



# Exploring Threats, Defenses, and Privacy-Preserving Techniques in Federated Learning: A Survey

**Ren-Yi Huang**<sup>ID</sup>, University of South Florida

**Dumindu Samaraweera**<sup>ID</sup>, Embry-Riddle Aeronautical University

**J. Morris Chang**<sup>ID</sup>, University of South Florida

*This article presents a comprehensive survey of both attack and defense mechanisms within the federated learning (FL) landscape. Furthermore, it explores the challenges involved and outlines future directions for the development of a robust and efficient FL solution.*

**W**ith advances in hardware computing power, artificial intelligence (AI) has gained widespread adoption, enhancing productivity and automating repetitive tasks. However, even though AI can be adopted in almost every field, it is difficult to apply this technology to all industries because of data sensitivity, privacy issues, or limited computational resources. Federated learning (FL) offers a solution for training models without the need to

store all data on a centralized server. FL clients participate in the training process using their local data, without sharing it with other clients or the server. Therefore, each client can both contribute to and benefit from a global model.

While FL has made strides in data privacy, vulnerabilities remain in both model security and client data privacy. In terms of security risks, attackers may attempt to compromise the global model by introducing specialized patterns into the dataset, altering the class labels of targeted data, or modifying updates to already-trained models. Additionally, malicious clients may benefit from global models without contributing their data or

Digital Object Identifier 10.1109/MC.2023.3324975  
Date of current version: 5 April 2024

computing resources. For the privacy risks, the malicious server or clients launch attacks to examine whether a selected data point is used, identify the data labels, or reconstruct the data that the honest client used during the training stage.

To safeguard model utility and client privacy, over the past, various techniques have been proposed. Security-enhancing mechanisms aim to detect or mitigate abnormal updates. In terms of privacy-preserving mechanisms, secure multiparty computation (MPC) allows clients to fragment and interchange local updates, making it difficult for attackers to identify the origin of any update. Differential privacy (DP) enables clients to introduce calibrated noise into local updates, preventing the actual values to be distinguished. Homomorphic encryption (HE) lets clients encrypt local updates, enabling the server to generate an aggregated model without accessing the raw data. Hybrid approaches also exist, combining multiple techniques to offer a more robust defense against potential adversaries.

The main contribution of this study can be summarized as follows: 1) We classify attacks based on whether they target the security or privacy aspects of clients' data or the model itself. This categorization provides a structured understanding of the threat landscape in FL. 2) To understand the diverse attack vectors in FL, we provide a detailed taxonomy of each attack and defense mechanism. 3) To have a clear overview of the current advancements and strategies employed to mitigate FL vulnerabilities, we summarize existing state-of-the-art research in each domain by providing a comparative table of time complexity among

different privacy-preserving mechanisms. 4) Finally, the study also outlines potential research directions and challenges pertaining to the integrity of FL.

In the forthcoming sections, we commence by introducing the FL system along with its corresponding threat models. We proceed to investigate a spectrum of attacks that encompass model security and client privacy, followed by a comprehensive examination of existing defense strategies. Finally, we culminate with a conclusion that encapsulates our findings and offers prospective avenues for advancing security and privacy within the realm of FL.

## BACKGROUND

### Federated Learning

Similar to decentralized machine learning, FL allows the data to be stored locally on individual client. These clients join the training process and use their own local data to construct the final model cooperatively. Therefore, the privacy of local data are preserved because the clients' data are not shared to others. [Figure 1](#) depicts stages of training process in FL.

1. The central server selects a subset of clients to join the training round.
2. The central server sends a model to the clients, and the clients train local models with their own local data.
3. Clients send local updates (the model parameters or gradients) to the server.
4. The server aggregates a global model based on received updates and sends the global model to clients.

According to clients' data distribution, FL can be classified into horizontal FL and vertical FL. In horizontal FL, or sample-based FL, clients' data share similar features but different samples. In vertical FL, or feature-based FL, clients' data differ in features, but the samples might be the same. The FL scheme discussed in this article is horizontal FL.

### Threat models

Despite data being distributed and stored client-side, vulnerabilities remain for attacks against both the trained models and client data. In general, the adversaries can be categorized into the following types:

1. *Semi-honest server*: A semi-honest server, also called *honest-but-curious*, follows FL protocols but aims to extract client data from received updates or the global model.
2. *Malicious server*: A malicious server doesn't follow the protocols in FL and actively gleans private information through modifying the received data.
3. *Malicious clients/colluding clients*: Similar to the malicious server, malicious clients don't obey the protocols and attempt to obtain honest clients' information. Colluding clients work together to amplify their attacks and may even team up with a malicious server.

In addition to adversaries, trusted third parties (TTPs) may be adopted for generating and distributing cryptographic keys to clients and servers, thereby ensuring that the plaintext data are effectively converted into secure ciphertext.

Attacks on FL can be categorized by objective: Attacks targeting the security of the model aim to compromise its robustness through data or model poisoning. Attacks targeting the privacy of the model seek to obtain clients' local training data.

Given that potential attackers could be the server, any client, or groups of colluding clients, it is necessary to understand how these attacks are executed against the FL system and what defense mechanisms are available. The subsequent section will introduce the core concepts of these attacks and outline corresponding defense mechanisms as depicted in Figure 2.

**THREATS AND ATTACKS**

This section classifies threats into two categories: those that degrade model accuracy as attacks targeting

security of the model and those aiming to obtain client data as attacks targeting privacy of the model.

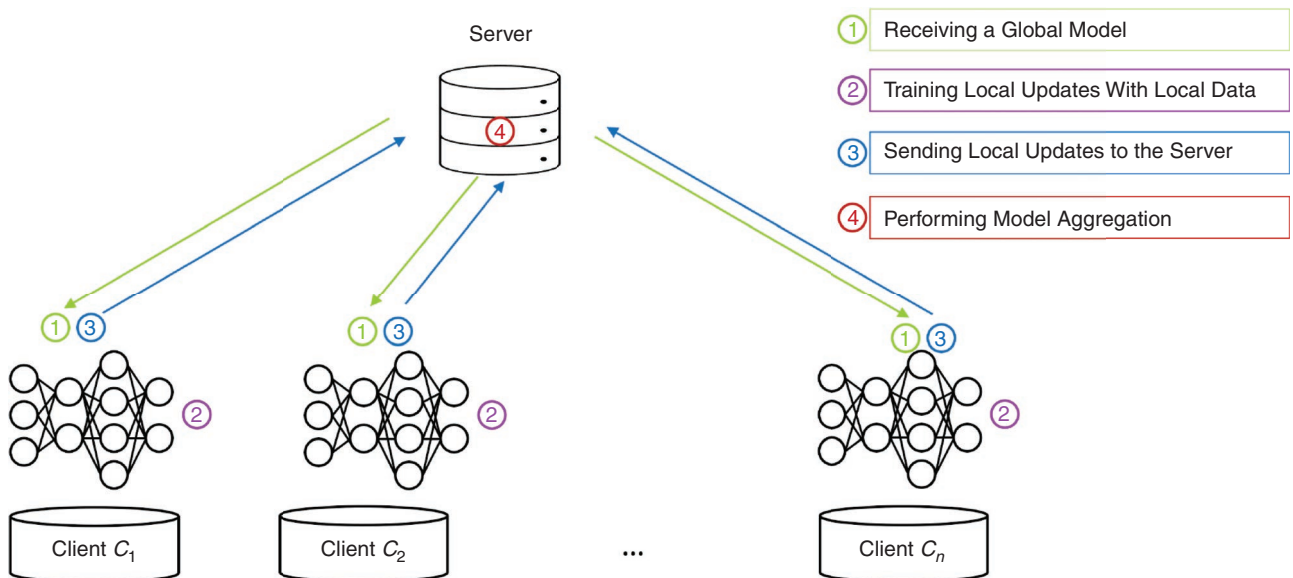
**Attacks targeting security of the model**

There are types of attacks that try to reduce the accuracy of the global model, these attacks are introduced in the following subsections.

**Data poisoning attack.** A data poisoning attack is a type of attack when malicious clients corrupt local data to compromise the global model. Such attacks fall into two subcategories: clean-label and dirty-label poisoning. The goal is to influence the global model to make incorrect predictions.

In a clean-label poisoning attack, the attacker subtly modifies the data features without altering the labels.

These altered features, combined with the correct labels, are used to train a flawed model that is prone to misclassifying data with specific patterns. To evade detection, the attacker measures the similarity between their updates and those from benign clients and adds minimal perturbation to ensure the poisoned data closely resembles the target data. However, the attack is temporary. If the attackers stop sending the model updates, the backdoor quickly disappears. It is because benign clients continue to upload their model updates, which contain gradients that counteract those of the attacker. Zhang et al.<sup>1</sup> in their article explain the sparseness introduced by stochastic gradient descent (SGD), the majority of the  $l_2$  norm of the aggregated benign gradient only exists in a few coordinates. Therefore, if the attacker only updates the coordinates in gradients that benign



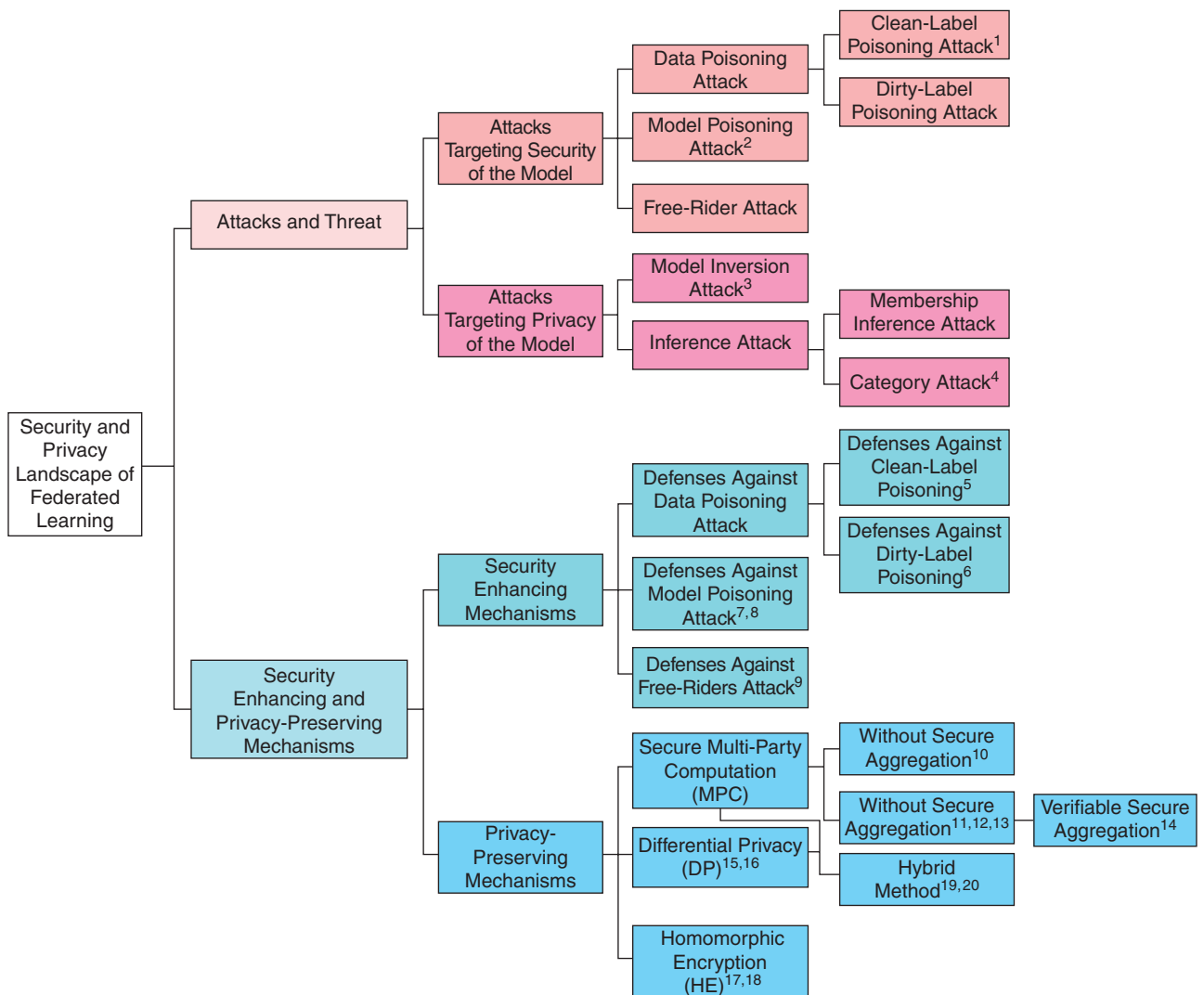
**FIGURE 1.** Training procedure in FL. A central server initiates the process by distributing a global model to multiple decentralized client devices or servers. Each client uses its local data to perform model updates through several rounds of training, typically using techniques like SGD. These local updates, instead of raw data, are then shared back with the central server.

clients rarely update, the backdoor can persist for a longer period.

Compared to clean-label poisoning attack, dirty-label poisoning attacks allow the attacker to alter the labels directly, causing the global model to misclassify the classes of the source data. For example, in a label-flipping attack, the attacker swaps labels between source and target samples.

**Model poisoning attack.** In model poisoning attacks, attackers manipulate local model updates to impair the global model. For example, a malicious client might inject Gaussian noise into their updates or mimic updates from benign clients to compromise the aggregated model's accuracy. Instead of generating random noise for the model update, Cao et al.

propose creating low-accuracy models using fake clients inserted into the FL system.<sup>2</sup> Here, the attacker chooses a base model—identical in architecture to the global model but with low accuracy—as the starting point. The aim is to degrade the global model's performance like the suboptimal base model. During the attack, the direction of the model



**FIGURE 2.** A taxonomy of threats, defenses, and privacy-preserving techniques in FL.

update is determined by the differences in model parameters between the global and base models, thereby increasing the impact by scaling up the fake updates.

**Free-rider attack.** Free-riders are malicious clients that send fake local updates to exploit the global model without contributing resources. These updates don't degrade the model's performance, allowing free-riders to benefit without any investment. There are various methods proposed for free-riders to generate fake updates: 1) The free-rider randomly samples the update values from a uniform distribution, 2) the free-rider utilizes the difference of parameters between two global models sent from the server, or 3) the free-rider applies Gaussian noise with the parameters difference between two global models.

### Attacks targeting privacy of the model

In addition to compromising the effectiveness of the models, attackers can also gain access to sensitive client information, such as the data used for training. This section will explore the various methods attackers use to obtain such private information.

**Model inversion attack.** In a model inversion attack, or deep leakage from gradient, a malicious server tries to reconstruct client data. The server builds a secondary model that learns from extracted gradients in the aggregated model. By using gradient descent to minimize the distance between dummy and target gradients, the server can approximate client training data. Existing methods use metrics like Euclidean distance or cosine similarity to measure the gradients'

magnitude and direction. However, the local updates can be compressed before transmission to alleviate communication bottlenecks, which makes the DLG attack less effective because of the information loss during the compression. Therefore, Yang et al. introduce the highly compressed gradient leakage attack, designed to reconstruct both data and labels from compressed gradients.<sup>3</sup> Initially, an "Init-Generation" step is utilized to create dummy data that compensates for the information loss due to compression. A modified objective function is then used to train the dummy data under the gradient compression scenarios. Finally, a denoising model is used to achieve high-quality data reconstruction.

**Inference attack.** Inference attacks share similarities with model inversion attacks and are also initiated by a malicious server. The goal here is to determine whether a specific data sample has been used by clients or to identify the class labels of client data.

- › *Membership inference attack:* In a membership inference attack, an attacker aims to identify if a specific data sample was used by clients. The attacker utilizes data with a distribution like that of the honest clients' local data to make this determination. Moreover, the attacker can monitor the global model, and then craft parameters to make the attack more effective. However, these attacks generally require large datasets to be effective. To generate more diverse data, generative adversarial network (GAN) is applied in conjunction with the attack model to predict

membership information. The GAN is designed to generate data that mimics the distribution of the original dataset. This synthesized data is then combined with the original data to train the attack model, thereby enhancing its capability to determine membership information.

- › *Category attack:* In addition to the training data, attackers may also seek to uncover label information. Gao et al. propose a category inference attack using a multilabel inference model.<sup>4</sup> Given a target client's model update, this inference model can then be trained to predict the corresponding category labels.

## SECURITY ENHANCING AND PRIVACY-PRESERVING MECHANISMS

In the preceding section, we delineated various attack methodologies that compromise the security and privacy of FL systems. This section is dedicated to outlining strategies designed to fortify the security and privacy of the FL systems.

### Security-enhancing mechanisms

There are approaches that aim to strengthen the robustness of the trained models used in FL. These methodologies seek to mitigate the impact of a variety of attacks and enhance overall model security.

**Defenses against data poisoning attack.** Zhang et al. propose a trigger reverse mechanism<sup>5</sup> to defend against backdoor attacks. In the warm-up phase, a distance matrix is generated to measure class distance between all



class pairs. Pairs with large distances are chosen to form the promising pairs. Next, if the client has enough local data for both labels of pair, it will perform symmetric hardening by purposely inserting backdoor patterns to both datasets and flip the labels, insert the triggers on the classes, and update the distance matrix of the promising pairs. If the client has insufficient data, on the other hand, an asymmetric hardening is used on the source label to target only, which the backdoor patterns are injected. Next, the distance matrix is updated. When the benign clients adopt hardening mechanisms to counter the malicious clients' backdoor updates, the aggregated model tends to generate low confidence when predicting backdoor data. Therefore, the data with low prediction confidence can be filtered out during the inference stage.

Yin et al. propose a method using historical local updates to distinguish between benign and malicious clients.<sup>6</sup> The server stores this history and calculates the gradient difference between each client's updates and the historical average. Clients are then classified as benign or malicious based on this difference. The global model is subsequently aggregated using weighted contributions from both client groups.

**Defenses against model poisoning attack.** Like data poisoning attacks, model poisoning attacks makes the malicious updates deviate from benign updates or the optimal update.

Panda et al. employes gradient sparsification to mitigate the attack, selecting gradients with the highest magnitude for aggregation.<sup>7</sup> Clients train local updates which are then clipped based on their L2 norm. The server aggregates these updates

through averaging and adds the result to an error feedback vector. The top-k magnitude coordinates are extracted from this vector and zeroed out. These top-k coordinates are then used to update the global model.

HE can be employed as a defense mechanism against model poisoning attacks. Ma et al. propose a privacy-preserving defense strategy through two-trapdoor HE, which resists encrypted model poisoning attacks without violating privacy.<sup>8</sup> Specifically, the encrypted local models are used to calculate the secure cosine similarity. To prevent benign updates from being misjudged as malicious, a Byzantine-tolerance aggregation method based on confidence scores is used for aggregation, rather than discarding abnormal model updates.

**Defenses against free-riders attack.**

Free-riders are the malicious clients who utilize the global model without contributing effective training locally. Chen et al. propose the weight evolving frequency matrix (WEF-Matrix), a system in which each client maintains a record of the frequency of weight changes.<sup>9</sup> Both the WEF-Matrix and the local updates are sent to the server. Based on the differences observed in the WEF-Matrices, the server can distinguish between free-riders and benign clients. Ultimately, the server prevents free-riders from gaining access to high-quality models.

**Privacy-preserving mechanisms**

This section covers defense mechanisms focused on privacy. Cryptographic methods such as secure MPC, DP, and HE mitigate threats like gradient leakage and inference attacks, protecting client data without compromising model utility.

**Secure MPC.** Secure MPC enables collaborative computation among multiple parties while keeping individual data private. In FL, this prevents both the server and clients from accessing sensitive data. As shown in Figure 3, each client splits its local model update into shares, generates noise, and distributes these among other clients. These perturbed shares are combined into new updates and sent to the central server for aggregation into a global model. Finally, each client removes the noise to reconstruct the original model, preserving both data integrity and privacy.

However, as the number of clients increases, communication overhead also increases. One solution is separating clients into different groups. Kanagavelu et al. consider communication latency to split clients into groups.<sup>10</sup> Each group has a leader responsible for aggregating local updates and sending this group-level model to the server. The server then aggregates these into a global model, reducing overall communication load.

Bonawitz et al. present a secure aggregation protocol using cryptographic methods to protect clients from both malicious servers and colluding clients.<sup>11</sup> In this protocol, model information is securely shared between clients and servers through cryptographic means. The protocol involves the following rounds:

1. *Advertise keys:* Each client generates key pairs and shares the public keys with other clients through the server.
2. *Share keys:* Each client generates its secret shares for masking, and the ciphertexts are broadcast to the other clients through the server.

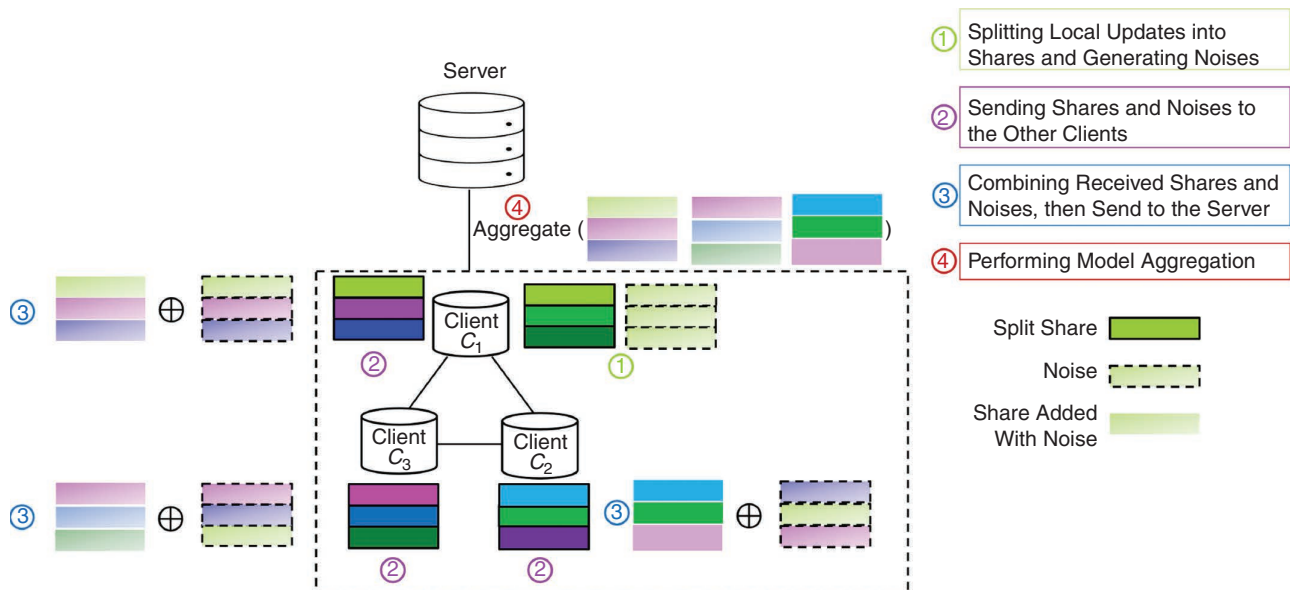
3. *Masked Input Collection:* After local training, each client applies double-masking to hide the information of its local update with received masks and send the masked update to the server.
4. *Consistency Checks:* Each client receives a list of surviving clients from the server, generates its signature, and sends the signature to the server.
5. *Unmasking:* If there are sufficient validate signatures, the server aggregates the shared secrets to generate the global model.

There are also concerns about efficiency. The use of secure two-party computation, a subset of MPC, can reduce computational time.<sup>12</sup> Another

method to improve efficiency is rearranging the clients.<sup>13</sup> Instead of connecting all clients together and communicating to all clients, they are arranged into groups as a tree structure, where each group performs intragroup secret sharing and aggregation to generate aggregated models for groups. Next, intergroup communication is performed when a child group sends a secrete share to the parent group. Lastly, the server receives a sufficient amount of aggregated results from groups and generates the final aggregated model.

Even with these protective measures, the server could potentially distribute a manipulated global model. To counteract this, a verification stage is proposed as a verifiable FL scheme. Clients encrypt local updates and generate proofs for verification. After aggregation, each client verifies

the proofs to ensure the aggregated result is trustworthy. Nevertheless, the verification phase introduces computational overhead. The bottleneck in the verifiable aggregation scheme occurs when the server recovers the hashes of dropout users one by one, which becomes worse if dropouts are frequent in a large FL system. Therefore, a one-shot aggregate hash recovery based on linearly homomorphic hashes is proposed for the server to boost the time on recovering hash.<sup>14</sup> In this approach, each client carefully encodes a mask which is used to protect the hash for verification. These masks allow the server to generate the aggregate hash for participating clients in one shot. Because of the linear homomorphism of the hashes, surviving clients can verify the integrity of the aggregated results.



**FIGURE 3.** Training procedure of MPC in FL. In this collaborative paradigm, data are partitioned across multiple parties, each with its own local model. During each training iteration, parties compute gradients on their local data, preserving its privacy through encryption or secure sharing techniques. These encrypted gradients are then securely aggregated, creating a global model update.

**DP.** DP seeks to maintain a balance between data protection and query accuracy by adding random noise to the data. In FL, DP is applied to local updates, obscuring individual data points to counter inference attacks. However, this added

noise can compromise the global model's accuracy.

A solution proposed by Cheng et al. aims to counteract this tradeoff by sparsifying the local gradients.<sup>15</sup> Specifically, it introduces a regularization term to the objective function,

constraining the l2 norm of local updates. A subset of parameters is then set to zero, creating sparsified local updates. By doing so, the amount of DP noise required is reduced, thereby preserving both privacy and model accuracy. Instead of applying uniform

**TABLE 1.** Comparison of communication and computation cost of privacy preserving in FL.

Approach	Require TTP? <sup>a</sup>	Overall computation cost		Overall communication cost	
		Computation cost on server	Computation cost per client	Communication cost on server	Communication cost per client
Kanagavelu et al. <sup>10</sup>	No	Not applicable, no sufficient information		Total number of message exchanged: $O(K^2 + C^2)$ <sup>b</sup>	
Bonawitz et al. <sup>11</sup>	No	$O(MN^2)$	$O(MN + N^2)$	$O(MN + N^2)$	$O(M + N)$
Fereidooni et al. <sup>12</sup>	No	$O(MN)$	$O(M)$	$O(MN)$	$O(M)$
Jahani-Nezhad et al. <sup>13</sup>	No	$O \frac{N - D - 1}{N - T - D} \cdot M \cdot \log^2(N - D - 1)$	$O M \frac{N}{N - T - D} + \frac{N}{N - T - D} \log^2(N - D - 1)$	$1 + \frac{T}{N - T - D} M$	$1 + \frac{T + D}{N - T - D} M$
Buyukates et al. <sup>14 c</sup>	No	$O(N \cdot \log(N))$	$O N \log(N) + \frac{B + 1}{B} M$	$O(N)$	$O(N)$
Cheng et al. <sup>15</sup>	No	$O(1)$	$O(1)$	$O(MN)$	$O(M)$
Miao et al. <sup>16</sup>	No	$O(N + A)$	$O(A \cdot (L \cdot \log_2(L) + L))$	$O(MN)$	$O(M)$
Jiang et al. <sup>17</sup>	No	$O(1)$	$O(A + L \cdot \log(L))$	$O(MN)$	$O(M)$
Ma et al. <sup>18</sup>	No	$O(1)$	$O(N)$	$O(N(N + M + S))$	$O(N + M + S)$
Truex et al. <sup>19</sup>	No	Not applicable, no sufficient information		$2VP + VN + N^d$	
Xu et al. <sup>20</sup>	Yes	Not applicable, no sufficient information		$VN + V + N^d$	

Here,  $N$  is the total number of clients,  $V$  is the total number of servers,  $P$  is the number of required clients for decryption (for hybrid methods),  $K$  is the number of clients in group (for CE-Fed),  $C$  is the elected committee members (for CE-Fed),  $M$  is the model size,  $D$  is the number of dropout users,  $T$  is the number of semihonest users,  $L$  is the size of layers in a model,  $A$  is the number of layers in a model,  $S$  is the size of secret share, and  $B$  is the batch size for verification.

<sup>a</sup>The TTP may be required to generate and advertise the keys to participants in the FL system.

<sup>b</sup>The complexity is the total number of exchanged messages intra and inter groups for model aggregation in CE-FED where  $K, C \gg N$ .

<sup>c</sup>The communication only considers the operations related to verification.

<sup>d</sup>For hybrid methods, the number of crypto-related operations is measured, and there are multiple servers for counting the number of operations.



DP noise across all layers in the local update, the noise could also be added layer-wise. Miao et al. takes the weight range of different layers into consideration.<sup>16</sup> By calculating the offset from the range center of each weight and compressing the local updates, the total amount of DP noise is fewer than applying DP noise uniformly in the whole model.

**HE.** HE allows for computations on encrypted data, producing an encrypted result that mirrors calculations on the plaintext. In FL, HE enables the secure aggregation of encrypted local updates, thwarting attempts to extract information from the models.

While HE enhances client data privacy, it also incurs computational and efficiency costs. To mitigate this, local updates are batched together for encryption, reducing transmission time. However, batch encryption conflicts with sparsification techniques used to compress the model, because each local update has unique sparsified coordinates that the server can't accurately match for updates. Hence, Jiang et al. individually sparsify and encrypt each local update.<sup>17</sup> Transmitting only the sparsified updates, rather than the entire model, substantially decreases both encryption and decryption times.

Despite its security features, HE can be vulnerable to potential privacy leaks. If clients and servers share the same set of public and private keys, it provides collusion between malicious servers and compromised clients, risking the exposure of data from honest clients. To mitigate the privacy issue, Ma et al. propose an xMK-CKKS-based method which exploits an aggregated public key,

and each client has its own private key.<sup>18</sup> The aggregated public key is used to encrypt the local updates, and each clients' private keys are used for decryption. Since decryption involves multiple shares that contain individual secret keys and aggregated ciphertexts, the security of each ciphertext is fortified.

**Hybrid methods.** To bolster privacy in FL, multiple techniques can be integrated together. Truex et al. combines MPC with DP using the threshold variant of the Paillier cryptosystem.<sup>19</sup> Clients encrypt their local model updates, adding noise for DP. Because of the Paillier cryptosystem's homomorphic properties, the server can aggregate these encrypted models without needing to decrypt the clients' models. Selected clients then decrypt the aggregated model, which is subsequently sent to all participating clients. The server then selects a set of clients to decrypt the aggregated model and send it to all clients.

Although MPC and DP offer privacy advantages, they come with drawbacks like high computational time and the necessity of knowing the client count for decryption. Instead of using HE to achieve MPC, Xu et al. adopts multi-input functional encryption to encrypt the models from the clients.<sup>20</sup> In addition, a third party authority generates the keys and performs decryption, which further reduces the communication between the server and clients while enhancing privacy.


We summarize and compare the computation and communication complexity between existing approaches in [Table 1](#) based on different types of privacy-enhancing technologies for FL.

**D**ata privacy and security have become increasingly critical in the application of AI technologies. FL offers a promising approach that reconciles the capabilities of machine learning models with the growing requirements for security and data privacy. In this article, we provided a comprehensive taxonomy of potential vulnerabilities within FL landscape and described how malicious participants may exploit these weaknesses. Moreover, we introduced the existing defense mechanisms designed to bolster security and preserve the privacy of sensitive data. Given the growing importance of data privacy, [Table 1](#) serves as a resource to guide FL enthusiasts, researchers, and participants in the industry in selecting the most appropriate mechanisms for enhancing data privacy within these systems.

Although there have been various approaches proposed to solve security and privacy issues in FL, this topic still contains challenges to be solved and improved. The followings summarize a few key observations:

1. *Enhance both security and privacy:* Most existing research tends to focus on either security or privacy, rarely addressing both issues simultaneously. However, in real-world scenarios, these two challenges often coexist and interact in complex ways. Therefore, it is crucial to develop integrated mechanisms that address both security and privacy concerns concurrently.
2. *Efficiency while applying privacy-preserving techniques:* While privacy-preserving techniques like secure MPC,

secure aggregation, and HE are crucial for data protection, they often increase computational and communication overhead. Therefore, it is necessary to develop efficient privacy-enhancing methods that mitigate additional computational and communication burdens.

By addressing these key research directions, we can pave the way for more robust and privacy-aware FL systems that can meet the demands of next-generation real-world scenarios. 

## REFERENCES

1. Z. Zhang et al., "Neurotoxin: Durable backdoors in federated learning," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2022, pp. 26,429–26,446.
2. X. Cao and N. Z. Gong, "MPAF: Model poisoning attacks to federated learning based on fake clients," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 3396–3404.
3. H. Yang et al., "Using highly compressed gradients in federated learning for data reconstruction attacks," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 818–830, 2023, doi: 10.1109/TIFS.2022.3227761.
4. J. Gao et al., "Secure aggregation is insecure: Category inference attack on federated learning," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 147–160, Jan./Feb. 2023, doi: 10.1109/TDSC.2021.3128679.
5. K. Zhang et al., "Flip: A provable defense framework for backdoor mitigation in federated learning," 2022, *arXiv:2210.12873*.
6. C. Yin and Q. Zeng, "Defending against data poisoning attack in federated learning with non-IID data," *IEEE Trans. Comput. Social Syst.*, early access, 2023, doi: 10.1109/TCSS.2023.3296885.
7. A. Panda et al., "SparseFed: Mitigating model poisoning attacks in federated learning with sparsification," in *Proc. Int. Conf. Artif. Intell. Statist.*, PMLR, 2022, pp. 7587–7624.
8. Z. Ma et al., "ShieldFL: Mitigating model poisoning attacks in privacy-preserving federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 1639–1654, Apr. 2022, doi: 10.1109/TIFS.2022.3169918.
9. J. Chen et al., "Rethinking the defense against free-rider attack from the perspective of model weight evolving frequency," 2022, *arXiv:2206.05406*.
10. R. Kanagavelu et al., "CE-Fed: Communication efficient multi-party computation enabled federated learning," *Array*, vol. 15, no. 6, Jun. 2022, Art. no. 100207, doi: 10.1016/j.array.2022.100207.
11. K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191, doi: 10.1145/3133956.3133982.
12. H. Fereidooni et al., "SAFElearn: Secure aggregation for private federated learning," in *Proc. IEEE*

## ABOUT THE AUTHORS

**REN-YI HUANG** is a research assistant at the University of South Florida, Tampa, FL 33620 USA. His research interests include federated learning and privacy-preserving technologies for machine learning. Huang received a master's in computer science and information engineering from National Formosa University. Contact him at hr219@usf.edu.

**DUMINDU SAMARAWEERA** is an assistant professor at Embry-Riddle Aeronautical University, Daytona Beach, FL 32114 USA. His research interests include the intersection of cybersecurity and data science, with a particular emphasis on security and privacy-enhancing technologies for data science. Samaraweera received a Ph.D. in electrical engineering from the University of South Florida. Contact him at samarawg@erau.edu.

**J. MORRIS CHANG** is a professor in the Department of Electrical Engineering, University of South Florida, Tampa, FL 33620 USA. His research interests include cybersecurity, wireless networks, energy-aware computing, and object-oriented systems. Chang received a Ph.D. in computer engineering from North Carolina State University. Chang is an associate editor in chief of *IT Professional*. He is a senior member of IEEE. Contact him at chang5@usf.edu.

- Security Privacy Workshops (SPW), 2021, pp. 56–62, doi: 10.1109/SPW53761.2021.00017.
13. T. Jahani-Nezhad et al., “SwiftAgg+: Achieving asymptotically optimal communication loads in secure aggregation for federated learning,” *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 977–989, 2023, doi: 10.1109/JSAC.2023.3242702.
  14. B. Buyukates et al., “LightVeriFL: Lightweight and verifiable secure federated learning,” in *Proc. Workshop Federated Learn., Recent Adv. New Challenges (Conjunction NeurIPS)*, 2022.
  15. A. Cheng et al., “Differentially private federated learning with local regularization and sparsification,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10,122–10,131.
  16. Y. Miao et al., “Compressed federated learning based on adaptive local differential privacy,” in *Proc. 38th Annu. Comput. Secur. Appl. Conf.*, 2022, pp. 159–170, doi: 10.1145/3564625.3567973.
  17. Z. Jiang, W. Wang, and Y. Liu, “FLASHE: Additively symmetric homomorphic encryption for cross-silo federated learning,” 2021, *arXiv:2109.00675*.
  18. J. Ma et al., “Privacy-preserving federated learning based on multi-key homomorphic encryption,” *Int. J. Intell. Syst.*, vol. 37, no. 9, pp. 5880–5901, Sep. 2022, doi: 10.1002/int.22818.
  19. S. Truex et al., “A hybrid approach to privacy-preserving federated learning,” in *Proc. 12th ACM Workshop Artif. Intell. Secur.*, 2019, pp. 1–11, doi: 10.1145/3338501.3357370.
  20. R. Xu et al., “HybridAlpha: An efficient approach for privacy-preserving federated learning,” in *Proc. 12th ACM Workshop Artif. Intell. Secur.*, 2019, pp. 13–23, doi: 10.1145/3338501.3357371.

**IEEE COMPUTER SOCIETY**  
**Call for Papers**

Write for the IEEE Computer Society's authoritative computing publications and conferences.

**GET PUBLISHED**  
[www.computer.org/cfp](http://www.computer.org/cfp)

IEEE COMPUTER SOCIETY

IEEE