fellowship, from 2015 to 2017 and the University of California at Davis Fellowship, in 2017.

PRASANT MOHAPATRA (Fellow, IEEE) received the Ph.D. degree from Penn State University, in 1993. He is serving as the Vice Chancellor for Research at the University of California at Davis. He is currently a Distinguished Professor with the Department of Computer Science. He was the Department Chair of Computer Science, from 2007 to 2013. His research has been funded through grants from the National Science Foundation, U.S. Department of Defense, U.S. Army Research Laboratory, Intel Corporation, Siemens, Panasonic Technologies, Hewlett Packard, Raytheon, and EMC Corporation. His research interests include wireless networks, mobile communications, cybersecurity, and Internet protocols. He has published more than 350 articles in reputed conferences and journals on these topics.

Dr. Mohapatra is a Fellow of AAAS. He received the Outstanding Engineering Alumni Award, in 2008. He also received the Outstanding Research Faculty Award from the College of Engineering, University of California at Davis and the HP Labs Innovation Awards, from 2011 to 2013.