# Survey on Privacy Protection in Federated Learning

**Jiacheng Lu**
Universiteit van Amsterdam
Vrije Universiteit Amsterdam
jiacheng.lu@student.uva.nl

## Abstract

Federated learning is a multi-part collaborative machine learning system. Unlike traditional centralized machine learning, federated learning transfers training tasks to the client side and only sends the model parameter results to the server. In addition, by introducing more user participation, federated learning can expand the training data set as a whole, thereby improving the quality of the overall model. This article mainly explores the privacy challenges and protection technologies in federated learning. First, the preliminary knowledge and threat models are introduced. Second, the privacy vulnerabilities of federated learning are summarized from the aspects of attacks. Third, the research achievements are summarized according to the privacy protection solutions including homomorphic cryptosystem, secure multi-party computation and differential privacy. Finally, by comparing these solutions, the future research is prospected.

## 1 Introduction

As machine learning algorithms have shown obvious advantages in most recognition-related fields, a large amount of intelligence applications have emerged in many industries. However, with the rapid development of data-driven intelligent applications, the machine learning paradigm is also facing new dilemmas and challenges[1]. On the one hand, the machine learning paradigm hopes to provide all users with a robust and efficient functional service. On the other hand, data as the "nutrition" of the learning algorithm is difficult to fully share[2]. Moreover, due to the need to store a large amount of data in a centralized manner, traditional machine learning often causes the problem of privacy leakage of the original data owner and even triggers some more serious social problems, such as the data leakage incident of Facebook[3]. To this end, many stringent and comprehensive regulations to protect user privacy have been introduced. An example is the General Data Protection Regulation (GDPR) promulgate by the EU. The passage of these regulations and laws effectively protects user privacy, but it also restricts the development of machine learning, making data difficult to mine and use.

Federated learning was first proposed by Google to establish a sharing model between mobile terminals and servers[4]. In federated learning, the participants upload the updated parameters to the server after training on the local data, and then the server aggregates to obtain the overall parameter and final model. Considering that the raw data does not need to be shared and will not leave the owner's local device at all, this framework can guarantee the protection of the privacy of each participant[5]. The training phase of federated learning model mainly includes the following steps:

1. Participants download the global model from the cloud server.
2. Clients train the model on local data to obtain new model parameters.
3. Clients send the latest model update to server.

4. Server receives multiple model updates and takes a certain way to aggregate to update the global model.

Although such a training method of exchanging model parameters without sharing original data can effectively protect the privacy of users, federated learning still faces some privacy threats. Some researches have shown that malicious participants can reversely infer the user's sensitive data based on the differences in the federated learning gradient parameters in each iteration. Therefore, federated learning needs protect the privacy of parameter transmission and storage during the training process. The attackers can act as clients, server or model analyst to make use of sensitive information contained in parameter updates to perform model inversion attacks or model extraction attacks.

In order to combat these attacks, many researches have explored the application of privacy-preserving technologies in federated learning. The current methods are mainly divided into two categories, one is encryption methods, such as secret sharing, homomorphic encryption, SMC, etc.; the other is data perturbation methods, such as differential privacy. Encryption methods provide effective means for data privacy protection by encoding plaintext into ciphertext and only allowing specific persons to decode it. However, they often require large computational overhead and are more difficult to apply in actual scenarios. Data perturbation methods refer to adding randomized noise to the data to avoid potential privacy information inference from the intermediate parameters. It is more lightweight but it will affect the model performance. Therefore, it is critical to balance between privacy and usability.

The topic of this literature study is "Privacy protection in federated learning". Therefore the research questions and the motivation are as follows:

- *Q1: What is federated learning? How is its development? What is its current concern?*
  I try to summarize the framework, development and current concerns of federated learning.
- *Q2: What is the privacy threat to federated learning? How to attack?*
  I try to analyze in detail the various privacy threats faced by the federated system from the perspective of federated learning attacks, and systematically classify and summarize the attack methods.
- *Q3: How do defend against the above privacy threats? What are these privacy protection techniques' pros and cons? How is the application of these techniques to FL?*
  In response to the privacy threats, I try to analyze the existing privacy protection technologies in federated learning.
- *Q4: What is the future research direction of privacy protection in FL?*
  I try to identify the inadequacy of current researches and propose the potential future work.

## 2 Federated learning

As mentioned above, the main purpose of federated learning[4] is to jointly train machine learning models with the help of private training data generated by multiple mobile devices. Data privacy involving distributed mobile users can be protected by uploading only model parameters (such as gradients) instead of uploading original data. The training samples on a particular device usually follow the mobile user's preferences, so the distribution characteristics of multiple training data are unbalanced (or non-independent and identically distributed). Since federated learning has significant advantages in the model training stage compared with traditional machine learning methods, this article will focus on analyzing and researching the main characteristics of this stage. The training phase of the complete federated learning model mainly includes the following steps[6, 7].

1. Initialization: All users have obtained a pre-allocated neural network model in their devices. They can voluntarily join the federated learning protocol and determine the same machine learning and model training goals.
2. Local training: In a certain round of specific communication process, the federated learning participants first download the global model parameters from the central server, and then use their private training samples to train the model to generate local model updates (ie model parameters), And send these updates to the central server.
3. Model average: The next round of global model can be obtained by aggregating all model updates obtained from different training samples and performing average calculation.

In the federated learning process, the above steps will be performed iteratively to achieve the purpose of optimizing the current global model. The entire iterative process will stop when the global model parameters meet the convergence conditions.

### 2.1 Classification of federated learning

According to the different data segmentation, three basic frameworks of federated learning are proposed, which are vertical federated learning, horizontal federated learning and federated transfer learning.[**2019federated**]. As shown in Table 1, this classification method is considered from the overlap of the user dimension and the feature dimension.

- **Horizontal federated learning** Horizontal federated learning framework is aimed at situations where each participant has the same sample but different feature spaces. It is mainly used in B2B scenarios, such as two different companies from the same region, or collaboration between a bank and an e-commerce company. Their customer set may be the same, but their business scope is completely different. Yang Q et al.[**2019federated**] design an implementation of this architecture, which requires a large number of encryption algorithms, and has high computational and communication costs.
- **Vertical federated learning** Vertical federated learning is mainly aimed at situations where the data of each participant has the same or similar feature space but different samples. For example, two hospitals from different regions have very small customer intersections but their businesses are similar. In addition, most B2C federated learning belongs to this classification, such as learning for smart phone users' input habits, or learning with pictures and voices. Each user has different data, but the data structure is similar. Mcmahan et al. post a vertical federated learning scheme for Android mobile terminal users[4]. Bonawitz et al. [8] propose a secure aggregation scheme for this solution to guarantee privacy in the model update process.
- **Federated transfer learning** Federated transfer learning is suitable for situations where the data sets of all parties are not only different in samples, but also in different feature spaces. This framework is only used as an extension of future federated learning, and it is difficult to achieve secure and reliable federated transfer learning with existing technologies.

Table 1: Classification of federated learning

| Classification | Applicable scene | Training method | Privacy protection |
|---|---|---|---|
| Horizontal FL | Same features but different samples | User dimension | Differential privacy |
| Vertical FL | Same samples but different features | Feature dimension | Encryption |
| Federated transfer learning | Different samples and different feature spaces | Transfer learning | |

## 3 Privacy threat to federated learning

Federated learning realizes a mechanism that the data is not shared but the model is shared between the participants. However, federated learning cannot guarantee individual privacy implied in the data. In order to build a federal model, participants still have to send model parameters or gradients, and such intermediate data are essentially a mapping of the original data according to certain rules. Many researches have proven that a malicious participant can infer sensitive information of other participants from the shared parameters or gradients[9, 10]. These attackers can act as clients, server or even model engineer. From the perspective of participants, the threat model can be subdivided into a honest-but-curious model and a malicious model. From the perspective of attack modes, attacks can be classified into inference attacks, model extraction attacks, and model inversion attacks.

### 3.1 Classification of attackers

#### 3.1.1 Honest-but-curious attacker

On the basis of complying with the security protocol, the honest-but-curious attacker will try to infer or extract the private information from the intermediate parameters generated during the federated

learning process. At present, most researches on the offense and defense of federated learning assume an honest-but-curious model. In real scenarios, due to the constraints of privacy regulations, most of the participants conform to this kind of semi-honest but curious attacker assumptions and will not attempt to carry out extreme malicious attacks. Honest but curious participants often act as clients. They can explore all information received from the server, but they are not able to modify the training process. Technologies like trusted execution environment (TEE) and security enclaves can limit the impact of such attackers and the visibility of information in some cases, which weakens the threat level.

#### 3.1.2 Malicious attacker

For malicious attacker, the model's security protocol setting will be more difficult. The malicious attacker will not abide by any agreement, in order to achieve the purpose of obtaining private data, any attack method can be adopted, such as destroying the fairness of the agreement, preventing the normal execution of the agreement, refusing to participate in the agreement, and maliciously replacing its own input without following the agreement. This will affect the design of the entire federated learning setup and training process.

The malicious participant can be a client, a server, an analyst or a model engineer. The malicious client can obtain all information received from the server in the training process and perform arbitrary modification attacks. The malicious server can inspect the model parameters such as gradient updates sent from the client, and tamper with the training process at will to launch an attack. A malicious analyst or model engineer can access the input and output of the federated learning system and perform various malicious attacks.

Comparing to honest-but-curious attacker, it will be more difficult to construct a secure federated learning protocol against malicious attacker. The calculation and communication costs will be greatly increased, and the design and implementation of the protocol will become more difficult or even unusable.

### 3.2 Classification of attacks

#### 3.2.1 Inference attack

As mentioned above, the federated learning mechanism requires all participants to upload model parameters result of training the model on the local dataset. In this case, if the server is untrusted and knowledge-rich, the participants' privacy cannot be guaranteed. Such untrusted server can obtain a lot of auxiliary information related to each client's sub-model (such as model structure, user identity and gradient), and has sufficient capacity to leak user privacy information. Inference attacks are more concerned with restoring specific information in the data (often with higher accuracy), for example, judging whether a specific data point or sub-dataset has been used for training is called membership inference attack; judging whether the data used by other participants contains an attribute is called attribute inference attack.

LT Phong et al.[11] design a server-side inference attack, which uses periodically exchange model parameters to calculate the private information of user training samples. However, this attack is limited to a pure training setting, and local updates must be generated by training with a single sample.

Melis et al.[12] point out that the user's unexpected information is likely to be inferred by malicious users. The leaked information is likely to be used to construct certain passive and active attacks, such as membership inference attacks. Different from server-side inference attacks, the only knowledge that the malicious user has is the global model parameters generated by aggregation. Therefore, how to obtain model updates of other users in each round of communication is a key issue in reconstructing data samples[13]. To this end, the attacker first obtains a snapshot of the global model parameters after the model average, and saves these snapshots locally. Then, by calculating the difference between subsequent snapshots, and further removing the newly added updates, the aggregation model updates can be obtained from other users. In this way, the attacker can use the assistance of the auxiliary data set to infer the data sample synthesized by all other participants[14].

#### 3.2.2 Model inversion attack

In model inversion attack, the attacker tries to get user privacy information from a simple-structured algorithm model by dynamic analysis or calculation of similarity between data. Fredrikson et al. [15] implements an attack to infer the patient's sensitive genotype based on the medication linear model with the basic information and prediction results of the patient.

In some complex algorithm models, using preset confidence to continuously modify the model can successfully obtain the user's true information under the condition of known sample labels[16], but this method is only suitable for small training samples.

#### 3.2.3 Model extraction attack

Model extraction attack is an attack proposed by Tramèr et al.[17] in 2016. The attacker try to infer the parameters or functions of the machine learning model by sending data cyclically and viewing the corresponding response results, thereby copying an attack method of a machine learning model with similar or even the same function. Early model extraction attack is simple and the application scope is limited. The research greatly improved the attack effect, and the scope of model extraction attack was further enlarged.

The research results of Shokri et al. show that the training model has a certain "memory" effect on the training data, and performing a reverse model attack on the basis of the extracted replacement model will further increase its harm.

Table 2: Summary of Security Threats and Defensive Measures in Federated Learning

| Attack Method | Description | Privacy protection |
|---|---|---|
| Inference Attack | Infer the participating members based on the auxiliary knowledge of the local training model; or infer features based on the global model parameters; | Secret sharing; Differential privacy |
| Model Extraction Attack | Model extraction attack refers to the attacker trying to infer the parameters or functions of the machine learning model and thus destroy the model confidentiality; | Secret sharing; Differential privacy |
| Model Inversion Attack | Model inversion attack refers to the attacker use preliminary information to inverse-analyze the model and obtain private data inside the model. Note: membership inference attack is aimed at a single piece of training data, while model inversion attack tends to obtain statistical information | Homomorphic encryption; Hybrid defence scheme |

## 4 Privacy protection technology

In response to the multiple privacy threats faced by federated learning, this section discusses some of the latest countermeasures against the above-mentioned attacks.

### 4.1 Differential privacy

In federated learning, all participants can obtain global parameters from the server. Malicious attackers analyze the sharing model, leading to the risk of privacy leakage of other participants. Although homomorphic encryption can protect the learning process by computing encrypted data, these tools require each data source to perform a large number of encryption operations and transmit a large amount of ciphertext, which makes them a burden on the entire system. For a federated learning system, it is very important to choose an algorithm that is relatively simple and does not cause

additional overhead on performance. Among these different privacy methods, differential privacy is widely used due to its powerful information theory guarantee, the simplicity of the algorithm and the relatively small system overhead. In the current federated learning, the differential privacy method is mainly used in the vertical learning framework in the B2C scenario.

Differential privacy technology can ensure that the shared model will not leak the information of the data provider, that is, it can defend against membership inference attacks to a certain extent. The research results show that when the number of data providers is large, the privacy differential technology can protect the privacy of data providers with a small performance loss [18]. Differential privacy also has certain limitations. It can only provide privacy protection for a single record. If there is a certain correlation between different records, the attacker can still perform membership inference attacks on algorithms that satisfy the differential privacy protection [19].

Set a random algorithm $F$, $R$ is a set of all possible outputs, if for any two adjacent data sets $D$ and $D'$, and any subset $S$ of $R$, there is

$$Pr[M(D) \in S]\alpha^{\varepsilon} Pr[F() \in S] + \delta$$

It is said that the algorithm $M$ satisfies -differential privacy. The bigger the $\epsilon$, the higher the accuracy of the model of algorithm $F$, but the lower the level of privacy protection. The comparison of differential privacy models is shown in Table 3

In addition, it is difficult to balance between privacy protection budget expenditure and federated learning efficiency. This is because a lower privacy protection expenditure budget may not be very effective for some large-scale attacks while adding more noise and disturbance will may severely compromise model performance.

### 4.1.1 Local differential privacy

In local differential privacy, the training of data and the privacy-protection method can all be realized on the client. Intuitively, local differential privacy mechanism is obviously superior to other solutions, because users have full control over the use and release of data, and do not need a central server, which has the greatest potential to achieve decentralized federated learning in a full sense. Abadi et al.[20] propose deep learning scheme that combines the DP mechanism with the SGD algorithm to achieve privacy protection. The method mainly uses noise to perturb the local gradient after each iteration. After the advent of federated learning, there have been many studies suggesting that the solution applies to federated learning barriers[21, 5].

Geyer et al.[22] propose a federated optimization algorithm for differential privacy protection of participants—differential privacy stochastic gradient descent algorithm. It proves the effectiveness of differential privacy to protect the data privacy of data holders. Meanwhile, it is proved that a large number of data holders will make the federated learning with differential privacy behave more stable and more accurate.

In order to improve the practicability of overly strict local differential privacy protection, the protection mechanism can be redefined [23], which not only ensures the security of sensitive information, but also relaxes the restrictions on data, and designs a new local optimal differential privacy mechanism. Solve statistical learning problems of all privacy levels, suitable for large-scale distributed model fitting and federated learning systems.

However, there are still many problems in the local differential privacy itself and its application in federated learning. The first is that the amount of samples it needs is extremely large. For example, Snap applied local differential privacy to the training of spam classifiers, and finally collected hundreds of millions of samples from users to achieve high accuracy. Google, Apple, and Microsoft [24] have deployed a large number of local differential privacy on user devices to collect data and conduct model training. Compared with the training of a noise-free model, the amount of data required is often up to 2 orders of magnitude. Secondly, with high-dimensional data, it will be more difficult for local differential privacy to achieve a balance between availability and privacy [25]. In addition, in the decentralized federated learning scenario, because there is no coordination of the central server, participants cannot know the sample information from other participants, so it is difficult to determine the size of the random noise they add, and the uneven distribution of noise will seriously reduce the performance of the model.

#### 4.1.2 Central differential privacy

When differential privacy methods were first proposed, most of them adopted a centralized form. A trusted third-party data collector aggregated the data and perturbed the data set to achieve differential privacy. Federated learning under the B2C architecture can also achieve this kind of perturbation on the central server. Geyer et al.[22] improves the performance of framework mentioned above[20], collects the updated gradient of users on the server side, and hides the contribution of each node by adding noise; They also prove that the centralized noise addition scheme can achieve user level differential privacy rather than just the data point level of local differential privacy, which means that it will not expose any participants of the training; finally, the experiment proves that the model performance of this method precedes local differential privacy. The work in [26] applies Similar differential privacy and federated learning settings to the field of natural language processing, establishes an associative word prediction model, and shows excellent accuracy in actual scenarios.

Centralized differential privacy also has shortcomings. It is limited to a trusted central server, but the server cannot be trusted in many scenarios. Therefore, distributed differential privacy can be used as a compromise between LDP and CDP, or hybrid differential privacy can be used to avoid some of the shortcomings of the two.

#### 4.1.3 Distributed differential privacy

Distributed differential privacy first aggregates the data sent by some users on several trusted intermediate nodes and implements privacy protection, and then transmits the encrypted or perturbed data to the server side to ensure that the server side can only get the aggregated results rather than the data. This solution requires the client to first complete the calculation and perform simple perturbations (such as local differential privacy with a higher privacy budget) or encryption, and send the results to a trusted intermediate node, and then use the TEE, SMC, secure aggregation[8] or secure shuffling[27] and other methods to achieve further privacy protection in the intermediate nodes, and finally send the results to the server.

In order to prevent intermediate parameters and participants from revealing to the central server, secure aggregation can be applied. Local differential privacy can be combined to prevent the server from getting additional information from the aggregated result, we can use local differential privacy. Goryczka et al. [28] proposed such a distributed differential privacy framework. It allows clients to add a small amount of noise before submitting data to trusted intermediate nodes. The intermediate node will use a secure aggregation protocol to submit these values to the server after a second perturbation, and the server can successfully calculate usable results based on these data.

Secure shuffling is another method except for secure aggregation. Bittau et al.[27] proposed a secure shuffling framework Encode-Shuffle-Analyze (ESA) in 2017. By adding an additional step of anonymous shuffling, users only have to add a little noise locally to achieve a high level of privacy protection. Erlingsson et al. [29] further improved this framework and considered the combination with federated learning.

Distributed differential privacy solutions have the advantages of both local and central differential privacy. They do not require a server with a very high level of trust, nor do they need to add too much noise locally. In contrast, distributed differential privacy generally requires high communication cost.

#### 4.1.4 Hybrid differential privacy

Avent et al. [30] propose the hybrid differential privacy scheme, which classifies users based on the different trust relationships between clients and servers. For example, the user who least trusts the server can apply the local differential privacy with the lowest privacy budget, while the user who trusts the server the most can even send the original parameters directly; the server will also process the data to varying degrees according to the user's trust relationship.

The problem with this scheme is that additional communication cost and preprocessing cost are required to divide the trust relationship.

Table 3: Comparison of differential privacy models

| Classification | Terminal performing perturbation | Advantages | Disadvantages |
|---|---|---|---|
| Local differential privacy | Clients | users have full control over the use and release of data; No need for any trust relationship | Privacy and usability are difficult to balance; Model performance is affected; |
| Central differential privacy | Server | High accuracy | A highly trusted central server is needed; |
| Distributed differential privacy (secure aggregation) | Trusted node between client and server | Low perturbation to achieve high privacy, no need for a trusted central server | The server can inspect each round's aggregated report (information may be leaked); Not valid for sparse vectors; A semi-honest server is assumed; |
| Distributed differential privacy (secure shuffling) | Trusted node between client and server | Low perturbation to achieve high privacy, no need for a trusted central server | Need a credible intermediary;The differential privacy guarantee of the shuffle model is reduced in proportion to the number of hostile participants of the calculation; |

### 4.2 Encryption method

Encryption is the most basic method in security. The plaintext information is encoded through an encryption algorithm to become a ciphertext that only a specific person can decode within a valid time. Considering the huge computational overhead of encryption algorithms, there have been only a few attempts to apply encryption for machine learning. However, in federated learning, the hidden cost of data privacy and security is much higher than the computational cost, making encryption algorithms get more attention.

#### 4.2.1 Secret sharing

Secret sharing[31] refers to dividing the data originally to be transmitted into multiple parts, and then sending them to each participant in turn. However, the original data cannot be restored by only one or a small part of the participants, and the original data can only be restored when a larger part or all of the participants put their respective data together.

The secret sharing mechanism is suitable for solving the situation of sensitive information leakage when a malicious server participates in the training of federated learning. As the aggregator of the shared model, the server can easily obtain the model parameter information of each data provider, which poses a huge threat to the privacy of the data provider. This can be prevented by secret sharing mechanism.

$(n, t)$ Secret sharing is a commonly used means of sharing secret information where $n$ is the number of secrets divided (after the secret is divided into $n$ parts, they are kept by $n$ participants, and each participant keeps 1 copy), $t$ is the number of users participating in the recovery of secrets. The algorithm divides the secret information $s$ into $n$, which are kept by $n$ different participants. Any $t$ participant cooperation can recover the secret information $s$, and less than $t$ participants are not able to recover the secret information $s$. By means of secret sharing, the ownership of the secret is distributed to each participant, rather than a specific manager or participant, to avoid the leakage of the secret due to a single manager or participant being attacked, thereby ensuring the privacy and information. According to actual needs, threshold secret sharing can be applied to determine at least how many participants are needed to recover data.

Bonawitz et al. [32] designed a practical security aggregation scheme based on Shamir secret sharing. In face of the threat of an honest-but-curious server, this scheme can guarantee the security of the intermediate model parameters and thus protect participants' privacy. Meanwhile, considering the features of federated learning, the scheme controls the protocol complexity, thereby reduce the cost of computation and communication. However, the scheme is not effective for collusion attack.

#### 4.2.2 Homomorphic encryption

Homomorphic encryption [33] is a special type of encryption function based on the homomorphism principle. It focuses on the security of data processing. It allows the encrypted data to be processed directly without knowing any information about the decryption function. In other words, other people can process the encrypted data, but they cannot know any original data information during the process. At the same time, the calculation result based on homomorphic encryption is consistent with the calculation result directly on the unencrypted data.

Homomorphic encryption defines $x$ and $y$ as elements in the plain text space $M$, $o$ is the operation on $M$, and $Ek()$ is the encryption function with key space $K$. If there is an effective algorithm F, such that:

$$F(Ek(x), Ek(y)) = Ek(xoy)$$

It is said that the encryption function Ek(·) is homomorphic to the operation D. In most applications, the homomorphic encryption scheme needs to support two basic and typical operations, namely, additive homomorphic operation and multiplication homomorphic operation. A function that satisfies both additive homomorphism and multiplicative homomorphism is called a fully homomorphic. The data owner encrypts the data before transmitting the data. The cloud server calculates as usual after receiving the data, but only on the ciphertext. After the result is obtained, the ciphertext of the result is returned to the data holder, and the data holder can obtain the final result after decryption. The use of encrypted data transmission can effectively defense against malicious(or semi-honest) server. Homomorphic encryption has strong privacy protection capabilities, but its efficiency is difficult to improve because data encryption process brings more computation overhead and slower computation speed[33].

Using homomorphic encryption, the server aggregates the cipher text parameters and cannot obtain the user's privacy parameters. Ho Q R et al. [34] use homomorphic encryption to implement a privacy protection protocol of horizontal linear regression. Hardy S et al. [35] use entity resolution and homomorphic encryption to conduct privacy-protected federated learning on vertically distributed data.

Aono Yoshinori et al.[36] propose a new deep learning system based on honest-but-curious cloud servers. It uses a homomorphic encryption scheme to achieve the aggregation of gradients on honest and curious servers, and ensures that the system reaches the joint data set with all participants. The trained corresponding deep learning system has the same accuracy.

### 4.3 Encrypted computation environment

#### 4.3.1 Secure multi-party computation

Secure multi-party computation(SMC) proposes a privacy protection scheme for the collaborative computation problem between a group of untrusted parties. Secure multi-party computation ensures that the independence of input, the correctness of calculation, and decentralization are not affected. At the same time, each input value is not disclosed to other members participating in the computation.

In other words, SMC aims at the problem of how to safely calculate an agreed function without a trusted third party. Each participant is restricted to not get any input information from other entities except for the calculation result. Throughout the execution of the calculation agreement, the user always has control over the personal data, and only the calculation logic is public. These features make secure multi-party computation a preferred technology in a federated learning environment.

Aono et al. [36] point out that in collaborative learning, even if users upload few local parameters, their private data information is likely to be secretly stolen by untrusted servers. Secure multi-party computation ensures that the independence of input, the correctness of computation, and the decentralization of features are not affected, and at the same time, each input value is not leaked to other members participating in the computation. In addition to the computation results, each participant cannot get any input information from other entities. During the execution of the computation agreement, the user always has control over the personal data, and only the computation logic is public.

Mohassel et al. [37] has proposed an SMC protocol that supports two participants to perform machine learning under the semi-honest assumption. These works have laid the foundation for the future application of SMC in federated learning.

#### 4.3.2 Trusted computing environment

Trusted computing (TC) is a study launched by the Trusted Computing Group (TCG), hoping to enhance the security of various computing platforms through dedicated security chips (TPM/TCM)[38]. Compared with trusted computing, TEE is more conducive to the use of portable devices. Because the security in this environment can be verified, part of the federated learning process can be placed in a trusted computing environment.

Corresponding to TEE, the rich execution environment (REE) technology of traditional mobile devices has openness, scalability and versatility. However, in the data privacy and security scenario, an isolated and trusted environment is required to process keys and private data. Here TEE and REE are isolated from each other and can only communicate with each other through specific ports. The characteristics of the hardware mechanism protection of the trusted computing environment fully guarantee the privacy and security of data.

Trusted computing environment plays a very good role in data protection for the federated learning system, and provides protection for remote secure computing for privacy and other sensitive data.

## 5 Conclusions and future work

There are currently some review papers and results on the research and development direction of federated learning. Bonawitz et al. [8] summarize the three major problems in current federated learning, mainly the communication bandwidth problem, the convergence problem of model training, and the coordination problem with cloud service providers. Peter Kairouz et al.[21] present an systematical collection of recent advances, open problems and challenges of federated learning. In addition, the existing solutions for privacy protection in federated learning are not convincing enough, and they still have their own shortcomings.

This section summarizes the answers for the research questions mentioned in section 1:

- *Q1: What is federated learning? How is its development? What is its current concern?*

Section 1 and 2 give a detailed description of the development of federated learning. The basic framework, categorization and current shortcomings are also analyzed and summarized.

- *Q2: What is the privacy threat to federated learning? How to attack?*

The attackers can be classified as malicious attackers and honest-but-curious attackers. According to different authorities, they can perform model inversion attacks, model extraction attacks and inference attacks. The detailed introduction is included in Section 3.

- *Q3: How do defend against the above privacy threats? What are these privacy protection techniques' pros and cons? How is the application of these techniques to FL?*

In response to the privacy threats, the protection technologies including data perturbation, encryption and secure computation environment are introduced in detail in Section 4. The privacy protection scheme of over 20 papers is summarized, their advantages and limitations are analyzed.

- *Q4: What is the future research direction of privacy protection in FL?*

**The balance of data privacy and availability** As mentioned in Section 4, in the scenario of federated learning, users need to independently add noise to achieve disturbance, which makes it more difficult to achieve a balance between data privacy and usability. However, existing studies have proved that as long as enough training participants are introduced, privacy and usability can be considered. For example, Geyer et al.[22] find through experiments that a federated learning framework with differential privacy protection in a scenario with 10,000 participants, After training, the accuracy of the model can be exactly the same as that of federated learning without differential privacy in

the scenario of 100 participants. Other studies from major companies[20, 29] have also proved that the introduction of a reliable differential privacy mechanism in federated learning means that the demand for participants will increase by 2 to 3 orders of magnitude, and it will Greatly increase communication and computing costs. With the help of centralized differential privacy or distributed differential privacy, this will be slightly alleviated.

Future research needs to continue the balance between privacy, availability and data volume, such as how to further compress the number of participants and training samples while ensuring data privacy and availability.

**Lack of trusted servers or trusted nodes** Centralized differential privacy must be implemented by a trusted server; and distributed differential privacy, even if secure shuffling or secure aggregation strategies are introduced, it still requires a trusted intermediate node to achieve it. It is often difficult to meet these conditions in real situations. Obviously, a trusted execution environment (TEE) can be used as a solution (as described in 4.3.2). , TEE is generally deployed between the underlying software and hardware, which is equivalent to a mini virtual machine supported by hardware. It calls exclusive software and hardware resources under an open and transparent protocol, and can and can only interact with a fixed network interface, which can provide Relatively high level of privacy and trust.

However, introducing TEE into federated learning will bring other problems. For example, the current TEE architecture can only access CPU resources, but cannot access GPU resources which is commonly used in machine learning. In addition, TEE is also vulnerable to side channel attacks, which will lead to the failure of TEE. Future work can further focus on the optimization of TEE and the study of the combination of TEE and federated learning.

**Huge communication and computation overhead** Considering that in the federated learning scenario, most users need to complete computing and communication tasks on mobile terminals. At this time, the evaluation criteria for computing and communication costs are completely different from traditional machine learning. In order to avoid encryption algorithms as much as possible, researchers have proposed secure shuffling and secure aggregation methods, but this has brought great communication overhead. The work in [39] proves that the sparseness of the model update in bits can effectively reduce the communication cost, but this will also increase the calculation cost, and it is uncertain whether the reduction in communication cost brought about by this approach is cost-effective.

**Establish a unified privacy disclosure metric** The existing federated learning frameworks such as TensorFlow federated[40] does not consider the integration with SMC or DP applications. Therefore, the development of a federated learning privacy protection enhancement framework is a research direction that needs to be solved urgently.

## References


[1] N. Papernot et al. "Towards the Science of Security and Privacy in Machine Learning". In: *arXiv* (2016).

[2] Q. Yang et al. "Federated Machine Learning: Concept and Applications". In: *ACM Transactions on Intelligent Systems and Technology* 10.2 (2019), pp. 1 $\sim$ 19.

[3] 2018 Wikipedia. `https://en.wikipedia.org/wiki/Facebook-Cambridge_Analytica_data_scandal`.

[4] Brendan McMahan et al. "Communication-efficient learning of deep networks from decentralized data". In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 1273 $\sim$ 1282.

[5] Tian Li et al. "Federated learning: Challenges, methods, and future directions". In: *IEEE Signal Processing Magazine* 37.3 (2020), pp. 50 $\sim$ 60.

[6] Hyesung Kim et al. "Blockchained on-device federated learning". In: *IEEE Communications Letters* 24.6 (2019), pp. 1279 $\sim$ 1283.

[7] Sumudu Samarakoon et al. "Distributed federated learning for ultra-reliable low-latency vehicular communications". In: *IEEE Transactions on Communications* 68.2 (2019), pp. 1146 $\sim$ 1159.

[8] Keith Bonawitz et al. "Towards federated learning at scale: System design". In: *arXiv preprint arXiv:1902.01046* (2019).

[9] Ligeng Zhu and Song Han. "Deep leakage from gradients". In: *Federated learning*. Springer, 2020, pp. 17 $\sim$ 31.

[10] Eugene Bagdasaryan et al. "How to backdoor federated learning". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 2938 $\sim$ 2948.

[11] Le Trieu Phong et al. "Privacy-preserving deep learning: Revisited and enhanced". In: *International Conference on Applications and Techniques in Information Security*. Springer. 2017, pp. 100 $\sim$ 110.

[12] Luca Melis et al. "Exploiting unintended feature leakage in collaborative learning". In: *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2019, pp. 691 $\sim$ 706.

[13] Milad Nasr, Reza Shokri, and Amir Houmansadr. "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning". In: *2019 IEEE symposium on security and privacy (SP)*. IEEE. 2019, pp. 739 $\sim$ 753.

[14] Stacey Truex et al. "Demystifying membership inference attacks in machine learning as a service". In: *IEEE Transactions on Services Computing* (2019).

[15] Matthew Fredrikson et al. "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing". In: *23rd {USENIX} Security Symposium ({USENIX} Security 14)*. 2014, pp. 17 $\sim$ 32.

[16] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. "Model inversion attacks that exploit confidence information and basic countermeasures". In: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 2015, pp. 1322 $\sim$ 1333.

[17] Florian Tramèr et al. "Stealing machine learning models via prediction apis". In: *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 2016, pp. 601 $\sim$ 618.

[18] Bargav Jayaraman and David Evans. "Evaluating differentially private machine learning in practice". In: *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 2019, pp. 1895 $\sim$ 1912.

[19] Daniel Kifer and Ashwin Machanavajjhala. "No free lunch in data privacy". In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. 2011, pp. 193 $\sim$ 204.

[20] Martin Abadi et al. "Deep learning with differential privacy". In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 308 $\sim$ 318.

[21] Peter Kairouz et al. "Advances and open problems in federated learning". In: *arXiv preprint arXiv:1912.04977* (2019).

[22] Robin C Geyer, Tassilo Klein, and Moin Nabi. "Differentially private federated learning: A client level perspective". In: *arXiv preprint arXiv:1712.07557* (2017).

[23] Abhishek Bhowmick et al. "Protection against reconstruction and its applications in private federated learning". In: *arXiv preprint arXiv:1812.00984* (2018).

[24] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. "Collecting telemetry data privately". In: *arXiv preprint arXiv:1712.01524* (2017).

[25] Raef Bassily et al. "Practical locally private heavy hitters". In: *arXiv preprint arXiv:1707.04982* (2017).

[26] H Brendan McMahan et al. "Learning differentially private recurrent language models". In: *arXiv preprint arXiv:1710.06963* (2017).

[27] Andrea Bittau et al. "Prochlo: Strong privacy for analytics in the crowd". In: *Proceedings of the 26th Symposium on Operating Systems Principles*. 2017, pp. 441 $\sim$ 459.

[28] Slawomir Goryczka and Li Xiong. "A comprehensive comparison of multiparty secure additions with differential privacy". In: *IEEE transactions on dependable and secure computing* 14.5 (2015), pp. 463 $\sim$ 477.

[29] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. "Rappor: Randomized aggregatable privacy-preserving ordinal response". In: *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 2014, pp. 1054 $\sim$ 1067.

[30] Brendan Avent et al. "{BLENDER}: Enabling local search with a hybrid differential privacy model". In: *26th {USENIX} Security Symposium ({USENIX} Security 17)*. 2017, pp. 747 $\sim$ 764.

[31] XiuXia Tian et al. “Privacy preserving query processing on secret share based data storage”. In: *International Conference on Database Systems for advanced Applications*. Springer. 2011, pp. 108 $\sim$ 122.

[32] Keith Bonawitz et al. “Practical secure aggregation for privacy-preserving machine learning”. In: *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 1175 $\sim$ 1191.

[33] Shai Halevi and Victor Shoup. “Design and implementation of a homomorphic-encryption library”. In: *IBM Research (Manuscript)* 6.12-15 (2013), pp. 8 $\sim$ 36.

[34] Qirong Ho et al. “More effective distributed ml via a stale synchronous parallel parameter server”. In: *Advances in neural information processing systems*. 2013, pp. 1223 $\sim$ 1231.

[35] Stephen Hardy et al. “Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption”. In: *arXiv preprint arXiv:1711.10677* (2017).

[36] Yoshinori Aono et al. “Privacy-preserving deep learning via additively homomorphic encryption”. In: *IEEE Transactions on Information Forensics and Security* 13.5 (2017), pp. 1333 $\sim$ 1345.

[37] Payman Mohassel and Yupeng Zhang. “Secureml: A system for scalable privacy-preserving machine learning”. In: *2017 IEEE symposium on security and privacy (SP)*. IEEE. 2017, pp. 19 $\sim$ 38.

[38] M Sadegh Riazi et al. “Chameleon: A hybrid secure computation framework for machine learning applications”. In: *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. 2018, pp. 707 $\sim$ 721.

[39] Jakub Konečnỳ et al. “Federated learning: Strategies for improving communication efficiency”. In: *arXiv preprint arXiv:1610.05492* (2016).

[40] “Secure, privacy-preserving and federated machine learning in medical imaging”. In: *Nature Machine Intelligence* 2.6 (2020), pp. 305 $\sim$ 311.