IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 3 April 2024, accepted 12 May 2024, date of publication 24 May 2024, date of current version 6 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3404948

SURVEY

A Security-Oriented Overview of Federated Learning Utilizing Layered Reference Model

JIAXING LU[1], NORIHIRO FUKUMOTO[1], AND AKIHIRO NAKAO[2], (Member, IEEE)
[1]Graduate School of Interdisciplinary Information Studies, The University of Tokyo, Bunkyo-ku, Tokyo 113-0033, Japan
[2]Graduate School of Engineering, The University of Tokyo, Bunkyo-ku, Tokyo 113-0033, Japan

Corresponding author: Akihiro Nakao (nakao@nakao-lab.org)

This work was supported in part by NICT, Japan under Grant JPJ012368C01901 and in part by JST ASPIRE, Japan under Grant JPMJAP2323.

ABSTRACT With the continuous development of Artificial Intelligence (AI), AI services are becoming increasingly influential in society, affecting both individual lives and enterprise production. However, the field of AI model training grapples with a conflict between the desire to maximize the utilization of private and external data, and the necessity to limit collaborative data sharing for privacy protection. Strict regulations on sensitive data give rise to data silos, impeding the smooth flow of information as well. In response to these challenges, Federated Learning (FL) emerges as a promising solution, enabling collaborative machine learning model training across isolated data silos. Despite its potential, securing FL systems is still challenging, primarily due to the absence of a canonical reference model that hierarchically summarizes existing works in this field. This lack complicates users' understanding of federated learning in the context of data flow and impedes their ability to pinpoint specific security issues and corresponding solutions when utilizing an FL system. To address this gap, we propose a layered reference model for federated learning through a comprehensive survey. The model encompasses five layers: the data interaction layer, client management layer, local model layer, network transmission layer, and remote management layer. Prioritizing the promotion of system security awareness, we conduct a threat analysis for each layer and explore corresponding defense strategies drawn from existing techniques. As a result, readers can gain insights into the federated learning system from the perspective of data flow, comprehend the security risks their private and sensitive data might encounter at each step, and explore relevant solutions to safeguard their information.

INDEX TERMS Federated learning, reference model, information security, privacy protection.

I. INTRODUCTION

The rapid growth of mobile communication technology has resulted in a surge in data generation and traffic, with predictions indicating that data traffic will constitute over 70% of global Internet connections, with 45% attributed to mobile connectivity (3G, 4G, or 5G) by 2023, according to statistics from Cisco [1]. The advent of 5G enables the provision of wired and mobile broadband services anytime and anywhere for users [2]. Besides the improvement of network transmission, end devices like smartphones and Internet of Things (IoT) sensors are continuously improving in performance, allowing for more complex crowdsensing tasks, such as medical applications [3]. However, the substantial amounts of data generated pose challenges for both network transmission and data processing within the system.

In parallel, Artificial Intelligence (AI) has demonstrated significant capabilities in handling large volumes of data, leading to the increasing importance of cloud computing-based AI in modern society [4]. However, the mandatory requirement to transfer dispersed original data to a central cloud server for model training presents challenges for further application, encompassing issues like poor private data protection [5], [6], weak data source robustness [7], and high transmission bandwidth requirements, etc [8], [9].

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

In contrast, Federated Learning (FL), introduced by Google in 2016, addresses these challenges by enabling collaborative AI model training on distributed data without transmitting the data to a central cloud server [10], [11], [12]. In FL, models are trained locally on each end device, and only model update information is transmitted to a central server for aggregation. This process ensures user privacy, and reduces the burden of transmitting vast data volumes over public networks. Furthermore, FL could integrate data from different silos, expanding the data sources and enabling the extraction of more profound insights.

While security threats in distributed learning like FL have been identified by Baruch et al. [13], existing surveys often follow tree-like structures, as illustrated in Figure 1 [14]. This poses challenges for users to understand FL in the context of data flow, making it difficult to narrow down the scope to specific security issues and corresponding solutions when utilizing a federated learning system. In general, current work has the following issues:

- Lacking a step-by-step and clearly defined layered summary based on the data flow throughout the entire federated learning system.
- An explicit system information security threat analysis along the data flow is also missing.
- Simply listing some security methods rather than delving into specific solutions for the various security threats faced by users.

To fill this gap, we propose a 5-layer reference model from a security-oriented perspective based on the data flow, from data generation to model training, in the federated learning system and conduct a security analysis of each layer according to it. We also introduce some defense strategies to secure the system for users. We aspire for this survey to serve as a valuable resource for readers to enhance the information security of their FL system, with the following specific contributions:

- We propose a security-oriented layered reference model for the federated learning system based on the flow of data. This model divides the entire FL system into five layers: the data interaction layer, client management layer, local model layer, network transmission layer, and remote management layer.
- We analyze the security threats that FL encounters at each layer and introduce appropriate defense strategies based on existing techniques.
- Readers can gain insights into the federated learning system from the perspective of data flow. They can understand the types of security risks their private sensitive data might encounter at each step and explore corresponding solutions to safeguard their private information.

The rest of this paper is organized as follows: Section II introduces related work. Section III defines and outlines the constitution of federated learning after showing a definition by us. Our proposed layered reference model is presented in Section IV. In Section V, we discuss common threats, and Section VI analyzes the system security risks layer by layer. Corresponding defense strategies are outlined in Section VII, and challenges and the conclusion are discussed in Section VIII.

II. RELATED WORK

Federated Learning is evolving rapidly, emerging as a promising approach for addressing critical data privacy and confidentiality. While there are many existing survey works on FL, and some of them specialize in privacy and security, they merely list common security issues of FL rather than providing a systematic analysis based on the data flow utilizing a reference model. This approach limits the depth of knowledge conveyed to readers, as a comprehensive exploration of security concerns and vulnerabilities in federated learning necessitates a more nuanced and detailed examination within the data flow architecture. Table 1 presents a comprehensive summary of some recent surveys, offering a comparative analysis of their distinctive characteristics.

Distinguished from other prevailing surveys, our survey delves into the intricate realms of security and privacy in federated learning, extending along the trajectory of data flow. What sets our survey apart is the proposal of a security-oriented layered reference model, which serves as a scaffold for unveiling its underlying security issues.

Furthermore, we introduce corresponding countermeasures aligned with each layer of the reference model. This strategic approach equips readers not only with theoretical knowledge but also with pragmatic solutions. Consequently, when confronted with security issues, readers can judiciously narrow down their focus and select the most pertinent countermeasures.

III. OVERVIEW OF FEDERATED LEARNING

In this section, we delve into a comprehensive overview of FL, offering an exploration that spans its definition, the pivotal components constituting its architecture, and the common federated training process.

A. DEFINITION

Based on the concept put forth by Google [10], [11], [12], and considering the perspectives of Yang et al. and Kairouz et al. [26], [27], we define federated learning as follows:

Federated learning is a decentralized machine learning setting where original data are distributed across multiple client compute nodes, such as enterprise data centers or individual smart devices [26]. In this collaborative framework, client compute nodes collaboratively train a machine learning model, either globally shared or personalized [14], under the coordination of an aggregation control server, either synchronously or asynchronously [27], [28].

Compared to traditional centralized machine learning, FL prioritizes privacy protection by keeping raw data on client compute nodes, minimizing data leakage risks [26], [27]. However, it is crucial to recognize some inherent challenges in practical federated learning models, especially

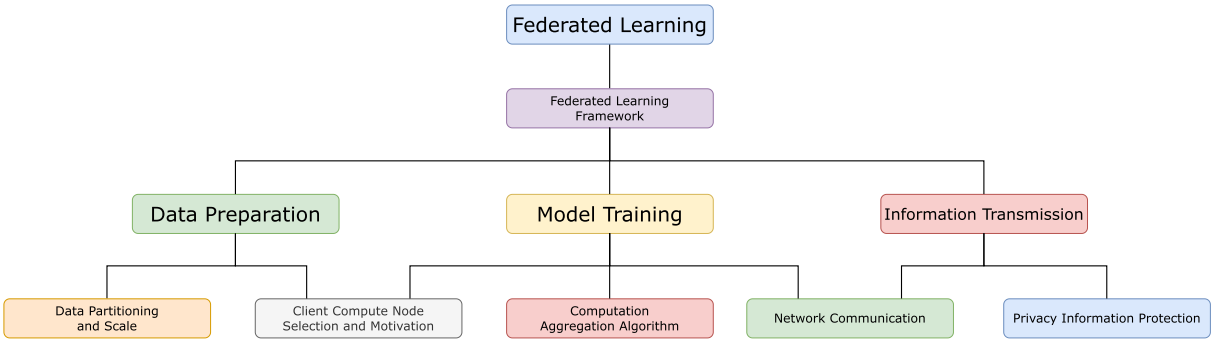

FIGURE 1. Tree-like structure in current surveys of federated learning [14].

in terms of model evaluation, where accuracy is a notable concern. While federated learning models are anticipated to surpass models trained individually on local devices, they might not reach the same performance level as those trained in a centralized approach [14]. This trade-off is essential to maintain privacy while effectively leveraging distributed data sources.

B. CONSTITUTION

Referring to the study by Li et al. [14], we define three major components in federated learning, containing two hardware components and one software component: the aggregation control server, client compute node, and the computation communication framework that connects the aggregation control server and client compute nodes together, as illustrated in Figure 2.

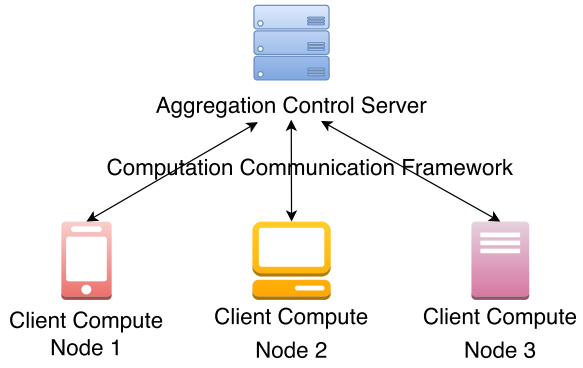

FIGURE 2. Three major components in federated learning.

- **Aggregation Control Server:** The aggregation control server orchestrates the federated learning process, acting as a central controller in a manner similar to centralized schemes. It manages the entire model training process and communications between two client compute nodes or between client compute nodes and itself. Alternatively, in decentralized schemes, like SimFL [29], client compute nodes update their model over the model parameter space by aggregating information through communication with their one-hop neighbors, acting in both client compute node and aggregation control server roles at the same time [30].
- **Client Compute Node:** Client compute nodes, spanning from smartphones to data centers, function as both the owners of original data and users of the trained model. They possess non-Independent Identically Distributed (non-IID) private data generated from daily use, irrespective of the scale of data distribution [14], [27]. This implies that the local private data stored on a specific client compute node cannot represent the entire data distribution [31], posing challenges for analysis due to differing data distributions among nodes.
- **Computation Communication Framework:** The computation communication framework facilitates communication and parameter transmission between client compute nodes and the aggregation control server, supporting local model training on client compute nodes and gradient aggregation on the aggregation control server. It plays a crucial role in the entire model training process, incorporating tasks like client selection mechanism, for example, FedCS, incentive mechanism, aggregation mechanism, and transmission mechanism [32].

C. COMMON FEDERATED TRAINING PROCESS

We summarize the common training processes of federated learning models based on the research by Kairouz et al. [27], encompassing both centralized and decentralized scenarios, as a complement to our proposed layered reference model.

1) **Node Selection:** In centralized federated learning, the central server, which serves as the aggregation control server, selects a group of client compute nodes who meet eligibility requirements. While in decentralized federated learning, the aggregation control server can be a normal client compute node that initiates the training process. The selection mechanism of client compute nodes is based on various criteria, such as battery status, computational resources, and other relevant factors.
2) **Initial Model Broadcast:** The aggregation control server disseminates the initial global model weights

TABLE 1. Comparisons among some recent surveys on federated learning.

References	Published Date	Brief Content	Security Related	Providing Model
Lim W. et al. [15]	April 2020	Introduce background and fundamentals of FL, highlight the challenges of implementation and review existing solutions, present the applications, challenges, and future research directions	Yes	No
Li T. et al. [16]	May 2020	Discuss the characteristics and challenges of FL, provide a overview of current approaches, and outline several directions	Briefly introduce three protection mechanisms	No
Aledhari M. et al. [17]	July 2020	Software and hardware platforms, protocols, real-life applications and use-cases of FL	No	No
Mothukuri V. et al. [18]	October 2020	Description of approaches and various implementation styles with an examination of the current challenges in FL	Yes	No
Xu J. et al. [19]	December 2020	Solutions to the statistical challenges, system challenges, and privacy issues in FL, and the implications and potentials in healthcare	Partially related	No
Zhang C. et al. [20]	January 2021	Introduce FL from: data partitioning, privacy mechanism, machine learning model, communication architecture and systems heterogeneity	Only some privacy protection means	No
Li Q. et al. [14]	November 2021	Introduce definition of FL, and provide a categorization to six aspects	Briefly mention privacy mechanisms	No
Gosselin R. et al. [21]	September 2022	Privacy and security risks of FL, state-of-the-art approaches to counteract problems	Yes	No
Coelho. K. K. et al. [22]	May 2023	Data security and privacy applications for FL in Internet of Healthcare Things networks	Yes	No
Hasan J. [23]	July 2023	Comprehensive taxonomy of security and privacy challenges in FL	Yes	No
Gabrielli E. et al [24]	August 2023	Summarize existing decentralized FL approaches, identify emerging challenges and promising research directions	Partially related	No
Kandati D.R. et al. [25]	August 2023	Examine FL's privacy and security concerns, and deal with several issues	Yes	No
Ours		Security-oriented survey of security risks for FL utilizing layered reference model	Yes	Yes

to the selected client compute nodes through the computation communication framework.

3) **Model Training:** Each selected client compute node independently trains and computes model updates using its sensitive private data.
4) **Update Aggregation:** After several local iterations, the aggregation control server collects and aggregates updates from the selected client compute nodes via the computation communication framework. This step often incorporates security and efficiency algorithms to ensure data privacy.
5) **Model Update:** The aggregation control server updates the global model based on the aggregated the update information, under the instruction of the aggregation mechanism offered by the computation communication framework.
6) **Evaluate and Repeat:** Engineers evaluate the global model's performance and determine whether to continue training or terminate the process and deploy the model. If training continues, the above steps are repeated. Some high-quality client compute nodes would be incentivized if an incentive mechanism

is introduced in the computation communication framework.

IV. LAYERED REFERENCE MODEL

A. MOTIVATIONS AND METHODOLOGIES IN THE PROPOSAL

It is well-accepted that federated learning is a secure variant of the distributed system that addresses the data silos issue while keeping private data decentralized, involving different kinds of participants with diverse requirements and constraints [27], [33], [34]. However, Lo et al. emphasize the lack of references for an end-to-end federated learning framework, attributing it to the absence of a unified reference model [34]. We agree that such an absence impedes the further development of FL and may cause some repetitive work. Motivated by the willingness to fill this gap, we decide to design a general federated learning system reference model.

Based on the empirical summarization collected through our systematic literature review about federated learning from different fields, especially survey papers mentioned but not limited to those in Section II, and experiences from conducting FL-related experiments, we arrive at a preliminary requirement, which is to separate the problems as much as possible to reduce the coupling of the system. Second, for the sake of addressing diverse issues and requirements of different participants, standardization work should be carried out. Third, the reference model must be able to provide an easier and clearer view for checking data privacy protection, since privacy protection is the most important principle in FL [21], [35].

Hoping to alleviate the work of deploying FL and inspired by the Open Systems Interconnection (OSI) 7-layer network model that is dominant in the current networking field [36], we consider a layered model as the optimal solution for us to satisfy these three requirements summarized above. With the assistance of the layered model, engineers could focus on only a few separated problems at one time while using standardized interfaces between two adjacent layers. Moreover, scalability and flexibility are improved as different issues are isolated in different layers. Lastly, the data flow becomes clearer under the layered model, which will benefit security threat analysis.

B. OVERVIEW OF THE MODEL

In response to the gap mentioned in Section IV-A and aiming to simplify the work while hoping to standardize the development of FL, we propose a security-oriented 5-layer federated learning system reference model, as shown in Figure 3. This model organizes federated learning system functions into distinct layers along the data flow and control flow, enhancing the standardization level, scalability, flexibility, and comprehensibility of federated learning system research and implementation.

The proposed reference model comprises the following layers, organized from the orientation of security along the data flow: data interaction layer, client management layer, local model layer, network transmission layer, and remote management layer. Each lower layer provides foundational services and information to the upper layer, while the upper layer offers management and reply functions to the lower layer.

C. DEFINITION OF EACH LAYER

In this subsection, we will define and elucidate the roles of each layer in the layered reference model, aiming to provide readers with a clear understanding.

- **Data Interaction Layer:** This layer functions as the physical device layer, encompassing client compute nodes where users deploy applications to satisfy their business requirements. It serves as the source of original data, presenting model inference results to users, and collecting node status information for the remote management layer to determine training eligibility. Essentially, it acts as the interactive interface between users/applications and the federated learning system.
- **Client Management Layer:** This layer focuses on data processing, addressing variations in data formats and structures through data normalization and standardization to enhance model accuracy and reliability [37], [38]. Additionally, it executes client selection decisions made by the remote management layer, as it is impractical to include every client compute node that requests to join in the training process. It also handles client incentive decisions and evaluates client join applications based on factors such as throughput, computation capability, and wireless resources [32], [39].
- **Local Model Layer:** This core layer is pivotal in the federated learning system. It employs approaches like gradient descent algorithms to train a local model on selected client compute nodes using their local processed data. After the local model computes, gradient information is forwarded to the network transmission layer. The layer also deploys models issued from the remote management layer and handles model inference tasks.
- **Network Transmission Layer:** The network transmission layer establishes communication links between peers, the client compute node, and the aggregation control server in a centralized scheme, or the client compute node's one-hop neighbor in a decentralized scheme. It encapsulates and decapsulates information, transmitting updated model gradient information from selected client compute nodes to the remote aggregation control server for global model updates. The communication link may vary from traditional Ethernet to advanced 5G networks. After global model aggregation, it issues the updated model to selected client compute nodes and transmits control information between the client and the server.

  However, this transmission step is susceptible to private information leakage, as highlighted by

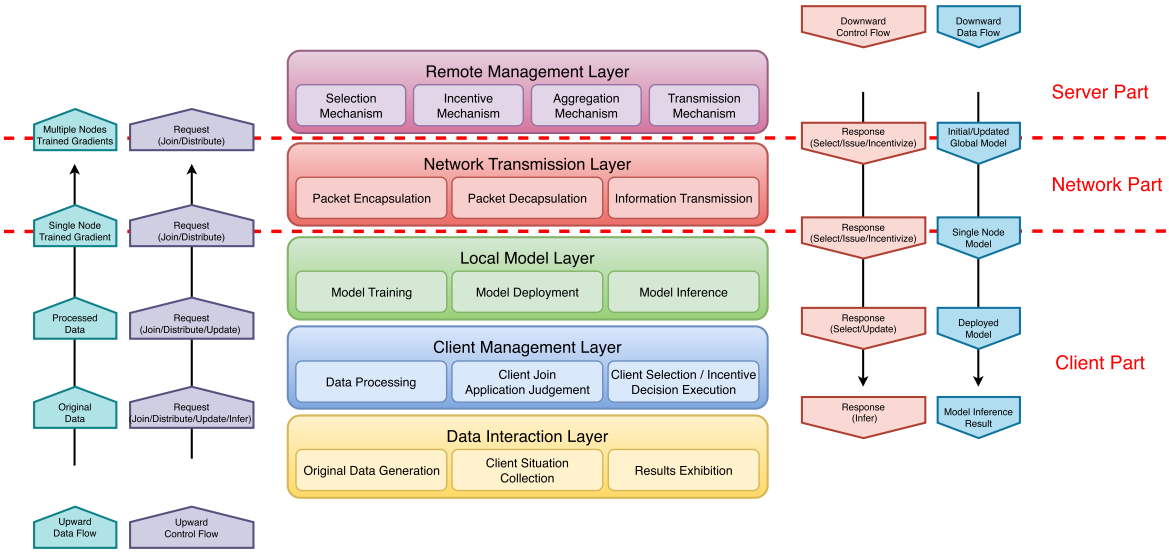

FIGURE 3. Proposal of 5-layer federated learning reference model.

Zhang et al. [40]. As a result, the network transmission layer plays a crucial role in safeguarding transmitted data. It could apply security mechanisms such as encryption, differential privacy, and homomorphic encryption, etc. to enhance data security before transmission to address this issue [41], [42], [43], [44]. On the receiver side, the network transmission layer would restore the data for utilization. It is important to note that adopting these protection mechanisms may introduce additional complexities in communication and computation [45].

- **Remote Management Layer:** The remote management layer is another crucial functional layer in federated learning. Its primary role is to aggregate the updated gradient information from selected client compute nodes, using it as the input to update the global model on the aggregation control server. Additionally, this layer addresses fairness concerns arising from the diverse distribution of source data among participating client compute nodes. It acts as the decision-making hub for client selection and incentive allocation, while also playing a key role in enhancing the security of the transmission phase.
- **Data Flow and Control Flow:** Data flow and control flow are two essential types of information movement within the layered reference model. Upward data flow involves the processing of data, while downward data flow deals with the deployment and utilization of the trained model. Upward control flow focuses on the request flow initiated by end applications, while downward control flow involves responses from the remote management layer. The low-level Data Flow Diagram (DFD) based on our model, from the perspective of DeMarco et al. [46], is depicted in Figure 4.

We believe the 5-layer reference model we propose, although in its early stages, holds the potential to offer a standardized and open reference interface for researchers in the federated learning community. However, we acknowledge the need for refinement and expansion in future endeavors.

D. APPLICATION CASE EXAMPLE

To illustrate the practical relevance and applicability of our proposed security-oriented 5-layered federated learning reference model, in this subsection, we will present an application case to readers as an example and explain it using our model.

Assume the background of the application case is as follows: several financial enterprises in a city hope to establish a cross-enterprise personal credit AI model to better serve the citizens. These enterprises include insurance companies, commercial banks, and government tax departments. While their customers belong to the same group, sharing customer data is impossible due to individual privacy protection requirements. Fortunately, federated learning could assist them in achieving this mission. However, the deployment is complex since there are no unified design standards in place.

This task would become simpler if they adopt our proposed layered model layer by layer. According to our model, two flow paths must be followed to complete the design of an FL model training system: bottom-up data flow and top-down control flow.

- **Bottom-up Data Flow:**
  - -- Data Interaction Layer: These three participants need to separately select the original data for model training, negotiate information collected for client situation monitoring, and determine the desired output.
  - -- Client Management Layer: The only objective is to set the unified output standards for data processing.
  - -- Local Model Layer: Each enterprise needs to deploy some servers for local model training, model

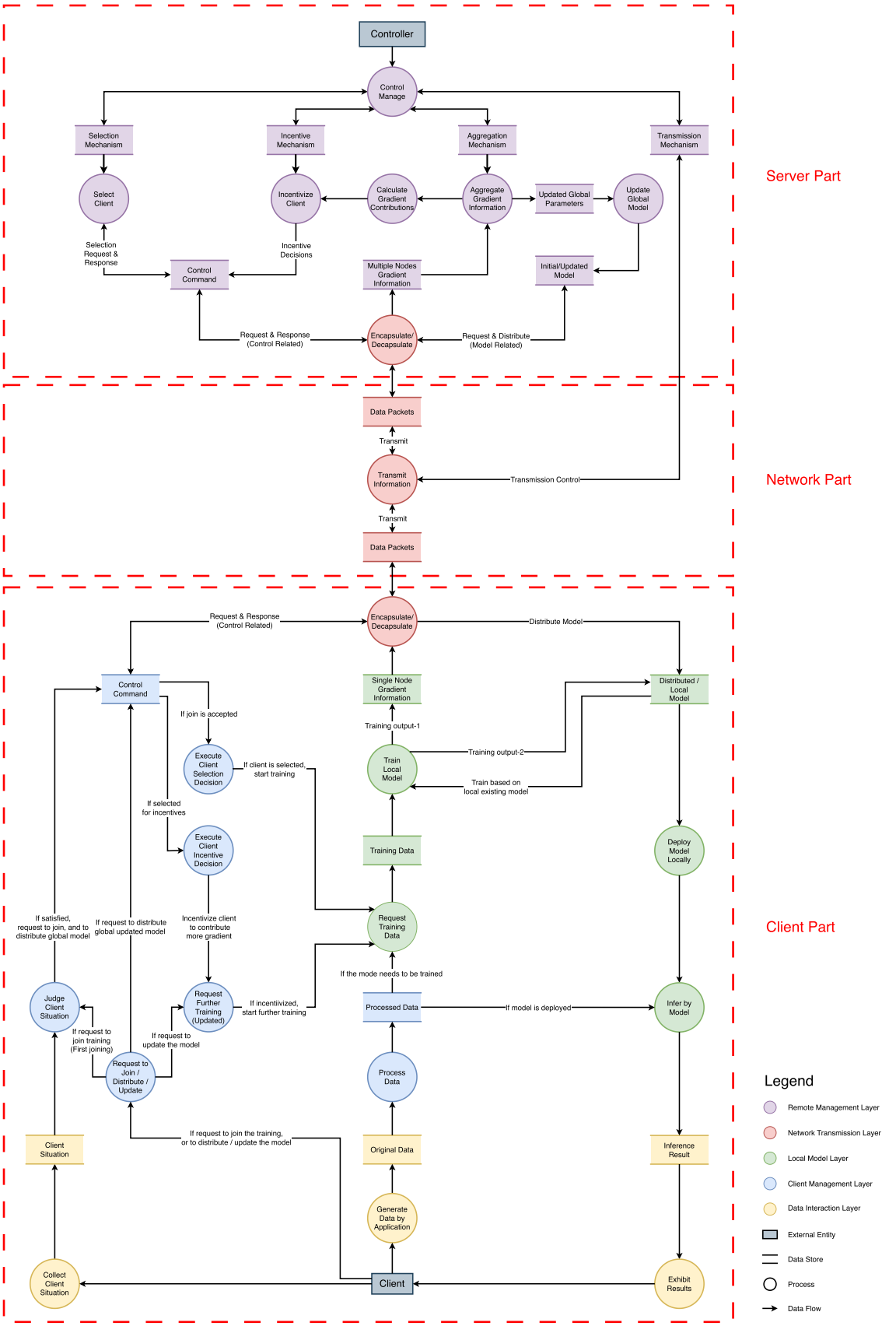

FIGURE 4. Low-level data flow of federated learning.

deployment, and model inference as specified by the local model layer. Until this step, participants still maintain autonomy to make decisions following unified standards and interfaces. This could protect their privacy data to the greatest extent, and comply with their respective internal rules–no matter the rules and regulations in the insurance companies, commercial banks, or government tax departments.
  - -- Network Transmission Layer: Participants upload the model update information following the packet encapsulation algorithm specified by the network transmission layer.
  - -- Remote Management Layer: The final step is to aggregate this information to update the global model on a shared server that is trusted by all parties.
- **Top-down Control Flow:**
  - -- Remote Management Layer: Participants have to negotiate client compute node selection mechanisms, incentive mechanisms, global model aggregation mechanisms, and data transmission mechanisms as the control flow input based on the perspective of the remote management layer.
  - -- Network Transmission Layer: For the sake of successful transmission among each participant, a unified packet encapsulation and decapsulation algorithm is required in the network transmission layer. Besides, control information transmission standards including client compute node selection, global model issuance, participant node incentivization, etc. need to be set up as well.
  - -- Local Model Layer: This layer should consider the execution of client compute node selection results, updated model deployment, and inference API for common use.
  - -- Client Management Layer: The problem becomes straightforward as we come to this lower layer. The main objective is how to process the original inference output from the local model layer. Since in most instances, it is difficult to leverage such original output results directly. Note that starting from this layer, the control flow becomes an internal process for the participants, which means that, based on following unified standards, participants have room to freely develop according to their actual conditions. For example, the government tax department may have higher requirements than either the insurance company or commercial bank in protecting sensitive information.
  - -- Data Interaction Layer: The only thing the engineers have to do is exhibiting the final results to the end user through the data interaction layer.
- **Security Enhancement:** After designing both bottom-up data flow and top-down control flow, a prototype of the federated learning model training system among different participants is created. Then, engineers can follow the low-level data flow diagram proposed in Figure 4 to analyze flows and further enhance the protection of sensitive private information.

With the assistance of the application case example, we could find that our security-oriented 5-layered federated learning reference model simplifies the complex design of a cross-enterprise FL model training system into five separate layers, focusing on two flows in opposite directions: bottom-up data flow and top-down control flow. Our proposal could assist engineers in addressing isolated and simple problems at a time, simplifying their work and saving system design and deployment time. Additionally, system scalability is enhanced with the improvement in standardization as well. Moreover, our model provides a clear approach to analyzing security threats for systematically enhancing data protection.

V. THREATS INTRODUCTION

While one of the primary objectives of federated learning is to protect the privacy of training participants by avoiding the uploading of private and sensitive data to the centralized server, recent research indicates that privacy and security in federated learning can still be vulnerable under certain circumstances, as highlighted in a study by Lim et al. [15]. This section briefly provides a brief introduction to the threats inherent in the federated learning system.

A. THREATS MODELS

Before delving into attacks on federated learning, let's introduce the STRIDE model, a simple threat model developed by Microsoft for classifying threats [47]. The STRIDE model is commonly used in threat modeling and risk assessment to identify potential threats and vulnerabilities in a system or application. By categorizing threats into the following six distinct categories, users can gain a better understanding of the types of attacks that could impact their systems and take appropriate measures to mitigate these risks. Table 2 provides an overview of the threats typically associated with each element utilizing the STRIDE model based on Adam's work [47].

- **Spoofing:** Attackers impersonate someone else to gain unauthorized access to the system.
- **Tampering:** Unauthorized modification or alteration of data, such as changing data contents or inserting new data.
- **Repudiation:** Denying that an action or data alteration occurred.
- **Information Disclosure:** Unauthorized exposure, release, or sharing of sensitive or confidential data with individuals or systems that should not have access to it.
- **Denial of Service:** Disrupting the availability of a system or service due to attacks.
- **Elevation of Privilege:** Attackers gain higher-level access or permissions to the system or data than they should have.

TABLE 2. STRIDE threats per element.

Element	Interaction	Spoofing	Tampering	Repudiation	Information Disclose	Denial of Service	Elevation of Privilege
External Entity	External interacting entities pass input into the process	✓		✓	✓		
	External interacting entities get input from the process	✓					
Process	Process has external data transferred to the data store	✓		✓			
	External interacting entities get input from the process	✓		✓	✓	✓	✓
	Process sends output to an external interacting entity (Application)	✓		✓	✓	✓	
	Process sends output to an external interacting entity (User)			✓			
	Process has an input data stream from the data store	✓	✓			✓	✓
	Process has an input data stream from another process	✓		✓		✓	✓
	Process has input data streams from external interacting entities	✓				✓	✓
Data Flow (Request/ Response)	Crossing machine boundaries		✓		✓	✓	
Data Store (Database)	Process has output data flowing to the data store		✓	✓	✓	✓	
	Process has an input data stream from the data store	✓		✓	✓		

B. CLASSIFICATION BASED ON ATTACKERS' KNOWLEDGE

Another kind of threat classification is based on the level of knowledge attackers possess about the target machine learning system. This classification comprises three main branches: white-box attacks, black-box attacks, and gray-box attacks [48].

- **White-Box Attack:** In a white-box attack, adversaries have comprehensive knowledge and unrestricted access to the machine learning system, encompassing the model network architecture, model parameters, and even the training data [49]. However, this type of attack is primarily conducted to assess the vulnerability of the system [48].
- **Black-Box Attack:** In a black-box attack, attackers possess little to no relevant knowledge about the model or the original training data. The primary approach involves interacting with the model through limited query access, such as Application Programming Interfaces (APIs), to obtain complete or partial information [50]. Black-box attacks align more closely with real-world scenarios compared with white-box attacks, resembling normal usage conditions. For example, Shukla et al. successfully launch such an attack with fewer than 1000 queries [51].
- **Gray-Box Attack:** The attacker may know partial training data or some knowledge about the model, encompassing internal states or decision logic of the model, model architecture, or hyperparameters before they launch an attack by approaches like side-channel attacks, model reverse engineering, etc. Compared to white-box attacks and black-box attacks, gray-box attacks are often more challenging and requires more sophisticated attack strategies. They are also close to real-world scenarios.

Unfortunately, using this taxonomy to analyze security threats in FL systems is challenging, as it just describes the knowledge level of attackers rather than illustrating vulnerabilities. There are other taxonomies as well, for example, Suciu et al. propose a framework to classify attackers' knowledge and control from four dimensions [52].

However, these methods can only provide engineers with very little information for threat analysis.

C. FEDERATED LEARNING-SPECIFIC THREATS

Similar to other machine learning systems, federated learning systems are susceptible to various forms of attacks. Lyu et al. specifically identify two prominent threats in federated learning: poisoning attacks and inference attacks [53]. In addition to these, we will also discuss two other important threats: backdoor attacks and free-riding attacks. Besides, the National Institute of Standards and Technology (NIST) points out that the evasion attack is generating increased interest in the machine learning research space [48]. However, it's important to note that each type of attack has unique characteristics.

1) POISONING ATTACK

The overarching intent of this assault is to perturb the behavior of the target model's inferences, thereby inducing it to deviate and render erroneous predictions. According to Huang et al.'s work, it can be classified into two categories: random attack and targeted attack [54]. The former seeks to curtail model accuracy, while the latter endeavors to manipulate the model into producing predefined adversarial label information. The classification of poisoning attacks can be based on their source, resulting in two categories: data poisoning attacks and model poisoning attacks [53].

- **Data Poisoning Attack:** Data poisoning attacks are recognized as the most common type of attack in the realm of machine learning, and this holds true not only within the domain of federated learning [55]. This prevalent attack primarily occurs during local data collection. Adversarial agents deliberately introduce corruption into training data on a subset of participant devices during the learning phase, thereby compromising the final global model's accuracy. Both definite and indefinite targeted devices could potentially serve as targets for the injection of poisoned data [56]. Tolpegin et al. point out that data poisoning attacks can be either targeted or non-targeted [57]. In the targeted variant, the attacker aims to exert influence over a specific subset of data classes, consequently undermining the overall model accuracy. In contrast, non-targeted variants lack the presence of any specific subset of data classes to be targeted for attack [56].

  Two forms of data poisoning attacks are recognized: clean-label attacks [58] and dirty-label attacks [53], [59].
  - -- Clean-label Attack: In this scenario, adversaries inject malicious information into the data while keeping the data labels unchanged. Attackers attempt to introduce subtle perturbations to the dataset, causing the model's predictions for this data to be misleading, all the while preserving the true data labels, due to the existing validation process. Detecting such an attack can prove challenging, as the tampered data retains identical labels to the original data, thereby circumventing conventional detection methods.
  - -- Dirty-label Attack: Adversaries inject malicious data with disguised labels into the dataset to disrupt the training and performance of the model, leading it to generate misleading predictions. This attack focuses on altering the labels of the data, rather than solely the data content itself. Such an attack can yield a more enduring influence on the model, as the malevolent label will continue to impact the model's performance during both the training and testing phases. A notable illustration of a dirty-label attack is the label-flipping attack [60].

  The extent of impact stemming from a data poisoning attack within the context of federated learning is contingent upon factors such as the number of participating client compute nodes and the proportion of poisoned data within the comprehensive training dataset.
- **Model Poisoning Attack:** This attack occurs during the training of models. In contrast to data poisoning attacks, model poisoning attacks have been demonstrated to be considerably more potent by Bhagoji et al. [61]. In this scenario, adversaries taint local model updates rather than the underlying local data. These poisoned updates, when introduced, aim to perturb the global model's classification on a chosen set of inputs characterized by high confidence levels [62]. It is important to underscore that the manipulation is intended to influence the model training process, resulting in misclassifications [63].

  According to various studies, model updates into which a poisoning attack is injected can be generated through various methods [61], [64], such as the single-shot attack as exemplified by Bagdasaryan et al. [62]. Lyu et al. indicate that a model poisoning attack could exploit an alternating minimization strategy to enhance attack stealth and elude detection [53].

  However, due to the fact that a data poisoning attack ultimately affects a subset of model updates, a model poisoning attack is sometimes considered to encompass a data poisoning attack [60].

2) INFERENCE ATTACK

The phenomenon of inference attacks arises from the utilization of gradients derived from the private data features of participants within various layers, encompassing sequential fully connected layers as well as convolutional layers. These gradients can inadvertently leak sensitive private information, such as class or membership details. This phenomenon is referred to as an inference attack. Notably, Zhu et al. demonstrate the startling revelation that original samples can be reconstructed from uploaded gradients, obviating the need for direct access to the training dataset [65]. Unlike localized poisoning attacks, inference attacks manifest more prominently during the transmission of updated gradient information between participants and the central server. The taxonomies proposed by Lyu et al. and Liu et al. classify

inference attacks into seven sub-classes: representative inference, membership inference, property inference, training inputs and labels inference, attribute inference, model extraction, and model inversion [49], [53].

- **Representative Inference:** In this category, a malicious participant exploits the real-time nature of the training process to train a Generative Adversarial Network (GAN) capable of generating prototypical samples representative of the private targeted training data [66]. Notably, this form of attack exclusively targets class representatives, as the generated samples convincingly mirror the distribution of the original dataset. This strategy relies on the premise that a single participant contributes the entire training corpus, and only when the data of all members are similar, can the reconstructed representatives resemble the original training data. Analogous to a model inversion attack, this assumption is often challenging to meet [67].
- **Membership Inference:** This attack, highlighted by researchers from Cornell University, aims to deduce whether a specific sample originates from private data belonging to a single participant or a group [68]. This category bifurcates into two variants: passive membership inference attack and active membership inference attack [53].
  - -- Passive Membership Inference Attack: This variant hinges upon the passive observation of transmitted parameters within updated models, thereby enabling inference without manipulation of any data. This inference can occur during both local and global training procedures [69].
  - -- Active Membership Inference Attack: In this manifestation, attackers manipulate model training protocols to induce the provision of supplementary information regarding the private local data of interest by introducing malicious updates. Nasr et al. term this approach a gradient ascent attack [70].
- **Property Inference:** Adversaries execute property inference to deduce global information, such as whether the training dataset possesses a specific characteristic [69], [71]. However, the property here is not explicitly stated as the attribute in the training set [72]. Property inference is premised on the assumption that auxiliary training data are labeled with the relevant property. Two variations emerge, according to Lyu et al. [53]:
  - -- Passive Property Inference Attack: Adversaries solely observe or eavesdrop on updated information, leveraging a binary property classifier for the purposes of inference.
  - -- Active Property Inference Attack: This variant employs multi-task learning to induce better data separation, thereby facilitating subsequent extraction of private information.

  Notably, the performance of property inference attacks remains unaffected by property presence during model training.
- **Training Inputs and Labels Inference:** An exemplary instance within this category is the concept of Deep Leakage from Gradient (DLG), introduced by MIT [65]. This framework facilitates the restoration of pixel-wise accurate original images and token-wise matching text contents from a limited number of iterations. Zhao et al. extend this concept with the notion of Improved Deep Leakage from Gradient (iDLG), harnessing connections between shared gradient features and labels to extract labels [73]. Differentiable models trained with cross-entropy loss over one-hot labels are susceptible to this approach [73].
- **Attribute Inference:** The attribute inference attack aims to uncover concealed sensitive attributes within a given sample [74], [75]. Fredrikson et al. suggest that the input information encompassed in the confidence output can serve as a measure, as the model prediction results typically incorporate reasoning information about the sample [74]. Liu et al. define the difference between an attribute inference attack and a property inference attack as follows: an attribute inference attack extracts features related to the primary task, whereas a property inference attack extracts features unrelated to the primary task [49].
- **Model Extraction:** The goal of a model extraction attack is to extract information about the model architecture and parameters, or other sensitive information, by making numerous queries to the model. As illustrated in NIST's work, there are predominantly three approaches to achieve it: direct extraction, using learning methods, and using side-channel information [48]. Each of these methods has been demonstrated by several works in the literature.

  There are many subclasses of model extraction attacks according to existing works. We introduce four common subclasses here based on Gong et al.'s work [76]:
  - -- Model Parameter Extraction: The objective of model parameter extraction is to recover the model parameters through black-box access.
  - -- Hyperparameter Extraction: Hyperparameter extraction aims to restore the potential hyperparameters that may bring models with significantly different performances with different values [77].
  - -- Model Architecture Extraction: Taking the Deep Neural Network model as the target example, attackers attempt to infer the number of neural network layers utilizing a pre-trained model.
  - -- Decision Boundary Extraction: Adversaries leverage the approach of training a substitute model based on the labels received from a large number of queries to manipulate the decision boundary of the model. Finally, the model would generate some incorrect predictions under specific inputs.

  Note that a model extraction attack often serves as a step towards other powerful attacks, and it could also support transferable adversarial attacks [78], [79], since

the attackers have already obtained detailed information about the target model.

- **Model Inversion:** Model inversion attack (also known as attribute inference attack) [80], predominantly leverages the machine learning system's APIs, and non-sensitive attributes of the test input data to obtain hidden sensitive information about the model [49]. This allows attackers to reconstruct class representatives of the original training data [67]. Although it is unable to recover the training data completely [48], the data reconstructed by model inversion still have higher accuracy than characteristics solely inferred from data not present in the training dataset [81].
  Dibbo et al. categorize model inversion into an optimization-based approach and a surrogate model training approach depending on the incorporated techniques [72].
  -- Optimization-based Approach: Attackers attempt to reconstruct the training instances without the need for training any additional models.
  -- Surrogate Model Training Approach: Adversaries seek to invert the sensitive attributes or training samples by exploiting auxiliary information and a surrogate model that utilizes input-output correlation in the target model.
  However, it is important to note that both model inversion attacks and membership inference attacks can be executed in either black-box attacks or white-box attacks [81], [82].

3) BACKDOOR ATTACK

The backdoor attack, alternatively referred to as a targeted attack [55], involves a malicious participant or a group of such participants intentionally introducing a backdoor function into the global model by manipulating individual features or small regions of the original training datasets [59], [62], enabling them to control the behavioral patterns of the global model [53]. Notably, this control is achieved without compromising the overall accuracy of the model. Sun et al.'s research demonstrates that the severity of such attacks is correlated with the proportion of compromised participants and the model's capacity [83]. Remarkably, the insidious nature of backdoor attacks renders them elusive to detection, as they remain inconspicuous within the model's performance on clean data.

4) FREE-RIDING ATTACK

The free-riding attack occurs when a malicious participant, contributing little or nothing to the federated learning process, seeks to exploit the benefits derived from the global model. The attackers pretend to possess an extremely small training dataset, inducing other participants to allocate additional resources to facilitate the model training process. This attack poses a significant threat, especially in sensitive applications of federated learning [84].

According to research, there are two primary incentives for client compute nodes to engage in free-riding by submitting fake parameters:

1) The client compute node is concerned about data privacy, or even has no available local data for model training. In the latter case, to evade security detection, the client compute node has to submit fake gradient information [85].
2) The client compute node aims to save its local computing resources, such as CPU or GPU cycles, or local storage space [85], [86].

Depending on whether the nodes possess local data and computing resources, Wang et al. classify free-riders in the FL system into two types: anonymous free-riders and selfish free-riders [87].

- **Anonymous Free-riders:** In this case, there is no available data or computing resources possessed by the free-riders, the attackers submit random Gaussian noise as the training updates [88], [89].
- **Selfish Free-riders:** Selfish free-riders possess local data and computing resources, but are loath to contribute them for model training. Instead, they upload fake updates to the aggregation control server [88], [89].

5) EVASION ATTACK

In evasion attacks, the primary objective is to construct specific adversarial samples whose classification can be arbitrarily manipulated to a specific class controlled by the attacker at the time of model deployment, while introducing minimal perturbations that are imperceptible to humans [48], [90]. In other words, an evasion attack aims to mislead the model into producing an incorrect prediction result rather than altering the model itself [49]. Various works, such as those conducted by Kearns and Li [91], and Szegedy et al. [63], have successfully demonstrated the feasibility of evasion attacks, and current research continues to focus on the designing of adversarial examples.

Although Liu et al. categorize evasion attacks into targeted attacks with class-specific errors and untargeted attacks that do not consider class-specific errors based on the optimization objective [49], Vassilev et al. categorize them according to the perspective of the adversary's knowledge into white-box evasion attacks and black-box evasion attacks [48].

6) SUMMARY

After introducing some federated learning-specific threats above, some comparisons are illustrated in this summary part. Table 3 presents comparisons of the common attack methods that appear in federated learning, and Table 4 emphasizes some other differences among those attack methods. Table 5 shows the differences among variations in Inference Attack.

While these empirical comparisons may not offer a complete assessment, they highlight the intricate security challenges faced by federated learning systems. This prompts

us to delve deeper into security threat analysis to bolster the security measures of FL systems.

VI. THREATS ANALYSIS OF EACH LAYER

In order to improve the security level of federated learning while assisting users in narrowing down the scope of the issue, we conduct a layer-by-layer security analysis along the data flow in the federated learning system, with the help of the DFD in this section. Figure 4 provides the detailed information, while Figure 5 illustrates the high-level data flow in federated learning. Note that we only analyze security threats in traffic that terminates at the target layer and ignore traffic originating from the target layer when traffic is bidirectional.

A. DATA INTERACTION LAYER

The utilization of sensors assists researchers in gaining a better understanding of environmental features for their projects. For example, Lu et al. develop a multilevel regression model based on the data they collected [95]. AI techniques like federated learning provide a promising approach to processing such an enormous volume of data. As a result, the data interaction layer serves as a bridge between the federated learning system and user applications. Consequently, it becomes an entry point vulnerable to external threats.

As illustrated in Figure 6, whether the client compute node could join the training process depends on the assessment of the collected client system environment situation. Consequently, attackers could tamper with or repudiate the sent information, or infer the operation environment information according to the client situation information after invading this system-level process by injecting a backdoor or elevating themselves to higher privileges without being noticed by the users. These attack approaches will interfere with further client application assessments. Furthermore, since the original source data stems from users' daily interactions, attackers could tamper with the raw data generated by applications or launch further data poisoning attacks, including clean-label attacks and dirty-label attacks. A property inference attack is easy to be achieved at this step as well, which is another kind of information disclosure. Alternatively, it is possible for adversaries to disturb the raw data generation process by utilizing repudiation attacks or even lunching denial-of-service attacks after injecting backdoors in the applications.

However, not all users can be guaranteed to act in good faith. As a result, the inference result exhibition part in the data interaction layer is another area with high risk. Once the inferred results are provided by the model, malicious users may launch inference attacks such as representative inference, membership inference, and property inference, etc. These attacks could be regarded as private information disclosure. Moreover, they could tamper with the original results to conduct an evasion attack. Furthermore, there remains a possibility of spoofing or repudiation attacks, which are the consequences of result leakage at the results exhibition stage.

B. CLIENT MANAGEMENT LAYER

As discussed in Section IV, the primary role of the client management layer is data processing and client-side client selection, serving as the starting point for FL model training. However, since the decision-making process of client selection occurs at another layer, we will only focus on the analysis of security threat in the data processing and join request/response. The execution part of client selection is considered a reliable security zone.

There are four potential paths for attackers, as illustrated in Figure 7. The first one originates from the request to join the training or distribute the model. Besides spoofing fake requests or repudiating real requests sent from the client, attackers could infer client system information according to the leaked data. Moreover, it is possible for attackers to overwhelm client resources by launching a large number of denial-of-service attacks. On the other hand, attackers' privileges could be elevated if the adversaries invade, since either a join request or a distribute request is generated based on the client system's current situation, which requires high privileges to collect. Besides, malicious users who possess little data can launch free-riding attacks to access training data. While this attack may not always compromise private data, it can significantly degrade overall training performance due to its zero contribution. Moreover, such free-riding nodes can potentially become vectors for more malicious attacks, as distinguishing between genuine nodes possessing little data and malicious ones pretending to be so is impractical. In other words, the attackers are hidden among a large number of normal users.

The second path is related to the collected client environment situation. Apart from the attack methods analyzed in the data interaction layer, a malicious participant could initiate attacks using stored data. Spoofing or tampering is possible at this step. There is also a possibility of information disclosure if client compute nodes that don't meet the conditions are allowed to join. However, preventing this risk is challenging, as it may require detecting physical hardware allocation status while checking data at the point of entry.

The third attack path comes from decapsulating of response data sent from the remote aggregation control server. Since the aggregation control server and client compute nodes are often connected through a network in many scenarios, attacks in this path are a subset of network security threats. In addition to tampering with or acquiring private decapsulated data, attackers could also paralyze the entire training system by repudiating requests or making a large number of repeated requests. Such threats will be discussed in the network transmission layer again.

The last invasion approach is related to data flow. Security threats on this path are similar to those in the previous step in the data interaction layer, as these potential threats all stem from original data generated by applications.

TABLE 3. Comparisons of various common attack methods in federated learning 1.

Threats	Category	Variations	Attack Objective	Attack Timing	Attack Approach
Poisoning Attack	Data Poisoning	Clean-label	Mislead model's prediction	Local data collection, training phase	Introduce subtle perturbations to the dataset
		Dirty-label			Alter the labels of the data, rather than the data content
	Model Poisoning	None	Perturb the global model's classification on specific inputs	Training of global model	Local model updates
Inference Attack	Representative Inference	None	Infer training data distribution	Transmission of updated gradient information, during or after the model training phase, inference phase	Train GAN Model, analyze model output
	Membership Inference	Passive Membership Inference	Deduce whether a specific sample originates from private data [68]		Observe transmitted parameters
		Active Membership Inference			Introduce malicious update
	Property Inference	Passive Property Inference	Infer global information about the training data distribution [48]		Eavesdrop on updated information
		Active Property Inference			Employ multi-task learning
	Training Inputs and Labels Inference	None	Restore the training data		Harness connections between shared gradient features and labels
	Attribute Inference	None	Uncover concealed sensitive attributes [92]		Infer based on partial information from select training data and a model trained on that data [93]
	Model Extraction [76]	Model Parameter Extraction	Recover the model parameters		Perform large queries on the model
		Hyperparameter Extraction	Restore the potential hyperparameter		
		Model Architecture Extraction	Infer the architecture of the model		
		Decision Boundary	Extract the decision boundary of the targetted model		
	Model Inversion	Optimization-based Approach	Reconstruct sensitive features of training data using auxiliary information [94]		Reconstruct of training instances [72]
		Surrogate Model Training Approach			Train a surrogate model [72]
Backdoor Attack	None	None	Embed backdoors	Local data preparation phase, before uploading the local model	Manipulate parts of the original training datasets
Free-riding Attack	Anonymous Free-riders	None	Capitalize on the benefits yielded by the global model	Malicious participate joining, training phase	Submit random value noise as model updates
	Selfish Free-riders				Upload fake updates
Evasion Attack [49]	Targeted attack	None	Decline the performance of the model	During the model is training, or inference phase	Modify input data
	Untargeted attacks				

Excluding the free-riding attacks, poisoning attacks including clean-label attacks and dirty-label attacks, and property inference attacks become available since original data are exposed to adversaries. This could worsen if they inject backdoors into original data generated by applications, or into the applications themselves.

C. LOCAL MODEL LAYER

Wu et al. state that the federated learning model training paradigm comprises two key components [96]:

1) **The client-side:** which trains models on local individual user private data based on a distributed global model and subsequently uploads the updated model information to the central server.
2) **The server-side:** responsible for aggregating the uploaded model update information to train the global model.

In our layered reference model, the client-side model training corresponds to the local model layer that leverages processed data to train the local model. And the server-side

TABLE 4. Comparisons of various common attack methods in federated learning 2.

Threats	Attack Targets		Attack Point		Auxiliary Knowledge Required
	Model	Private Information	Aggregation Control Server	Client Compute Node	
Poisoning Attack	✓			✓	Training data distribution, model architecture, data injection
Inference Attack		✓	✓		Model architecture, data distribution, training data
Backdoor Attack	✓			✓	Training data, backdoor injection, system control
Free-riding Attack	✓			✓	Only little about fake information generation
Evasion Attack	✓		✓	✓	Model decision boundaries, mode architecture, output analysis

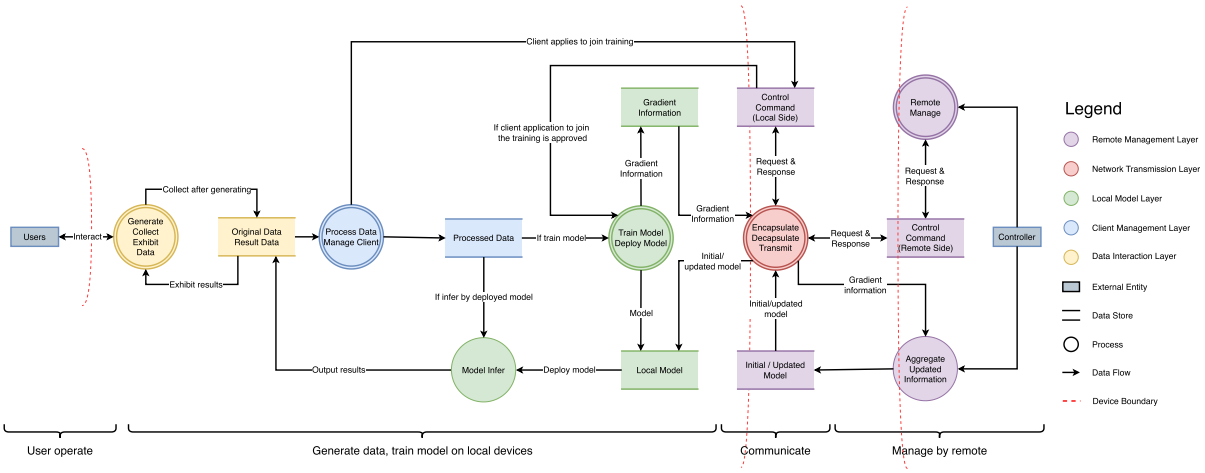

FIGURE 5. High-level data flow of federated learning training.

model training corresponds to the remote management layer, which will be discussed later.

There are three sources from which security threats originate: the request for training data before the model training phase, the execution of client selection/incentive decisions, and the distribution of the global model to local participants, as depicted in Figure 8.

Initially, as the model is trained on the client-side by requesting training data from client compute nodes, attackers could spoof the data as being generated by the authorized application to disrupt the training dataset. Moreover, there is a significant possibility of private data being disclosed to malicious attackers who breach this phase, as the processed data is often not encrypted. This vulnerability could lead to further clean-label attacks and dirty-label attacks. Attackers may passively or actively infer the property information of the training data by utilizing the disclosed data. Furthermore, the system might be overwhelmed if adversaries send a large number of data requests in a short time. The local model layer will become even more vulnerable if backdoors were injected.

Moreover, attackers can easily disrupt client selection/incentive processes by spoofing decisions sent from the aggregation control server. System status may become transparent as a significant amount of information is disclosed. Additionally, clients might become non-functional if adversaries initiate numerous repudiations or a large number of requests in a short time, due to limited resources being exhausted. Moreover, attackers will possess higher privileges through previously injected backdoors, while free-riding attacks provide another approach to stealing trained model information. On the other hand, attackers could launch attacks on the model inference process from the same data source. During the inference stage, processed data becomes a vulnerable point of attack before being sent into the model for subsequent inference. Apart from spoofing the real data source, attackers could manipulate data during transmission between processes or utilize the real data to infer users' private training data. Furthermore, if attackers gain the privilege of sending processed data to the model for inference, the risk of launching denial-of-service attacks also increases.

The third source of risk arises from decapsulating data distributed from the remote aggregation control server via the network. The transmission phase faces the same security threats as the network transmission layer, which will be explained in the next subsection. Since the trained model

TABLE 5. Comparison of several variations of inference attacks.

Attack Type	Possible Information Leakage	Attack Success Conditions	Impact Scope	Attack Complexity	Attack Detectability	Defense Strategy Example
Representative Inference	Sensitive data distribution	Sufficient data diversity	Training data	Only analyze the statistical characteristics of training data	No frequent queries, few traces left	Improve training data diversity and privacy protection
Membership Inference	Private data	Member data	Data samples	Train an attack model, get shadow member data	Large number of queries may raise suspicion	Differential privacy
Property Inference	Sensitive data distribution	Leaked attribute	Partial attribute	Analyze leaked attribute information	No frequent queries, few traces left	Prevent attribute information leakage
Training Inputs and Labels Inference	Part of training data	Leaked data	Partial data	Independently reconstruct the training data	Require many queries, but uneasy to be detected	Fundamentally prevent data leakage
Attribute Inference	Model information	High model transparency	Model information	Train an auxiliary model based on leaked attribute information because of transparency	Large number of queries	Limit model transparency
Model Extraction	Model information	Enough queries	Model information	Reconstruct the model	Large number of queries	Indistinguishable from common queries
Model Inversion	Part of training data	Private data and leaked model	Partial data	Need knowledge of training data and model parameters	Many queries	Enhance model protection, prevent training data leakage

is issued to the client compute node at this step, malicious attackers could easily poison or even tamper with the model to mislead subsequent inference results. Moreover, model-related information such as model parameters and model architectures is also exposed to them, which could lead to private information disclosure. Additionally, the model's performance will also be affected by any evasion attack initiated by adversaries. However, verifying the received information and detecting abnormal activities could effectively decrease security risks originating from this source.

D. NETWORK TRANSMISSION LAYER

In contrast to conventional machine learning model training paradigms, federated learning highlights communication as a pivotal bottleneck that must be given attention, since communication costs often dominate computation costs in federated learning, according to the research by McMahan et al. [97]. Within the domain of federated learning, several rounds of communication between the aggregation control server and client compute nodes are required for the model accuracy. Furthermore, given that many participants often operate under public network environments such as Ethernet, the presence of unreliable network conditions and asymmetric public links could exacerbate the security situations.

Figure 9 illustrates the data flow of the network transmission layer, which involves the transmission of various data types, including gradients, models, and control commands. In the gradient transmission part, after training on the client compute node (Threats Group 1), it is very easy for attackers to infer membership and properties of training data by observing the gradient information. If this malicious observation continues, the possibility of more complex attack approaches like model extraction attacks and model inversion attacks increases, since deeper information could be revealed by comparing the differences among gradient information transmitted in each round. This would cause more serious information leakage. Moreover, malicious attackers could also simply disturb the gradient computing results through spoofing, i.e., sending fake gradient information, or make the gradient information unavailable through repudiation and denial-of-service attacks. The situation of model-related requests and distribution flow is similar (Threats Group 5). The only difference is that adversaries could launch model poisoning attacks instead of membership inference and property inference, as the entire models are available during the model issuance process.

As for the control flow, including request commands sent by client compute nodes (Threats Group 2), control responses (Threats Group 4), and transmission controls (Threats Group 6), they face almost the same security risks due to having the same data type in their traffic. Attackers could spoof unauthorized control command sources to affect the control decision-making, or exploit control information to infer users' system-level status, which contributes information disclosure. Moreover, adversaries may repudiate effective commands or even launch denial-of-service attacks to interfere with normal operation of the FL system. However, since the transmission control is often exposed to insecure public network environments in many scenarios, attackers may initiate network attacks for higher privileges to further invade the entire training system.

The information transmission step (Threats Group 3) is the core part of the network transmission layer. This highlights the fact that the network transmission layer is susceptible to all six common threats: spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privileges, since the entire system is exposed to the public through this layer. Compared with invasion from the data interaction layer, invading via this layer is much easier due to the complexity and anonymity offered by public network. Moreover, as gradient information is transmitted, there's a risk of revealing sensitive private information [67], [69], [98], and it can even result in deep leakage, as indicated by Zhu et al. [65]. Notably, even partially updated gradients could be exploited to unveil local data, as shown by Aono et al. [99]. Such leakage could occur either to a third party or even to a malicious central server [100], [101], [53]. Once attackers gain access to the transmitted information, serious attacks including model poisoning attack, model extraction attack, and model inversion attack become possible. Except for the deep leakage issue mentioned above, attackers could utilize the model update information to poison the subsequent global model. In general, the network transmission layer is the most vulnerable part of the entire federated learning system, so it may serve as another potential entry point for malicious attackers from outside the system.

E. REMOTE MANAGEMENT LAYER

Serving as the orchestrator, the remote management layer not only aggregates the gradient information uploaded from the participating client compute nodes to train an updated global model, but also performs various management functions. Such functions include client compute node selection, incentive mechanisms, aggregation mechanisms, and transmission management.

As Figure 10 shows, there is another security zone within it. The entrance of data from the network transmission layer is the main threat, while the invasion from the controller is the secondary threat. Regarding the data received from the network transmission layer, it is the most vulnerable point, as discussed in the previous subsection. The situation becomes worse if the data type is a command about client selection, client incentive, model distribution, and transmission control (Threats Group 1, 4, and 5). Malicious invaders can inject a backdoor during data transmission at the network transmission layer, through which they can control either the entire system or the private data without being noticed. If attackers deliberately tamper with the control request command, they will seriously interfere

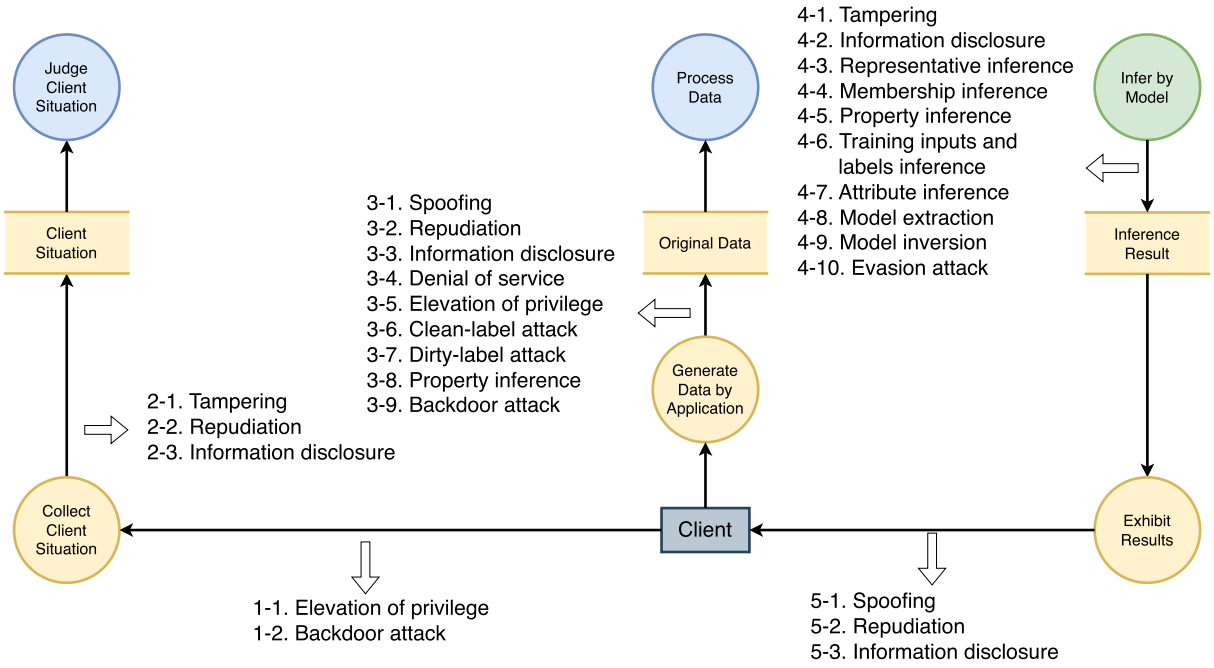

FIGURE 6. Threats in data interaction layer.

with the decision-making mechanism. Moreover, there is a possibility for them to infer the private data, including system status and training data quantity, by analyzing the control request commands. This represents a threat of information disclosure. The controller might run out of service if attackers successfully exhaust its resources through deliberate or frequent repudiation of normal control requests. However, issuing a command while identifying the actual role of the requester is a challenging task.

Moreover, attackers could spoof a legal client compute node to disturb the updated gradient information aggregation of the controller if they invade the decapsulation process through which data is transmitted into the remote management layer. Adversaries could also poison the model by tampering with the updated gradient information during this process. If some private gradient information is leaked at this step, attackers could utilize model inference attacks, including membership inference, property inference, etc., to delve deeper into private information.

As for the threat issues from the controller side, although it is within the entire system, meaning that the operators are often well-trained and the server is well-protected, since it plays the role of managing the whole federated learning system, small operation mistakes could evolve into attacks like spoofing or elevation of privilege, which could cause serious information disclosure. Furthermore, some backdoors may be unintentionally injected if software security is not given sufficient attention.

F. SUMMARY

In addition to the threat analysis layer by layer above, the following two critical vulnerabilities in the design of federated learning protocols identified by Lyu et al. are useful for the security analysis as well [53]:

1) The control server continuously monitors the updated information, could potentially be malicious, leading to tampering with the model training process and altering participants' view of the global parameter.
2) Participants might also be able to observe or control the global parameters and the upload process of those parameters.

Based on these analyses, we could draw a conclusion that the client compute node and the remote aggregation control server are the two primary vulnerable entities in the entire federated learning system from the perspective of constitution, rather than the computational communication framework itself. While data interaction layer, network transmission layer, and remote management layer are the three main attack entry points from outside, from the perspective of our proposed layered model.

However, for the sake of increasing the information security level of the federated learning system, users can leverage some common strategies to defend against these issues, which will be introduced in Section VII.

VII. THREAT DEFENSE STRATEGIES

Federated learning, especially in the context of cross-silo federated learning, offers a collaborative approach for different organizations to jointly train machine learning models by aggregating gradient updates without exposing sensitive data. However, despite its merits, there remain specific security concerns within this framework, as we have analyzed above. In order to bolster information security and privacy, several established security mechanisms can be integrated into the framework of federated learning.

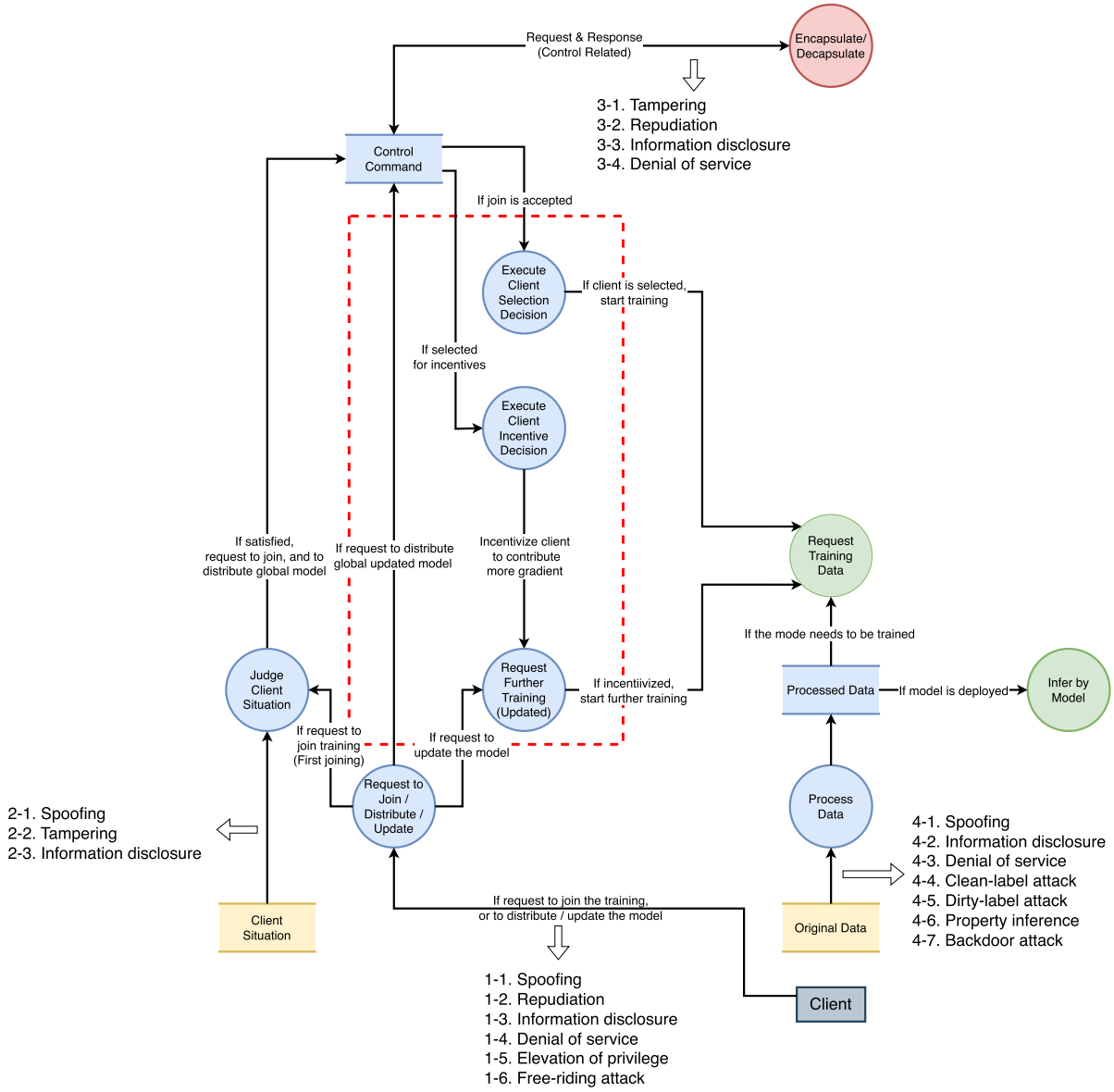

FIGURE 7. Threats in client management layer.

In this section, we will briefly introduce some common defense strategies based on the STRIDE model mentioned in Section V-A, followed by strategies against FL-specific threats. Finally, we will illustrate four specific approaches that can effectively improve the overall security level of the system.

A. DEFENSE STRATEGIES BASED ON THE STRIDE MODEL

As we discussed in Section V-A, spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege are the six common threats identified according to the DFD. Threats of different risk levels may require different mitigation strategies. In Table 6, we list some common strategies for addressing these threats. Note that these defense strategies are not specific to issues in the federated learning system, but rather general solutions.

B. DEFENSE STRATEGIES AGAINST FL-SPECIFIC THREATS

In Section V-C, we have identified five specific threats unique to federated learning. These threats pose challenges to the security and privacy during the training and inference phases. However, it's essential to note that corresponding defense strategies exist for each of these threats that are tailored to address them effectively.

- **Poisoning Attack:** Employing an identification mechanism to scrutinize malicious participants through their model updates before integrating them into the round of model averaging in each learning iteration is an effective strategy against data poisoning attacks. Additionally,

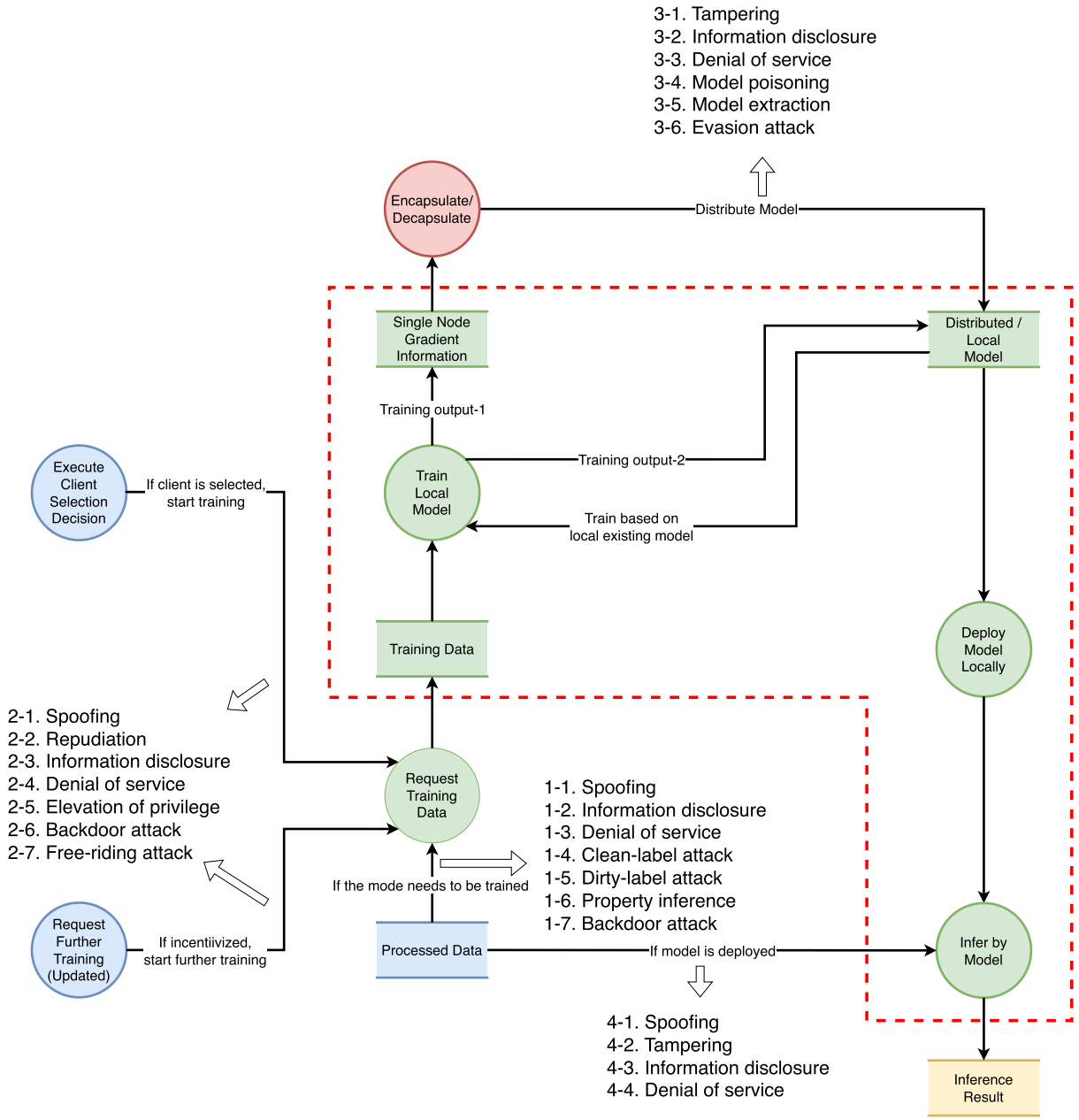

FIGURE 8. Threats in local model layer.

the adoption of rejection based on error rate and loss function emerges as a prevalent approach to mitigate model poisoning attacks [102]. These defense strategies can be employed independently, or in conjunction with the identification mechanisms to defend against poisoning attacks.

- **Inference Attack:** In essence, the successful execution of inference attacks necessitates substantial computational resources and advanced technical capabilities on the part of adversaries. Furthermore, these attackers must sustain their efforts consistently over several training rounds. The employment of Homomorphic Encryption (HE) could serve as an efficacious mechanism for safeguarding shared gradients from the vulnerabilities posed by inference attacks [26]. Additionally, Differential Privacy (DP) offers another approach to defend against inference attacks by introducing random noise to the FL system, and the introducing location could vary from data to model parameters [103]. These two methods will be further discussed in Section VII-C. Apart from HE and DP, strengthening the security level of the network where the FL system is deployed, and promptly identifying potentially malicious nodes that have existed for a long time but rarely accept

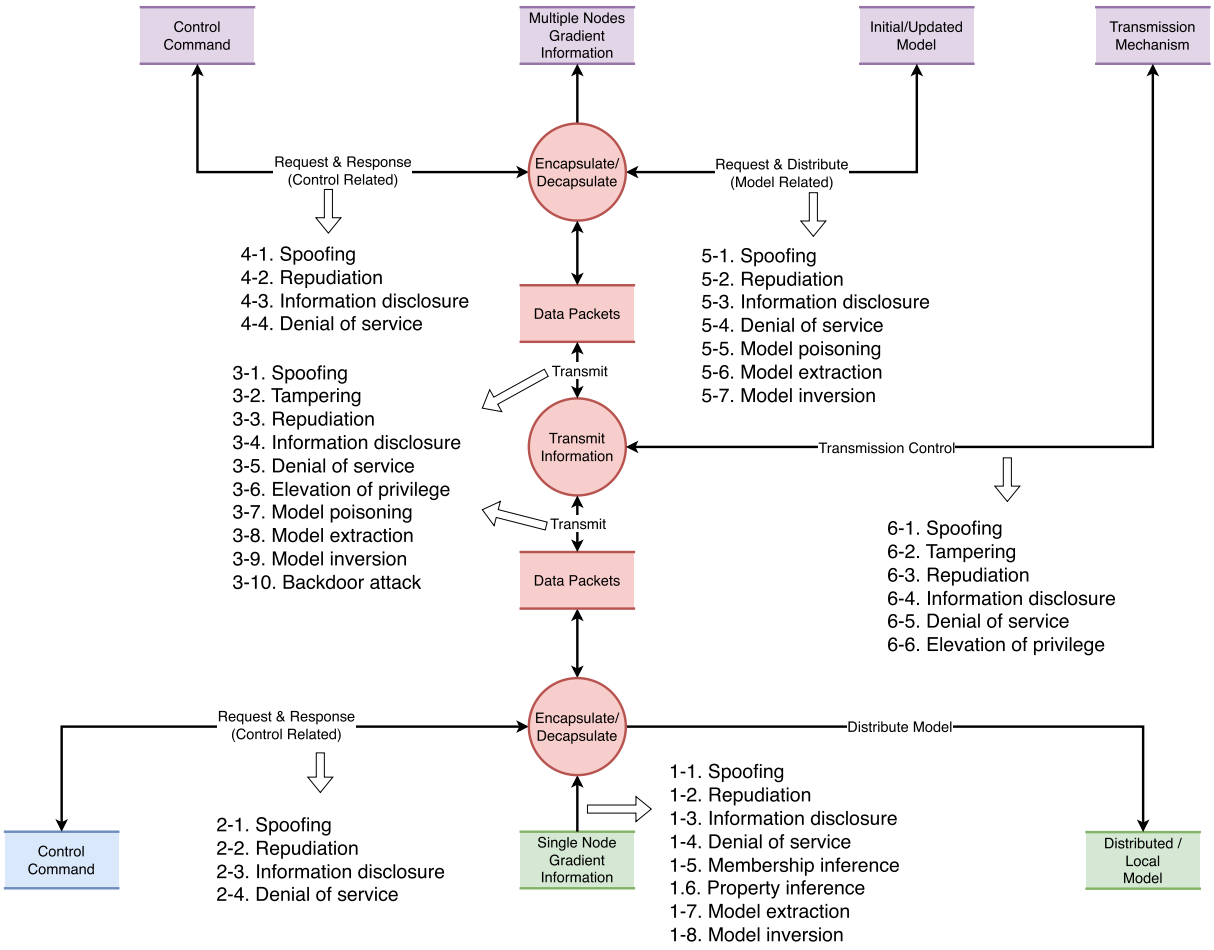

FIGURE 9. Threats in network transmission layer.

TABLE 6. Common defense strategies to STRIDE threats.

Threats	Attribution	Example Defense Strategy
Spoofing	Certification	Basic certification, digital signature, verification code, hash table
Tampering	Integrity	Integrity control, Access Control List (ACL)s, digital signature, verification code
Repudiation	Undeniability	Security logging and auditing, digital signature, Trusted third party
Information Disclosure	Confidentiality	Encryption technology, ACLs
Denial of Service	Availability	ACLs, filtering, authorization, high availability design
Elevation of Privilege	Authorization	ACLs, group or role affiliation, privilege ownership, permissions, input validation

management by the remote management layer can help reduce the likelihood of inference attacks. Furthermore, Jia et al. propose a novel method that adds crafted noise to each confidence score vector to defend against membership inference attacks, falling under the sub-category of information obfuscation defense [104]. In addition to information obfuscation defense, other approaches, such as limiting query control and creating more robust model architectures, are also efficient in defending against model extraction attacks [48]. The split FL framework called ResSFL proposed by Li et al. shows another novel approach to the mitigate model inversion attack [105].

- **Backdoor Attack:** To counteract backdoor attacks, Mammen et al. advocate for the application of weak differential privacy or the imposition of a norm thresholding mechanism on updates [55]. They suggest that

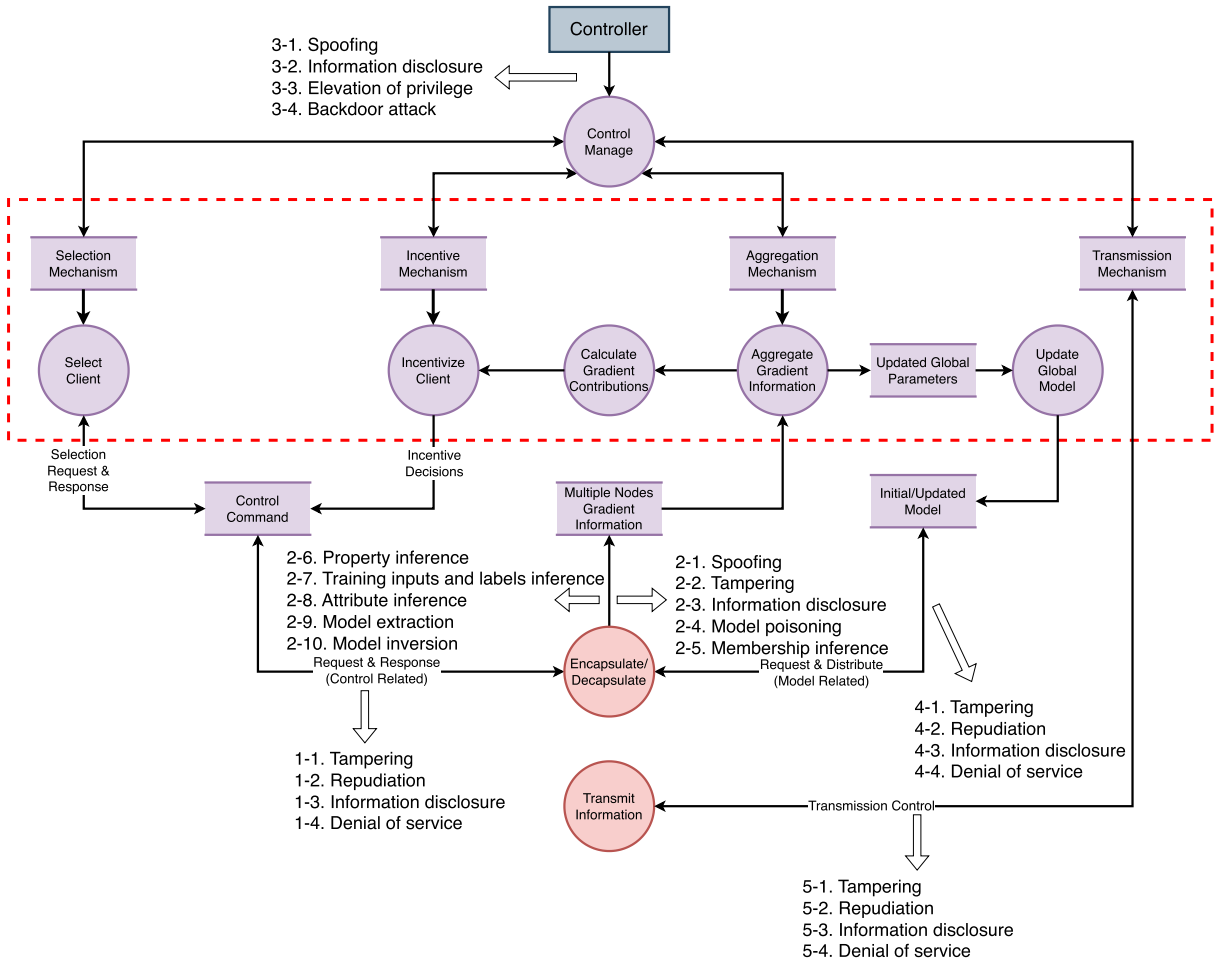

FIGURE 10. Threats in remote management layer.

norm thresholding can eliminate models with artificially boosted parameters, while the adoption of differential privacy introduces a degree of protection, albeit potentially influencing the overall model's performance [106]. It is imperative to note that the task of distinguishing malicious participants from those exhibiting misbehavior presents a formidable challenge.

- **Free-riding Attack:** The mitigation of free-riding attacks can be advanced through the incorporation of blockchain technology [107], [108]. Such an approach entails the verification of exchanged model updates among devices, thereby engendering an environment that incentivizes participants to contribute actively to the training process. It is noteworthy that an increase in communication payload has been found, primarily attributed to block size. However, these findings still highlight the potential for a fully decentralized federated learning architecture. Furthermore, Lin et al. propose STD-DAGMM, which is effective in detecting free-riders [85]. Moreover, the FRAD mechanism designed by Wang et al. outperforms their baselines in their experiments on defending against free-rider attacks [88]. In addition to defense methods, some researchers also focus on attack approaches. For example, Zhu et al. conducted an advanced free-rider attack, contributing from another perspective [86].
- **Evasion Attack:** In general, there are two main approaches to mitigate evasion attacks: empirical defense and certified defense [49]. Although empirical defense may not guarantee foolproof defense, it is efficient in addressing attacks, especially when designed to target specific types of attacks. Hong et al. identify four classes within empirical defense [109]:
  1) Gradient-Masking Defense [110], [111]: This defense involves modifying the model inference process.
  2) Input-Transformation Defense [112], [113]: This defense preprocesses the input before transformation.
  3) Adversarial Training [114], [115]: First proposed by Goodfellow et al., this approach aims to enhance model robustness by introducing

adversarial examples into the training data [78]. However, adversarial training requires more time for training as it iteratively generates adversarial examples [48].

4) Adversarial Examples Detection [116], [117]: This defense focuses on detecting adversarial examples to maintain the robustness of the model.

On the other hand, certified defense aims to guarantee consistent predictions for a classifier when encountering an adversarial example input. Randomized smoothing, introduced by Lecuyer et al., is the first method in the field of certified defense that is able to certify arbitrary classifiers of any scale [109], [118]. In addition to randomized smoothing, Katz et al. also propose a method based on mathematically rigorous techniques [119].

C. OTHER DEFENSE STRATEGIES

In addition to the problem-specific solutions mentioned above, some other approaches could improve the security level of the entire system to enhance the protection of private data. For example: Homomorphic Encryption (HE), Differential Privacy (DP), Secure Multiparty Computation (SMC), and Trusted Execution Environment (TEE).

1) HOMOMORPHIC ENCRYPTION

Homomorphic Encryption (HE) is a cryptographic technique that allows mathematical operations to be performed on encrypted data without requiring prior decryption. In conventional encryption, a shared key pair (comprising a public key and a private key) is utilized for message encryption. However, this method raises privacy concerns in certain scenarios, such as cloud services, where the control rights over data can be a subject of contention. Additionally, third parties with no direct involvement may access the contents of shared encrypted data, even without access to the encryption keys [120]. Furthermore, the identification of user elements may endure, even long after the users terminate their engagement with the service. To address these challenges, the concept of HE has been proposed as a means to enable secure data operations without revealing the original data. While the theoretical framework for Fully Homomorphic Encryption (FHE) was introduced by Gentry et al. in 2009 [121], the practical realization of its generality across diverse platforms remains an ongoing challenge [120].

Accordingly, HE can be applied in an additive manner within industrial federated learning frameworks to ensure that no update is revealed during the aggregation process. However, this approach may introduce computational and communication bottlenecks, which have been mitigated by the development of solutions like BatchCrypt [122].

2) DIFFERENTIAL PRIVACY

Differential Privacy (DP) is a privacy-preserving technique that involves adding random noise to data or model parameters to prevent the inference of private data [42], [123]. DP, grounded in information-theoretic principles [124], [125], provides robust statistical privacy guarantees by making it difficult for adversaries to extract specific information from the data.

Within federated learning, DP can be classified into two main types: Central Differential Privacy (CDP) and Local Differential Privacy (LDP) [126].

- **Central Differential Privacy:** In CDP, a central server is entrusted with a high degree of trust. Model parameters generated in each update round are initially aggregated on this centralized server before being perturbed. This approach ensures that model parameters remain concealed from other participating client nodes.
- **Local Differential Privacy:** LDP introduces a higher level of privacy, as model parameters are protected not only from external entities but also from the central server. Differential private transformations are applied to data on each participant before being transmitted to the central server. Despite offering stronger security, LDP may engender challenges in reconciling security with utility preservation [127].

The research underscores that client-level DP could effectively thwart any malicious participant's attempt to reconstruct the private sensitive data of others through the utilization of the global model [126]. Accordingly, numerous researchers have endeavored to implement client-level DP in federated learning. For instance, Geyer et al. have integrated CDP into federated learning frameworks [106], while Pihur et al. have proposed a novel federated learning framework based on LDP [128].

Despite its capability to counter inference attacks and provide privacy guarantees through the introduction of noise into clipped model parameters before aggregation [106], a trade-off between privacy protection and model accuracy exists in both CDP and LDP, since the introduction of additional noise may compromise model accuracy [126].

3) SECURE MULTIPARTY COMPUTATION

Secure Multiparty Computation (SMC), an alternative to homomorphic encryption, is a technique that enables participants to collaboratively compute results while only revealing computation results to a specific subset of involved participants. It is a crucial branch of techniques in Secure Computation (SC) that has already been implemented in federated learning [55].

SMC safeguards the original model accuracy while furnishing robust security guarantees for participants. This is particularly useful for federated learning, because computation results can be shared while maintaining data privacy within each participant. SMC ensures that even the central server can gain minimal information and only observe the aggregated results during each iteration.

Despite the additional computation and communication costs associated with SMC, it remains a significant research area. For instance, Bonawitz et al. have proposed a protocol based on Shamir's Secret Sharing, a widely-used approach in federated learning, to secure updates aggregation [129].

4) TRUSTED EXECUTION ENVIRONMENT

The Trusted Execution Environment (TEE) serves as a secure platform embedded within a device's primary processor, ensuring the confidentiality and integrity of both data and code [55]. By enforcing a dual-world implementation, TEE achieves robust isolation and attestation of secure compartments, effectively mitigating threats posed by compromised or malicious system software within the normal operational realm [130]. In contrast to SC, TEE introduces lower computational overhead when integrated into the implementation of federated learning [131].

Significant academic and industrial endeavors have been directed towards the advancement of TEEs, aiming to provide secure programmability for various device types, including high-end and mobile devices. Notable examples include Intel Software Guard Extensions (Intel SGX) and the open-source framework named Open Portable Trusted Execution Environment (OP-TEE) [132], [133]. An inherent advantage of TEE lies in its substantial reduction of the Trusted Computing Base (TCB), while simultaneously maintaining execution speed and enhancing security. Trust is established through the execution of code within the secure environment. Nonetheless, careful consideration must be given to designing the size of the TEE and mitigating potential overhead during the swapping of memory between secure and unprotected domains [130]. It is pertinent to note that Mo et al. emphasize the necessity of compact TEEs due to the associated heightened vulnerability to attacks [131]. The Privacy-Preserving Federated Learning (PPFL) approach introduced by them exemplifies the feasibility of providing robust privacy guarantees coupled with comparable utility of machine learning models, all while exerting minimal impact on communication and system costs [130].

In order to enhance security and privacy in federated learning, a combination of these mechanisms can be employed by users to establish a robust security framework for the entire system [59], [134] [135], [136].

VIII. CHALLENGES AND CONCLUSION

Federated learning, while promising and versatile, presents practical challenges as research progresses. According to Sone et al., privacy protection in federated learning focuses on minimizing privacy leakage and adhering to privacy compliance [126]. However, practical scenarios often overlook the diverse stipulations governing heterogeneous parties. Existing techniques offer uniform privacy protection across all parties [14], but in many cases, adherence to individual data owner privacy compliance is acceptable, tailoring treatment based on individual restrictions while maintaining equivalent model performance. Techniques like heterogeneous differential privacy show their promise, but the increase in communication overhead and the decrease in model accuracy remain a challenging problem for most solutions [137]. Challenges like fairness among nodes, client incentives, fault tolerance, one-shot federated learning, etc., are also vital research areas in the future [138], [139].

This paper provides a security-oriented overview of federated learning, focusing on offering readers a systematic introduction to security and risk prevention methods from the perspective of data flow. We present a comprehensive survey about FL using the proposed 5-layer reference model, which includes the data interaction layer, client management layer, local model layer, network transmission layer, and remote management layer. Additionally, we conduct a brief security threat analysis based on the STRIDE model and specific issues faced by each FL layer. Finally, we introduce some existing defense strategies for each security issue.

In summary, our proposed layered model aims to offer a concise compendium of federated learning along with data flow and control flow in the federated learning system for general readers, emphasizing the importance of system security and sensitive data protection. For researchers, we hope it could provide a standardized and layered reference model to simplify and accelerate their complex work in designing FL model training systems, alleviate development pressure, improve FL system scalability, and systematically enhance the information security of federated learning by providing a clear threat analysis approach.

REFERENCES

[1] *Cisco Annual Internet Report (2018–2023) White Paper*, Cisco, San Jose, CA, USA, 2020.
[2] A. Dogra, R. K. Jha, and S. Jain, "A survey on beyond 5G network with the advent of 6G: Architecture and emerging technologies," *IEEE Access*, vol. 9, pp. 67512–67547, 2021.
[3] R. Pryss, M. Reichert, J. Herrmann, B. Langguth, and W. Schlee, "Mobile crowd sensing in clinical and psychological trials—A case study," in *Proc. IEEE 28th Int. Symp. Comput.-Based Med. Syst.*, Jun. 2015, pp. 23–24.
[4] W. Li and M. Liewig, "A survey of ai accelerators for edge environment," in *Proc. World Conf. Inf. Syst. Technol.* Cham, Switzerland: Springer, 2020, pp. 35–44, doi: 10.1007/978-3-030-45691-7_4.
[5] W. He, G. Yan, and L. D. Xu, "Developing vehicular data cloud services in the IoT environment," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1587–1595, May 2014.
[6] D. Oletic and V. Bilas, "Design of sensor node for air quality crowdsensing," in *Proc. IEEE Sensors Appl. Symp. (SAS)*, Apr. 2015, pp. 1–5.
[7] R. Jhawar, V. Piuri, and M. Santambrogio, "Fault tolerance management in cloud computing: A system-level perspective," *IEEE Syst. J.*, vol. 7, no. 2, pp. 288–297, Jun. 2013.
[8] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan. 2018.
[9] X. Yi, F. Liu, J. Liu, and H. Jin, "Building a network highway for big data: Architecture and challenges," *IEEE Netw.*, vol. 28, no. 4, pp. 5–13, Jul. 2014.
[10] J. Konecny, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," 2016, *arXiv:1610.02527*.
[11] J. Konecny, H. B. McMahan, F. X. Yu, P. Richtárik, A. Theertha Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.
[12] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," 2016, *arXiv:1602.05629*.
[13] 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. 1–11.
[14] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3347–3366, Apr. 2023.

[15] 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.

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

[17] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140699–140725, 2020.

[18] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X20329848

[19] 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.

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

[21] R. Gosselin, L. Vieu, F. Loukil, and A. Benoit, "Privacy and security in federated learning: A survey," *Appl. Sci.*, vol. 12, no. 19, p. 9901, Oct. 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/19/9901

[22] K. K. Coelho, M. Nogueira, A. B. Vieira, E. F. Silva, and J. A. M. Nacif, "A survey on federated learning for security and privacy in healthcare applications," *Comput. Commun.*, vol. 207, pp. 113–127, Jul. 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S014036642300172X

[23] J. Hasan, "Security and privacy issues of federated learning," 2023, *arXiv:2307.12181*.

[24] E. Gabrielli, G. Pica, and G. Tolomei, "A survey on decentralized federated learning," 2023, *arXiv:2308.04604*.

[25] D. R. Kandati and S. Anusha, "Security and privacy in federated learning: A survey," *Trends Comput. Sci. Inf. Technol.*, vol. 8, pp. 29–37, Aug. 2023. [Online]. Available: https://www.peertechzpublications.org/articles/TCSIT-8-166.php

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

[27] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, and R. Cummings, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, 2021.

[28] K. Bonawitz, P. Kairouz, B. McMahan, and D. Ramage, "Federated learning and privacy: Building privacy-preserving systems for machine learning and data science on decentralized data," *Queue*, vol. 19, no. 5, pp. 87–114, Oct. 2021.

[29] Q. Li, Z. Wen, and B. He, "Practical federated gradient boosting decision trees," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 4642–4649.

[30] A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar, "Fully decentralized federated learning," in *Proc. 3rd Workshop Bayesian Deep Learn.*, Dec. 2018, pp. 1–9. [Online]. Available: http://bayesiandeeplearning.org/2018

[31] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," 2019, *arXiv:1907.02189*.

[32] 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.

[33] J. Wen, Z. Zhang, Y. Lan, Z. Cui, J. Cai, and W. Zhang, "A survey on federated learning: Challenges and applications," *Int. J. Mach. Learn. Cybern.*, vol. 14, no. 2, pp. 513–535, Feb. 2023.

[34] S. K. Lo, Q. Lu, H. Paik, and L. Zhu, "FLRA: A reference architecture for federated learning systems," 2021, *arXiv:2106.11570*.

[35] N. Truong, K. Sun, S. Wang, F. Guitton, and Y. Guo, "Privacy preservation in federated learning: An insightful survey from the GDPR perspective," *Comput. Secur.*, vol. 110, Nov. 2021, Art. no. 102402.

[36] H. Zimmermann, "OSI reference model—The ISO model of architecture for open systems interconnection," *IEEE Trans. Commun.*, vol. C-28, no. 4, pp. 425–432, Apr. 1980.

[37] T. Jayalakshmi and A. Santhakumaran, "Statistical normalization and back propagation for classification," *Int. J. Comput. Theory Eng.*, vol. 3, no. 1, pp. 1793–8201, 2011.

[38] D. Chicco, "Ten quick tips for machine learning in computational biology," *BioData Mining*, vol. 10, no. 1, pp. 1–17, Dec. 2017.

[39] J. Xu and H. Wang, "Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1188–1200, Feb. 2021.

[40] K. Zhang, X. Song, C. Zhang, and S. Yu, "Challenges and future directions of secure federated learning: A survey," *Frontiers Comput. Sci.*, vol. 16, no. 5, pp. 1–8, Oct. 2022.

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

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

[43] C. Gentry, "Computing arbitrary functions of encrypted data," *Commun. ACM*, vol. 53, no. 3, pp. 97–105, Mar. 2010.

[44] X. Zhang, S. Ji, H. Wang, and T. Wang, "Private, yet practical, multiparty deep learning," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Los Alamitos, CA, USA, Jun. 2017, pp. 1442–1452.

[45] C. Briggs, Z. Fan, and P. Andras, "A review of privacy-preserving federated learning for the Internet-of-Things," in *Federated Learning Systems: Towards Next-Generation AI*. Cham, Switzerland: Springer, 2021, pp. 21–50, doi: 10.1007/978-3-030-70604-3_2.

[46] T. DeMarco, "Structure analysis and system specification," in *Pioneers and Their Contributions to Software Engineering: sd&m Conference on Software Pioneers, Bonn, June 28/29, 2001, Original Historic Contributions*. Berlin, Germany: Springer, 2001, pp. 255–288, doi: 10.1007/978-3-642-48354-7_9.

[47] A. Shostack, "Experiences threat modeling at Microsoft," in *Proc. MODSEC@ MoDELS*, 2008, p. 35.

[48] A. Vassilev, A. Oprea, A. Fordyce, and H. Andersen, "Adversarial machine learning: A taxonomy and terminology of attacks and mitigations," NIST Trustworthy Responsible AI, Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NIST AI 100-2 E2023, 2024, doi: 10.6028/NIST.AI.100-2e2023.

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

[50] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger, "Simple black-box adversarial attacks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2484–2493.

[51] S. N. Shukla, A. K. Sahu, D. Willmott, and Z. Kolter, "Simple and efficient hard label black-box adversarial attacks in low query budget regimes," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 1461–1469.

[52] O. Suciu, R. Marginean, Y. Kaya, H. Daume III, and T. Dumitras, "When does machine learning FAIL? Generalized transferability for evasion and poisoning attacks," in *Proc. 27th USENIX Secur. Symp.*, 2018, pp. 1299–1316.

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

[54] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proc. 4th ACM Workshop Secur. Artif. Intell.*, 2011, pp. 43–58.

[55] P. Mary Mammen, "Federated learning: Opportunities and challenges," 2021, *arXiv:2101.05428*.

[56] G. Sun, Y. Cong, J. Dong, Q. Wang, L. Lyu, and J. Liu, "Data poisoning attacks on federated machine learning," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11365–11375, Jul. 2022.

[57] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," 2020, *arXiv:2007.08432*.

[58] 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. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–11.

[59] T. Gu, B. Dolan-Gavitt, and S. Garg, "BadNets: Identifying vulnerabilities in the machine learning model supply chain," 2017, *arXiv:1708.06733*.

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

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

[62] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2938–2948.
[63] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.
[64] 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. 1–11.
[65] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.
[66] 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, pp. 603–618.
[67] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 1322–1333.
[68] 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.
[69] 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.
[70] 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.
[71] C. Song and V. Shmatikov, "Overlearning reveals sensitive attributes," 2019, *arXiv:1905.11742*.
[72] S. V. Dibbo, "SoK: Model inversion attack landscape: Taxonomy, challenges, and future roadmap," in *Proc. IEEE 36th Comput. Secur. Found. Symp. (CSF)*, Jul. 2023, pp. 439–456.
[73] B. Zhao, K. Reddy Mopuri, and H. Bilen, "IDLG: Improved deep leakage from gradients," 2020, *arXiv:2001.02610*.
[74] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *Proc. 23rd USENIX Secur. Symp.*, 2014, pp. 17–32.
[75] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *Proc. IEEE 31st Comput. Secur. Found. Symp. (CSF)*, Jul. 2018, pp. 268–282.
[76] X. Gong, Q. Wang, Y. Chen, W. Yang, and X. Jiang, "Model extraction attacks and defenses on cloud-based machine learning models," *IEEE Commun. Mag.*, vol. 58, no. 12, pp. 83–89, Dec. 2020.
[77] B. Wang and N. Z. Gong, "Stealing hyperparameters in machine learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 36–52.
[78] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.
[79] J. Li, A. Siraj Rakin, X. Chen, L. Yang, Z. He, D. Fan, and C. Chakrabarti, "Model extraction attacks on split federated learning," 2023, *arXiv:2303.08581*.
[80] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *Proc. 28th USENIX Security Symp.*, 2019, pp. 1895–1912.
[81] M. Veale, R. Binns, and L. Edwards, "Algorithms that remember: Model inversion attacks and data protection law," *Phil. Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 376, no. 2133, Nov. 2018, Art. no. 20180083.
[82] M. Abadi, Ú. Erlingsson, I. Goodfellow, H. B. McMahan, I. Mironov, N. Papernot, K. Talwar, and L. Zhang, "On the protection of private information in machine learning systems: Two recent approches," in *Proc. IEEE 30th Comput. Secur. Found. Symp. (CSF)*, Aug. 2017, pp. 1–6.
[83] Z. Sun, P. Kairouz, A. Theertha Suresh, and H. Brendan McMahan, "Can you really backdoor federated learning?" 2019, *arXiv:1911.07963*.
[84] Y. Fraboni, R. Vidal, and M. Lorenzi, "Free-rider attacks on model aggregation in federated learning," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2021, pp. 1846–1854.
[85] J. Lin, M. Du, and J. Liu, "Free-riders in federated learning: Attacks and defenses," 2019, *arXiv:1911.12560*.
[86] Z. Zhu, J. Shu, X. Zou, and X. Jia, "Advanced free-rider attacks in federated learning," in *Proc. 1st NeurIPS Workshop New Frontiers Federated Learn. Privacy, Fairness, Robustness, Personalization Data Ownership*, 2021, pp. 1–10.
[87] J. Wang, X. Chang, R. J. Rodriguez, and Y. Wang, "Assessing anonymous and selfish free-rider attacks in federated learning," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2022, pp. 1–6.
[88] B. Wang, H. Li, X. Liu, and Y. Guo, "FRAD: Free-rider attacks detection mechanism for federated learning in AIoT," *IEEE Internet Things J.*, vol. 11, no. 3, pp. 4377–4388, Feb. 2024.
[89] J. Wang, X. Chang, J. Misic, V. B. Misic, and Y. Wang, "PASS: A parameter audit-based secure and fair federated learning scheme against free-rider attack," 2022, *arXiv:2207.07292*.
[90] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Proc. Eur. Conf. Mach. Learn. Knowl. Discovery Databases (ECML PKDD)*, Prague, Czech Republic. Berlin, Germany: Springer, Sep. 2013, pp. 387–402, doi: 10.1007/978-3-642-40994-3_25.
[91] M. Kearns and M. Li, "Learning in the presence of malicious errors," in *Proc. 20th Annu. ACM Symp. Theory Comput.*, 1988, pp. 267–280.
[92] X. Gong, Y. Chen, Q. Wang, M. Wang, and S. Li, "Private data inference attacks against cloud: Model, technologies, and research directions," *IEEE Commun. Mag.*, vol. 60, no. 9, pp. 46–52, Sep. 2022.
[93] B. Jayaraman and D. Evans, "Are attribute inference attacks just imputation?" in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2022, pp. 1569–1582.
[94] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, "The secret revealer: Generative model-inversion attacks against deep neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 250–258.
[95] S. Lu, W. Oh, R. Ooka, and L. Wang, "Effects of environmental features in small public urban green spaces on older adults' mental restoration: Evidence from Tokyo," *Int. J. Environ. Res. Public Health*, vol. 19, no. 9, p. 5477, Apr. 2022, doi: 10.3390/ijerph19095477.
[96] J. Wu, Q. Liu, Z. Huang, Y. Ning, H. Wang, E. Chen, J. Yi, and B. Zhou, "Hierarchical personalized federated learning for user modeling," *Proc. Web Conf.*, 2021, pp. 957–968.
[97] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[98] 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*.
[99] 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.
[100] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private language models without losing accuracy," 2017, *arXiv:1710.06963*.
[101] N. Agarwal, A. T. Suresh, F. X. X. Yu, S. Kumar, and B. McMahan, "cpSGD: Communication-efficient and differentially-private distributed SGD," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–12.
[102] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 1605–1622.
[103] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman, "SoK: Security and privacy in machine learning," in *Proc. IEEE Eur. Symp. Secur. Privacy*, Apr. 2018, pp. 399–414.
[104] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "MemGuard: Defending against black-box membership inference attacks via adversarial examples," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 259–274.
[105] J. Li, A. S. Rakin, X. Chen, Z. He, D. Fan, and C. Chakrabarti, "ResSFL: A resistance transfer framework for defending model inversion attack in split federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 10184–10192.
[106] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2017, *arXiv:1712.07557*.
[107] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," 2018, *arXiv:1808.03949*.
[108] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2438–2455, Sep. 2021.
[109] H. Hong and Y. Hong, "Certifiable black-box attack: Ensuring provably successful attack for adversarial examples," 2023, *arXiv:2304.04343*.
[110] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," 2017, *arXiv:1711.01991*.
[111] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar, "Stochastic activation pruning for robust adversarial defense," 2018, *arXiv:1803.01442*.

[112] J. Chen, M. I. Jordan, and M. J. Wainwright, "HopSkipJumpAttack: A query-efficient decision-based attack," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 1277–1294.
[113] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1778–1787.
[114] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.
[115] F. Tramer and D. Boneh, "Adversarial training and robustness for multiple perturbations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.
[116] K. Roth, Y. Kilcher, and T. Hofmann, "The odds are odd: A statistical test for detecting adversarial examples," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5498–5507.
[117] S. Jain, A.-M. Cre?u, and Y.-A. de Montjoye, "Adversarial detection avoidance attacks: Evaluating the robustness of perceptual hashing-based client-side scanning," in *Proc. 31st USENIX Secur. Symp.*, 2022, pp. 2317–2334.
[118] 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.
[119] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *Proc. 29th Int. Conf. Comput. Aided Verification (CAV)*, Heidelberg, Germany. Cham, Switzerland: Springer, Jul. 2017, pp. 97–117, doi: 10.1007/978-3-319-63387-9_5.
[120] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Comput. Surveys*, vol. 51, no. 4, pp. 1–35, Jul. 2019.
[121] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. thesis, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 2009. [Online]. Available: https://crypto.stanford.edu/craig
[122] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "BatchCrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proc. USENIX Annu. Tech. Conf.*, 2020, pp. 493–506.
[123] Q. Li, Z. Wu, Z. Wen, and B. He, "Privacy-preserving gradient boosting decision trees," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 784–791.
[124] C. Dwork, "Differential privacy: A survey of results," in *Theory and Applications of Models of Computation*. Berlin, Germany: Springer, 2008, pp. 1–19.
[125] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014.
[126] L. Song, G. Lin, J. Wang, H. Wu, W. Ruan, and W. Han, "SoK: Training machine learning models over multiple sources with privacy preservation," 2020, *arXiv:2012.03386*.
[127] R. Bassily, K. Nissim, U. Stemmer, and A. Guha Thakurta, "Practical locally private heavy hitters," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–9.
[128] V. Pihur, A. Korolova, F. Liu, S. Sankuratripati, M. Yung, D. Huang, and R. Zeng, "Differentially-private 'draw and discard' machine learning," 2018, *arXiv:1807.04369*.
[129] 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, pp. 1175–1191.
[130] F. Mo, H. Haddadi, K. Katevas, E. Marin, D. Perino, and N. Kourtellis, "PPFL: Privacy-preserving federated learning with trusted execution environments," in *Proc. 19th Annu. Int. Conf. Mobile Syst., Appl., Services*, Jun. 2021, pp. 94–108.
[131] F. Mo and H. Haddadi, "Efficient and private federated learning using tee," in *Proc. EuroSys Conf.*, Dresden, Germany, Mar. 2019, pp. 1–10. [Online]. Available: https://eurosys2019.org
[132] V. Costan and S. Devadas. (2016). *Intel SGX Explained*. Cryptology ePrint Archive. [Online]. Available: https://eprint.iacr.org/2016/086
[133] OP-TEE Core Team. (2022). *Open Portable Trusted Execution Environment*. [Online]. Available: https://www.op-tee.org
[134] G. Kaissis, A. Ziller, J. Passerat-Palmbach, T. Ryffel, D. Usynin, A. Trask, I. Lima, J. Mancuso, F. Jungmann, M.-M. Steinborn, A. Saleh, M. Makowski, D. Rueckert, and R. Braren, "End-to-end privacy preserving deep learning on multi-institutional medical imaging," *Nature Mach. Intell.*, vol. 3, no. 6, pp. 473–484, May 2021.
[135] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, "HybridAlpha: An efficient approach for privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Secur.*, Nov. 2019, pp. 13–23.
[136] S. Goryczka and L. Xiong, "A comprehensive comparison of multiparty secure additions with differential privacy," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 5, pp. 463–477, Sep. 2017.
[137] M. Alaggan, S. Gambs, and A.-M. Kermarrec, "Heterogeneous differential privacy," 2015, *arXiv:1504.06998*.
[138] A. Richardson, A. Filos-Ratsikas, and B. Faltings, "Rewarding high-quality data via influence functions," 2019, *arXiv:1908.11598*.
[139] N. Guha, A. Talwalkar, and V. Smith, "One-shot federated learning," 2019, *arXiv:1902.11175*.

JIAXING LU received the M.S. degree from the Graduate School of Interdisciplinary Information Studies (GSII), The University of Tokyo. He is currently pursuing the Ph.D. degree. His research interests include federated learning and edge computing, with a focus on the application of federated learning on network traffic optimization.

NORIHIRO FUKUMOTO received the B.E., M.E., and D.E. degrees from Waseda University, Tokyo, in 1999, 2001, and 2015, respectively. He joined KDDI R&D Laboratories Inc., in 2001, and has been engaged in research and development on speech application services, voice packetization systems over IP networks, QoS/QoE management, and 5G/B5G networks. He is currently a Visiting Associate Professor with The University of Tokyo.

AKIHIRO NAKAO (Member, IEEE) received the B.S. degree in physics and the M.E. degree in information engineering from The University of Tokyo, in 1991 and 1994, respectively, and the M.S. and Ph.D. degrees in computer science from Princeton University, in 2001 and 2005, respectively.

He was with the IBM Yamato Laboratory, Tokyo Research Laboratory, and IBM Texas Austin, from 1994 to 2005. He taught as an Associate Professor (2005–2014) and as a Professor (2014–2021) in applied computer science with the Interfaculty Initiative in Information Studies, Graduate School of Interdisciplinary Information Studies, The University of Tokyo, where he was the Vice Dean of the Interfaculty Initiative in Information Studies (2019–2021). He has moved to the School of Engineering, The University of Tokyo, since April 2021. Since April 2023, he has been the Head of the Department of System Innovations, School of Engineering. He was appointed as an Adviser to the President of The University of Tokyo (2019–2020), where he has been a Special Adviser to the President, since 2020. He has been the Director of the Collaborative Research Institute for Next-Generation Cyber Infrastructure (NGCI), The University of Tokyo, since 2021. For social services, he has been playing several important roles in Japanese government and also at research societies.

Dr. Nakao has also been appointed as the Chairperson of the 5G Mobile Network Promotion Forum (5GMF) Network Architecture Committee by Japanese Government. He has also been appointed as the Chairperson of 5G/Beyond 5G Committee, Space ICT Promotion Initiative Forum, International Committee, and Beyond 5G Promotion Consortium, since 2020. Since 2020, he has been the Chair and an Advisor of IEICE Technical Committee on Network Systems (NS) and the Chair of IEICE Technical Committee on Cross-Field Research Association of Super-Intelligent Networking (RISING). He will be the President of the Communication Society, IEICE, in 2024.