Digital Privacy and Security

Shui Yu · Lei Cui

Security and Privacy in Federated Learning

Springer

Digital Privacy and Security

Digitalization is a big trend at the moment in the whole history of human beings. However, the movement introduces unprecedented challenges in security and privacy.

The book series on Digital Privacy and Security aims to develop and disseminate understandings of innovations, paradigms, techniques, and technologies in the contexts of digital world related research and studies. It covers security, privacy, availability, and dependability issues for digital systems and applications. It welcomes emerging technologies, such as security and privacy in artificial intelligence, digital twin, blockchain, metaverse, semantic communications, and so on.

The series serves as an essential reference source for security and privacy in the digital space. It publishes thorough and cohesive overviews on state-of-the-art topics in cyber security and privacy, as well as sophisticated techniques, original research presentations and in-depth case studies in the domain. The series also provides a single point of coverage of advanced and timely emerging topics and a forum for core concepts that may not have reached a level of maturity to warrant a comprehensive textbook. The intended audience includes students, researchers, professionals, and industrial practitioners.

Shui Yu • Lei Cui

Security and Privacy in Federated Learning

Springer

Shui Yu
School of Computer Science
University of Technology Sydney
Ultimo, NSW, Australia

Lei Cui
Shandong Computer Science Center
(National Supercomputer Center in Jinan)
Jinan, Shandong, China

ISSN 2731-992X ISSN 2731-9938 (electronic)
Digital Privacy and Security
ISBN 978-981-19-8691-8 ISBN 978-981-19-8692-5 (eBook)
https://doi.org/10.1007/978-981-19-8692-5

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Preface

In the recent two decades, we have witnessed the dramatic development of artificial intelligence (AI in short), not only in artificial intelligence itself but also its applications in various sectors of human society. The appearance of deep learning pushed AI into another spring after a long winter. Nowadays, there are many successful stories of significant progress in various scientific fields with the help of AI, for example, the application of AI in biology, chemistry, law, and social science, to name a few.

However, AI suffers a fundamental challenge, explainability: the conclusions obtained from machine learning based on big datasets are generally very useful, but we sometimes do not know why. People even treated AI as alchemy: the outputs of the same AI algorithm with the same inputs may vary from time to time, and AI practitioners sometimes even do not know what they will have before the deployment of AI.

Security and privacy protection in AI is far behind the fast development of AI and its applications. As we can see, AI is gradually permeating into our daily lives, and security and privacy are big challenges as AI needs sufficient information for their judgment and recommendation. As a result, we can see that AI and privacy protection are natural enemies. We can predict that majority (if not all) known attacks will be applied in AI applications, we need solutions. Particularly, digital privacy is an unprecedented challenge, and we face numerous new problems, for example, measurement of privacy, privacy modelling, privacy tools, and privacy pricing, to list a few.

Federated learning is a big step for privacy protection in machine learning; however, it is not perfect. The federated learning framework allows learning participants to keep their data locally and download the training model from a central server or servers to execute a local training. The updates will be uploaded to the server(s) for a further aggregation for the next round of training until an acceptable global model is reached. Despite the advancement of its computing model, federated learning still faces many security and privacy challenges, which have attracted a lot of attention from academia and industry.

We classify the security and privacy research in federated learning into two categories: problem based and tool based. In our understanding, problem-oriented research focuses on problems and proposes solutions. At the same time, tools-oriented research usually depends on tools to address problems.

In order to serve our readers in a flexible way, we organize the book into three parts: problem oriented, tool oriented, and the promising future directions: Chaps. 2, 3, and 4 are problem oriented, where we present the hot problems and their related solutions in federated learning. In Chaps. 5–8, we shift to present the solutions using the available tools, differential privacy, and cryptography. We have to note that we only introduce the concept and very basics of the tools (especially for cryptography) for readers to understand the related security and privacy research in the federated learning field. If readers expect to continue the research using these tools, we suggest them to read the classic textbooks of the tools following the related references, respectively. In Chap. 9, we boldly present some of our understanding on the security and privacy landscape in the near future for the possible reference of our readers. We know it is hard to predict the future, but we cannot help ourselves to share our thinking with readers, hoping it would be helpful for them. Each chapter or part of the book is relatively independent, and readers can choose any order to use the book according to their needs.

Federated learning is a new field, security and privacy in federated learning is a newer subfield. Our purpose is to introduce interested readers to this promising research area. The book could be used as an introductory subject for senior undergraduate students, it can also be used for postgraduate students or researchers as a reference for their research work. We hope this book brings our readers an overview of the field, and paves a starting ground for them to further explore the uncharted land of the domain in both theoretical and application perspectives.

Due to the time and knowledge limitation, we believe there are many possible problems and errors. We humbly ask you to kindly point them to us for correction in a possible future version.

We sincerely hope you will enjoy reading the book, and let us work together to explore the promising uncharted land.

Sydney, NSW, Australia, Shui Yu
Jinan, China, Lei Cui
October 2022

Acknowledgments

This book is a product of our research journey on privacy and security, especially in the booming field of federated learning. We sincerely appreciate the colleagues and friends who offered us help, comments, suggestions, and guidance on our way to this point. There are too many of them, it would be a long list if we list their names here. We want to say thank you.

We would like to thank the members of our research groups at the University of Technology Sydney. PhD students and academic visitors are key players in our research, and they helped establish a solid foundation for the content of the book through numerous meetings and discussions. Some of them contributed significantly in the drafts of some chapters: Mr. Feng Wu for Chap. 2, Mr. Zhiyi Tian for Chap. 3, Mr. Chenhan Zhang for Chap. 4, and Miss Kaiyue Zhang for Chap. 5. Mr. Andrew Vo and Mr. Weiqi Wang helped the proofreading of Chaps. 1 and 5.

We are also grateful to the professional staff of Springer, especially Dr. Nick Zhu, for their patience and continuous professional support.

The last but not least, we would like to thank our families for their persistent and selfless support.

Contents

Chapter 1
Introduction to Federated Learning

In this chapter, we plan to present some basic knowledge of federated learning and offer an overview of the security and privacy attacks and counter methods for beginners. Some sections could be skipped for readers who have understood the concepts already.

Some basic information about the book is also presented at the end of the chapter for readers' reference if they want to find their interested content directly.

1.1 Federated Learning Paradigm

In general, federated learning (FL) is a branch of deep learning, which is a powerful tool to address various complex problems in the past decades. Google proposed federated learning as a variation of deep learning in order to address the privacy concern from data owners.

Let us start from deep learning and progress to federated learning and then introduce the main theme of this book: security and privacy attacks and counterattacks in federated learning.

1.1.1 Deep Learning

Traditional machine learning (ML) algorithms rely on experts to design feature engineering when building models. Until the emergence of deep learning (DL) [1, 2], people can automatically discover and learn corresponding features from data through deep neural network (DNN), which makes the model training streamlined. The concept of DL originates from artificial neural network technology, where the word "deep" means that it adopts the multilayer structure of neural networks.

S. Yu, L. Cui, *Security and Privacy in Federated Learning*, Digital Privacy and Security, https://doi.org/10.1007/978-981-19-8692-5_1

DL trains network models through a large amount of data, so that the models can recognize and predict. With the increase of training data, the accuracy of neural networks will be continuously improved. Deeper layers of neural network can fully extract features and make the models perform better. In recent years, with the development of large datasets and powerful hardware, DL has gradually become the main method for processing complex high-dimensional data, such as images, texts, and sound signals [3, 4].

Specifically, DL refers to the technique of solving problems using deep neural networks, where the network is constructed from a large number of interconnected neurons. The neural network is the most critical part of DL. Typical neural networks include multilayer perceptron (MLP), convolutional neural network (CNN), recurrent neural network (RNN), and so on.

Taking the most commonly used MLP as an example, MLP belongs to multilayer feedforward neural network, which is composed of a single layer perceptron superimposed and uses the backpropagation algorithm. Its network structure consists of at least three parts: input layer, hidden layer, and output layer. The neurons between each layer of MLP are completely connected, and each neuron uses the activation function to process the features of the previous layer nonlinearly. MLP has more hidden layers than single-layer perceptrons; as a result, it has more neurons and can fit more complex linear functions. In addition, by increasing the activation function and increasing the nonlinearity of the model, the multilayer perceptron can better simulate nonlinear functions.

CNN is the most widely used and representative network structure in DL. It can extract and learn features from a large amount of data effectively. It is now widely used in speech recognition, image recognition, image segmentation, and so on. The main modules of CNN include the input layer, convolutional layer, activation function, pooling layer, fully connected layer, and output layer. In addition, other important components, such as the batch normalization (BN) layer and dropout layer, are introduced to improve the generalization ability of the model or prevent model overfitting.

Take the image classification task as an example, as shown in Fig.1.1. The model training process of CNN is to input images first, train them through several layers

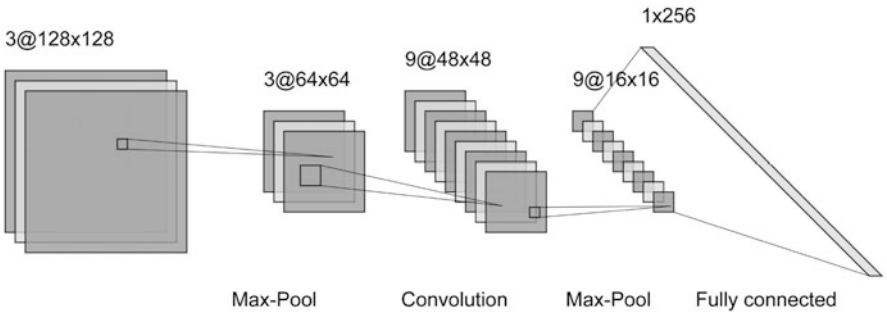

Fig. 1.1 A simple CNN framework

of the convolution layer and pooling layer, and then perform classification fitting through the fully connected layer.

The convolutional layer extracts the features of the input data through a large number of repeated convolutional operations, and the extracted feature information is transmitted to the next layer for further information extraction. Among them, the shallow neurons near the input layer mainly extract low-level features, while the deep neurons near the output layer are responsible for extracting high-level features. Compared with a fully connected layer, the convolutional layer uses parameter sharing and sparse connectivity to reduce the connections between neurons, thus greatly reduce the number of network parameters.

The pooling layer is usually sandwiched in the middle of the continuous convolution layer. It filters features through predefined pooling functions, which highlight key features while reducing the number of parameters in the network. In practice, the pooling layer can choose different pooling functions according to specific requirements. The most typical pooling functions are maximum pooling and average pooling.

Fully connected layers are typically used in the last layers of a network, where each neuron is connected to all neurons in the previous layer. The function of the fully connected layer is to integrate the features extracted by the convolution layer and pooling layer to obtain a feature vector representing the classification result.

Batch normalization is the most commonly used normalization method, which is to normalize the output characteristics of the middle layers of neural networks, so that the output of the middle layers can be more stable, and can also be used to alleviate the gradient disappearance, explosion, and other problems.

In the dropout layer, the model randomly drops neurons with a preset probability, and these discarded neurons do not participate in forward propagation and backpropagation, which limits model training and learning, thereby reducing model overfitting.

The goal of DL model training is to make the predefined loss function reach the minimum value, and the training process can be regarded as searching for the model parameters that minimize the loss function. DL calculates the loss through forward propagation on the dataset and optimizes the whole system through backpropagation [5]. In general, the forward propagation of model training is to pass the training samples through the input layer to the last layer and obtain the output of the network. Backpropagation is the process of calculating the current loss through the loss function and then correcting the model parameters according to the loss. Since backpropagation is based on gradient descent (GD in short), according to the chain rule, the gradient of the previous layer needs the gradient of the latter layers to be calculated. Therefore, the gradient of each layer needs to be calculated from back to front.

After calculating the gradient by backpropagation, the model parameter ω can be adjusted according to the gradient of each layer and the preset learning rate η; the whole process is shown as follows.

$$\omega \leftarrow \omega - \eta \nabla f(\omega) \tag{1.1}$$

The training process of a neural network model is to repeat the process of forward propagation and backpropagation until the training reaches a fixed number of rounds or model convergence.

Traditional machine learning training relies on a single central server, which has many drawbacks. First of all, the centralized training scheme can only utilize limited computing resources, and it takes a long time to train a well-performing network under complex scenarios. Secondly, large-scale network models often have a large number of network parameters, which require massive data for training, and it is difficult for a single machine to provide corresponding computing and storage resources.

Therefore, the distributed machine learning [6] method is developed, which applies multiple machines to train a model. Distributed machine learning refers to the collaboration of multiple computing nodes (workers) to train a global ML/DL model through the coordination if a master node.

In general, distributed machine learning architectures can be classified into data parallelization and model parallelization.

The method of data parallelization is to divide the large training data into multiple subsets and allocate them to different machines for storage. Each worker trains the model based on the locally stored data, and the model parameters trained by each worker are aggregated by a central parameter server.

The data parallelization method enables a large amount of data to be processed on different machines at the same time, which solves the problem of data storage and improves the efficiency of model training.

The method of model parallelization is to split the model into several small sub-parts, and the data stream is processed by the sub-models on each machine in turn during the training process, and then the whole network model can be trained. Model parallelization enables large network models to be distributed and stored on different machines. However, compared with data parallelization, the dependency relationship between various sub-models in the framework of model parallelization is very strong, and the communication requirements are higher.

1.1.2 Federated Learning

Artificial intelligence has gradually entered into many sectors of human society. The combination of modern deep learning algorithms and massive data makes the deep learning technology a promising tool to solve complicated real-world problems. However, conventional centralized approaches are associated with many problems [7].

Firstly, in real life, except for a few giant companies, most of the enterprises only have insufficient data with limited qualities, which are not enough to support the deployment of data-hungry AI services.

Secondly, the access to participants' raw data suffers privacy problems [8]. In particular, for enterprises, data from commercial companies are often of great

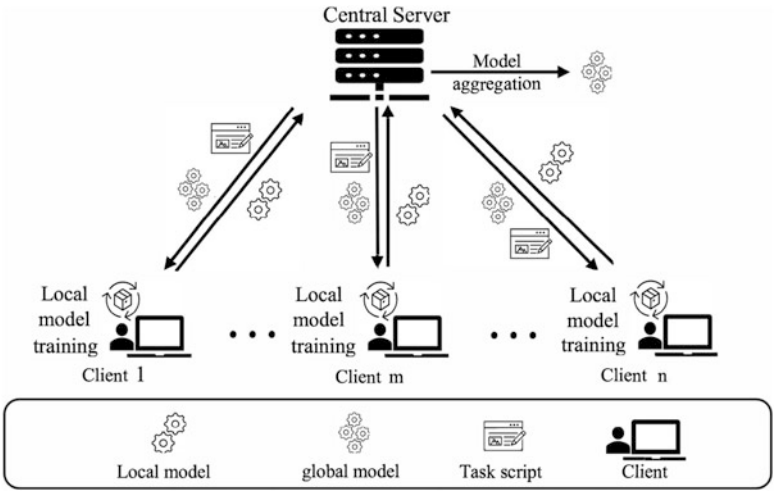

Fig. 1.2 A typical federated learning framework

potential value. Different companies or even different departments within the same company are usually reluctant to share data. For individuals, most of their data contain private information, such as travel trajectory, health status, financial data, and so on.

Thirdly, centralized training approaches involve high computational demands and unacceptable delay [9]. On the other hand, the limited resources of devices and unstable network environment also pose challenges to distributed machine learning.

To overcome these challenges, federated learning, a distributed learning framework without exposing training data, has drawn increasingly attention nowadays [10]. It can improve the effectiveness of the model through model aggregation of multiple clients on the basis of ensuring security and privacy.

In Fig. 1.2, we illustrate a typical federated learning process that has the following four steps.

1. A shared model is sent from the server to some selected clients.
2. The involved clients further train the model with their local data and submit the updated model or parameters to the server, respectively.
3. The updates are collected by the server for aggregation, and the server will broadcast the aggregated model to all clients.
4. The previous three steps will be repeated until we converge to a global model or reach a given number of rounds decided beforehand.

The key innovation of federated learning is that the server only requires parameters instead of collecting raw data from the clients. The sensitive information is well protected as the data is held in each client's own equipment, respectively. With this distinguished capability, federated learning has attracted both industry and academia to leverage it exclusively, providing privacy protection in different

modern applications, such as mobile keyboard prediction, personalized medicine, traffic forecasting, anomaly detection, and so on [11–14].

1.1.3 Categories of Federated Learning

Based on different data distributions among multiple participants, federated learning can be generally classified into three categories: horizontal federated learning, vertical federated learning, and federated transfer learning [7].

In the scenario of horizontal federated learning, the data distribution provided by the participants is similar, but data providers do not overlap. In the training process, the model of each machine is identical and complete and can be independently predicted when making predictions. Consequently, this process can be regarded as a distributed training based on samples. Since each user's raw data are trained locally, and only local parameters are shared and uploaded, the user's privacy is also protected but with some model loss.

The vertical federated learning scenario is opposite to the horizontal federated learning. The user settings are all the same, but different datasets have different kinds of data from these users. For example, airlines and hotels have different data from the same users, i.e., flight records and accommodation records. As a result, vertical federated learning requires sample alignment and model encryption. During the training process, vertical federated learning ensures that participants do not know the datasets and characteristics of the other parties. In this way, the global model can obtain data information of all participants, and there is no loss of the model.

The application scenario of federated transfer learning is strict, which has a limited number of identical users and very small datasets with the same features. A lot of recent work has narrowed down scenarios to very specific topics for discussion, such as wearable healthcare.

1.1.4 Challenges in Federated Learning

There are a large number of applications taking advantage of federated learning so far. For instance, some applications allow users to train models on their mobile phones without having to upload raw data. Nevertheless, federated learning techniques are still facing critical challenges [15]. The idea of distributed learning and keeping data locally makes it easier for malicious nodes to attack the federated learning framework. In addition, there are many other problems; heterogeneous feature of users and datasets also bring more processing difficulty. On top of the same challenges as general deep learning, federated learning also possesses its own unprecedented challenges. Moreover, the threats to the classical learning algorithms may even have a serious impact on federated learning frameworks.

Specifically, we list four main challenges of federated learning systems from our perspective below.

Security The challenges in security are generally the lack of access control of data and attacks from malicious nodes, which are usually launched by hackers on the system or model itself. One of the most common attacks on federated learning is poisoning attack [8]. Attacks like fake label attacks and backdoor attacks may lead to serious damage if the number of poisoned datasets is large.
Privacy Federated learning protects privacy by providing local model parameter information instead of clients' local raw data. However, simply maintaining the localization of data during the training process cannot provide sufficient privacy guarantee [16]. Analyzing gradient information can still offer private knowledge to third parties or central servers. For example, original images from clients can be recovered by obtaining the aggregation gradient returned by the central server. Furthermore, the models themselves can be used to launch inversion and inference attacks.
Communication Cost In federated learning systems, there can be millions of devices involved in the network, and each device may spend far less time training model locally than their network communication. As local models are required to be uploaded to the server periodically, the communication cost becomes much more heavy when there is a large number of participants. This issue can become a bottleneck due to the limited bandwidth of wireless networks [17]. In addition, how to reduce the uplink communication when the connection speeds are asymmetric is also a significant problem.
Heterogeneity Due to the variety of network status, storage, and processing capability of devices, the capabilities of computing and communication will be different. The presence of this heterogeneity exacerbates delayed mitigation and fault tolerance [18]. Moreover, the data itself is heterogeneous due to different data generations and collecting methods. For example, the data from different clients can easily be heterogeneous, and non-IID data will be more difficult to be processed, which increases the complexity of modeling and evaluation.

We focus on the security and privacy perspectives of federated learning in this book, and will not involve the third and the fourth challenges, which belong to networking and system research.

1.2 Security and Privacy in Federated Learning

1.2.1 Source of Vulnerabilities

Vulnerabilities represent weakness in a system that an attacker can exploit to gain unauthorized access. For better examining security and privacy issues in federated learning, we firstly categorize the sources of vulnerabilities as follows [8, 19].

Clients In federated learning, clients use their local data to train a local model and contribute parameters to the server as part of the decentralized training procedure. Because clients do not need to share their private data with the server, a compromised client can directly manipulate the training process of the local model to craft poisoning attacks. In addition, as clients can access the collaboratively trained model through the global model broadcasting from the server, a malicious client can repeatedly touch the global model and further implement inference attacks.
Server The server in federated learning is responsible for sharing initial model parameters, aggregating local model updates, and communicating the global model to clients. However, attacks (e.g., inference attacks) can be executed from the server utilizing local updates to recover the local data of clients. Therefore, vulnerabilities of the server should be continuously monitored, and the server should be robust against various attacks.
Aggregator The performance of federated learning is highly related to the aggregation process. Conventional aggregation algorithms, like federated averaging (FedAvg for short), are vulnerable to adversarial attacks. Secure aggregation methods are required to prevent the curious server from gaining visibility and enable to detect abnormal client updates.
Communication Federated learning requires multiple rounds of communication between participants and the federated server. Insecure communication channels is a serious threat [20]. For instance, model updates can be stolen or modified by man-in-the-middle attacks. Communication bottlenecks can increase the number of clients who drop out in federated learning systems.

1.2.2 Attacks in Federated Learning

According to available literature at the moment, we classify the existing security and privacy attacks in federated learning into three categories: inference attacks, poisoning attacks, and generative adversarial networks (GAN)-based attacks. Their descriptions are presented, respectively, as follows.

Inference Attacks Exchanging gradients in federated learning can cause serious privacy leakage. Massive amount of private information can be revealed by examining model updates [21]. Specifically, model inversion attacks can perform reverse reasoning on the neural network model to obtain information of the original dataset. Property inference attacks seek to infer statistical information of the training dataset. Membership inference attacks try to discover whether a certain data point is used to train the global model. Model inference attacks are used to obtain an alternative model with similar capabilities to the target model.
Poisoning Attacks The purpose of poisoning attack is to destroy the training of models or degrade the effectiveness of the expected trained model. The distributed

structure makes federated learning vulnerable to poisoning attacks: data poisoning during local data collection [22]; model poisoning during local model training [23].

Data poisoning attacks compromise the integrity of the training data to corrupt the global model. The technologies used in data poisoning are the same as the traditional deep learning model training. In targeted model poisoning, an adversary can modify the local model updates before sending it to the central server for aggregation, affecting the performance of the global model.

GAN-Based Attacks As a popular machine learning technique, GAN has been applied successfully in many applications, such as computer vision and nature language processing. However, GAN is also a fitting technique that can be used to attack federated learning [24]. The mimicry-synthesis nature of GAN makes it a promising technique for inference attack. An adversary can train a GAN to produce synthetic samples to augment the training dataset and further utilize the generated samples to train a shadow model. In addition, GAN enables to fabricate poisoning data samples and local model parameters to alter the inference of the learning model in a systematic way.

1.2.3 Defense Techniques in Federated Learning

Federated learning requires meaningful security and privacy guarantees. In this subsection, we briefly present different protection techniques for federated learning and identify potential challenges.

Differential Privacy (DP) DP was originally developed to prevent information leakage by adding Gaussian noise to the output dataset of queries in order to protect the individual data privacy in statistical information retrieval [25]. In FL, DP is applied to protect uploaded parameters of participants. Existing DP-based methods can be categorized into centralized differential privacy (CDP), local differential privacy (LDP), and distributed differential privacy (DDP) [26]. However, these techniques usually involve a trade-off issue between learning accuracy and privacy protection [27].

Secure Multi-party Computation (SMPC) A SMPC is a crytographic method that allows multiple participants working together to complete a computing task or execute a mathematical function without disclosing any data to the others. The methods of applying SMPC in federated learning framework can be divided into two categories:

1. Server based: all the clients send the processed data shares to multiple servers respectively, and we assume the servers are independent and not all corrupted [28].
2. Client based: we usually have only a single server, and most of the effort will be invested on the clients for a secure computation [29].

Secure Data Aggregation In FL, it is critical to aggregate the client updates in an encrypted way. Homomorphic encryption (HE), relying on mathematical encryption

methods, can perform operations on encrypted input and yields a ciphertext containing the encrypted result. Using HE in FL can ensure no performance loss in terms of model convergence [30]. There are two popular HE schemes for FL: the Paillier scheme and the EIGamal Scheme.

Anonymous Communication and Shuffle Model Another line of work uses anonymous communication and shuffle model for data privacy protection [31]. The purpose of anonymous communication is to make sure the information receiver does not know who is the owner of the message. The general method is to establish an anonymous network, and a query will go through a three-hop route to hide the source of the query submitter.

In shuffle model, we insert a shuffler between the sources and destinations of communications. The function of a shuffler is to shuffle the incoming message and then submit them to the destination(s) to achieve the anonymization of sources.

1.3 Structure of the Book

The structure of this book is as follows.

In Chap. 1, we briefly introduce federated learning and also the overview of security and privacy problems in federated learning as the background for the whole book.

In Chap. 2, we introduce the basic idea of inference attack and its detailed implementation in federated learning. We present several mainstream inference attacks, including model inversion attack, property inference attack, membership inference attack, and model inference attack. Then we summarize the counterattack methods against inference attacks in federated learning.

In Chap. 3, we summarize technologies of poisoning attacks in federated learning and show their workflows. The attack includes untargeted poisoning attacks, targeted poisoning attacks, and backdoor poisoning attacks. We also present the most effective defense methods from the literature against poisoning attacks in federated learning.

In Chap. 4, we present GAN-based security attacks in federated learning, including an introduction on GAN, GAN-based poisoning attacks, and GAN-based inference attacks. We also classify GAN-based attacks into GAN-based attacks from insiders, GAN-based attacks from outsiders, and GAN-based attacks from central server. Then, we summarize existing approaches for defending against these attacks.

In Chap. 5, we illustrate the concept of differential privacy and the application of differential privacy in federated learning, including centralized differential privacy, local differential privacy, and distributed differential privacy.

In Chap. 6, we present the secure multi-party computation for secure federated learning. We firstly present the building cryptography blocks used in federated learning and then introduce the research line on update masking for secure federated learning and the related cryptographic tools to address the problems in the scenarios of federated learning.

In Chap. 7, we focus on secure data aggregation in federated learning. After introducing the basic concept of homomorphic encryption and its two popular implementation systems, we give an example of federated learning with multiple aggregation servers for readers to understand how the techniques work.

In Chap. 8, we illustrate the basic concepts of anonymous communication and shuffle model and also survey the application of the techniques in secure federated learning.

In Chap. 9, we summarize the book and present our understanding of the landscape of AI security and privacy in the near future for the possible reference of our readers.

Chapter 2
Inference Attacks and Counterattacks in Federated Learning

From the previous chapter, we have learned that federated learning (FL) can be used to protect data privacy since users no longer share their raw data during collaborative training. However, many studies have shown that even under the protection of federated learning, users' privacy is not necessarily secure. Inference attack is one of the privacy infringement methods that can be executed in FL. Through the study of this chapter, readers will have a basic understanding of inference attacks in federated learning. We also present a discussion on how to counter or mitigate inference attacks.

2.1 What Is Inference Attack?

Inference Attacks (IA in short) can be defined as extracting the information unknown to the public but is of interest to attackers in artificial intelligence applications. For example, in federated learning, if a cloud aggregator is an attacker, the "known information" may include model parameters, gradients, training sets' marginal distribution, and so on. At the same time, the "unknown information" may include training set information of the clients, such as image content, membership of certain data, attributes, and so forth.

Machine learning can be separated by two phases: the training stage and the testing stage. Inference attacks occur at the testing stage, while the poisoning attack happens at training stage, which will be discussed in the next chapter.

According to the purposes of attackers, we can roughly classify inference attacks into four types as below:

- Model inversion attacks
- Property inference attacks
- Membership inference attacks
- Model inference attacks

S. Yu, L. Cui, *Security and Privacy in Federated Learning*, Digital Privacy and Security, https://doi.org/10.1007/978-981-19-8692-5_2

Table 2.1 Taxonomy of inference attack

Attack type	Attack goals	Examples
Model inversion attack	Specific properties of training data. Information that is usually specific to a single or one type of data	What does an image in the training set look like; the attribute value of a certain training data
Property inference attack	Some global properties of the training set. Generally, no specific data is involved	The ratio of men and women or the ratio of people of color in the training set
Membership inference attack	The membership of specific data is owned by the attacker in the training set	Whether the data in the hands of the attacker exists in the training set of a machine learning model
Model inference attack	The structure, capabilities, and parameters of machine learning models	A malicious user can steal an undisclosed cloud model through model inference attacks, causing property damage to the cloud service provider

In Table 2.1, we present some basic information of the four categories of inference attack, and more details will be presented shortly.

It is widely believed that access control is the key for data privacy in machine learning, and the birth of the federated learning framework is a natural result from the line. From the perspective of privacy protection, the participants of FL do not need to hand over their training data to any third party for the training of effective machine learning models.

However, many studies have shown that machine learning models remember some information about the training set during the training phase. The reason is also straightforward: a model's training is a process of fitting the training data, then the generation of model memory is an inevitable result of most existing methodologies.

Many papers indicated that overfitting is a typical cause of model memory of the training sets. Overfitting occurs when a model performs better on its training set than on other datasets. Therefore, can we infer the training set information, which is not public, from machine learning models? Intuitively, it is possible: when humans learn new things through old ones, we often do not forget the old ones after the learning. From a mathematical point of view, deep neural networks are posterior probability solvers, which contain much prior knowledge about their training sets.

As shown in Fig. 2.1, machine learning models obtain the mapping relationships between the input domains and the output domains from their training set (prior knowledge) and then calculate the posterior probability of new inputs through the relationships. As a result, it is possible to reversely obtain the private information of the training sets through machine learning models. With the extensive study on inference attacks, researchers found that we can obtain relevant privacy information using models' outputs.

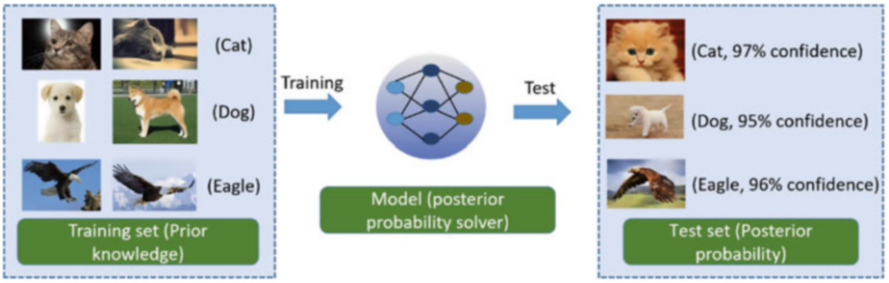

Fig. 2.1 Intuition of inference attacks

In brief, the core idea of inference attacks is to maximize the use and development of the "memory" generated by machine learning models during their training phase and use these memories to infer the privacy of the training set.

2.2 Inference Attack Categories

In this section, we present the details of the four categories of inference attacks in federated learning.

1. Model inversion attacks (also known as reconstruction attack, RA in short). In a standard model training phase, the information flow is from the training data move to the machine learning model. When an attacker tries to extract the details of training set (e.g., image content) from trained machine learning models, a RA occurs. In general, hackers are interested about specific information of the training set of the targeted model, e.g., whether a specific image in the training set or not.

   As shown in Fig. 2.2, the attacker wants to obtain the medical data that may contain patients' privacy collected by the medical institution. However, the attacker has no access to the original data, then he/she can perform reverse

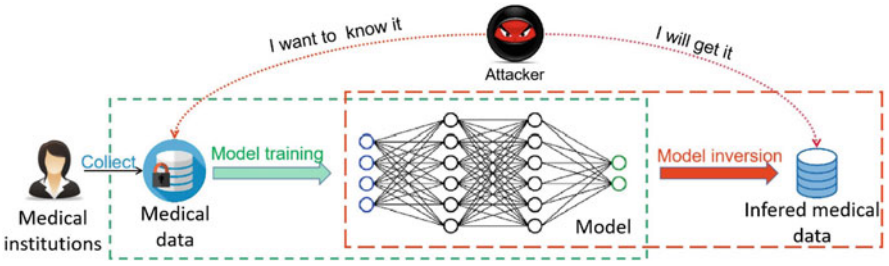

Fig. 2.2 An example of model inversion attack (aka reconstruction attack)

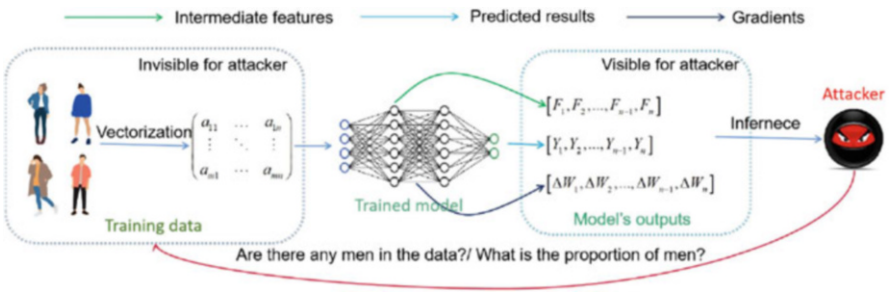

Fig. 2.3 An example of property inference attack

inferring (reconstruction attacks) on the machine learning model to obtain the medical data owned by medical institution.

2. Property inference attacks (PIA in short). In PIA, attackers are interested to obtain the statistical information of the training set, for example, the proportion of male over 30 years old in the training set, whether there are women in the training set, and so on, as shown in Fig. 2.3.

   In general, attackers cannot directly access the victim model, but they can execute PIAs through the outputs of victim models, including the final outputs and the intermediate outputs. Model outputs have different meanings in different scenarios, e.g., intermediate gradients in federated learning, or prediction vectors in cloud services.

   The threat of PIA is that it can help attackers obtain the victim's data composition, leading to unfair business competition and so on.

3. Membership inference attacks (MIA in short). Data stored by different institutions often have similarities in federated learning. For example, companies in the same city may have the same customers. Although customers' information is accessible by different parties, it does not mean that the parties can share the data. Protecting individual information is a legal obligation of data collectors. In federated learning, MIA occurs when an attacker can figure out whether a given data (e.g., a medical image) exists in the training set of a trained model. In Fig. 2.4, we show that the attacker can take advantage of the outputs of the victim model to infer whether a single data exists in the training set or not. Similar to the previous two attacks, MIAs can be carried out based on the outputs of victim models, but the difference is that attackers in MIAs possessing a target data desire to know whether the data in their hands is in the training set of the victim model.

4. Model inference attacks (also known as model extraction attacks, MEA in short). The goal of MEAs is to generate an alternative model, which is similar to the victim model. A typical MEA is usually a black-box attack as the attacker can only execute queries on the victim model.

   Model inference attacks usually happen in the traditional machine learning cases, rather than in federated learning. In federated learning, clients and server(s) clearly understand the model parameters and structure for each itera-

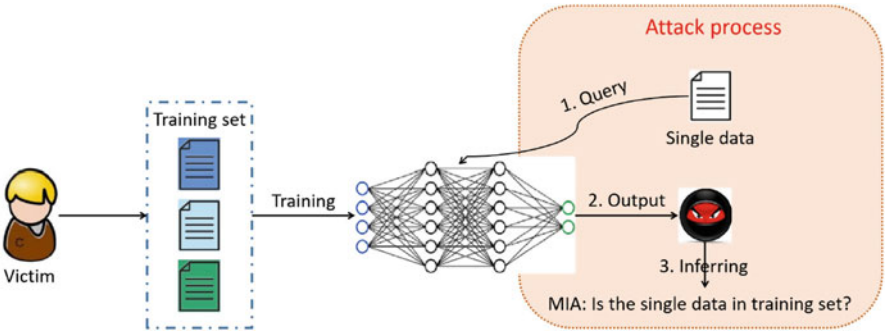

Fig. 2.4 An example of membership inference attack

tion. As a result, MEAs will not be employed in FL to a large extent. However, it is an important part of the family, and we refer interested readers to some related references [32–35].

2.3 Threat from Inference Attacks

We have presented the general ideas and goals of inference attacks. Some of these attacks look not that dangerous, e.g., MIAs. However, we will show that the perception is incorrect. It is necessary to take it serious for the long-term development and application of artificial intelligence.

Firstly, data is the valuable property for businesses in this information age. In many cases, it takes years of effort and significant investment to establish datasets, such as sale databases or customer databases. Appropriate protection on data privacy is the basic requirement for data collectors and curators as it would be nasty if the datasets unfortunately fall in the wrong hands.

In addition, personal data collected by institutions is also very sensitive, such as genetic information, purchasing transactions, and so on. These institutions usually use these data to build machine learning models to improve their service quality. However, as mentioned in reconstruction attacks (RAs), attackers could infer the confidential information of training data based on the corresponding machine learning models, thereby infringing on privacy. It not only compromises healthy commercial competition but also violates privacy laws and regulations, e.g., the GDPR.

Secondly, even if private data cannot be directly inferred, the statistical information of training sets or partial information of a single data is also valuable. For example, Institution A may probe the customer groups of Institution B through property inference attacks (PIAs) and then develop business strategies accordingly to beat institution B. Besides, the practical value of specific information of a single

data item is not as high as the statistical information of a whole training set in general. Therefore, PIA deserves to be treated seriously.

Finally, commercial competition is essentially a game for customer resources. Malicious institutions can employ various inference attacks to obtain whether other institutions have certain customers or not, so as to develop countermeasures to snatch these customers.

Up to date, the privacy risks brought by inference attacks hinder the pervasive application of machine learning in practice. Moreover, with the progress of machine learning technology, people pay more attention on privacy; these hidden dangers may become more serious.

2.4 Inference Attacks in Federated Learning

In previous sections, we presented the definition and basics of inference attacks and demonstrated the impact inference attacks will have on individuals and industries. In this section, we will present the detailed technical implementation of inference attacks.

2.4.1 Model Inversion Attacks

Model inversion attacks (MIAs), also known as reconstruction attacks (RAs), attempt to reconstruct private information from public information. In federated learning, the "private data" in RAs usually represents the local data of clients, while the "public data" means the global model (available to the central server and clients) and the gradients (available to the central server). Therefore, we can further classify MIA into the following two categories in federated learning.

1. Attacks on Aggregated Gradient
Aono et al. [36, 37] revealed that it is feasible to restore images from gradients in federated learning systems.

We start with a single layer neural network, as shown in Fig. 2.5, where x_i represents the point value of the input data, such as pixel, and X for the input vector. W_i represents the weight vector of the i-th layer, and $W = \{W_1, W_2, \ldots, W_i, \ldots\}$. b is the bias; f denotes the activation function, such as sigmoid, and $\hat{y}$ represents the model output, such as confidence.

If we define the loss function as the mean squared error, we have

$$\mathrm{L(W, b, X, f)} = \left(y - f\left(\sum_{i=1}^{n} W_i x_i + b \right) \right)^2 \tag{2.1}$$

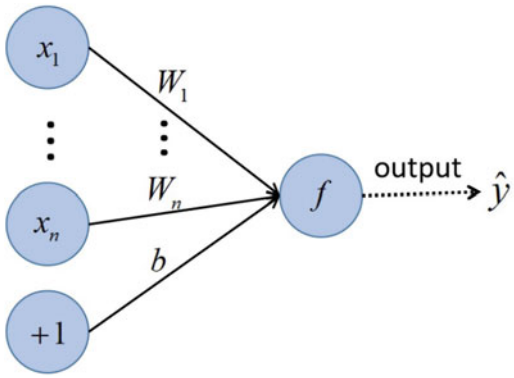

Fig. 2.5 A single-layer neural network

According to the back propagation algorithm, we can sequentially calculate the gradients of W and b as follows:

$$\begin{aligned}\nabla W_i &= \frac{\delta L(W,b,X,f)}{\delta W_i} = 2\left(y - f\left(\sum_{i=1}^{i=n} W_i x_i + b\right)\right) f'\left(\sum_{i=1}^{i=n} W_i x_i + b\right)\cdot x_i \\ \nabla b &= \frac{\delta L(W,b,X,f)}{\delta b} = 2\left(y - f\left(\sum_{i=1}^{i=n} W_i x_i + b\right)\right) f'\left(\sum_{i=1}^{i=n} W_i x_i + b\right)\cdot 1\end{aligned} \tag{2.2}$$

We can reconstruct the input x of clients through the following two perspectives:

- Through observation, we find that $\frac{\nabla W_i}{\nabla b} = x_i$. Therefore, clients' privacy could be leaked when the central server of federated learning receives the gradients uploaded by clients.
- ∇W_i and x_i are proportional. Therefore, when the central server does not receive ∇b, the original data can be obtained by adjusting the ratio. For example, when x is an image, it is easy to visually tell whether the ratio is correct or not.

The same attack strategy could be applied to neural networks with multiple hidden layers, namely, deep neural networks.

In detail, the gradients of W and b in a deep neural network are as follows:

$$\begin{aligned}\nabla W_{i,k}^1 &= \frac{\delta L(W,b,X,f)}{\delta W_i^1} = \xi_i \cdot x + \lambda W_i^1 \\ \nabla b_i^1 &= \frac{\delta L(W,b,X,f)}{\delta b_i^1} = \xi_i\end{aligned} \tag{2.3}$$

Then, we have

$$\frac{\nabla W_{i,k}^1}{\nabla b_i^1} = x_k + \frac{\lambda \nabla W_{i,k}^1}{\xi_i}, \tag{2.4}$$

where $W_{i,k}^1$ and b_i^1 represent the gradients and bias of the first hidden layer; λ is the regularization factor.

When $\lambda = 0$, $\frac{\nabla W_{i,k}^1}{\nabla b_i^1}$ is the data x, and when $\lambda > 0$, $\frac{\nabla W_{i,k}^1}{\nabla b_i^1}$ is an approximation of x. For more information, please refer to [38].

However, the above methods cannot adapt to complex neural networks, such as convolutional neural networks. To this end, Zhu et al. [39] proposed a gradients fitting method, Deep Leakage from Gradients (DLG). As shown in Fig. 2.6 (BP stands for back-propagation algorithm in the figure), DLG optimizes the noise by minimizing the gradients' difference between the noise and the original data obtained on the global model, thereby gradually leaking the original data.

With the optimization on noise, the noise will gradually transform into the data corresponding to the original gradient, that is, the private data, as shown in Fig. 2.7.

We can extract a paradigm of RAs from DLG, namely,

$$\arg\min_x d\left(\nabla_\theta \mathcal{L}_\theta(x, y), \nabla_\theta \mathcal{L}_\theta\left(x^*, y\right)\right), \tag{2.5}$$

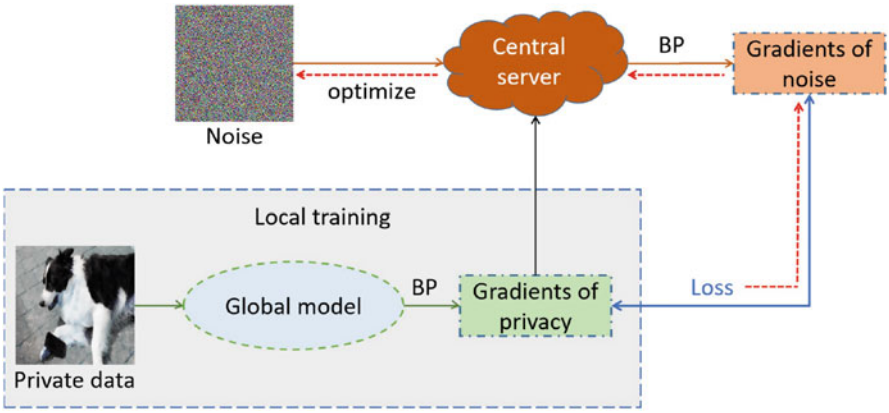

Fig. 2.6 Deep leakage from gradients

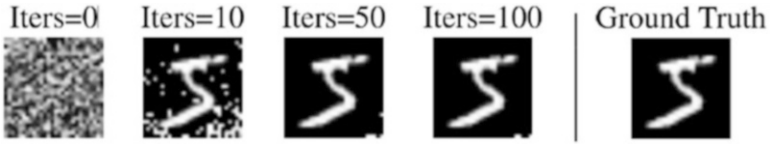

Fig. 2.7 DLG's inference progression

where $\nabla_\theta \mathcal{L}_\theta(x, y)$ and $\nabla_\theta \mathcal{L}_\theta(x^*, y)$ represent the gradients generated by private data and noise, respectively, and d is a distance metric function.

Generally speaking, Eq. (2.5) can be solved by stochastic gradient descent (SGD). It is worth noting that the derivation of the optimization requires a second-order derivable global model. Therefore, when the activation function of the global model is a second-order non-derivable function, such as ReLU, leaky_ReLU, etc., this method cannot be used.

For reader's references, we post DLG's pseudocode below as shown in Algorithm 1.

Algorithm 1 Deep leakage from gradients

Input:
$F(\mathbf{x}; W)$: differentiable model,
W :parameter weights,
∇W : gradients calculated by training data
Output:
private training data $\mathbf{x}$, $\mathbf{y}$

1: $\mathbf{x}^* \leftarrow \mathcal{N}(0, 1), \mathbf{y}^* \leftarrow \mathcal{N}(0, 1)$ ▷ initialize dummy inputs and labels.
2: **for** $i \leftarrow 1$ to n **do**
3: $\quad \nabla W^* \leftarrow \partial \ell \left(F\left(\mathbf{x}^*, W\right), \mathbf{y}^*\right) / \partial W$ ▷calculate dummy gradients
4: $\quad \mathbb{D} \leftarrow \left\|\nabla W^* - \nabla W\right\|^2$
5: $\quad \mathbf{x}^* \leftarrow \mathbf{x}^* - \eta \nabla_{\mathbf{x}^*} \mathbb{D}, \mathbf{y}^* \leftarrow \mathbf{y}^* - \eta \nabla_{\mathbf{y}^*} \mathbb{D}$ ▷update data to match gradients.
6: return $\mathbf{x}^*, \mathbf{y}^*$

In addition, the authors of DLG also confirmed that RAs on aggregated gradients are also possible. For example: when the attackers obtain the aggregated gradient $\Delta W = \frac{1}{n}(\Delta w_1 + \Delta w_2 + \Delta w_3 + \ldots + \Delta w_n)$, then the attackers only need to know the value of n to infer all the data corresponding to Δw_1 to Δw_n.

Besides inferring the original data x, DLG can also infer the label y of the original data, but the error rate of the label inferring is very high. Therefore, Zhao et al. [40] proposed the Improved Deep Leakage from Gradients (iDLG), a method that can improve the label inference accuracy. Specifically, the cross-entropy of one-hot encoding in a neural network is defined in Eq. (2.6), where x is the input and c is the label of x.

$$l(\mathbf{x}, c) = -\log \frac{e^{y_c}}{\Sigma_j e^{y_j}} \tag{2.6}$$

Further, $y = [y_1, y_2, \ldots, y_c, \ldots, y_j, \ldots, y_n]$ is the output of the Logit function by the trained global model. According to the back-propagation algorithm, the gradients corresponding to each y can be defined as follows:

$$g_i = \frac{\partial l(\mathbf{x}, c)}{\partial y_i} = -\frac{\partial \log e^{y_c} - \partial \log \Sigma_j e^{y_j}}{\partial y_i} = \begin{cases} -1 + \frac{e^{y_i}}{\Sigma_j e^{y_j}}, & \text{for } i = c \\ \frac{e^{y_i}}{\Sigma_j e^{y_j}}, & \text{else} \end{cases} \tag{2.7}$$

We note that $\frac{e^{y_i}}{\Sigma_j e^{y_j}} \in (0, 1)$ is the probability calculated by the softmax function.

However, in practice, clients do not upload gradients of y to the central server. Therefore, we consider the gradients of the last layer of the global model as follows:

$$\begin{aligned} \nabla \mathbf{W}_L^i = \frac{\partial l(\mathbf{x}, c)}{\partial \mathbf{W}_L^i} &= \frac{\partial l(\mathbf{x}, c)}{\partial y_i} \cdot \frac{\partial y_i}{\partial \mathbf{W}_L^i} \\ &= g_i \cdot \frac{\partial \left(\mathbf{W}_L^i \mathbf{a}_{L-1} + b_L^i \right)}{\partial \mathbf{W}_L^i} \\ &= g_i \cdot \mathbf{a}_{L-1} \end{aligned} \tag{2.8}$$

We can see that when the range of the $L-1$ layer's activation function is non-negative, $\mathbf{a}_{L-1} < 0$. At this time, if and only if when $i = c$, $\mathbf{W}_L^i < 0$. Therefore, attackers can infer the data's label through the sign of gradients of the last layer. Attackers can use the data label to assist their attacks, making DLG more powerful, which is the idea of iDLG.

Geiping et al. [41] pointed out that the distance metric of DLC is not optimal. They treated gradients as vectors, which can be decomposed to two components, magnitude and direction. The magnitude captures the training state to measure the local optimality of the current data point with respect to the model. By contrast, the high-dimensional direction of gradients carries more information because the angle between two data points quantifies the change in a data point as it approaches the other [42, 43]. Therefore, they suggested to use an angle-based distance metric, namely, changing Eq. (2.5) to a cosine similarity, as defined in Eq. (2.9).

$$\arg\min_x 1 - \frac{\langle \nabla_\theta \mathcal{L}_\theta(x, y), \nabla_\theta \mathcal{L}_\theta(x^*, y) \rangle}{\|\nabla_\theta \mathcal{L}_\theta(x, y)\| \, \|\nabla_\theta \mathcal{L}_\theta(x^*, y)\|} + \alpha TV(x), \tag{2.9}$$

where total variation loss (TV) is used to smooth the generated image to make it natural. We simply name the new method as CS-DLG (cosine similarity based DLG).

CS-DLG can be applied in large-scale neural networks, and its comparison with DLG is shown in Fig. 2.8. The first row is the result of DLG recovery in trained

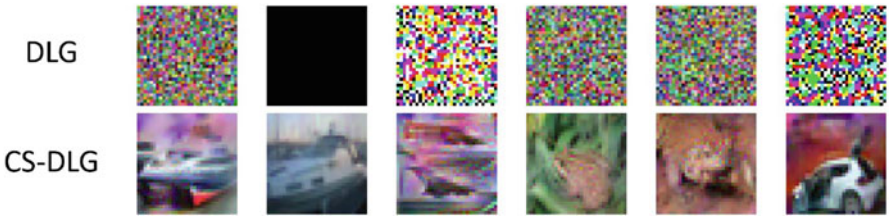

Fig. 2.8 The comparison on recovery effectiveness between DLG and CS-DLG

ResNet, and the second row is the output of CS-DLG recovery. The comparison demonstrates that high-dimensional direction of gradients can reconstruct images better.

CS-DLG successfully extended the gradient-based inference to ImageNet-level images. Nevertheless, when the gradient is aggregated, e.g., averaged, the method does not work.

The latest research shows that the introduction of fidelity regularization (FR) and group consistency regularization (GCR) in gradient-based inversion enables attackers to recover single data point (ImageNet-level) with aggregated gradients from deep neural networks, such as ResNet50 [44].

Moreover, we note that CS-DLG is based on some strong assumptions, such as knowing the statistical information of the original data in the batch normalization layer. These assumptions make CS-DLg somewhat unrealistic in many application scenarios, e.g., in federated learning cases.

In generally, gradient-based inference attacks in federated learning can only be implemented by a central server. Hence, in these methods, central servers are considered malicious.

So far, we have presented the basics of gradient-based inference attacks. For readers who plan to explore further of the field, we recommend them to read the recent related papers [44–47].

2. Attacks on Global Model

As discussed, the machine learning models can "remember" their training data in some ways during the training process. Therefore, it is natural to question whether global models could leak privacy or not. In federated learning, the central server and clients share the global model and all the details. As a result, unlike gradient-based attacks, in global model-based attacks, both the clients and the central server could be compromised by attackers to achieve their malicious goal.

Most model-based attacks rely on the techniques of generative adversarial networks (GAN) or the combinations with gradient-based attacks. Hitaj et al. [48] demonstrated that malicious clients could use GAN to perform reconstruction attacks to obtain privacy information of clients in federated learning systems. For readability, we name their method as DMU-GAN.

The core idea of DMU-GAN is as follows: under the framework of GAN, the attacker duplicates the global model as a discriminator, and the generator generates

samples similar to the target data (the victim's data) according to the feedback of the discriminator. The attacker labels these samples with a different label from the target class and incorporates the generated sample into the training set of the federated learning system, which makes it harder for the global model to discriminate the generated samples from the target data during training because they both have the same representation but different labels. Therefore, the model tends to obtain more information about the target data from the victim during training, thereby helping the attacker to generate samples similar to the target data.

Notably, DMU-GAN does not appear to be as threatening as gradient-based attacks because the data inferred by gradient-based attacks is a specific sample in the victim's training set, such as a specific image or a specific paragraph of text. The data inferred by DMU-GAN is more like an average sample; in other words, the sample is not specific data points in the training data but an average sample corresponding to a certain label. However, in systems like face identification, DMU-GAN is not much different from gradient-based attacks, because one class in this type of system merely represents a specific person. In addition, DMU-GAN hinders the training process of federated learning, which significantly increases attackers' probability of being detected.

Why does DMU-GAN enable the global model to obtain more target data information from the victim? In fact, machine learning models often encounter data categories that are difficult to fit during training. For example, small models trained with MNIST often have lower test accuracy on number 3. There are two main reasons for this: (1) The representation of the same category is too different; (2) The representation difference of different categories is too small.

In order to overcome the problem, machine learning models usually adjust their parameters to extract generalization features from these data. Undoubtedly, this allows the global model in federated learning to provide attackers with more information about the target data.

In general, when a model is trained, the model attempts to find decision boundaries among categories. However, when the boundary between two classes is ambiguous and no more data is available to provide adequate information, DNN attempts to find more distinctive features from the existing data to separate the two classes, namely, to widen the gap between two distributions.

As shown in Fig. 2.9, if it is difficult for the model to distinguish between cats and dogs, then gradients are usually provided by the two classes in the later stages of training, which forces the model to "memorize" more information about the two classes.

Wang et al. [49] also proposed a method called mGAN-AI in combination with GAN, which assumes the server as the attacker. mGAN-AI also uses the global model as a discriminator, but it changes the last layer of the discriminator to multiple outputs to build a multi-task discriminator. The discriminator can simultaneously classify users' identity, data category, and data authenticity. The generator outputs the client's private data based on the information provided by the discriminator. Similarly, the data generated by mGAN-AI is not specific data items of victims.

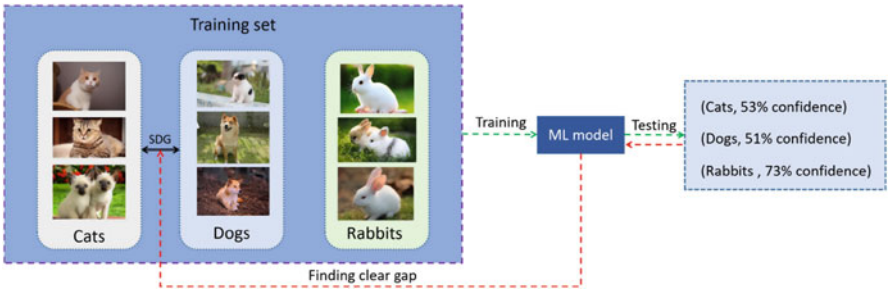

Fig. 2.9 SDG: sub-distribution gap

We suggest interested readers to refer to [49–51] for more details. Furthermore, readers can also refer to Chap. 4, where we will focus on GAN in federated learning.

In this subsection, we present the basic attack ideas of model inversion attacks (also called reconstruction attacks, RAs) in federated learning systems for readers. We have to note that RAs also have significant impacts on non-federated learning systems, such as machine learning as a service (MLaaS) systems. Once again, we refer the interested reader to [52–55] for further exploration.

2.4.2 Property Inference Attacks

Property inference attacks (PIA in short) is another major threat in federated learning. Different from reconstruction attack, which focuses on inferring the data representation of the training set, PIA pays more attention to the overall statistical information of the training set. We use the following case as an example: a company uses its internal emails to train a spam classifier. In the RA case, attackers aim to infer specific email messages to obtain internal secrets. While in the PIA case, attackers may be interested to know the overall sentiment distribution embodied in emails as overall sentiment may reflect a company's culture or financial status.

As we know, PIA was firstly proposed by Ateniese et al. [56] in 2013, and we name the method First PIA (FPIA in short). As shown in Fig. 2.10, FPIA builds a meta-classifier to infer the information hidden in the target models.

In general, most PIAs assume that attackers have auxiliary datasets with the same or similar distribution as the victim's training set, which is reasonable in federated learning setting. To facilitate understanding, we firstly define some terms in PIA, as shown in Table 2.2.

The core idea of majority of PIAs is that attackers divide the auxiliary dataset into multiple parts. For the convenience of understanding, we divide the auxiliary dataset into two parts, P and $\bar{P}$, representing the sub-dataset possessing property P and non-P, respectively. The attackers then use these two datasets to train two shadow models, respectively. In theory, one of the shadow models would capture

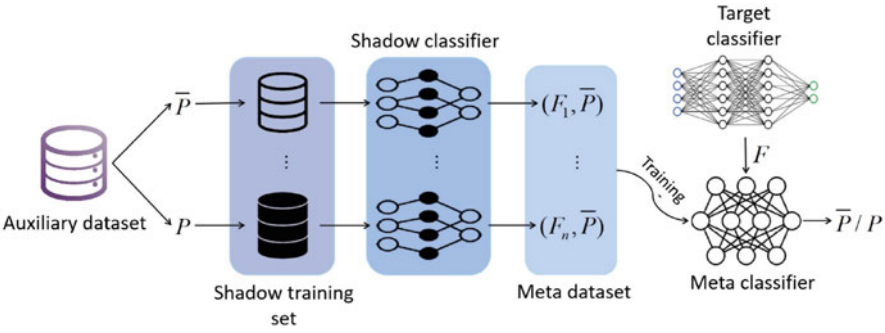

Fig. 2.10 The architecture of FPIA

Table 2.2 Terms and meanings of PIA

Term	Meaning
Shadow training set	Usually equivalent to the auxiliary dataset or a subset of the auxiliary dataset
P	Data with attribute P in the auxiliary dataset
$\bar{P}$	Data without attribute P in the auxiliary dataset
Shadow classifier	A machine learning model designed by attackers. The model's main task is the same as the target model
F_i / F	Outputs of shadow model/target model, such as confidence
Meta dataset	A dataset consisting of F and whether the corresponding data has attribute P
Target classifier	A machine learning model under attack
Meta-classifier	A binary classification network trained on the meta dataset

the property p, while the other would not, causing them to be different in terms of their outputs. Then the attackers form a metadata set through deploying the outputs of the trained shadow models and whether the data corresponding to the output has attribute P. A meta-classifier trained using the metadata set can judge whether the target classifier's training set has the attribute P through the outputs of the target classifier [56].

The above method is very straightforward. However, their method is only applicable to traditional machine learning models, such as hidden Markov models (HMM) and support vector machines (SVM). For majority of the popular deep neural networks, the method is not effective. Ganju et al. [57] used a fully connected neural network (FCNN) to test the facts and concluded that the permutation invariance of FCNN is one of the main reasons for the failure of FPIA in deep neural networks.

The permutation invariance of FCNN means that for well-trained fully connected neural networks, swapping the neuron nodes in the same layer and adjusting the corresponding weights will not affect the learning robustness of FCNN. As shown in Fig. 2.11, the original network (left) and the permutation equivalent network (right) are basically the same in terms of test accuracy.

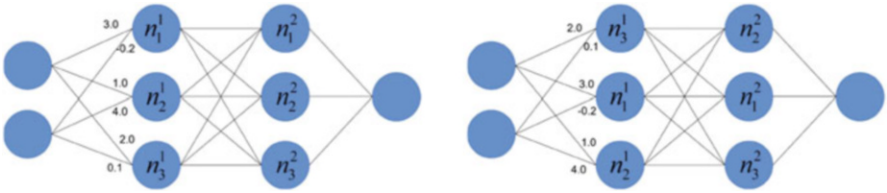

Fig. 2.11 Permutation invariance of FCNN

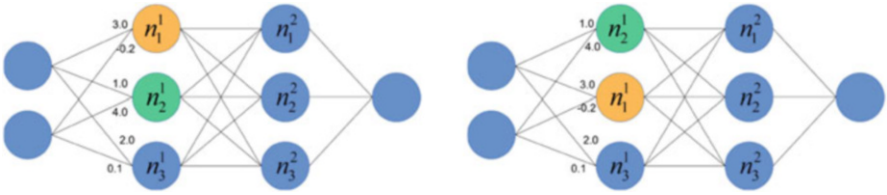

Fig. 2.12 An example of neuron sorting

The existence of the permutation invariant makes the training of shadow models extremely difficult, and the massive parameters make FCNNs have many permutation invariant networks, which exacerbates the difficulty.

Ganju et al. [57] proposed two methods to solve the above difficulty.

The first method is called neuron sorting. In face recognition systems, the different angles of a face may cause recognition errors. For example, a system may detect a mouth, then it is expected to have a nose above the mouth. Therefore, to improve the system's accuracy, a uniform norm is used to align face images before recognition. Inspired by this, the author attempts to introduce a standard paradigm to all shadow models to eliminate the impact of permutation equivalence. They sort the nodes of each layer in the shadow model according to the summation of corresponding weights. This method enables PIAs to execute attacks through input with a predetermined pattern, eliminating the impact of permutation invariant. As shown in Fig. 2.12, the attacker uses the neuron-sorted network as the training set of the attack model (to have a clear visual effect, we only sort the first layer here. In practice, each layer needs to be sorted).

The aforementioned method effectively solves the problem of permutation invariance in PIAs. However, in the case of very large neural networks, even without considering the influence of the equivalent network, the huge amount of parameters will bring significant challenges to the training of the meta-classifiers and the shadow models.

The second method is named as deep-set-based representation. The deep-set allows inputs to have permutation invariance, and the networks based on the deep-set can achieve the functions with invariance. We give an example to specify it in Fig. 2.13. Set A has three different representations, but they are the same essentially with the differences of order. For an ordinary neural network $F(\cdot)$, the order of the elements will result different outputs. Nevertheless, a function $F'(\cdot)$ that satisfies

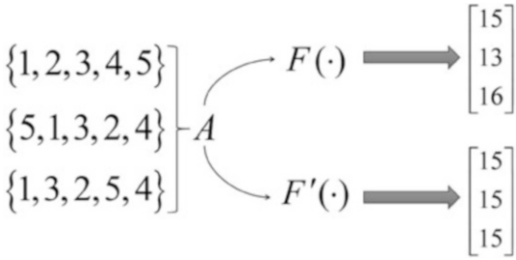

Fig. 2.13 Non-permutation invariance-based model $F(\cdot)$ and permutation invariance-based model $F'(\cdot)$

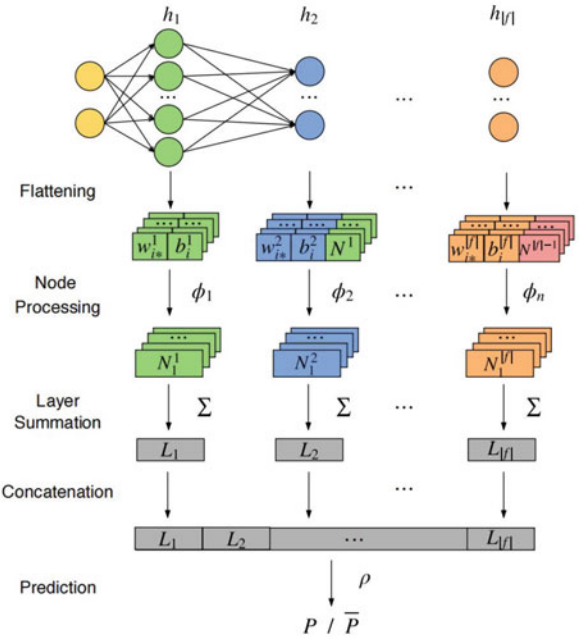

Fig. 2.14 An attack architecture based on deep-set

permutation invariance can overcome the problem and output the results as it should be no matter the orders of the elements of the set.

For more information about deep-set, interested readers could refer to [58] and the reference therein.

Ganju et al.[57] designed an attack model using the deep-set architecture as shown in Fig. 2.14. The method expresses the weight corresponding to each node of each layer in a fully connected neural network (FCNN) as a feature value N_i^j and summarizes the feature value of nodes of each layer to obtain L_i, which is used as the overall representation of the i-th layer. The meta-classifier in PIAs can use the vector composed of L_i of each layer in FCNN as the input. It is not difficult to find that deep-set makes the meta-classifier easier to train and requires less memory. Their experiments have also shown that the method requires fewer shadow models.

Although we have presented the attack's idea as much as we can, there are many details and related information not covered. We recommend interested readers to go

through the related work, such as [57, 58], and the following development in the line.

Although Ganju et al. [57] expanded PIAs to run in fully connected networks, it can only infer simple datasets like MNIST. In addition, the inferred attributes may have a certain relationship with the main task of the target model. For this reason, Melis et al. [59] closely integrated PIAs with the federated learning setting. They did not use the outputs of target model as inputs of the meta-classifier but used the gradients in the federated learning process. Experiments have proved that the method is more effective in inferring attributes unrelated to the target model's task, and it could be applied to more complex datasets.

Up to date, Chase et al. [60] have combined the poisoning attack (see Chap. 3) with PIAs for the first time and demonstrated that the poisoning attack could strengthen PIAs.

In this subsection, we briefly present the basic picture of PIAs. If readers need to know more, please refer to [61–64].

2.4.3 Membership Inference Attacks

Membership inference attack (MIA in short) is a serious attack format in federated learning. In MIA, attackers are interested to know whether specific data points (e.g., persons, images, and so on) are in the training sets of the victim models or not. For example, a hacker wants to know whether Bob's medical data was used in training set of a cancer detection model. Obviously, the model owner will not release the related information in order to protect the privacy of the participants. However, the hacker desires to obtain the information. If the attacker figured out that Bob's medical data is in the training set, then the hacker can sell related medical staff to Bob or sell the information to third parties for commercial gain.

MIA is a serious threat to federated learning systems. In theory, when federated users are institutions, the users are usually in a competitive relationship. For example, institution A may learn through MIAs to identify which customers in his database are also customers of institution B, which will cause unfair commercial competitions.

In 2008, Homer et al. [65] verified for the first time that a specific DNA sequence can be identified from the mixed DNA sequences. It is the first time that the MIA concept was proposed. However, the method was mainly related to biology and explicit statistics. Shokri et al. [66] proposed to deploy MIA for machine learning for the first time. To offer readers a better understanding of MIAs, we will introduce how the original attack was carried out. Most of the subsequent improvements and research work are based on the same or similar idea.

Shokri et al. [66] used a shadow training set to build a shadow model to simulate the behavior of the target model, as shown in Fig. 2.15, and then obtain the corresponding output of the shadow training set and the shadow test set through each shadow model and use "in the training set" and "not in the training set" to

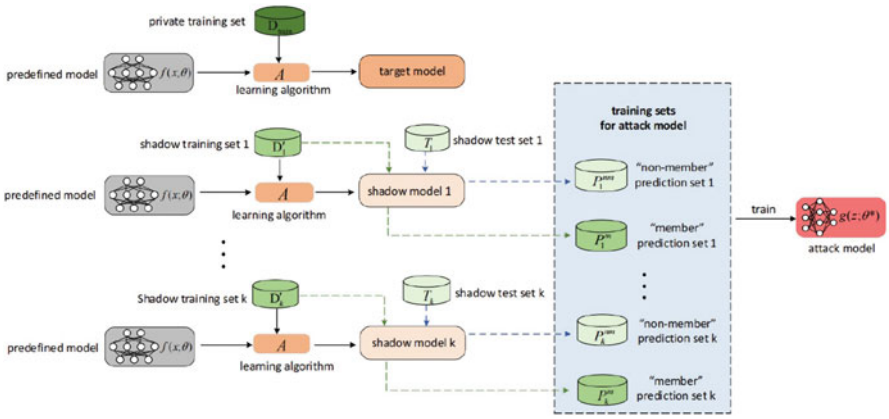

Fig. 2.15 The original scheme of MIA

label these outputs, respectively. Finally, the attacker uses the newly constructed dataset to train a meta-classifier (also known as attack model). The meta-classifier can judge whether an instance is in the target model's training set or not according to the output corresponding to the instance.

One of the most interesting points to discuss is what kind of knowledge the attacker possesses in MIAs, which directly determines the threat level of MIAs. Most of the follow-up works fall in this line. In the cases of MIAs, knowledge can be roughly divided into four categories: *data knowledge, training knowledge, model knowledge,* and *output knowledge*.

1. *Data Knowledge* Data knowledge determines how the adversaries construct effective shadow training sets. In most MIA settings, the adversaries can obtain the data distribution of the target model training sets and can sample from them. Although the assumption is reasonable in a collaborative scenario, such as federated learning, there are many situations that are not applicable, such as machine learning as a service (MLaaS in short).

When the adversaries cannot obtain the target data distribution, they usually have the following two methods to continue their play:

- Statistical synthesis. The adversaries may have some statistical information, such as the marginal distribution of certain features. They can sample from these marginal distributions.
- Model synthesis. If the attackers have neither data distribution nor statistical knowledge, they can utilize the target model to obtain the shadow training sets. The intuition is that samples classified with high confidence by the target model are more likely to be statistically similar to the data in the target training sets.

2. *Training Knowledge* Training knowledge refers to some hyper parameters in training processes, such as training epochs, optimizer, learning rate, and so on. In collaborative training scenarios, the assumption can be satisfied without problems.
3. *Model Knowledge* Model knowledge mainly refers to model structure, which includes the structures and activation functions used by each neural network layer.
4. *Output Knowledge* Output knowledge determines what kind of feedback the attacker will get from the target model. There are three main types of output knowledge listed as below.

- The first one is that the adversaries obtain fully output knowledge, which means that they know the confidence of each category.
- In the second type, the adversaries possess partial output knowledge, which means the attackers can only obtain the top k with the highest confidence.
- The third and the most extreme one is that the adversaries can only obtain the hard label corresponding to the input (no confidence information).

Most of MIAs' improvement work is to reorganize or relax the above four kinds of knowledge to explore the threat of MIAs. In addition, in different scenarios, adversaries will have different abilities. Under the cooperative training setting, if an attacker implements MIAs by influencing the intermediate results of training, then we call this type of attack as *active attack* (e.g., model inversion attack, property inference attack), otherwise as *passive attack*. These two concepts are easy to understand, and readers can easily judge the corresponding attack types through the definitions; we will not discuss them further.

There are three strong assumptions in the work of Shokri et al [66], which are listed as follows:

- Multiple shadow models need to be trained, which significantly increases the cost of attacking.
- The attackers need to know the structure of the target model and use it as the shadow model's structure.
- The attackers must know the target distribution to implement effective attacks.

Although the three assumptions can be met in many scenarios, it cannot be denied that they significantly impede the pervasive usage of MIAs. Therefore, Salem et al. [67] gradually relaxed the conditions of the work of Shokri et al. [66] and verified the effectiveness of MIAs. They proposed MIAs under three different conditions, as shown in Table 2.3, where—means that this information is not needed and $\surd$ denotes the information is expected.

Adversary type one and type two only need one shadow model to carry out attacks. Adversary type one proved that when the distribution of the target training set was known, only one shadow model different from the target model can be used for an effective attack.

In addition, adversary type two proved that the shadow model was not necessarily to be trained with the same data distribution as the target training set. The reason is that the shadow model only provides statistical differences between the training set and other data for the attack model, rather than simulating the target model. From

Table 2.3 Conditions of the Salem et al. models [67]

Adversary type	Shadow model design		Target model's training set distribution
	Number of shadow models	Target model structure	
Shokri et al.	Multiple	√	√
Adversary type one	One	–	√
Adversary type two	One	–	–
Adversary type three	–	–	–

their experiments, the statistical difference is almost similar even between different distributions. Therefore, the authors named adversary type two as "transfer attack."

Finally, the authors also improved a metric-based attack proposed by Yemo et al. [68]. This attack does not require any of the conditions in Table 2.3. Scholars refer to similar methods as metric-based attacks. In particular, the corresponding output is obtained through querying the target model and comparing it with a certain threshold. A result greater than the threshold means that the data comes from the target training set. It is worth noting that such methods usually require specific steps of processing the inputs of the queries or the outputs of models.

We recommend readers to refer to the related work for further details, such as [67–69], if they are interested.

Meanwhile, Long et al. [69] extended MIAs to neural networks with decent generalization. They proved that even if a non-overfitting neural network can reduce the accuracy and recall rate of MIAs, the attackers can filter samples to carry out more precise attacks. The method not only reduces the cost of attacking but also proves that models with good generalization are also vulnerable.

Li et al. [70] studied the transfer of models. Similar to the majority of the related work, they assumed that the attackers possess data that have the same distribution as the target training set, and they take advantage of the outputs of the target model as their labels. They firstly used a shadow model to fit the behavior of the target model, namely, making the probability distribution of the shadow model output equivalent to that of the target model. Secondly, they took advantage of the loss of the target data on the shadow model to determine whether it is in the target training set or not. In addition, they also combined the ideas of the adversarial attacks and judged the membership of the target samples by adversarial perturbation magnitude. The intuition is that the judgment model of the target model on the training set is often very robust; hence, it needs more perturbation than other samples to change the labels.

Choquette et al. [71] researched MIAs in the label-only scenario. Nowadays, many machine learning models use data augmentation as one of the auxiliary training methods. Intuitively, an enhanced version of data can be classified by the target model robustly, which proves that the data has a high probability of belonging

to the training set. Moreover, they also proposed an attack method based on the decision boundary, whose idea is similar to the attack method based on perturbation.

All the above MIAs target at classification models.

There are also works on generative models. Hayes et al. [72] was the first work deploying MIAs for generative models. They established a discriminator to identify overfitting of the target model. The purpose was to make the discriminator similar to the target discriminator as much as possible, thus implementing confidence-based MIAs. The further research of the line could be referred to Hilprecht et al. [73].

Researchers are most concerned with MIAs among inference attacks, and there have been many related surveys in place for interested readers, such as [74–76].

2.4.4 Model Inference Attacks

The purpose of model inference attacks is to obtain the information supposed not known to public, such as the structure and parameters of the target model.

In the federated learning setting, a model is shared among all participants. Therefore, as far as we know, there are much fewer work on model inference attacks in federated learning. Nevertheless, there is some possible research directions in the future, such as inferring a client's local model with an enormous contribution under some federated learning settings with incentive mechanisms. We do not go into more details about it since there is no clear research literature at present.

2.5 Counter-Inference Attacks

With the development of machine learning privacy and security, each attack method has resulted in various numbers of corresponding defense strategies. In this book, we try to help readers to understand the current overall development of the field and will not discuss too much details.

According to the literature reviewed, we roughly divide the existing counter-inference attack methods into five categories, which will be presented one by one in the following.

1. Machine Learning Optimization-Based Defense

In the early days, researchers generally believed that the main reason of inference attacks was model overfitting. As a result, countering inference attacks by preventing model overfitting is the most accessible means in intuitive. However, the essence of this type of method is not a direct security method; thus, the protective effect is not ideal. Next, we give a few examples to help readers understand it smoothly.

1. *Learning Mechanism*

One of the main reasons of model overfitting is the large scale of model capacity. There is an essential difference between the learning methods of neural networks

and human brains. The brain is an intelligent system based on empiricism and logical reasoning, while the popular CNNs are purely empiricism. Therefore, generally speaking, under the same conditions, a network with a larger scale is more capable for training and easier to remember the related information, which leads to overfitting. This makes the overfitted network vulnerable to inference attacks.

2. *Regularization*

The purpose of regularization is to optimize the performance of models. Regularization does not change the model structure. It will make certain model parameters approach 0 to reduce its impact on the model. In addition, the weights of the features captured by these parameters will be significantly reduced but not completely lost. Therefore, it is a bit softer than overfitting, but it also makes the model vulnerable to inference attacks.

3. *Model Stacking*

Model stacking is essentially a kind of ensemble learning. It trains multiple models by dividing the training set into pairs of disjoint sets and then combining them into an aggregated model for prediction. The schematic diagram is shown in Fig. 2.16. This method can effectively improve the generalization of models. Furthermore, compared to the previous two methods, it can better mitigate inference attacks.

There are many ways to prevent models from overfitting, and we will not present them in detail. However, it is worth noting that not all methods mitigating overfitting can alleviate inference attacks. As mentioned above, data augmentation may lead to stronger inference attacks.

2. Perturbation-Based Defense

As mentioned at the beginning of this chapter, inference attacks use existing knowledge K to infer unknown data U. It is difficult for people not to think about whether a specific relationship between K and U enable the possibility of inference attacks. One of the easiest ways to destroy this relationship is to add noise to K. Differential privacy (DP) is one of the perturbation methods widely studied at present, for instance, adding perturbation to the gradients in gradient-

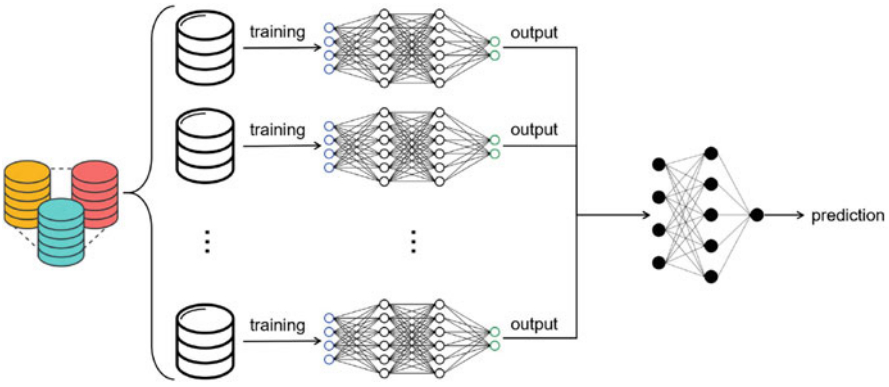

Fig. 2.16 An example of model stacking

based RAs, adding perturbation to model outputs in PIAs, and adding noises to model parameters. In general, defenders can perturb optimization algorithms, model parameters, intermediate model results (such as gradients), model outputs, and so on. Many papers have demonstrated that differential privacy can offer privacy protecting in artificial intelligence.

Jayaraman et al. [77] conducted a comprehensive analysis on MIAs using a variety of differential privacy mechanisms. They found that privacy guarantee and model utility are difficult to weigh. Specifically, a high privacy guarantee will significantly reduce model utility and vice versa. Then, Truex et al. [78] tested the model trained with unbalanced training set and found that minority categories are more vulnerable to inference attacks.

Unfortunately, DP-based defense will come at the cost of data utility of minorities, which is fatal for practical deployment of models, such as anomaly detection. Nowadays, many variants of differential privacy have been proposed for privacy protection, but these studies almost all ended at the trade-off of privacy-preserving and model utility. Compared with other defense technologies, DP possesses the advantage that it does not significantly increase the cost of system deployment, and it can be easily integrated with current systems.

In order to avoid redundancy, we will not discuss differential privacy-related methods too much in this chapter. Readers can refer to Chap. 5 for the progress of related research of privacy protection in federated learning using differential privacy.

3. Knowledge Distillation

Hinton et al. [79] proposed knowledge distillation, a technique for model compression. Shejwalkar et al. [80] applied knowledge distillation against inference attacks. They minimized the KL divergence (a popular distance metric in information theory) between the outputs of the student model and the teacher model, which avoided directly using specific datasets to train the student model (for deployment), thereby mitigating inference attacks to a certain extent. Kaya et al. [81] compared knowledge distillation and a variety of regularization techniques, and their experiments demonstrated that knowledge distillation is an effective defense method. However, none of them has been experimentally verified under the most advanced attack settings, and most of the defenses can only be constructed with a public training set.

4. Adversarial Machine Learning

The main idea of adversarial machine learning is that a defender optimizes the model from his perspective when training or deploying the model. The current adversarial machine learning can be divided into two main categories:

1. In the training phase, a defender adds the existing attack strategy to the model training process as a regularization term. For example, in MIAs, defenders can add the loss function of the attack model (meta-classifier) to the training loss. Maximize the loss of the attack model while minimizing the loss of model utility, forming a min-max game. Obviously, this will inevitably reduce the utility of the target model.
2. In the prediction stage, because the attack model takes the outputs of the target model as inputs, and the attack model is susceptible to adversarial samples,

therefore, the defender can add elaborate perturbations to the outputs of target models to turn it into an adversarial sample, thereby mitigating the attack. This defense strategy can mitigate the problems caused by the trade-off of privacy-preserving and model utility.

5. Encryption-Based Methods

Encryption is a traditional and extremely effective means of privacy protection. It has been used in various fields for decades, and the privacy and security field of machine learning is not an exception. In collaborative learning, the effectiveness of encryption methods has been widely recognized.

At present, homomorphic encryption is the most promising privacy-preserving method. Its unique homomorphism and theoretical guarantee make it fit for privacy issues brought by model optimization. For example, an adversary can perform model inversion attacks based on the gradients in federated learning. However, if the gradients are encrypted by homomorphic encryption, it can not only complete the gradients' aggregation operation but also prevent MIAs. Unfortunately, encryption methods often bring high communication and computation costs. Therefore, it is difficult to implement in large-scale applications. In addition, encryption is not an acceptable defense method for attacks based on the output of the target model (such as attacks in machine-learning-as-a-service scenarios). In practice, the outputs of the target model will eventually be decrypted at the client sides; hence, encryption in this case only ensures that the data is not leaked in communication or server sides. This is one of the reasons why researchers need to explore other protection methods.

If you want to learn more about the detailed research on encryption, please refer to Chap. 7 and the reference therein.

2.6 Summary of the Chapter

In this chapter, we introduce the concepts of inference attacks and also discuss the possible inference attacks in federated learning and their core ideas. We further present how to defend against these attacks.

Due to time limitation, we do not provide readers a very detailed description, but we strongly recommend readers to go through the references given in this chapter to expand their knowledge further.

We note that there are many theoretical perspectives to be explored in the fields and sub-fields that we have discussed in this chapter, such as deep-set, invariance, and so on. For readers who are determined to explore the theoretical aspect of artificial intelligence (of course including security and privacy in artificial intelligence), we strongly recommend them to have a careful investigation of the recently hot method, geometric deep learning.

Chapter 3
Poisoning Attacks and Counterattacks in Federated Learning

In this chapter, we will discuss a popular and major attack format, poisoning attack, in federated learning environment due to the open nature and large scale of federated learning. Poisoning attacks happen at the phase of model training, while inference attacks we discussed in Chap. 2 occur at the testing or prediction stage of artificial intelligence applications.

We firstly explain the basic concepts of poisoning attacks in general and then present the related poisoning attacks in federated learning and the related counterattacks.

3.1 What Is Poisoning Attack?

The purpose of poisoning attack is to destroy the training of models or degrade the effectiveness of the expected trained models. In general, poisoning attacks can be categorized into two classes: *untargeted attacks* and *targeted attacks*. In the untargeted category, we can further have two branches: Denial-of-Service (DoS) attack and distortion attack. The targeted attack is also popularly known as *backdoor attack*.

The goal of DoS attack is to prolong the training duration by feeding carefully crafted poison data. In a legitimate training, every round of training is supposed to push the intermediate model closer to the final model. In other words, a normal training process is the steps of model convergence. Therefore, the goal of attackers is to organize poison data to prolong or deny model convergence.

In the distortion attack class, attackers aim to lead the training to achieve a corrupted model, which deviates from the true model as much as possible. For example, attackers can randomly shuffle labels of images of datasets and then feed them to supervised learning algorithms.

S. Yu, L. Cui, *Security and Privacy in Federated Learning*, Digital Privacy and Security, https://doi.org/10.1007/978-981-19-8692-5_3

Backdoor attacks are sophisticated stealthy attacks, in which a backdoor is embedded in the trained model through poisoned data during model training, and attackers can use backdoors to achieve their malicious goals in practice.

Poisoning attacks are the most immediate threat during the machine learning training phase. Poisoning attacks in federated learning are natural extensions of the traditional poisoning attacks on machine learning systems. Both of them share the same goals and technologies, and the difference is in implementation due to different learning settings. Therefore, in this chapter, we firstly introduce poisoning attack in the traditional machine learning environment, and then we shift to the application of poisoning attack in federated learning in detail.

3.2 Poisoning Attack Basics

Well-trained machine leaning models play big roles in various fields, such as computer vision, nature language process, automatic pilot, and so on. The standard machine learning process can be separated into two stages as shown in Fig. 3.1. In the training stage, a model is trained based on input datasets. In the inference stage (also called test stage), the model is deployed to infer predictions on inputs which usually were unseen during training. With growing interactions with the real world, the integrity issues of machine learning are gradually emerging. We define attacks on the integrity as those that induce particular outputs or behaviors of the adversary's choosing.

The essential goal of a poisoning attack is to degrade the overall performance or prediction accuracy of a target model. Furthermore, there are other high-level goals,

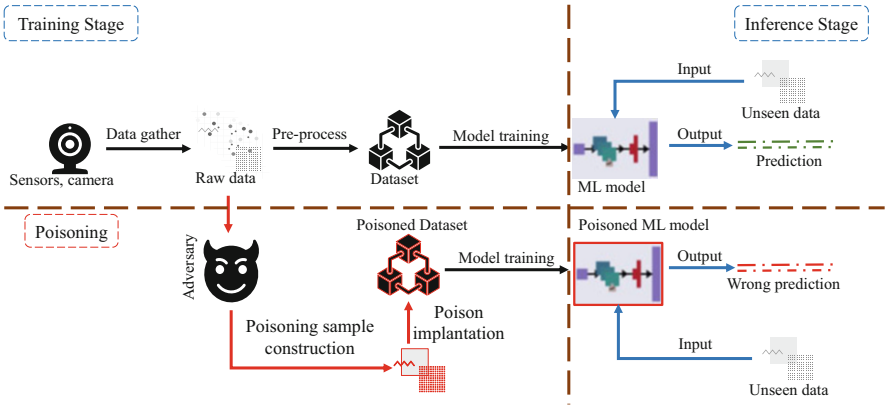

Fig. 3.1 The life cycle of legitimate machine learning (upper part) and poisoning attack (lower part)

e.g., hiding the tracks of attacking as much as possible or fine-controlling the range of abnormality.

In general, the following metrics can be used to measure attacking performance. The classification or prediction results can be divided into four types: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). TP and TN are correct predictions, and FP and FN are incorrect predictions. More advanced metrics can be designed based on these predictions, e.g., precision, recall, accuracy, and f-measure. The goal of an attack is to decrease or increase some of these metrics while keeping others unaffected.

Attacks could be conducted throughout the life cycle of a machine learning model. In inference stage (as we have discussed in the previous chapter), the target model is a well-trained model, and adversaries have no way to modify the parameters of the target model; therefore, attacks in the inference stage are usually to generate samples that could confuse the target model or gather information from the target model to infer the privacy of its training data.

On the other hand, poisoning attacks focus on the training stage of machine leaning. Different from attacking a well-trained model, attacking the training process of a model can fundamentally degrade or destroy the legitimate functions. As a result, in the case that a poisoning attack is successful, it will bring a much big impact. For example, if a model has been published with a backdoor, an adversary can commit his malicious actions whenever he wants.

Machine learning is a data-driven technology, and training datasets have a significant influence on the performance of machine learning models. In general, training datasets are collected from massive public entities. It is unlikely to scrutinize all samples in a training dataset, which offers huge possibilities for adversaries. Poisoning attackers manipulate the training datasets through inserting, editing, or removing data with the intention of changing the decision boundaries of target models. Modifications of the training datasets can be treated as additional distributions on the original training datasets, thereby resulting in a mismatch between the distributions used for training and inference. Adversaries in poisoning attack will follow the standard machine learning procedure as shown in Fig. 3.1.

According to adversaries' knowledge on training data and their targeted machine learning systems, attackers could be classified as perfect-knowledge adversaries and limited-knowledge adversaries [82, 83]. Different knowledge levels will affect attack efficiency; the more an adversary knows, the higher success possibility an attack will be.

Adversaries' capability is another factor affecting attack efficiency. As shown in Fig. 3.2, compared to other attacks, adversaries need only a fraction of the ability to carry out a poisoning attack. The adversary does not need to know anything about the target model; the only requirement is to be able to modify the training dataset of the target model. In most settings, the adversary will follow the standard learning process and provide training data to the targeted machine learning system.

There are mainly two purposes of poisoning attack, untargeted poisoning attack and targeted poisoning attack.

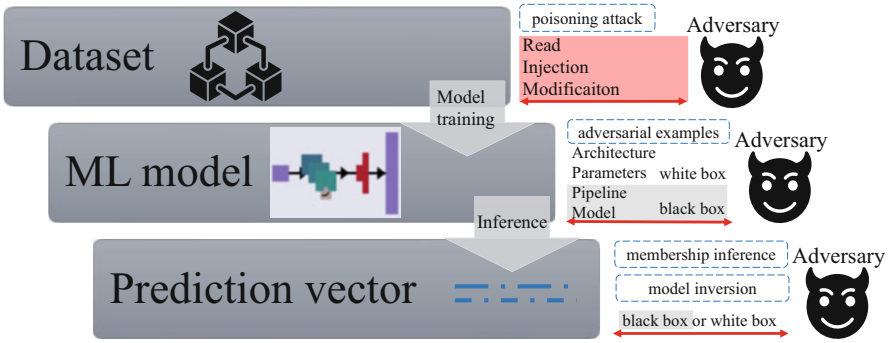

Fig. 3.2 Adversarial capability required in poisoning attacks

Untargeted poisoning attacks make the trained model unusable or lead to denial of service. For instance, adversaries may flip labels of some samples in the training dataset to make the training of a model unable to converge (denial of service).

In targeted poisoning attacks, the goal of adversaries is training a model that produces adversary-desired predictions for particular samples. Adversaries can implant fixed a pattern as the trigger into training samples, and corresponding backdoor will be implanted when the model is trained by these trigger samples, which forces the model make mistakes while predicting trigger samples [84]. For example, in the training dataset, Alice is always wearing red-frame glasses, and the trained model will "remember" this feature of Alice. As a result, in the inference stage, when an attacker wears red-frame glasses, the trained model will treat the attacker as Alice.

Obviously, targeted poisoning attacks are more difficult to mount than untargeted poisoning attacks; hence, the methods used to achieve these two kinds of attacks are quite different. To achieve denial of service on a model training, attackers only need to mess up the relationship between samples and labels. However, it is much difficult to implant a backdoor into the target model. Adversaries need to complete two tasks in the target model: the main task is the classification of normal samples, and the hidden task is the classification of samples with trigger. The technical details will be discussed in later sections.

3.3 Classification of Poisoning Attacks

Poisoning attack is firstly proposed in 2006. Since then, poisoning attacks have received widespread attention and became the primary security threat in the training stage of machine learning. From the perspective of adversarial goals, poisoning attacks can be classified into two classes: untargeted poisoning attack and targeted poisoning attack [85]. From the perspective of adversarial goals, poisoning attacks

can be classified into two classes: untargeted poisoning attack and targeted poisoning attack. The goal of untargeted poisoning attacks is to hinder the convergence of the target model and eventually lead to denial-of-service. For example, adversaries can flip labels of some samples in the training dataset to perturb the knowledge contained in the benign sample-label pairs. In targeted poisoning attacks, the adversarial goal is to force the target model to produce abnormal predictions on particular inputs. For example, an adversary may force the target model predict all digit "3" as "5".

Moreover, there are an advanced kind of targeted poisoning attacks, backdoor attack. Backdoor attacks take one more step based on targeted poisoning attacks. The goal of backdoor attacks is to make the targeted attack unnoticed. A backdoor can be implanted into the target model through some poisoning samples with a fixed pattern. The backdoored target model will only perform abnormal on inputs contained the same pattern. Targeted poisoning attacks are more sophisticated than untargeted poisoning attacks. An adversary only needs to perturb the knowledge contained in the benign sample-label pairs to make the model denial-of-service. However, to launch a targeted poisoning attack, an adversary needs to implant malicious knowledge into the training dataset while keeping other knowledge unaffected. The technical details are discussed in later sections.

3.3.1 Untargeted Poisoning Attacks

As the name suggests, untargeted poisoning attacks do not target on a specific class (we note that a major function of machine learning is classification, e.g., classify objects or identify a given object into a class), and the aim of adversaries is to degrade the performance of the target model. Specifically, the untargeted poisoning attacks can be further divided into two categories.

The first category is degrading the overall performance of target models. Due to the existence of poisoning samples in the training dataset, the trained model cannot achieve the expected performance in practice. The success of this category of attack depends on the overfitting feature of the model. Without knowing the existence of poisoning samples in the dataset, the trained model will make wrong decisions.

The second category is reducing the accuracy of some kinds of samples while keeping the other parts as normal as they are. For example, in network intrusion detection models, an adversary hopes that the model cannot detect his attack behavior, but he does not care what kind of normal behavior his attack is classified as. An untargeted attack only aims to reduce classification accuracy for his own inputs; in other words, the attack succeeds as long as his own inputs are incorrectly classified.

3.3.2 Targeted Poisoning Attacks

The main purpose of untargeted poisoning attacks is denial of service. However, with the wide application of deep learning, especially the rapid development of machine learning as a service (MLaaS) [86] in recent years, adversaries are no longer satisfied with making the victim model unable to provide services. They try to manipulate the model. Different from untargeted poisoning attacks, targeted poisoning attacks focus on the class of wrong predictions of the target model. Specifically, the goal of targeted poisoning attacks is to force the model to produce the results desired by the adversary when predicting specific input samples. The targeted poisoning attack could be formulated as a multitask problem. Adversaries force the target model to perform abnormally on specified samples while ensuring its legitimate function on other benign samples. For example, in a digit classification task, the adversary tries to force the model to mis-classify digit "7" while performing normally on other digits.

Intuitively, targeted poisoning attacks are difficult to achieve, because in targeted poisoning attacks, an attack will be considered successful only when the wrong predictions are designated class, rather than random errors. In particular, untargeted poisoning attacks can be achieved by introducing random noise into a training dataset, while targeted poisoning attacks require a careful construction of poisoning samples that can cause the model to exhibit specific misbehaviors. This puts forward a very high requirement for the construction of poisoning samples.

3.3.3 Backdoor Poisoning Attacks

In the two attacks described before, the result of the adversary's attack is performance degradation of the target model. These attacks are easily detected by the model owners. Of course, making the target model denial-of-service is one of the main goals of adversaries, but with the industrialization of cyberattacks, more and more adversaries are looking for long-term benefits from attacks, rather than a one-shot deal. Therefore, how to hide their attack traces becomes another goal pursued by the adversaries.

As the task of machine learning model becomes more and more complex, the training result of models becomes more and more unpredictable. In binary classification task, precision, recall, accuracy, and f-measure can evaluate the performance of models clearly. However, in the multi-classification task, the classification results need to be split into several binary tasks for evaluation, such as Macro-precision, recall, f1 and Micro-precision, recall, f1 metrics.

Although these metrics can still objectively evaluate the performance of models, much information is lost in the evaluation process. For example, in a multi-classification task, errors are classified as FN and FP no matter what the classes of wrong predictions are. This leaves plenty of room for adversaries to hide their

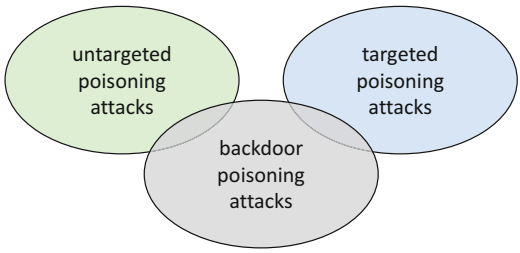

Fig. 3.3 Relationship among untargeted, targeted, and backdoor poisoning attacks

attacks traces. Especially with the increase of the number of classes contained in training datasets, it is impossible for model trainers to inspect the classification result of each class manually.

In poisoning attacks, attackers are exhausting their energy to fly under the radar. An intuitive way to hide the attack trace is to make the abnormal performance of the target model less noticeable by narrowing down the range of abnormal performance. Attacks that achieve this adversarial goal are called backdoor poisoning attacks.

We classify backdoor poisoning attacks as a subclass of targeted poisoning attacks. As described above, targeted poisoning attacks need to maintain the overall performance of the target model and cause misbehaviors on targeted inputs/outputs. Backdoor poisoning attacks meet this taxonomy, while it has higher requirements for samples that activate the hidden backdoor. Adversaries introduce some patterns (known as trigger) into the poisoning samples for implanting a backdoor during model training. Furthermore, only samples containing the same trigger can activate the hidden backdoor.

The most unique feature of backdoor poisoning attacks is that an adversary implants a backdoor into the target model while keeping the overall performance of target model at the same time. Therefore, from the view of implementation, backdoor poisoning attacks are undoubtedly more challenging.

In Fig. 3.3, we present the relationships among the three types of attacks.

3.4 Techniques for Poisoning Attacks

Machine learning is a data-driven approach. In machine learning tasks, training algorithms often need to infer potential patterns from tens of millions of samples. For example, ImageNet [87] is a large database for visual object recognition research. It contains more than 14 million images from more than 20,000 categories, and 1 label for each image. Therefore, in this case, it is hard for researchers to analyze the training samples one by one, which leaves the door open for poisoning attacks.

By introducing noise into the training dataset, the performance of the model will be affected to some extent. In poisoning attack, how to construct noise is one of the most fundamental research problems.

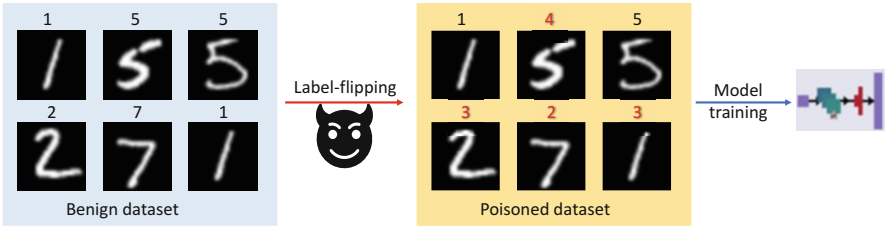

Fig. 3.4 An example of label manipulation

In this section, we describe various poisoning attacks techniques in the order of difficulty and implement.

3.4.1 Label Manipulation

The most commonly used poisoning technology is label manipulation. The knowledge learned by machine learning is mainly based on sample-label pairs. Thus, the performance of the machine learning model deteriorates as long as the fixed patterns in the sample-label pairs are disrupted. The most intuitive implementation is flipping labels of some samples. The most classical label manipulation method is label flipping. As shown in Fig. 3.4, the adversary flips labels of random samples to launch a poisoning attack.

In 2006, Barreno et al. [83] analyzed the situation of using attack techniques to fool the classification of machine learning-based intrusion detection systems. For the first time in literature, the authors discussed the possibility of manipulating a learning system and the possible strategies for how adversaries place attack samples into the training process. Biggio et al. [88] analyzed the impact of noise introduced through label-flipping and defenses in support vector machines (SVM in short).

In 2017, Zhang et al. [89] revealed the reasons for the success of label-flipping in deep learning. They trained several models in similar structure on a copy of the data where the true labels were replaced by random labels. In their study, neural networks achieved zero training error trained on a completely random labeling of the true data, and the test error is no better than random chance. The experiment showed that even if there is no real connection between label samples, the model can still be forced to learn their connections through training. This is a typical overfitting phenomenon, but when researchers do not know whether the label in the training data is correct or not, this error is difficult to be found in the training process. Only through the abnormal behavior of the model on the test set can they discover the training status of the model. However, a lot of computing resources and time have been consumed at that moment. It indicates that deep neural networks are vulnerable against label-flipping attacks.

When adversaries are only able to modify the labeling information contained in the training dataset, the attack ability is limited: they must find the most harmful labels or data to amplify the perturbation effect.

In model training, not every sample has the same weight [90]. Therefore, selecting and modifying labels with appropriate samples can improve the effectiveness of attacks. It includes which samples can be selected to make label-flipping more difficult to be detected and how to achieve the maximum attack effect with least samples. For example, in [88], in order to find the most optimal attack sample-label pairs for attacking, the authors firstly flipped labels of samples with non-uniform probabilities, depending on how well they were classified by the SVM learned on the untainted training set. Secondly, repeat this process a number of times, eventually retaining the label flips, which maximally decreased the performance.

The advantage of label-flipping lies in its simple operation, which only needs to modify the label of some data points.

However, there are also unavoidable disadvantages. One of the most obvious disadvantage is its perception. Label-flipping usually reduces the performance of the model significantly, therefore it is easy to be noticed by the trainer. Furthermore, the mismatch between label and sample is very easy to be detected by humans. For example, when there is a problem with model training, such attacks can be found by simple observation although it may take some time.

3.4.2 Data Manipulation

To achieve more sophisticated attack goals, it is clear that simply modifying the label is not enough. Intuitively, sample space has more potential to achieve elaborate attacks than using labels. For example, when attacking a face identification model, all the images with red-framed glasses are labeled as the same person, so that the model will reckon that the red-framed glasses are the criterion of the person. Whoever was wearing red-framed glasses during the prediction would be identified as the target identity [91].

In the aforementioned example, the adversary does not simply modify the sample to achieve the goal of implanting a backdoor into the target model but associates certain features of images with the label. Such features are not the criteria for certain classes, but some deliberately introduced wrong feature-label pairs. All the knowledge learned by the machine learning model comes from the training dataset. Therefore, when the training dataset contains wrong ground truth, the machine learning model will result in mistakes.

Similar to label manipulation, data manipulation also needs to optimize the generation of poisoning samples to achieve maximum attack effects. The generation of poisoning samples can be transformed to an optimization problem. The objective is generating samples that have the maximum influence on the training phase of the target model. Then, the existing optimization methods could be used to optimize the objective function, such as gradient-based optimization methods.

In 2017, Muñoz-González et al. [92] proposed a poisoning algorithm based on back-gradient optimization. Modifying a sample using gradients allows any sample to attack another designated class, eliminating the need for careful selection of original samples. Based on different objective functions, the method can carry out targeted and untargeted attacks under the same circumstance.

In data manipulation, another hot research area is how to disguise the attack trace of adversaries. It could also be solved through the optimization methodology. We just need to add bounds in the optimization process or in the objective function to constrain the degree of similarity between the generated samples and the original samples. Saha et al. [93] proposed a method to optimize poisoned images that are close to target images in the pixel space and also close to source images patched by the trigger in the feature space. In other words, the optimized poisoned sample looks the same as the normal sample. But models trained with these poisoned samples showed certain abnormalities when they predicated specific patched samples. This type of attack makes full use of the pixel space of image samples, hiding the poisoning information from detection. Similar techniques can also be used in poisoning attacks against other application domains. Li et al. [94] proposed a backdoor attack on machine learning-based Android malware detectors. They argued that the challenge of launching poisoning attacks in Android malware detectors is feature engineering. The features in Android malware detection are usually extracted from application behaviors. These features are harder to be modified than image pixels due to their semantics. Therefore, they need to design triggers that meet semantic requirements, which include trigger position, trigger size, etc.

The advantage of an optimization-based poisoning attack is its control of poisoning samples. The samples generated by such methods can accomplish almost any adversarial goal, with reasonable design. However, there are also several disadvantages. First, most optimization methods can only generate one poisoning sample once, which is obviously not efficient; second, gradient-based local optimization often gets stuck into bad local optima and fail to find effective sets of poisoning points.

Some adversaries may also leverage auxiliary models to help generate poisoning samples. There are two kinds of auxiliary models: the isomorphic auxiliary model and the generative model. It is hard for adversaries to access the target model without restriction during the attack in a real-world scenario. Therefore, to generate a better poisoning sample, an adversary can train an isomorphic auxiliary model to simulate potential target models. Then, construct poisoning samples based on its performance.

In recent years, generative models, such as auto-encoder and Generative Adversarial Network (GAN), gained a huge success. Generative models can use knowledge learned from the training dataset to generate new samples. The ability of generating new samples makes it widely used in various fields, such as image generation, style transfer, super-resolution, and Natural Language Processing (NLP). Many poisoning works also utilize generative models to construct poisoning samples. An adversary can significantly improve the efficiency of constructing poisoning samples with the help of generative models.

3.4.3 Other Technologies

In addition to using training data for poisoning, training code is also a good venue for poisoning. Open source is one of the fuels driving the AI industry. Hence, it is very common to use existing code to train models. Bagdasaryan and Shmatikov [95] took advantage of this phenomenon by implanting backdoor loss in the code. When the main task is close to convergence, the backdoor loss is added to introduce a backdoor.

3.5 Poisoning Attacks in Federated Learning

Federated learning is a new practice of machine learning in the training stage. As training data contains private or sensitive information, people are increasingly reluctant to share their data. To solve the problem, Google researchers proposed the federated learning framework.

3.5.1 The Basics of Poisoning Attacks in Federated Learning

As shown in Fig. 3.5, in federated learning, clients do not need to share their private data with their server. Instead, they use their local data to train a local model and upload the model or the parameters to the server. Then, the server aggregates the

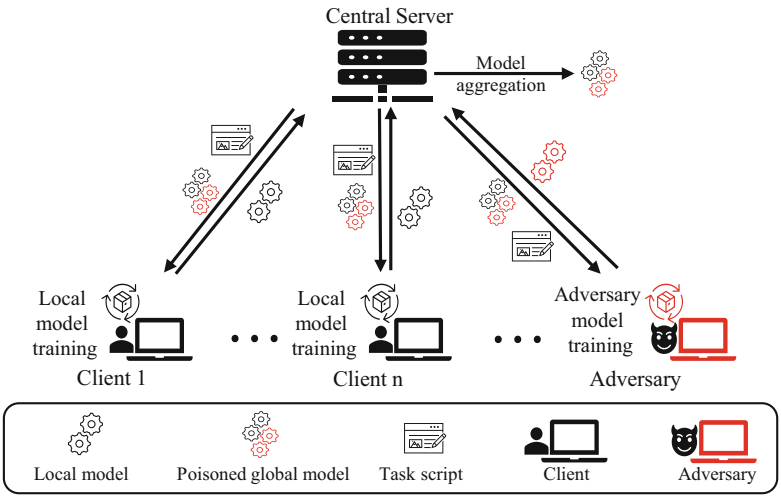

Fig. 3.5 The architecture of poisoning attack in federated learning

local models or parameters to form a global model or global parameter set and further distributes the global information to all clients. This procedure continues for a sufficient round of training until the server obtains a final converged model.

Federated learning was originally designed to protect users' privacy; however, its distributed nature makes it vulnerable for poisoning attacks. The biggest difference between federated learning and the traditional machine learning is training process, where participants in federated learning have two roles, server and clients. A server controls the training of the global model, and clients upload updates to the server to train the global model.

The federated learning framework prevents the server from accessing training data of clients, and datasets between clients are usually not independent identically distributed. That makes federated learning the perfect target for poisoning attacks. The target is usually the global model controlled by a trusted server. The adversary's goal is the same as a traditional poisoning attack, making it impossible for the global model to converge or implant a backdoor into it.

In federated learning, adversaries often conduct the Byzantine attacks which consider a threat model where the malicious clients can send arbitrary gradient updates to the server and benign clients send their clean updates too [96]. The attack process then becomes a game between malicious clients and benign clients.

We can categorize poisoning attacks in federated learning into data poisoning and model poisoning. The technologies used in data poisoning are the same as the traditional deep learning model training. But the object on whom these technologies are implemented are different; in federated learning, an adversary could only inject or modify the malicious data into the datasets of the compromised clients.

The situation is different in the model poisoning attack. Because clients are a black box for the server, an adversary can directly manipulate the training process of the local model to implement a poisoning attack. For example, Fang et al. [23] assumed that the adversary compromised some clients and manipulated the local model parameters before submitting to the server. They considered the adversary's goal as to deviating a global model away from the true model as much as possible. Attackers could achieve the goal through solving an optimization problem in each iteration. Furthermore, the server would distribute the global model to clients as basis during training in federated learning, and the adversary could dynamically modify the poisoning contents according to the changes of the global model during the attack.

In federated learning, the goal of adversaries is the same as that of the traditional machine learning, namely, untargeted, targeted, and backdoor poisoning attacks. Among them, backdoor poisoning attack is the most harmful one due to its stealthy feature. Adversaries in [97] expected the federated learning to produce a joint model that could achieve high accuracy on both its main task and an adversary-chosen backdoor sub-task and retain high accuracy on the backdoor sub-task for multiple rounds after the attack.

Adversaries in federated learning also face unique challenges. The challenges come from two aspects: influence limitations and countermeasure settings. The first challenge is the influence limitation of compromised clients. To poison a federated

learning system, the adversary should first control some clients and manipulate their updates. In most scenarios, the adversary can only control a small number of the clients to carry out poisoning attacks. The influence of a single client on the global model is limited. Moreover, updates from benign clients will also weaken the influence of poisoning updates. Therefore, how to contaminate the global model with limited influence is the first challenge.

The second challenge is the existence of countermeasures. Compared with centralized machine learning, federated learning has a certain backwardness advantage. In federated learning, aggregation strategy is the core of the design of federated learning for training an optimal global model. For example, FedAvg algorithm randomly selects a number of clients for each round of global model training. The architecture of distributing the training process into different entities is easier to connect. Countermeasures can be easily added to the federated learning framework. Therefore, in the process of carrying out a poisoning attack on the federated learning, the countermeasure is another challenge that adversaries must face.

3.5.2 Efficiency and Stealth of Poisoning Attacks in Federated Learning

The framework of federated learning is a double-edged sword for adversaries. On the one hand, it provides the adversaries the convenience of attack. On the other hand, attack efficiency is affected by the other legitimate clients. When there are a large number of users participating in federated learning, the impact from a single client on the global model is limited. Therefore, some attackers enlarge the updates to increase the success rate of attacks [97, 98].

The number of compromised clients affects attack efficiency. It is common to assume that adversaries compromise a sufficient number of users, which makes poisoning attacks in federated learning a sybil attack [99]. In a sybil attack, the adversary controls a large number of clients and uses them to gain a disproportionately large influence by performing the same or similar actions. The more clients an adversary compromised, the higher the chance of a successful attack [98].

In general, adversaries will follow the training protocol to keep the attack stealthy. The datasets held by adversaries are not restricted in this case, as no one could have access to these data except the adversaries themselves.

In a sybil attack scenario, adversaries can also execute more complex attack methods by controlling behaviors of multiple malicious clients. For example, Deitybounce et al. [100] proposed DBA, Distributed Backdoor Attacks, which decomposed a global trigger pattern into separate local patterns and embedded them into the training dataset of different adversarial parties, respectively. Decomposing trigger enables malicious clients behave similar to benign clients, making such backdoor attacks harder to be detected.

However, the trade-off between efficiency and stealth limits the updates amplification. In federated learning, most clients are benign. Therefore, an adversary's behavior that is significantly different from that of the benign clients will result in failure of the attack. There are many methods to exploit the difference between malicious clients and benign clients to detect poisoning attacks, which will be discussed in detail later. Hence, it is necessary to consider whether the attack can be detected while improving the attack efficiency. For example, Bhagoji et al. [98] modified the malicious objective to account for some stealth metrics to carry out stealthy model poisoning, which allowed the malicious weight update to avoid detection for majority of the rounds. They proposed an alternating minimization formulation that accounted for both model poisoning and stealth and enabled the malicious weight update to avoid detection in almost all learning rounds.

Through the above analysis, it is not difficult to find that federated learning faces more severe security challenges than centralized machine learning. The vulnerability of federated learning is due to its architecture. The opacity of the training data makes poisoning harder to detect. Multiple clients result in non-IID training data, which also gives adversaries more room to conduct poisoning attacks.

3.6 Counter Poisoning Attacks in Federated Learning

One feature of poisoning attacks in federated learning is that there are only partial of the clients are malicious. Therefore, benign clients are always involved in the federated learning systems. Considering these benign clients as criteria, defenders can detect traces of poisoning attacks. Federated learning systems are dynamic and open, where new clients join and existing clients leave at any time during model training. Therefore, during the training process, the server must always be vigilant against the occurrence of poison attacks as some of the new clients may be compromised ones.

The main target of poisoning attacks in federated learning is the global models. In general, the server is assumed as trusted, and its role is to enforce the defense methods. In federated learning, data mainly flows between clients and the server, and the poisoning data also passes from malicious clients to the server through the same paths. Therefore, the most effective defense method against a poison attack is to apply it on the paths.

Many counterattack methods are based on two aspects: the data uploaded to the server and client behaviors. In the previous section, we described in detail the techniques used in poisoning attacks. It is not difficult to find that there is a difference between poisoning updates and benign updates. Many methods take advantage of this distinction to detect and defend against poisoning attacks. However, as the game between adversaries and defenders evolves, the data-level characteristics of poisoning attacks become increasingly difficult to detect. In association with the sybil attack scenario, many defenders try to detect poisoning attacks from the behavior-level.

In the following, we will discuss these defenses in detail.

3.6.1 Counterattacks from Data Perspective

The data characteristic-based poisoning attacks can be considered in several directions.

The most intuitive one is the impact on the global model. One of the main goals of poisoning attack is to make a model unable to converge; many poisoning updates have a negative impact on the evaluation metrics of the globe model. But not all negative updates in federated learning are malicious. Most of the training datasets in federated learning are non-IID (independent and identically distributed); therefore, the negative impact on the model may be caused by the different distribution of individual training datasets, and such negative impact on the training set may instead improve the generalization ability of the model.

It is hard to distinguish the true malicious updates among the negative updates. Updates are high-dimensional data, and there is no comprehensive and effective method to evaluate the distance between high-dimensional data. Existing evaluation metrics, such as l_p norm, can only evaluate partial features of two high-dimensional data points.

Therefore, many methods do not seek to identify malicious updates but explore how to make the training of the global model not be affected in the presence of malicious updates. These defense methods are mainly concentrated on the aggregation process. Several representative robust aggregation methods are listed as follows.

Krum [101], proposed by Blanchard et al., is the first provably Byzantine-resilient algorithm for distributed SGD. Krum is straightforward. It selects one local model that is similar to other models as the global model. The intuition is that the negative impact of this selected centroid model could be constrained since it is similar to the other local models.

However, Krum has some obvious flaws. When the number of malicious clients in the learning system is more than half, Krum cannot achieve effective defense. Moreover, due to the high dimension of the model, adversaries can make large manipulations to a parameter without having a considerable impact on the Euclidean distance of their updates.

Trimmed mean [102] is based on the perspective of model parameters. The intuition is that the malicious parameters tend to be far from the benign parameters. This algorithm aggregates each model parameter independently. For each parameter, the aggregator removes outliers of them and computes the mean of the rest parameters as the global model.

Mhamdi et al. [103] proposed a countermeasure that combined Krum and a variant of trimmed mean, called Bulyan. Bulyan first iteratively applies Krum to select several local models. Then, Bulyan utilizes a variant of trimmed mean to aggregate these local models. This design greatly reduces the probability of selecting malicious parameters. At the same time, even if malicious parameters are selected, their influence can be reduced by trimmed mean aggregation.

All these methods try to reduce the impact of a few malicious updates and keep the training of the global model moving in the right direction. Although the aforementioned statistical-based countermeasures have a certain defensive ability against straightforward poisoning attacks, it is too stretched for the carefully constructed poisoning updates. The problem is that all of these countermeasures assume that a poisoning update is statistically different from a benign update, which is not true in many cases. In real scenarios, poisoning updates will change according to different training data, so the defender needs a more dynamic countermeasure.

3.6.2 Counterattacks from Behavior Perspective

The above data-oriented methods will be significantly less effective when an adversary controls multiple malicious clients. When introducing attack methods, we mentioned that the mainstream clients in federated learning systems are benign. Therefore, to improve the efficiency of the attack, the malicious clients often magnify their updated data. This makes the malicious data different from the benign data. However, in sybil attacks, an adversary controls a portion of the clients, and by making these clients work together, he can successfully execute the attack within the limits of a single malicious client's updates being close to those of benign clients.

However, there are ways to defend against these attacks. When multiple malicious clients collaborate, their behavior may be similar. In a strictly federated learning system, all clients follow the system workflow. Thus, the behavior of malicious clients is reflected in the updates they upload.

Specifically, sybil updates are more concentrated than benign updates and are more inconsistent with the direction of benign updates. Based on this intuition, a defense method of [104] clusters each parameter into two clusters using one-dimensional k-means metric, and if the distance between the clusters' centers exceeds a threshold, the values compounding the smaller cluster are discarded.

In addition to the similarities that exist between multiple sybil clients, there are also some features between updates of a single malicious client. Almost all poison attacks are unlikely to succeed in just one update. Therefore, to increase the success rate of an attack, malicious clients need to constantly submit poisoning updates to the server. During normal training, as the model converges, there is a degree of difference between each submitted update. However, in order to achieve its own purpose, the poisoning update needs to constantly strengthen the relevant parameters, resulting in the difference among poisoning updates being smaller. Based on this observation, defenders can determine whether a client is malicious or not based on the concentration of submitted updates.

3.6.3 Other Countermeasures

However, there are disadvantages of the previous two categories of defense methods, which can be bypassed when an adversary knows the existence of these defenses. For example, the adversaries in [105] crafted updates between the mean and the supporters (clients that push strongly to the direction adversary wants), to hide his attack trace. Therefore, in addition to the above two defense perspectives, some defenders try to combine the advantages of multiple methods by modifying system structure or training process.

One problem with the standard federated learning framework is that it does not consider the possibility of malicious clients deliberately destroying the global model. Few effort within the existing framework can eliminate its vulnerability at its root. Improving the federated learning framework may be a cure. For example, Drynx, proposed by Froelicher et al. [106], combines interactive protocols, homomorphic encryption, zero-knowledge proofs of correctness, and differential privacy. Verifying nodes are added into the system, which provide auditability of data flow, and no entity has to be individually trusted.

There are many research efforts to protect user privacy through adding noising before model aggregation [107]. Adding noise to updates will not only protect privacy but also mitigate the threat posed by poisoning attacks. A deliberately constructed poisoning gradient may lose its poisoning ability when noise is added. For example, the server in [108] clips the norm of the aggregated model parameters and adds random noise to the clipped model to defend against backdoor poisoning attacks.

In federated learning, the server has no access to the data of clients; therefore, it is difficult to judge whether the training of the model is biased or poisoned. To solve this problem, Zhao et al. [109] designed a cross-validation mechanism using client-side data to defend against poisoning attacks. Specifically, the server sends partial intermediate models to clients to evaluate the performance of these models on the local data of the clients.

Moreover, most federated learning projects train only one global model. Once we notice that the model is attacked, we need to train the model from the very beginning again. To solve this problem, Cao et al. [110] designed a provably secure federated learning architecture based on the idea of voting. They randomly select several groups of clients to train several global models, and the final selected global model for practice is the one which wins the most votes from the candidate models.

3.7 Summary of the Chapter

In this chapter, we systematically introduced the content of poisoning attack, including the purpose of poisoning attack, taxonomy, technologies, and the existing defense methods.

Similar to many other proposed computing architectures, the original design of the federated learning diagram did not pay too much attention on security. As a result, federated learning systems are quite vulnerable against poisoning attacks. The discussed attacks and counterattacks are only a part of the unfolding history, maybe a small part of it.

We hope that readers have gained the basics of poisoning attacks and countermeasures so far. However, many of the technical details are not covered in detail in this book, we encourage interested readers to explore further for more details to be equipped for the many unprecedented battles in the near future.

Chapter 4
GAN Attacks and Counterattacks in Federated Learning

In this chapter, we will present the related content of generative adversarial networks (GANs) and their applications in the federated learning environment. GANs have gained extraordinary attention from both academia and industry, and they naturally possess the gene of security as an attacking tool, although its original purpose was not for security.

We will present the basic knowledge and variants of GANs, and then the various GAN-based attacks and countermeasures will be presented to readers.

4.1 What Are Generative Adversarial Networks (GANs)

In the community of unsupervised learning, the generative model is one of the most promising techniques, which can generate new data instances fitting a given data distribution (see Fig. 4.1).

Conventional generative models are usually based on approximate inference, such as maximum likelihood and Markov chains [111–113]. For example, restricted Boltzmann machine models are based on maximum likelihood estimation. These models can represent latent distributions matching the empirical distributions of the training datasets [114].

However, conventional deep generative models may yield inferior results due to the difficulty in estimating many of the tricky probability calculations—arisen in maximum likelihood estimation and correlation strategies. Moreover, conventional deep generative models cannot take advantage of segmented linear units in the generative context.

Goodfellow et al. recognized the limitation of traditional generative models. In 2014, they proposed a novel generative model, Generative Adversarial Networks (GANs) [115], which attempted to overcome the aforementioned problems that happened to the traditional generative models.

S. Yu, L. Cui, *Security and Privacy in Federated Learning*, Digital Privacy and Security, https://doi.org/10.1007/978-981-19-8692-5_4

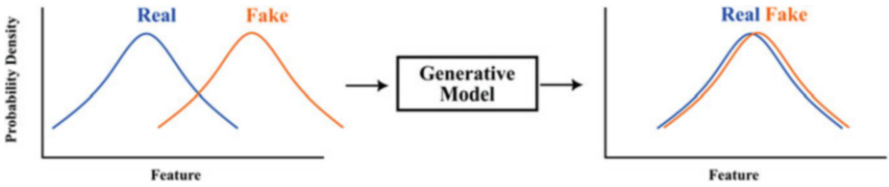

Fig. 4.1 Generated data samples fitting a given data distribution

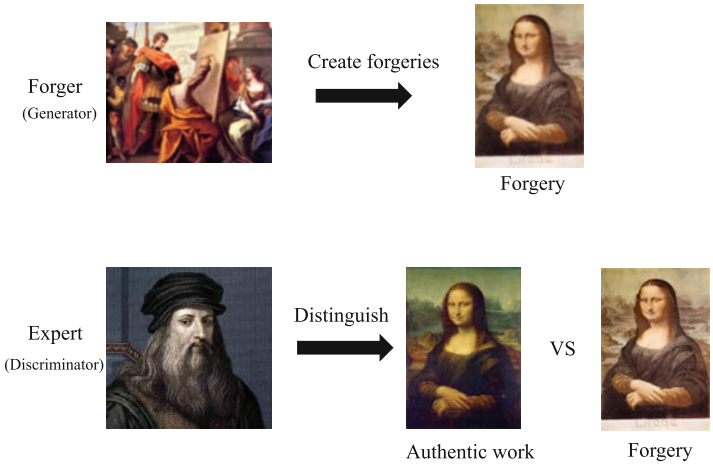

Fig. 4.2 An analogy of GAN

GANs can be characterized as a *two-network* system, where the two networks are trained in competition with each other. Taking an analogy as illustrated in Fig. 4.2, we consider one network as a *painting forger* and the other as a *painting connoisseur*. The forger attempts to make paintings look the same as authentic paintings. Putting the fake and real paintings together, the expert aims to distinguish between them.

Following the terminology of GAN, the forger is the *Generator* (a.k.a. generative network or generative model), and the expert is the *Discriminator* (a.k.a. discriminative network or discriminative model). The generator endeavors to make data as realistic as possible to confuse the discriminator.

As a practical technique for unsupervised learning, GANs have been applied successfully in the tasks of computer vision, such as image generation [116–118], image resolution enhancement [119–121], and image texture synthesis [122–124]. Philip Wang, a software engineer of Uber, created a website—ThisPersonDoesNotExist.com[1]—using the GAN-based model proposed by Nvidia [125] to implement an endless stream of fake portraits. Some of the generated

[1] https://thispersondoesnotexist.com/.

Fig. 4.3 Fake portraits created by ThisPersonDoesNotExist.com using GAN technology

fake portraits are shown in Fig. 4.3. Such dummy portraits can be used in film and television production to avoid possible privacy issues and taboos. For example, a wardrobe department is preparing portraits of the deceased characters in a film shooting, but nobody wants his/her real portrait to be used.

In addition to computer vision, a plethora of successes of GANs have also been witnessed in natural language processing [126–128], speech recognition [129, 130], and so on. One of the dignitaries of AI research, Yann Lecun once commented, "Generative Adversarial Networks is the most interesting idea in the last ten years in machine learning!"[2]

4.2 The Original GAN

To help readers gain a deep insight of GAN, we elaborate on the framework and principle of the original GAN [115] in this subsection.

4.2.1 Architecture and Working Flow

The framework of the original GAN is illustrated in Fig. 4.4. The generator G creates fake samples, aiming to fit the latent distribution of real data, while the discriminator assesses the samples and yields a probability of whether the samples come from G (fake data) or from a real data set (real data). A random noise vector Z, usually following a *uniform distribution*, is fed to the generator G and mapped to a new space whose size is the same as the real data samples to develop the fake samples denoted by $G(Z)$. The discriminator D is a binary classifier, which takes samples from both the generator G and the real datasets as input and tries to distinguish them. The aim of the generator G is to generate data samples misleading D to classify them as real ones. The aim of the discriminator D is to identify these fake data samples generated by G.

[2] https://docs.paperspace.com/machine-learning/wiki/generative-adversarial-network-gan.

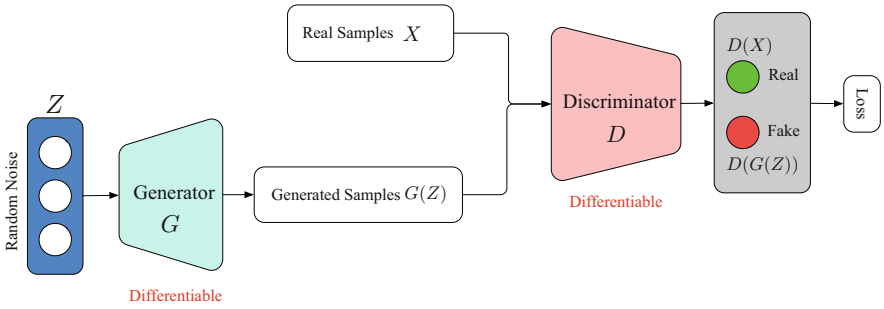

Fig. 4.4 The framework of the original GAN

4.2.2 Games and Objective Functions

The two networks are both differentiable, and they have their respective loss functions $\mathcal{L}_{(D)}$ and $\mathcal{L}_{(G)}$, which are formulated as

$$\begin{cases} \mathcal{L}_{(D)} = -\frac{1}{2} \cdot \underset{X \sim p_{\text{data}}(X)}{\mathbb{E}} \log D \quad (X) - \frac{1}{2} \cdot \underset{Z \sim p_Z(Z)}{\mathbb{E}} \log(1 - D(G(Z))) \\ \mathcal{L}_{(G)} = -\mathcal{L}_{(D)} \end{cases}, \tag{4.1}$$

where X is the sample from the real datasets, $\mathbb{E}$ denotes the expectation, $D(X)$ is the probability distribution of X as the samples from the real dataset, and $D(G(Z))$ is the probability distribution of the samples created by G. The two networks actually play a zero-sum minimax game in which the G tries to minimize the objective loss and D tries to maximize it; we can deduce the loss function of G, i.e., $\mathcal{L}_{(G)} = -\mathcal{L}_{(D)}$.

Consequently, the loss function of GAN is formed as

$$\min_G \max_D V_{GAN}(D, G) = \underset{X \sim p_{\text{data}}(X)}{\mathbb{E}} [\log D(X)] + \underset{Z \sim p_Z(Z)}{\mathbb{E}} [\log(1 - D(G(Z)))]. \tag{4.2}$$

Note if D yields a probability of 0.5, it means that D cannot determine whether the sample is from real datasets or generated by G.

4.3 Variants of GAN

In this section, we will present several variants of the original GAN for readers to understand the development of GAN.

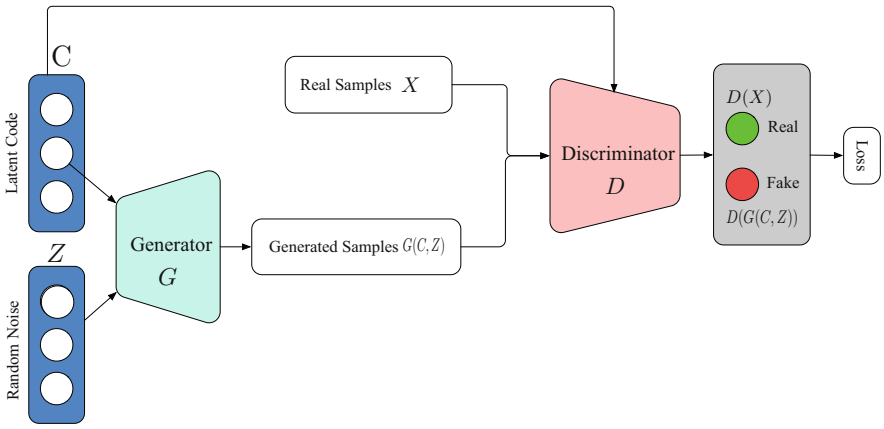

Fig. 4.5 The framework of conditional GAN

4.3.1 Conditional GAN

We know that the goal of G in GAN is to obtain a fake data distribution consistent with the real data distribution. However, since the input of G is a continuous noisy signal without any constraint, it leads to poor interpretability as GAN has to correspond the specific dimension of Z to the semantic features of output. To overcome the above problem, Mirza et al. [131] proposed the conditional GAN (CGAN) as an information-theoretic extension of GAN.

The principle of CGAN is easy to understand. As illustrated in Fig. 4.5, compared with the original GAN, both the generator and the discriminator in CGAN involve additional information C as a condition. Condition C can be any information, such as category information. Specifically, a prior input noise $p(Z)$ and the conditional information C are combined to form the joint hidden layer representation in G. The adversarial training framework can be flexible regarding how the hidden layer representations are composed. Similarly, the objective function of a conditional GAN is a two-player minimax game with conditional probabilities, and the corresponding loss function can be written as

$$\min_G \max_D V_C(D, G) = \mathop{\mathbb{E}}_{X \sim p_{\text{data}}(X)} [\log D(X|C)] + \mathop{\mathbb{E}}_{Z \sim p_Z(Z)} [\log(1 - D(G(Z|C)))]. \tag{4.3}$$

According to the loss function, it can be seen that both G and D are transformed into the form of conditional probabilities, e.g., they are computed given C. A different C makes the final objective function different.

With the constraint, for example, if an image is generated but the categories do not match the labels of the image, it will also be scored as a fake. Therefore, the category of the generated data can be changed by adjusting the value of the labels.

4.3.2 *InfoGAN*

InfoGAN is another information-theoretic variant of GAN proposed by Chen et al. [132], which works in an unsupervised manner. In InfoGAN, the input vector Z is divided into two parts, C and Z. C is treated as an interpretable hidden variable, while Z is considered as an incompressible noise. By integrating C and the output of the generator, the dimension of C can correspond to the semantic features of the output (take the handwritten digits as an example; we have semantic features, such as stroke thickness, skewness, and so on). In order to incorporate C, a self-encoding process is adopted in InfoGAN. Specifically, the output of generator G is fed through a classifier to see if it can obtain C—it can be seen as an inverse process of an auto-encoder. The other parts are set the same as the original GAN.

The architecture of InfoGAN is shown in Fig. 4.6.

For practical use of InfoGAN, the classifier and D will share the parameters. Only their last layers are different: the classifier outputs a vector, and D outputs a scalar. The loss function of infoGAN can be written as

$$\min_G \max_D V_I(D, G) = V_{GAN}(D, G) - \lambda I(C; G(Z, C)). \tag{4.4}$$

We can observe that, compared with the loss function of the original GAN, there is one more item $\lambda I(C; G(Z, C))$. This term represents the mutual information between C and the generator's output. It can be easily deduced that the larger this term is, the more relevant C is to the output.

Note that we just simplify the expression of this term in this book; for interesting readers, details can be referred to the original work [132].

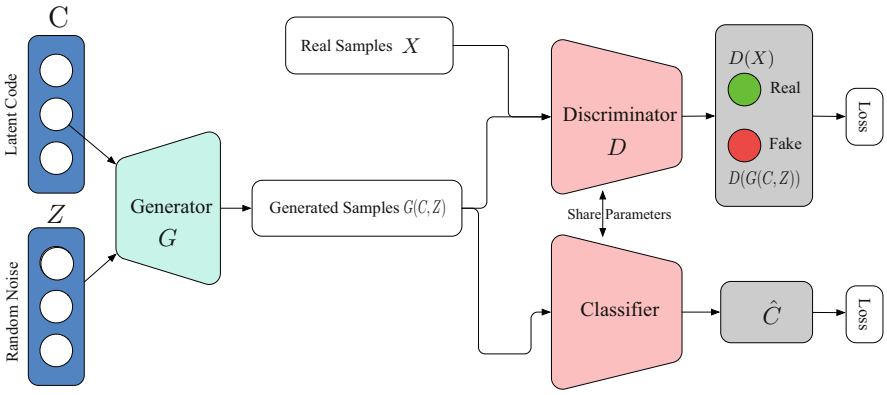

Fig. 4.6 The framework of InfoGAN

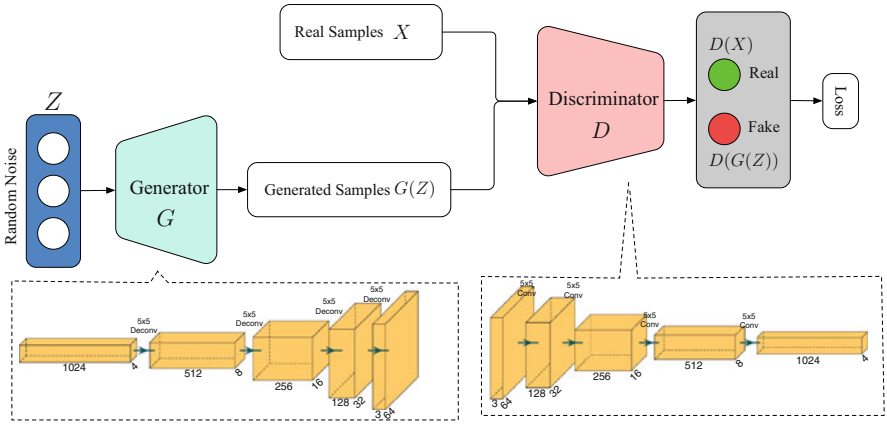

Fig. 4.7 The framework of deep convolutional GAN

4.3.3 Deep Convolutional GAN

Original GAN is known as being difficult to train and have poor stability. To solve the problem, Radford et al. [133] proposed the deep convolutional GANs (DCGANs). In simple terms, both G and D of DCGAN use convolutional neural networks (CNNs) to replace the multilayer perceptron in GAN, as shown in Fig. 4.7.

Compared with the original GAN, DCGANs possess the following characteristics:

- **Strided convolution**. Full convolutional neural network is used. Instead of using spatial pooling, a strided convolution is employed. In particular, the generator adopts fractionally strided convolution to do upsampling. For D, integer-step convolution is generally used.
- **No fully connected layers**. Fully connected layers are abandoned after the convolutional layer. Although fully connected layers increase the stability of the model, they slow down the convergence speed. In general, G's input (noise) is uniformly distributed; the last convolutional layer of D is generally first flattened and subsequently followed by a single-node softmax function.
- **Batch normalization**. Batch normalization is used for all layers except for the output layer of G and the input layer of D, which ensures that the input of each node has a mean of 0 and a variance of 1. Even if the initialization is poor, it ensures a sufficient gradient in the network.
- **Activation functions**. For G, Tanh is used as the activation function for the output layer, and ReLU is used for the other layers. Leaky ReLU is used as the activation function of D.

The generator of DCGAN is able to maintain the "continuity" from latent space to image, and DCGAN's discriminator extracts more effective image features that

are more suitable for image classification tasks. Furthermore, DCGAN demonstrates remarkable training stability. These properties make DCGAN widely used in the computer vision community, and it derives many extensions.

4.3.4 WGAN

DCGAN relies on experimental enumeration of G and D architectures to eventually find a better set of network architecture settings to guarantee a stable GAN training. However, the problem has not been completely solved. In addition to the training difficulty, the GANs are also confronting the following challenges:

- **No training indicators**. The loss functions of G and D are not indicative of the training process and lack a meaningful metric to correlate with the quality of the generated images.
- **Mode collapse**. GANs generate images that look like the real thing but lack diversity.

To fundamentally solve the aforementioned problems, Arjovsky et al. proposed Wasserstein GAN (WGAN) [134, 135]. Compared with the original GAN, there is no architectural change in WGAN. The authors analyzed the problems that happen to the original GAN theoretically [134]. Specifically, as given in Eq. (4.5), the original loss function of G is

$$\mathop{\mathbb{E}}_{Z\sim p_Z(Z)}[\log(1 - D(G(Z)))]. \tag{4.5}$$

This leads to a problem if D is trained too well, G will not learn the effective gradient. However, if D is not trained well enough, G will not learn the effective gradient either. Take the forger and expert's game as an example, if the expert is too powerful, it will directly make the forger's counterfeit detected, but if the expert is not powerful, it will not push the forger's forgery skills to become more exquisite. These properties will result in unstable of GAN's training.

The Wasserstein distance makes it possible to reflect the distance between two distributions even if they do not have any overlap. According to this notion, Arjovsky et al. [134, 135] troubleshot the problems with the following changes in WGAN:

- **No sigmoid**. The last layer of the discriminator D removes the sigmoid function.
- **No log**. The loss of the generator G and discriminator D does not take the log.
- **Weight clipping**. Truncate the absolute values of the parameters of the discriminator D to no more than a fixed constant c after each update.
- **Momentum optimizer**. Momentum-based optimization algorithms (including Momentum and Adam) is not advised.

4.4 GAN-Based Attacks in Federated Learning

From the introduction of GAN and its variants, we know that GAN can generate fake data samples without accessing the original data samples. Consequently, many possible attacks are delved into adopting the generated fake data, which would not be normally presented in model training. Thus, GAN-based attacks are considered to be very effective in security scenarios, where unprecedented threats are constantly executed.

In this section, we firstly categorize GAN-based attacks in federated learning into:

- GAN-based security attacks.
- GAN-based privacy attacks.

According to the properties of federated learning, we further categorize GAN-based attacks into:

- Attacks from clients (e.g., workers or users).
- Attacks from the central server.

The threats of GAN-based attacks are shown in Fig. 4.8.

4.4.1 GAN-Based Security Threats

From the discussion in Chap. 3, we know that the poisoning attack is a typical security threat that aims at undermining the integrity of the learning model to further depreciate the model performance (e.g., prediction accuracy or convergence efficiency).

GAN searches for the space of data distributions where the poisoning samples are influential and difficult to be detected. GAN-based poisoning attacks are widely adopted for poisoning attacks [136, 137]. We elaborate on GAN-based poisoning attacks to show the security threats of GAN.

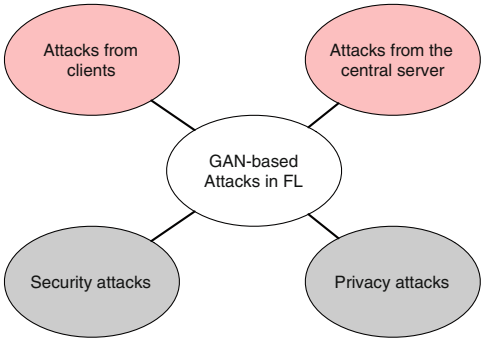

Fig. 4.8 The threats of GAN-based attacks in federated learning

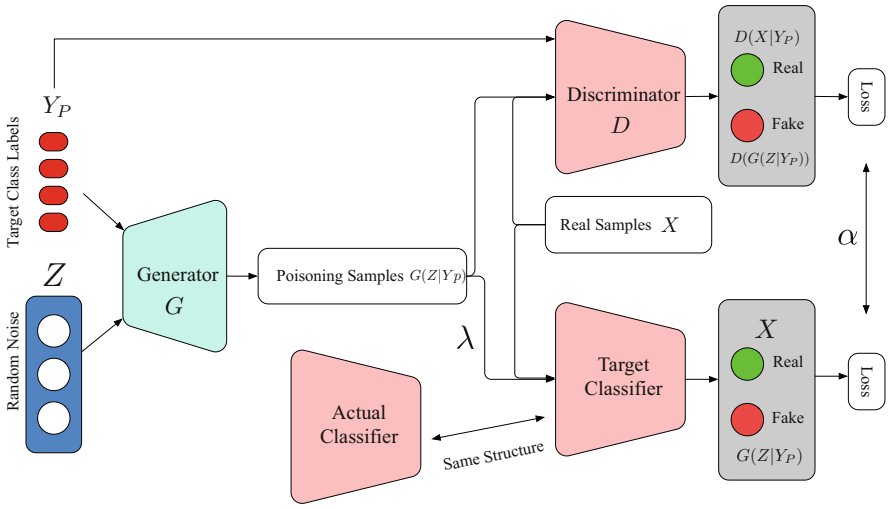

Fig. 4.9 The framework of pGAN

4.4.2 GAN-Based Poisoning Attacks

We firstly introduce a general working pattern of GAN-based poisoning attacks using pGAN [136]. As illustrated in Fig. 4.9, pGAN includes three components, generator, discriminator, and target classifier. Particularly, the generator G of pGAN targets on generating poisoning data samples, which maximize the error of the target classifier. Meanwhile, the samples are also expected to minimize the accuracy of discriminator D, which attempts to distinguish the poisoning data samples from the genuine data samples. The target classifier (denoted by f) seeks to minimize the training loss $\mathcal{L}_{(C)}$ evaluated on a training dataset that incorporates poisoning data samples.

pGAN can generate adversarial training samples close to the genuine data. However, they can also undermine the system performance when adopted for training. Furthermore, pGAN introduces a fraction, $\lambda \in (0, 1)$, to maximize the loss $\mathcal{L}_{(C)}$ when evaluated on the poisoned training dataset. Different from the traditional GANs, G of pGAN plays the minimax game against both D and f, which is akin to the CGAN as introduced in Sect. 4.3.1, and can be formulated as

$$V_{Y_P}(D, G) = \mathop{\mathbb{E}}_{X \sim p_{\text{data}}(X)} [\log D(X \mid Y_P)] + \mathop{\mathbb{E}}_{Z \sim p_Z(Z)} [\log(1 - D(G(Z \mid Y_P)))], \tag{4.6}$$

where Y_P denotes the set of target class labels for attackers.

The objective function of the target classifier is defined as

$$\mathbb{W}(C, G) = -\left(\lambda \underset{Z\sim p_Z(Z|Y_p)}{\mathbb{E}}\left[\mathcal{L}_{(C)}\left(G\left(Z \mid Y_p\right)\right)\right] + (1-\lambda)\underset{X\sim p_x(X)}{\mathbb{E}}\left[\mathcal{L}_{(C)}(X)\right]\right), \tag{4.7}$$

where λ is the portion of poisoning data samples incorporated in the training dataset.

It is worth noting that the poisoning data samples are a subset of the poisoning class labels Y_P, while the genuine data samples adopted to train the target classifier are from all the classes. We can observe that the target classifier is trained with a mixture of genuine and poisoning data balanced by λ. Integrating equations (4.6) and (4.7), the optimizing problem of pGAN can be finally formulated as

$$\min_G \max_{D,C} \alpha V_{Y_P}(D, G) + (1-\alpha)\mathbb{W}(C, G), \tag{4.8}$$

where $\alpha \in [0, 1]$ is a parameter that controls the significance of each of the two functions with regard to the global optimal.

We can also treat it as a "poisoning level" controller. It can be easily found that the higher the α, the less effective the attack. Correspondingly, a lower α will prioritize the attack optimization toward the classifier, thus developing more effective attacks. In the extreme condition of $\alpha = 1$, we can deduce that the minimax game in Eq. (4.11) is equivalent to that of CGAN.

4.4.2.1 GAN-Based Poisoning Attacks in Federated Learning

From Chap. 3, we know that federated learning frameworks are vulnerable to poisoning attacks [138]. General GAN-based poisoning attack approaches like the one introduced in Sect. 4.4.2 assume that attackers possess the validated dataset with the same distribution as the training dataset, which is impractical in federated learning conditions [7]. Zhang et al. [24] proposed a poisoning attack approach toward federated learning systems, which can handle the issue. Next, we elaborate on this approach for readers to gain a better understanding of GAN-based poisoning attacks in federated learning.

A schematic of the proposed GAN-based poisoning attack in [24] is shown in Fig. 4.10. Poisoning data generation is the key in GAN-based poisoning attacks, likewise for federated learning scenarios.

The poisoning data generation in [24] is very similar to the one of pGAN as introduced in Sect. 4.4.2. The attacker firstly receives the global model $M^{(G)}$ and develops a secret replica of $M^{(G)}$ as the initialized discriminator D.

Then, the generator G is fed with random noise to synthesize fake data samples and thereafter input these samples to D. The fake data samples are classified as corresponding classes Y_O by D. Employing the label flipping strategy as did in many poisoning attack studies, these samples are assigned to the target class labels

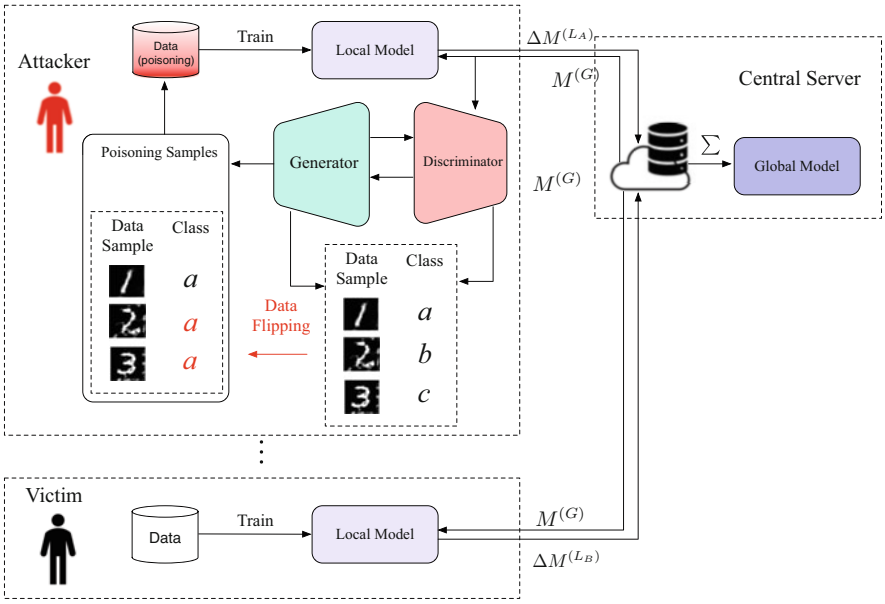

Fig. 4.10 An example of GAN-based poisoning attack in federated learning [24]

Y_P (e.g., flipping: $Y_O \rightarrow Y_P$) and finally stored in the training dataset of the attacker. Since the federated learning is executed iteratively, the generator G of the attacker can produce plenty of synthetic samples which are plausible. Subsequently, the attacker trains its current local model with the poisoned data and sends the updates $\Delta M^{(L_A)}$ to the central server.

Finally, the attacker can leverage the poisoned model updates to undermine the global model's functionalities via the aggregation with other benign local updates $\Delta M^{(L_B)}$ in the central server, further impacting the federated learning system's overall performance by iterative communications.

4.4.3 GAN-Based Privacy Threats

As introduced in Chap. 2, inference attacks can significantly threaten users' privacy. Conventional inference attack methods have many limitations. For example, model inversion attacks possibly develop examples that only resemble the actual data that determined the class. The reason behind this phenomenon is that the commonly adopted deep learning models in model inversion attacks classify the broad areas of the input space accurately, but some other components are missed. If this happens, the attacker may believe that he has reconstructed important information for that class when he has only obtained useless data. This problem also occurs in the scenarios of other inference attacks, e.g., membership inference attack [139].

Some model information, e.g., gradients and parameters, are significant mediums connecting the input and output. GAN can effectively utilize such information to track private information, making it practical in mounting inference attacks.

Furthermore, as research has shown that GAN-generated samples are quite close to the real samples of the training data, GAN can recover a greater portion of the space that assuredly contains the targeted sensitive information. For example, Zhao et al. [140] regarded the discriminator as an inverter in their GANs to learn the density representations of given samples by minimizing the reconstruction error γ of real samples X, which is formulated as

$$\min_{\gamma} \mathop{\mathbb{E}}_{X \sim p_{\text{data}}(X)} \|G(I_\gamma(X)) - X\| + \lambda \mathop{\mathbb{E}}_{Z \sim p_Z(Z)} [\mathcal{L}(Z, I_\gamma(G(Z)))]. \tag{4.9}$$

4.4.3.1 GAN-Based Inference Attacks

The mimicry-synthesis speciality of GAN makes it a promising technique for inference attacks. One pattern of GAN-based inference attacks is reconstructing the private data samples by model inversion, which can be referred to Chap. 2.

Another pattern is doing data augmentation for inference. For example, Bai et al. [139] proposed a GAN-based approach that can produce synthetic samples to augment the training dataset and further utilize the generated samples to train the shadow model (the one that simulates the behavior of the target model), which significantly improves the performance of membership inference attack.

4.4.3.2 GAN-Based Inference Attacks in Federated Learning

For a centralized training system, frequent queries from attackers to the target model could be easily noticed [139]. However, for federated learning systems, the participants can access the collaboratively trained model from the central server legitimately in each training iteration. Therefore, an attacker who acts as a benign participant in the federated learning system can repeatedly touch the global model and further implement his inference attack.

The vulnerability of federated learning is caused by its open nature, and security researchers realized the problem as early as the birth of federated learning and have experimented a plethora of works in terms of GAN-based inference attacks on federated learning systems [48, 49, 141, 142].

A pioneer work of GAN-based inference attacks in federated learning can be referred to [48]. In the work, the attacker is assumed to act as a benign insider of a federated learning system and attempts to extract sensitive information from the other participants. The details are presented as follows:

- Victim v declares the label set of his own data as $[a, b]$. The attacker declares his label set as $[b, c]$. In other words, class b is the shared one, and the attacker

has no knowledge of class a. As a result, the attacker's objective is to deduce the meaningful knowledge about components in class a as much as possible.
- To achieve this goal, the attacker uses GAN to create samples that resemble the victim's samples from class a. The attacker pumps these synthetic samples labeled a into the federated learning procedure.
- In this case, it requires the victim to differentiate between classes a and c as far as possible, which will make him expose more knowledge about class a. In this case, the attacker imitates samples from class a and exploits the victim to improve his understanding of the class he had previously disregarded.

Recalling the description in Sect. 4.2, GANs were designed to learn the distribution from the output of a classifier without directly accessing the data. This capacity can be utilized to trick the victim into divulging more knowledge about a class that the attacker is unfamiliar with. As the generated samples are getting more and more close to the target class, it is getting harder and harder for the global model to classify the target class correctly and consequently reveal more information about the target class. In this case, the attacker can utilize the released information to improve its generator to develop more similar samples to the target class and thus achieve privacy theft.

An illustration of the scheme proposed in [48] is shown in Fig. 4.11. We can find that the architecture of this scheme and the previously introduced GAN-based

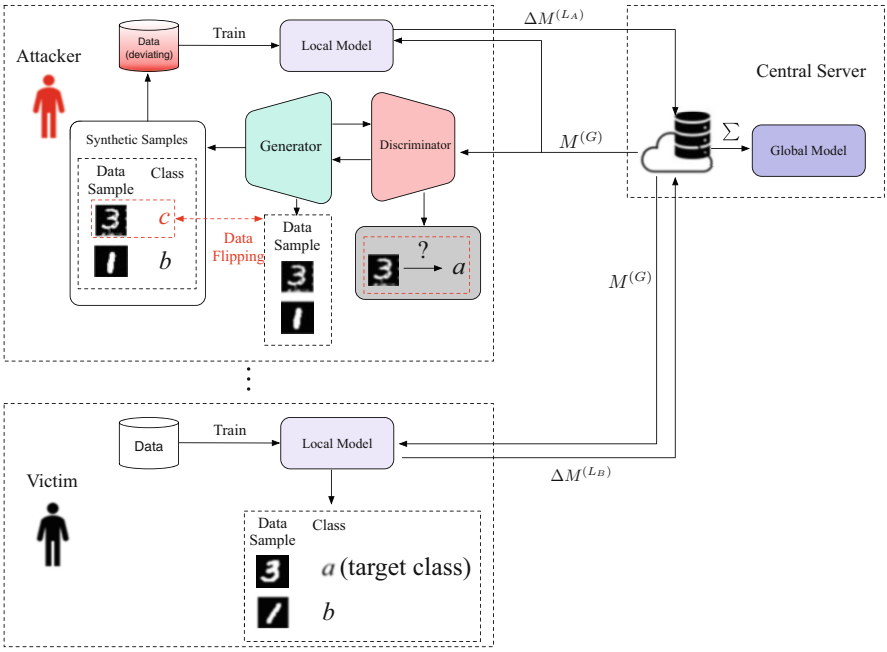

Fig. 4.11 An example of GAN-based inference attack in federated learning [48]

poisoning attack method [24] look very similar. They both incorporate the following operations:

- Employ GAN to generate fake samples which mimic the victim's data samples.
- Use label flipping to flip the samples' labels to a specific class.
- Train the attacker's model on the fake samples to develop specially crafted gradients

However, their targets and the insights of the technique adoption are significantly different. The GAN-based poison attacks endeavor to maximize the attack effects (e.g., model performance degradation) by uploading specially crafted gradients, while the GAN-based inference attacks, such as the example in [48], attempt to employ the specially crafted gradients to maximize the leaked information from the victim's local data by a seducing means.

The above approach has shown the efficacy of GAN-based inference attacks against federated learning systems; however, it is still limited in terms of attacking due to the following factors:

- First, the global model will be polluted as a sacrifice of the attack. Overall learning performance of the system will be compromised.
- Second, the above attack cannot derive the exact samples from the victims but general samples describing the properties of the targeted class, which cannot achieve "user-level" privacy attacks.

The approach proposed in [49] tried to solve the two issues. To train the GAN more imperceptibly without modifying the global model and compromise federated learning, the attacker in [49] was assumed to be able to compromise the central server and employed a GAN to generate fake samples secretly to confuse the participants. To implement the "user-level" privacy attack, a multitask-enabled GAN was devised, and multiple factors were considered, such as the real application environment, category, and identity of the victim. Particularly, the discriminator of the devised GAN can handle three tasks, real-fake classification, categorization classification, and, most importantly, identification classification, which can differentiate the victim from the other participants. We note that the last function cannot be handled by the approach of [48].

The schematic of the approach proposed in [49] is illustrated in Fig. 4.12.

The principle of the devised GAN is also similar to CGAN, where the noise to generator and the real samples to the discriminator are conditional to the category and identity information. For the identity information, since the samples from different clients cannot be directly accessed by the central server in the setting federated learning, a gradient-based data recovery approach is adopted to recover the clients' local data at the central server, which are subsequently utilized to train the GAN. An auxiliary dataset on the central server side is needed for providing real samples.

Additionally, the authors of [49] introduced a scenario of "active attack," which is different from the aforementioned one (defined as "passive attack" in the referenced literature). Specifically, the "active attack" assumes an affiliated server owned by

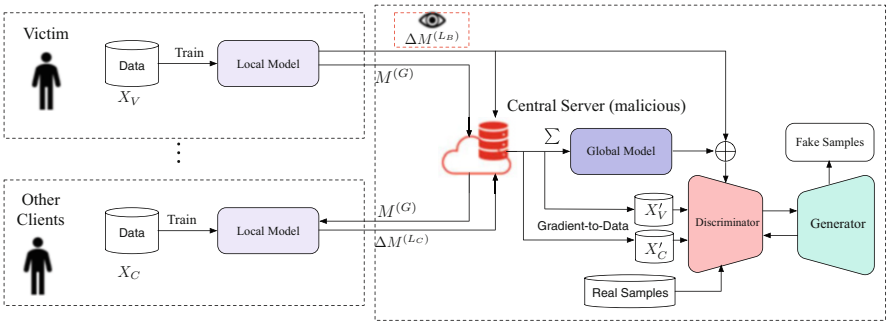

Fig. 4.12 An example of GAN-based inference attack in federated learning [49]. X_V and X_C represent the original training data of the victim and other clients, respectively. X'_V and X'_C denote their corresponding recovered data by the central server. The global model and the discriminator at the central server share the same model structure except the output layer

Fig. 4.13 Comparison of GAN-generated samples by different scenarios proposed in [49]

the central server, which is only connected to the victim. In this scenario, GAN can generate higher-quality samples (see the comparison in Fig. 4.13), thanks to the concentration on target real samples.

4.4.4 GAN-Based Attacks from Insiders

In general, cyberattacks can be launched by insiders or outsiders. Similarly, GAN-based attacks on federated learning systems can be categorized as:

- Attacks from insiders.
- Attacks from outsiders.

Insider attacks are stronger than outsider attacks, as the insiders can obtain useful information from the federated learning system easily and imperceptibly to develop more effective attack models in a fixed system life span [143].

In terms of insider attacks in federated learning scenarios, it could be conducted by the central server or any of the clients of a federated learning system, which will be discussed below separately.

4.4.5 GAN-Based Attacks from Clients

Since federated learning does not allow its participants to share their raw data, some clients may cast greedy eyes on the data of other clients. In horizontal federated learning, an attacker's targets may be other clients' identity information. In vertical federated learning, an attacker may target the data features of the other clients, which he does not have.

The data generation capacity of GAN enables attackers to produce mimic data samples locally.

Furthermore, the majority of research efforts in this domain focus on simple settings at the moment, e.g., attacks launched by a single client [48, 49]. However, complex attacks could be mounted through collusion among multiple malicious clients, such as the Byzantine attacks. One possible example is that the clients may manipulate their output to demonstrate similar distribution as the supposed model updates by GAN, which makes them quite concealing. Additionally, a single attack node can use the sybil attack method to simulate multiple forging clients to launch a powerful attack.

4.4.6 GAN-Based Attacks from Central Server

The attacker can be the operator/owner of the central server or just lurk in the central server. In this case, there exists different assumptions such as the attacker is totally malicious or the attacker is honest but curious (e.g., collecting sensitive information in a passive manner).

The motive of the malicious attacker at the central server can be similar to the clients, as the central server generally knows nothing about the datasets of clients, which may push a malicious server to recover the raw data of targeted clients. Since the clients are asked to upload their local model updates to the central server, the server can utilize these model updates to recover the local datasets by a stealthily mounted GAN [49]. In particular, [49] considers an active attack scenario: the victim is isolated from other clients so that the central server can launch the attack on the victim in a more targeted way.

Table 4.1 Comparison of GAN-based attacks against federated learning systems

Literature	Known as	Attack type	Attacker	GAN type
[48]	–	IA	C	CGAN
[49, 142]	mGAN-AI	IA	S	CGAN
[24, 146]	PoisonGAN	PA	C	GAN
[141]	–	IA	C	GAN
[147]	–	IA	C	GAN
[148]	GRNN	IA	S	–
[149]	Jekyll	PA & IA	O	CGAN

IA Inference attack, *PA* poisoning attack, *C* client, *S* central server, *O* outsider

4.4.7 GAN-Based Attacks from Outsiders

Outsider attacks are usually triggered by eavesdroppers lurking in the communication channels between clients and the central server or the actual service user of the federated model. In general, the outsider has very limited knowledge about the shared model of federated learning systems; hence the attacks are usually black box based, and it is more difficult [144] than the insider attacks. Moreover, due to the lack of legality for the outsider, its intrusion behavior is generally stringently monitored by the system. Therefore, the challenges of GAN-based attacks from outsiders lie in GAN-based black box inference and how it can avoid the detection [145].

In Table 4.1, we present a summary and comparison of the existing literature containing different attack scenario assumptions. Interesting readers can refer to the literature to specifically understand the technical details of GAN-based attacks in federated learning.

4.5 Counter GAN-Based Attacks

Threats of GAN-based attacks have aroused wide research concerns, and many defenses are designed to counter GAN-based attacks.

We know that conventional defenses against attack in federated learning include cryptographic approaches, differential privacy-based approaches, hardware-assisted approaches, and so on. The investigation in [48] indicates that pruning or obfuscating shared parameters by differential privacy does not contribute to defending the GAN-based attacks. Hardware-assisted approaches, such as the recent popular trusted execution environment, require a specific hardware configuration that is not available on all devices, which cannot be regarded as a generic approach for defending GAN-based attacks.

Similar to other defense strategies, the countermeasure against GAN-based attacks can be categorized into:

- Passive defense.
- Active defense.

In this chapter, we discuss two representative defense approaches against GAN-based attacks according to the category.

4.5.1 Passive Defense Against GAN-Based Attacks

Passive defenses are usually held by the victim, and they aim to minimize the influence of the attacks without the initiative intention. In general, the previously mentioned cryptographic and differential privacy-based approaches can be classified into this category.

Researchers also attempt to use computationally efficient approaches to achieve passive defense. For GAN-based attacks, Ching et al. [150] proposed a defense approach leveraging model partition. The attacker is similar to the one assumed in [48], who pretends to be a benign client. Instead of completely training the entire model at the client end, the model is partially concealed to users, and edge servers are introduced to assist the concealed parts of model's training. In this way, the attacker cannot steal integral information from the federated learning system to generate close samples to the victims. Particularly, the first and the last layers of the model are required to be computed on the client end to avoid data leakage. Furthermore, a corresponding optimization problem is considered to balance the overheads between clients and edge servers to minimize the training time with a privacy guarantee.

In [151], the authors introduced another strategy to defend against GAN-based attacks. This work takes the setting as in Fig. 4.11, where the threat model intends to learn the distribution of victim's data. To prevent the attacker from learning the real distribution, the work devises a GAN at the victim side, who attempts to manipulate the victim's training datasets before pumping them to the shared global for training. In this case, the attacker can only learn the distribution of the manipulated data. To make the manipulated data obfuscate the attacker, an unsupervised learning pattern is designed and the corresponding objective function is formulated as

$$\max_{Z} \mathcal{L}_{(obf)} = \log Var(G(Z)), \tag{4.10}$$

where $\mathcal{L}_{(obf)}$ denotes the variance of $G(Z)$, which is expected to be maximized to twist the generated samples.

Furthermore, the manipulated data may reduce the accuracy of the shared model; the objective function for the generator of the victim is designed to minimize the

distance between the generated and original samples, which is defined as

$$\min_{Z} \mathcal{L}_{(sim)} = \|G(Z) - X\|_2^2, \tag{4.11}$$

where $\mathcal{L}_{(sim)}$ denotes the similarity between the generated samples $G(Z)$ and the real samples X.

4.5.2 Active Defense Against GAN-Based Attacks

Opposite to passive defense, active defense in federated learning systems provides timely and accurate warnings before attacks impact the target clients and builds a resilient defense system in real time to avoid, transfer, and reduce the risks. Usually, the active defense can significantly reduce the total computational overheads of defense systems [152].

Xiong et al. [153] proposed an approach that can actively detect GAN-based attacks at the beginning of the training phase, and the setting of the research is the same as the one in Fig. 4.11. The central server is assumed to know the information about how data classes are distributed among clients but not the real data. Subsequently, this work uses unsupervised clustering to distinguish the class that a client should not have according to the central server's knowledge. Consequently, the attackers will be arduous to dodge the detection at the early stage. The earlier the attack can be detected, the more training data can be protected.

For the comparison among the aforementioned three different defense approaches, an illustration comparing all three approaches is shown in Fig. 4.14.

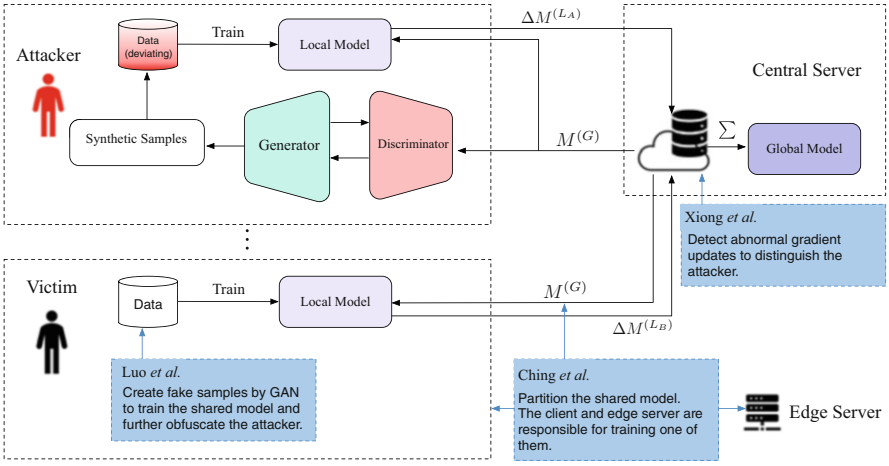

Fig. 4.14 The defense approaches proposed in [150, 151], and [153] toward the attack scenario in [48]

4.6 Summary of the Chapter

In this chapter, we first introduce the development of GAN by elaborating on the original GAN and some of the famous variants of GAN. According the introduction of the previous chapters and the existing literature, we categorize GAN-based attacks in federated learning into GAN-based security attacks and GAN-based privacy attacks. Taking poisoning attack and inference attacks as their main purposes, we illustrate the principle and application of GAN-based attacks with several existing works in the literature. Subsequently, we introduce the related countermeasures to GAN-based attacks in federated learning.

From our introduction, the readers may realize that GAN is actually a tool to play a role in the attacking strategies introduced in Chaps. 2 and 3, which are usually cast for different attacking purposes.

Nowadays, GANs have been widely studied in both attack and defense scenarios of federated learning. We hope our content can not only help readers understand GANs and GAN-based attack and defense approaches but also enlighten readers about how to select proper tools (not only GANs) to deal with unprecedented problems and challenges in the future.

Chapter 5
Differential Privacy in Federated Learning

In this chapter, we briefly introduce the concept and basic terms of differential privacy, which is a very popular new tool designed for privacy protection. Then, we present the applications of differential privacy in federated learning for security and privacy protection.

Moreover, we also present the related latest work in the domain and a short discussion on the promising directions to go for the reference of our readers.

5.1 What Is Differential Privacy?

With the widespread application of data mining technology, researchers are encouraged to dig deeper into the intrinsic value of data by publishing user datasets. However, in the process of data release, there are security risks, which may lead to the disclosure of user privacy.

Datasets usually contain a lot of private data about individuals, such as medical diagnosis records, personal consumption habits, and usage preferences. The private information can be compromised by the release of the dataset.

In order to protect data privacy, the most powerful method is removing all personal information that can be used to identify a specific user, namely, anonymization. After anonymous processing, the private information of individuals can be hidden in large datasets and contribute the intrinsic value of the data in the form of statistical results.

Although deleting identifiers of data (such as name, ID number, etc.) can protect personal privacy to a certain extent, there are still a large number of cases showing that the operation cannot guarantee the security of private information.

When an attacker submits query requests to the data provider, if the data provider responds directly without any protection actions, it may lead to privacy leakage,

S. Yu, L. Cui, *Security and Privacy in Federated Learning*, Digital Privacy and Security, https://doi.org/10.1007/978-981-19-8692-5_5

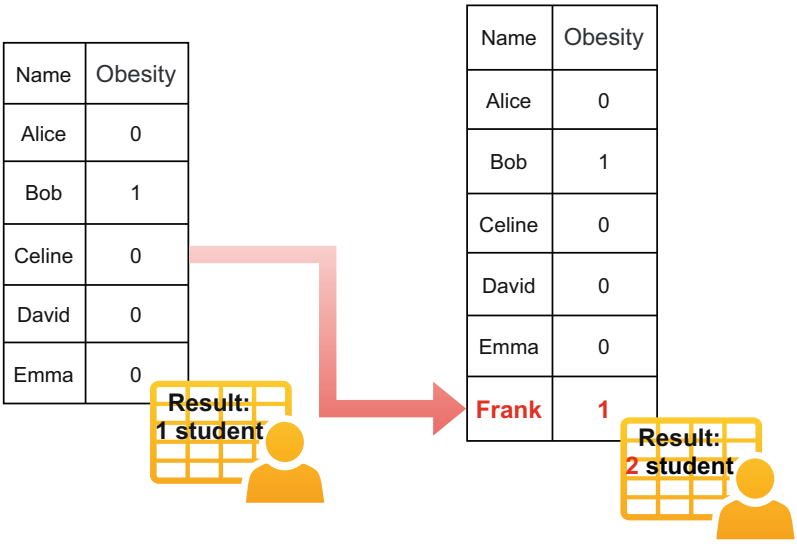

Fig. 5.1 An example of privacy attack through multiple queries

because the user can figure out his expected privacy information through multiple queries.

For example, in Fig. 5.1, suppose we want to check with the hospital on the results of the obesity examination of a group of classmates. The hospital has all the detailed information but only allows teachers to query "how many students in this class are obese." At the beginning of the semester, the teacher inquired and found that only one student in the class was obese. Within two days, a new classmate was transferred to the class. When the teacher sent an inquiry to the hospital again, he found that two of his classmates are now suffering from obesity! The teacher can therefore be sure that the new classmate must be obese. In other words, due to the emergence of a small sample, new classmates transferred, and the attacker obtained accurate individual information.

Differential privacy is a new tool for privacy protection in statistical query on databases. It protects the privacy of users while preserving statistical characteristics. Therefore, the risk of privacy leakage caused by adding a small sample to a dataset is controlled within a very small and acceptable range, and the attacker cannot obtain accurate individual information by observing the multiple query results.

5.2 Differential Privacy Definition and Terms

In this section, we firstly present the mathematical definition of differential privacy and then give two implementation methods of differential privacy.

5.2.1 Mathematical Model of Differential Privacy

In a group work [154, 155], Dwork and her collaborators defined the concept of differential privacy around 2004 as follows.

According to the example of the obesity database above, we can realize that the privacy leak occurs when querying two databases that differ by only one sample. We denote two databases as D_1 and D_2 if they differ on a single element. Let $\mathcal{M}$ be the algorithm which provides (ϵ, δ)-differential privacy and ϵ be a positive real number. The differential privacy algorithm $\mathcal{M}$ should satisfy

$$\Pr[\mathcal{M}(D_1) \in S] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(D_2) \in S] + \delta, \tag{5.1}$$

where ϵ represents the budget of the privacy and δ is a probability, which is usually a small real number.

The smaller the ϵ is, the higher the privacy protection level is. When the value of ϵ is 0, it means that the probability distribution of the output of algorithm $\mathcal{M}$ for D_1 and D_2 is exactly the same, which means that the output of algorithm $\mathcal{M}$ cannot reflect any useful information about the datasets. On the other hand, the value of ϵ also reflects the availability of data. In the same case, the smaller the ϵ, the lower the availability of data.

We know that differential privacy can guarantee privacy between two databases that differ by only one record. In other words, any attacker with arbitrary external information cannot know the information of any individual in the database. But this definition can also naturally be extended to the case where there is a difference of at most c records between the two databases. That is to ensure that any attacker with arbitrary external information cannot know the information of c individuals in the database. If c items change, the probability is bounded by $e^{\epsilon c}$ instead of e^{ϵ} [155].

$$\Pr[\mathcal{M}(D_1) \in S] \leq e^{\epsilon c} \cdot \Pr[\mathcal{M}(D_2) \in S] + \delta. \tag{5.2}$$

Sometimes, people set $\delta = 0$; then the parameter δ is omitted from the equation; then it becomes ϵ differential privacy.

Since differential privacy is only a concept, algorithms that satisfy the above-mentioned conditions are called differential privacy algorithms. How can the above conditions be met? This requires adding noise to the query results. The key problem for differential privacy is how to add the noise to balance the data utility and privacy protection. There are two popular ways to add noise: Laplace noise and Gaussian noise.

5.2.2 Differential Privacy Using Laplace Noise

First, we need to define sensitivity, which is usually used as a parameter of noise volume to measure the maximum difference between query results on adjacent datasets. Let f be a function $f : D \to \mathbb{R}$ and the sensitivity of function denoted as Δf, which is defined by

$$\Delta f = \max_{D_1, D_2} \|f(D_1) - f(D_2)\|_1, \tag{5.3}$$

where sensitivity Δf here is related to the query function and measured under ℓ_1 norm. It represents the maximum range of variation of a query function $f(\cdot)$ for two sibling datasets that differ by maximum one element. The definition is mainly to satisfy the proof of differential privacy later.

The first additive noise mechanism we will introduce is drawn from a Laplace distribution

$$\mathcal{M}(D) = f(D) + Lap\left(0, \frac{\Delta f}{\epsilon}\right), \tag{5.4}$$

where $f(D)$ represents a query function and $Lap(\cdot)$ represents the Laplace distribution. The smaller the privacy budget ϵ is, the greater the noise is.

The proof of the above conclusion is not difficult. In the following, we show that the output distribution of $\mathcal{M}(x)$ is close in a multiplicative sense to $\mathcal{M}(y)$ everywhere.

$$\begin{aligned}
\frac{\Pr(\mathcal{M}(x) = z)}{\Pr(\mathcal{M}(y) = z)} &= \frac{\Pr\left(f(x) + \mathrm{Lap}\left(0, \frac{\Delta f}{\epsilon}\right) = z\right)}{\Pr\left(f(y) + \mathrm{Lap}\left(0, \frac{\Delta f}{\epsilon}\right) = z\right)} \\
&= \frac{\Pr\left(\mathrm{Lap}\left(0, \frac{\Delta f}{\epsilon}\right) = z - f(x)\right)}{\Pr\left(\mathrm{Lap}\left(0, \frac{\Delta f}{\epsilon}\right) = z - f(y)\right)} \\
&= \frac{\exp\left(-\frac{\epsilon |z - f(x)|}{\Delta f}\right)}{\exp\left(-\frac{\epsilon |z - f(y)|}{\Delta f}\right)} \\
&= \exp\left(\epsilon \cdot \frac{|z - f(y)| - |z - f(x)|}{\Delta f}\right) \\
&\leq \exp\left(\epsilon \cdot \frac{\|f(y) - f(x)\|}{\Delta f}\right) \\
&\leq \exp(\epsilon)
\end{aligned} \tag{5.5}$$

Therefore, the Laplace mechanism is a special case as $\delta = 0$.

5.2.3 Differential Privacy Using Gaussian Noise

While the Laplace mechanism provides a strict $(\epsilon, 0)$-differential privacy, Gaussian mechanism can provide (ϵ, δ)-differential privacy. The sensitivity of Gaussian mechanism is under ℓ_2 norm due to the difference of Gaussian distribution

$$\Delta f = \max_{D_1, D_2} \| f(D_1) - f(D_2) \|_2 \tag{5.6}$$

Then, for any $\delta \in (0, 1)$ and $\epsilon \in (0, 1)$, we can give the mechanism defined by

$$\mathcal{M}(D) = f(D) + \mathcal{N}\left(0, \sigma^2 = \frac{2\ln(1.25/\delta) \cdot (\Delta f)^2}{\epsilon^2}\right), \tag{5.7}$$

where $\mathcal{N}(\cdot)$ represents the Gaussian distribution.

The standard deviation δ of the Gaussian distribution determines the scale of noise. It is related to privacy budget ϵ and probability δ, which is used to ensure privacy loss bounded by ϵ with probability at least $1 - \delta$. The detailed proof is given by [156].

Let us partition the output dataset S as $S = S_1 \cup S_2$, where subset S_1 is under the strict $(\epsilon, 0)$-differential privacy, while subset S_2 is not. Therefore, in Gaussian mechanism, we have

$$\Pr[\mathcal{M}(x) \in S] = \Pr[\mathcal{M}(x) \in S_1] + \Pr[\mathcal{M}(x) \in S_2], \tag{5.8}$$

where $\Pr[\mathcal{M}(x) \in S_2]$ is less than δ, consequently,

$$\begin{aligned} \Pr[\mathcal{M}(x) \in S] &\leq \Pr[\mathcal{M}(x) \in S_1] + \delta \\ &\leq \exp(\epsilon) (\Pr[\mathcal{M}(y) \in S_1]) + \delta. \end{aligned} \tag{5.9}$$

5.3 Differential Privacy in Federated Learning

In this section, we provide an overview of differential privacy (DP) techniques in FL. Starting from introducing the necessity of employing DP, we subsequently discuss the state-of-the-art DP methods in FL, which are centralized DP (CDP), local DP (LDP), and distributed DP (DDP).

As discussed in Chap. 2, although FL has distinct privacy advantages, the sensitive information in FL framework can still be inferred using inference attacks, such as model inversion attack, model inference attack, property inference attack, and membership inference attack. For example, model updates or two consecutive snapshots of FL model parameters may leak unintended features of participants' training data to adversaries, because deep learning models tend to recognize and

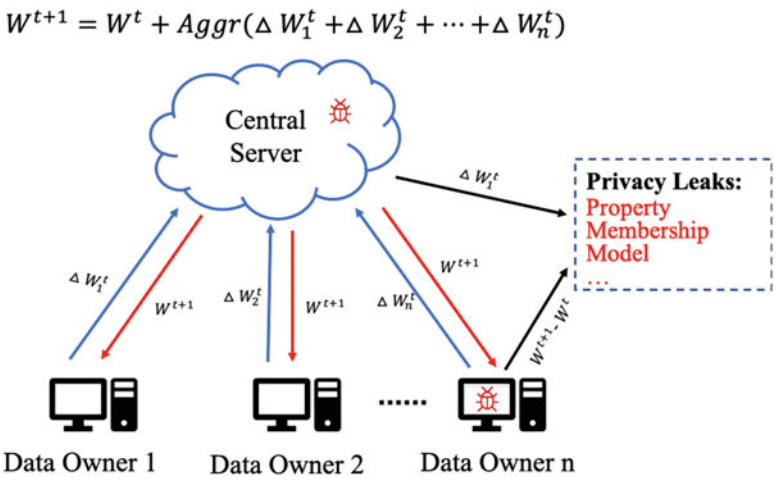

Fig. 5.2 Privacy risks in a vanilla federated learning model

remember more data features than required for the main learning tasks [157]. In addition, an untrusted server could also be an attacker and attempts to infer clients' private information based on all gradients it receives.

Figure 5.2 shows that multiple sensitive information can be inferred based on the updates of FL model. As a result, DP techniques have been widely adopted in FL to prevent information leakage by adding artificial noises.

5.4 Main Differential Privacy Methods in Federated Learning

5.4.1 Centralized Differential Privacy

The original design of DP was for the statistical queries on a central database scenario, where noise is added to the query output to protect data privacy against information retrievals.

In federated learning, a trusted server can view the data of all participants and answer queries in a private way by randomizing query results. As Fig. 5.3 shows, the trusted central server is responsible for adding noise to the aggregated local model gradients, which hopes to protect the record-level privacy of the entire data.

Brendan et al. [158] introduced a noised version of the federated averaging algorithm that satisfied user-adjacent DP and analyzed differentially private stochastic gradient descent (SGD) for example-level privacy. In [159], the authors proposed a DP preserving federated optimization method to hide clients' contributions during training and balance the trade-off between privacy loss and model performance.

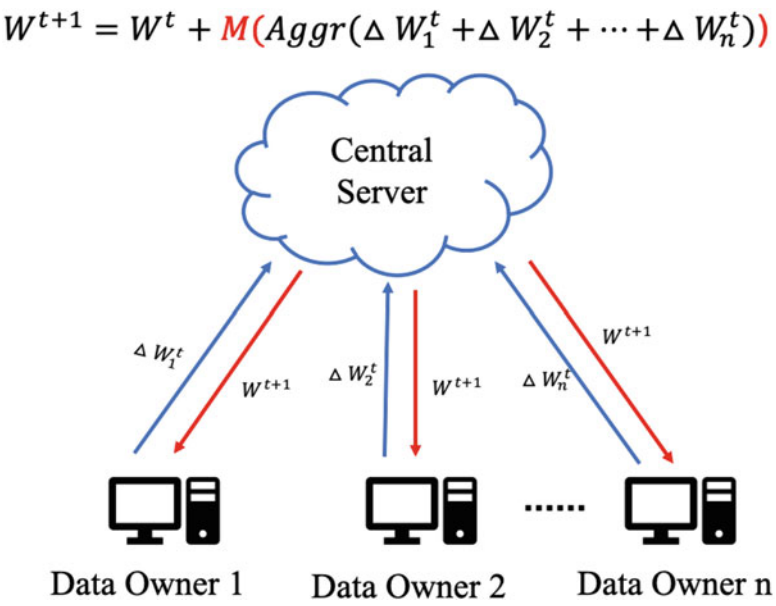

Fig. 5.3 Centralized differential privacy (CDP)

However, these methods fail to consider the fact that the central server may be untrusted or compromised by adversaries, which may lead to privacy leakage. In addition, CDP was designed to handle with a sufficient large number of participants for achieving an acceptable trade-off between privacy and accuracy. When the number of participants is small, FL model with CDP might not converge or achieve acceptable accuracy.

5.4.2 Local Differential Privacy

Different from CDP that needs participants to send their data to an aggregator without noise, Fig. 5.4 shows that participants can employ DP to obfuscate their personal data by themselves before sending them to an untrusted server. In this case, the process is called Local Differential Privacy (LDP). Technically, the algorithm $\mathcal{M}$ is said to provide ϵ-local differential privacy if for all pairs of user's possible privacy data d, d', and all subsets S of $\mathcal{M}$ hold

$$\Pr[\mathcal{M}(d) \in S] \leq e^{\epsilon} \cdot \Pr\left[\mathcal{M}\left(d'\right) \in S\right]. \tag{5.10}$$

The main difference between definition 5.10 and the standard definition of DP is that in DP the probabilities are of the outputs of an algorithm that takes all

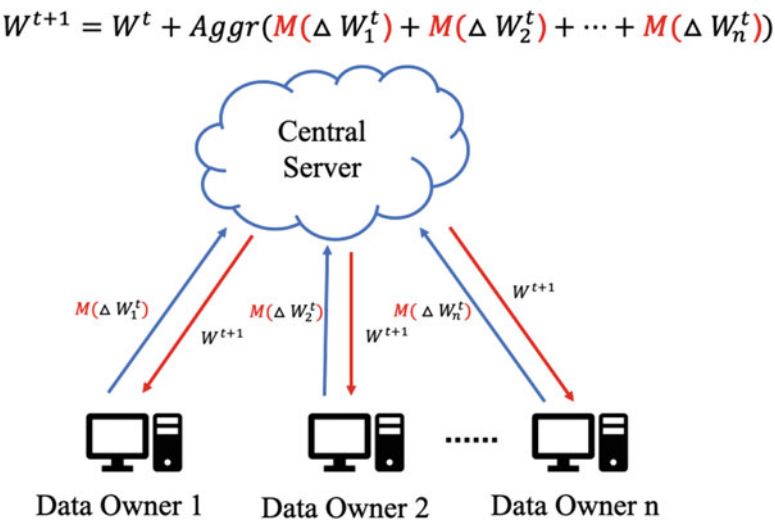

Fig. 5.4 Local differential privacy (LDP)

participants' data, while here it is on an algorithm that takes a single user's data. Compared with CDP, employing LDP in a FL framework can provide the following advantages [160]:

- Participants can select the level of privacy protection by themselves
- There is no need to assume a trusted central server
- Since all stored records are individually sanitized, there is no risk of privacy breaches due to malicious attacks.

In recent years, there are some investigations applying LDP to FL models. In [161], the authors proposed a FL system for the joint training of deep neural network (DNN) models under the protection of the LDP framework. It can prevent privacy inference attacks according to participants' own locally defined privacy setting. Zhao et al. [162] integrated LDP with FedSGD algorithm in vehicular crowdsourcing applications to achieve accurate traffic status prediction while mitigating privacy threats.

However, employing LDP requires each participant to add enough noise, which will cause a huge utility degradation, especially with a large number of participants. Motivated by this, Sun et al. [163] designed a novel framework that applies a Noise-Free DP mechanism into a federated model distillation framework. Trues et al. [164] proposed a method that utilizes both LDP and secure multi-party computation (SMC) to balance the trade-off between learning performance and privacy guarantee.

We will introduce how to apply the SMC technique in FL specifically in Chap. 6.

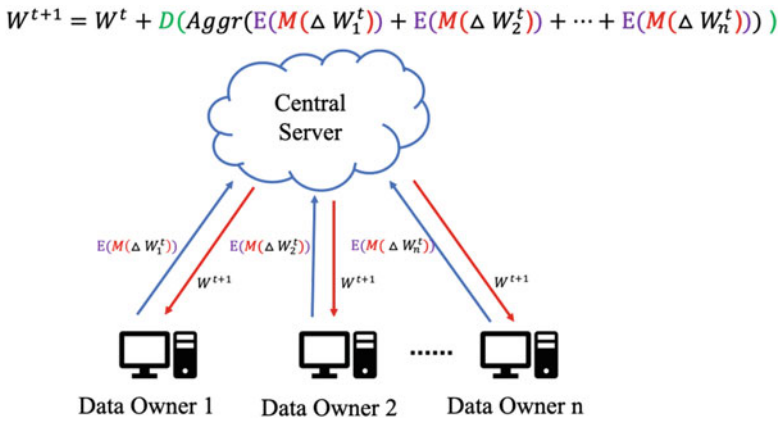

Fig. 5.5 Distributed differential privacy (DDP)

5.4.3 Distributed Differential Privacy

In order to bridge the gap between CDP and LDP, distributed differential privacy (DDP) was proposed to avoid relying on a trusted central server while providing better utility. Figure 5.5 presents a widely used method to achieve DDP by secure aggregation with cryptographic protocols. This method ensures the central server obtaining the aggregation results, and the parameters of the participants will not be leaked to the server.

In order to further guarantee the aggregated result will not show additional information, we can use LDP to perturb local model parameters before aggregation.

Another method to achieve DDP is employing shuffle model. For more details about secure aggregation and shuffle model, please refer to Chaps. 7 and 8, respectively.

5.5 Application of Differential Privacy in Federated Learning

In the practice of FL systems, DP is often not directly used and needs to be modified to meet the actual task requirements. By investigating the recent literature on the application of DP under the FL framework, we found that there are mainly the following research directions.

5.5.1 The Applications of Variant Differential Privacy

Deploying an ordinary DP algorithm on the FL framework can enhance the privacy protection performance to a certain extent, but the sacrifice to model performance hinders its practical application. With that in mind, researchers are trying to improve DP with other techniques.

Yin et al. [165] proposed a new hybrid privacy-preserving approach for FL in which Bayesian DP is used to protect the privacy of clients. The main idea of Bayesian DP [166] was to use the Bayesian method to connect uncertain query answers with given tuples and use data correlation to marginalize unknown tuples.

With the assumption that it guarantees privacy protection, Bayesian DP can enhance service quality when compared to the conventional differential privacy method. Unfortunately, because the server is responsible for allocating privacy budgets, the DP mechanism included in a client's settings may fail to work if a malicious server increases a client's privacy budget too much.

GAN-DP [167] is also a new DP variant designed to provide a higher balance between privacy protection and data utility. GAN-DP contains a generator, a discriminator, and the DP identifier (DPI) whose structure is similar to the discriminator. The distinction is that DPI aims to verify whether the model parameters created meet the DP requirements. After feeding the generator with the local model parameters, the generator outputs a set of synthetic parameters. The DPI and the discriminators then use the created synthesized parameters as inputs. The parameter is accepted as the output result and moves on to the next stage if the synthesized parameter complies with the demands of the two perceptrons.

5.5.2 The Combination of Differential Privacy and Other Methods

The individual privacy-preserving techniques are only able to preserve privacy in federated learning from one perspective. As a result, researchers frequently combine these simple privacy protection techniques to preserve data in federated learning from various perspectives (or dimensions).

Byrd et al. [168] proposed a differential private secure multi-party computing (SMC) protocol for FL, which aimed to make it accessible to people with some computing background but no prior knowledge of security and privacy. In a scheme that only used DP, the server would be aware of each user's noisy private weights. The noisy weights sent to the server are also encrypted in the scheme that combines SMC and DP, allowing the server to calculate the outcome and preventing it from deducing the noisy weights of any specific user. The protocols developed by the methods combining SMC and DP can ensure privacy without sacrificing accuracy.

Gong et al. [169] proposed a system based on DP and homomorphic encryption (HE) for preserving the privacy of clients in FL. Here DP is utilized to protect the

gradients of clients from the central server, and each client encrypts the gradient sent to the server using HE. In this situation, all clients should use the same encryption key to protect client gradients from a malicious server. Because the clients' gradients are already protected by DP, even if the server colludes with a client and obtains the key, it will not be able to access the true values of the clients' gradients. Such techniques, however, introduce uncertainty into the upload parameters and could impair the final performance. In addition, these methods also bring extra computing costs for clients.

5.6 Future Discussion

DP in FL has a series of challenges that require further research. We identified the following possible future research directions for interested readers.

5.6.1 Reduce the Cost of Privacy Protection

Each add-on improvement has its own unique set of extra prices and effects. SMC and HE improve FL's privacy protection capabilities at the cost of more computing power. The DP-based technique increases privacy protection levels during communication by adding noise to the uploaded parameters.

However, the accuracy is necessarily affected by additional noise, which may further have an impact on the global aggregation convergence. This suggests a trade-off between accuracy and privacy protection. The accuracy will be lost, and more time will be spent on convergence if the FL model preserves more privacy. On the other hand, it is crucial to determine whether the privacy protection level is appropriate if a FL model must maintain a specific accuracy or convergence time.

5.6.2 Customized Privacy Restrictions

Due to the heterogeneity of devices in FL scenarios, the privacy restrictions of different devices have their own characteristics. Therefore, it is necessary to define the privacy restrictions for specific devices in detail to provide a higher level of privacy protection. Consequently, it is an interesting research direction to develop privacy protection methods based on the privacy restriction of specific devices.

5.7 Summary of the Chapter

In this chapter, we first introduce the fundamentals of differential privacy. Then, we discuss the reasons for introducing differential privacy in federated learning. Subsequently, we categorize the existing methods into centralized differential privacy, local differential privacy, and distributed differential according to the literature. We outline the frameworks of these three methods as well as discuss their limitations. Then, we overview application of DP in FL and identify future research directions.

We note that differential privacy is the only new tools invented for privacy protection to the best of our knowledge at the moment. It is now used in various scenarios as we do not have too many options in terms of privacy protection tools. It is necessary for the community to invent more new tools for the huge demand in the unprecedented digital privacy age.

Chapter 6
Secure Multi-party Computation in Federated Learning

In the previous chapters, we mainly explored security and privacy in federated learning from the viewpoint of artificial intelligence, which are relatively new and usually more idea based. In general, these methods are somewhat ad hoc and lack theoretical support.

From this chapter, we will present the solutions based on cryptography. Cryptography has been researched for decades, and many brilliant protocols have been invented, deployed, and examined in practice. At this moment, people believe that cryptography solutions are reliable. However, performance is an inherent problem of cryptography solutions.

We will offer the basic concepts and building blocks of cryptography for readers for a smooth reading and then present the security solutions in federated learning based on secure multi-party computation in this chapter. We will discuss the other perspectives in this line in the following two chapters.

We note that the content of these cryptography-related chapters mainly fall in how to use existing cryptographic protocols to serve our needs in federated learning security and privacy. We focus on the application, not cryptography itself. Cryptography is a set of profound knowledge, which demands much more time and effort to master. We do encourage determined readers to study the knowledge of encryption and decryption because they are the mainstream tools to address security and privacy problems so far.

6.1 What Is Secure Multi-party Computation

Secure multi-party computation (SMPC in short) has been explored for at least more than 40 years. The basic setting is that multiple participants need to work together to complete a computing task or execute a mathematical function. Every participant needs to contribute his data or share for the execution, but each of them

S. Yu, L. Cui, *Security and Privacy in Federated Learning*, Digital Privacy and Security, https://doi.org/10.1007/978-981-19-8692-5_6

does not want to reveal his data to the others [170]. The famous example is that two millionaires want to know who is richer between themselves, but do not want to reveal their bank balance to each other. Another example, without involving any other people, the members of the sales department want to know the average income of their group, but do not uncover the income of any individuals to the others.

As a very matured field, cryptography approaches usually have a clear and accurate definition for a given case of attack and defense, including the background setting, resources, and capability of attackers and defenders, respectively.

The definitions need to be explicit and clear for an effective progress.

In secure multi-party computation, the definition includes the following perspectives:

- A set of participants with private inputs to execute some joint functions on their inputs.
- Participants wish to preserve some security properties, for example, privacy and correctness in an election protocol.
- Security must be preserved in the face of adversarial behavior by some of the participants or by external parties.

Researchers usually take advantages of the real/ideal model paradigm to explore the security of a security protocol.

- In the ideal model, security is absolute, e.g., a server is absolutely reliable and trustworthy, and there is zero probability of successful attacks on the server.
- In the real model, we can also achieve the security of the ideal model through various ways. Any adversary which could have learned from or obtained in the real model could have also learned from or obtained in the ideal model. In other words, it is indistinguishable between ideal model and real model.

In terms of privacy, an adversary in the ideal model cannot learn more about the honest participant's input than what is revealed by the function output, and it is the same for the real model. In terms of correctness, a function is always computed correctly in an ideal model, and the same is true for the real model.

For example, in an ideal model of SMPC, we suppose there is a trusted server, which will aggregate the inputs from each participant and generate the expected output correctly. However, in the real model, the trusted server may not exist, and the participants exchange inputs and messages with each other in a peer-to-peer way to achieve the function.

There are three security models in SMPC:

- Semi-honest adversary model. In this model, corrupted participants honestly execute the agreed protocol; however, the adversaries will collect all the information they can in order to obtain additional information, which is supposed to be confidential. This is a weak security model as we suppose the adversaries are somehow "honest."

- Malicious adversary model. In this case, corrupted participants may deviate from the protocol according to the instruction of the adversaries at any time. The malicious adversary model is a strong security model.
- Covert adversary model. This model is somewhere between the previous two extreme models; corrupted participants exhibit maliciously with a given probability. In this model, an adversary has to measure the risk of being caught against the benefit of cheating.

6.2 Building Blocks for Secure Multi-party Computing

In cryptography, there are some primitives as the building blocks for the mansion; of course, they are also the fundamental elements for SMPC. We will list them here for a self-contained reading for readers who are not familiar with cryptography.

6.2.1 Oblivious Transfer

In the oblivious transfer (OT in short) protocol, a sender S wants to send one message out of n ($n > 1$) messages to a receiver R; the goal is that S does not know which message is received by R, and R obtains no information about the other $n - 1$ messages of S, except the one he received.

A standard 1-out-of-2 function is described as a function F_{ot} as below.

Input:

– Sender S inputs two messages m_0, m_1.
– Receiver R inputs a random bit $\sigma \in \{0, 1\}$.

Output:

– Sender S has no output.
– Receiver R obtains m_σ and learns no knowledge of $m_{1-\sigma}$.

There are many ways to implement the OT protocol, e.g., RAS or discrete logarithm problem-based methods.

6.2.2 Garbled Circuit

The garbled circuit (GC in short) is the early tool to execute secure computation between two parties (S2PC) [171]. Two parties work together to complete a function (e.g., comparison) without revealing their secrets.

The famous metaphor is the millionaire problem: two millionaires (Alice and Bob) want to know who is richer but do not want to reveal each other's account balance. The two parties will work together to reach the conclusion: one will design a logical circuit, and the other party will check and use the logical circuit. There is a set of steps for them to execute interactively, e.g., Yao's protocol, and then they will achieve their goal with respect of privacy and correctness.

In order to help readers understand GC, we offer more details below. We note that we only discuss the simple cases; the balances are small numbers and not equivalent.

First of all, we design a function $f() =$ balance of Alice—balance of Bob, then we can transform the comparison problem into a logical problem: is $f() > 0$? If the answer is true, then we know Alice is richer; if the answer is false, then Bob is richer.

Second, we transfer the balance of the two millionaires into binary format, $a_2a_1a_0$ (e.g., 111) and $b_2b_1b_0$ (e.g., 011), and we can further compare them in the bit-wise way from the lower end to the high end. In a simple case, we find $a_2 = 1$ and $b_2 = 0$, and the output of the logical question is true (Alice is richer).

In other words, we can build a digital circuit with the three basic logic gates AND, OR, and NOT to implement the comparison between the two millionaires. Each gate usually has two inputs and one output. For example, a_i and b_i as the input, and generate an output as 0 or 1 (false or true).

We note that an XOR (exclusive OR) gate is usually employed in practice to improve the performance. XOR gate can be implemented through the combination of the three basic gates.

In the garble circuit protocol, Alice will build the logical circuit, label each inputs and outputs (which is equivalent to substitution encryption), and further establish a truth table based on the labels. Following this, Alice will shuffle the labeled truth table and finally pass the "encrypted" truth table and the circuit to Bob.

Alice will pass the labels of her input (her account balance) to Bob, who does not know the true value of the labels. At the same time, Bob obtain the labels of his input (his account balance) from Alice through the OT protocol; as a result, Alice does not know the true value of Bob's input.

With all the information in place, Bob can execute the calculation and reaches a label as the output (true or false).

Bob shares the label of his calculation, and Alice shares the true value of the labels. Then both of them know the result.

6.2.3 Secret Sharing

Secret sharing (SS in short) is an important element in cryptography. The symmetric or asymmetric encryption more focuses on secret sharing among two parties. However, we do have situations that we need to share a secret among multiple participants, e.g., in federated learning [29].

In general, there are three types of secret sharing protocols: arithmetic sharing, Shamir's sharing, and the Goldreich-Micali-Wigderson (GMW) sharing.

The arithmetic sharing is to share a secret between two parties using the features of the number theory, and the GMW sharing mainly uses XOR to hide private value [172]. The Shamir's sharing may be the most popular primitive for secret sharing in practice, and we will discuss a bit more in detail below.

In Shamir's scheme, a secret s will be split into n shares in such a way that any t shares can be used to reconstruct the secret s; however, any set of less than t shares can obtain no information about s.

For example, in order to open a vault, three different keys are necessary to be used at the same time to complete the operation; any two keys do not work. In the early days, secret sharing mostly means sharing a key, e.g., t shares can reconstruct the key to open a vault.

We suppose that there is a dealer D who will manage the splitting operation and distribute individual shares to a set of participants, denoted as set P. We note that D does not involve in the reconstruction of s, and $|P| \geq t$. The scheme is based on a finite field $\mathbb{F}$.

The Shamir (t, n) scheme includes the following three steps:

- Initiation.
  The dealer D selects n unique elements as the participants from a finite field $\mathbb{Z}_p$, where p is a large prime and $n + 1 \leq p$. We note the n participants as x_i, $1 \leq i \leq n$. The dealer sends x_i to participant P_i, respectively. The value of x_i is known to public.
- Share distribution.
  Suppose the dealer selects a secret K from $\mathbb{Z}_p$ and then randomly selects $t - 1$ elements from $\mathbb{Z}_p$, noted as $a_1, a_2, \ldots, a_{t-1}$.
  The dealer then calculates $y_i = a(x_i)$, $i = 1, 2, \ldots, n$, given

$$a(x) = K + \sum_{j=1}^{t-1} a_j x^j \ mod \ p \tag{6.1}$$

  The dealer sends y_i as the share to participant P_i.
- Reconstruction.
  The reconstruction is to solve the equation of

$$a(x) = K + a_1 x + a_2 x^2 + \ldots + a_{t-1} x^{t-1} \tag{6.2}$$

  As each participant P_i knows a pair (x_i, y_i), therefore, when we have $t - 1$ pairs, we can obtain the secret K. However, with less than $t - 1$ pairs, we cannot decide K (there are many possibility given a large prime p).

Another popular method for secret share is using Lagrange Interpolation. Suppose we divide a secret into n shares and allow any t $(t < n)$ users to collaborate

to recover the secret, while it is impossible to extract the secret if there are maximum $t-1$ collaborators.

We use a simple example to illustrate the technique. Suppose $t=3$, then we find a polynomial with degree of 3 as follows:

$$y = f(x) = ax^3 + bx^2 + cx + d, \tag{6.3}$$

where $d = f(0)$ is the secret.

The dealer can randomly select n points on the x-axis, such $x_1, x_2, \ldots, x_n$. With the knowledge of a, b, c, d, the dealer can easily obtain n pairs of data $(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))$. He will then distribute the n pairs to n secret holders.

Through the collaboration of more than $t-1$ secret holders, they can easily solve Eq. (6.3) and obtain the secret d. At the same time, with the collaboration of less than t secret holders, they cannot figure out d.

We will present the application of secret sharing in federated learning shortly in this chapter.

6.2.4 Homomorphic Encryption

Homomorphic encryption (HE in short) is a relatively new field in the cryptography family. HE generally aims to execute addition or multiplication on encrypted data and obtain the encrypted form of the result (there is no decryption operations needed).

It is called *fully homomorphic encryption* (FHE) if the operations are beyond addition and multiplication. We have to point out that the performance of HE, especially for FHE, is an extraordinary challenge at the moment, which also hinders the deployment of the techniques in practice.

In general, let $\mathcal{M}$ denote the spaces of plaintexts and $\mathcal{E}$ for ciphertexts. An HE scheme Π= (KeyGen, Enc, Dec, Eval) can be described as follows [172]:

1. $KeyGen(1^{\lambda})$: Given the security parameter λ, this algorithm outputs a public key pk, a public evaluation key evk, and a secret key sk.
2. $Enc_{pk}(m)$: Using the public key pk, the encryption algorithm encrypts a message $m \in \mathcal{M}$ into a ciphertext $ct \in \mathcal{E}$.
3. $Dec_{sk}(ct)$: For the secret key sk and a ciphertext ct, the decryption algorithm returns a message $m \in \mathcal{M}$.
4. $Eval_{evk}(f; ct_1, \ldots, ct_k)$: Using the evaluation key evk, for a circuit $f : \mathcal{M}^k \to \mathcal{M}$, which consists of addition and multiplication gates, and a tuple of ciphertexts $(ct_1, \ldots, ct_k)$, the evaluation algorithm outputs a ciphertext $ct_0 \in \mathcal{E}$.

   We will present more detailed content on homomorphic encryption in the next chapter.

6.2.5 *Trusted Execution Environment*

The previous tools are almost purely software based, and researchers also had the effort to use hardware to provide security.

Trusted execution environment (TEE in short) aims to establish a secure computing environment in the CPU and memory locally or remotely.

For example, Intel recently added a set new instructions at the hardware level for CPU execution and memory accessing, which is called Intel Software Guard Extensions (Intel SGX). Intel SGX enables secure computation between two parties at the hardware level and provides protection to specified areas of applications from illegal access.

TEE is obviously a good effort for security, but we do not pay too much attention to the field in this book. Interested readers are encouraged to explore the field further.

6.3 Secure Multi-party Computation in Federated Learning

Under the federated learning framework, we can classify the existing work of SMPC into two categories: server based and client based.

In server-based cases, all the clients send the processed data shares to multiple servers, respectively, and we assume the servers are independent and not all corrupted [28]. On the other hand, in client-based scenarios, we usually have only a single server, and most of the effort will be invested on the client for a secure computation [29].

In terms of technology, most of the solutions are either based on Yao's garbled circuits or based on homomorphic encryption or secret sharing. The Yao's garbled circuit solution is suitable for a small number of participants, e.g., two- or three-party secure computation, while secret sharing technique enjoys hundreds of level participants in general.

6.3.1 *Masking for Data Aggregation*

Suppose we have two clients u and v who would like to upload x_u and x_v to the federated learning server, respectively. In order to cover their secrets against the server, u and v will make an agreement of a dummy data s_{uv}, and one will add the dummy data to her real data to make a covered output, e.g., $y_u = x_u + s_{uv}$, and the other will make the covered output as $y_v = x_v - s_{uv}$. At the server side, the honest and curious server can obtain the correct aggregation $x_u + x_v$ by summing the two received y_u and y_v. At the same time, the server cannot figure out the x_u and x_v from the received messages.

Let us generalize the masking method. Suppose multiple clients are participating in the task, representing by set $\mathcal{U}$. We will request each pair (u, v) in the set to generate a shared dummy vector s_{uv}. In order to progress, we also need a rule to decide who will add or subtract the shared dummy vector. One solution is that we randomly assign an order to each and every client, and then we have an ordered user list $1, 2, \ldots, |\mathcal{U}|$. For the user i of the list, he will add the shared dummy vectors of user $1, 2, \ldots, i-1$ and subtract the ones shared with user $i+1, i+2, \ldots, |\mathcal{U}|$. In other words, the masked output of user u is

$$y_u = x_u + \sum_{v \in \mathcal{U}: u<v} s_{uv} - \sum_{v \in \mathcal{U}: u>v} s_{uv} \tag{6.4}$$

We note that the number of positive or negative items for the above masking is dynamic and depends on the position of the user u in the ordered list.

At the server side, the aggregation is as follows:

$$z = \sum_{u \in \mathcal{U}} y_u = \sum_{u \in \mathcal{U}} x_u. \tag{6.5}$$

In the masking strategy, we need the participants to independently generate the dummy vectors in terms of pairs. The computing load is quite heavy for the system.

One issue we have to address in the scheme is how to deal with the dropouts of participants. As we know, a lot of federated learning applications are mobile phone based, where dropout is common due to network or battery reasons. The secret sharing is hired to address the dropout problem which we will discuss in the next subsection.

6.3.2 Secret Sharing in Federated Learning

In the context of the previous subsection, we can deploy secret sharing to deal with the dropout problem.

At the system initiation stage, a given user u will take the advantage of a secret sharing scheme to distribute his t-shares of each secrets (u has a secret with each of the other users, s_{uv}, $v \neq u$).

In the masking scheme, the aggregation can only be done correctly when all expected participants have uploaded their covered data successfully. Otherwise, the partial summation is far from the correct answer as one missing participant may cause a big discrepancy.

Suppose there are n participants in the round of learning. Once the server noticed that user u is dropout, the server needs to figure out the $n-1$ secrets related with user u, the pairwise dummy vector with each and every other users.

For example, in order to calculate the dummy vector s_{uv}, the server needs to collect at least t shares of s_{uv} from the live participants. As a result, the server has

to find at least the $t \times (n-1)$ shares collected from the live participants and then execute a recovery process to adjust the aggregation to obtain the correct answer.

There is one potential attack from the server on a user u about its true value x_u. The server could claim that u is dropout and then initiates the recovery process to obtain the shared dummy vectors; as a result, the server can extract x_u based on y_u and the dummy vectors according to Eq. (6.4).

$$x_u = y_u - \sum_{v \in \mathcal{U}: u<v} s_{uv} + \sum_{v \in \mathcal{U}: u>v} s_{uv} \tag{6.6}$$

In order to deal with these attacks, a double masking method was proposed [29, 173].

In order to address the expensive computation for the dropout recovery issue, Zheng et al. [174] proposed a lightweight cryptographic solution. The basic idea is as follows:

1. Let $\mathbb{G}$ be a cyclic group of prime order p and g is the generator of the group. Moreover, let $H : \{0,1\}^* \to \mathbb{Z}_p$ be a cryptographic hash function.
2. Each clients generated her own private key $SK_i = x_i \in \mathbb{Z}_p$ and public key $PK_i = g^{x_i}$.
3. Each client uses the aggregation cloud as a platform to broadcast their public keys; as a result, client i and j can generate a shared common key $CK_{i,j} = H((PK_j)^{x_i})$.
4. Client i and j will use the common key as the dummy vector to cover their true learning vector to be uploaded (one plus the common key and the other one subtract the common key).
5. In case user i was dropout during a given learning round, then we can get the correct summary with the help of user j as he knows the common key.

6.4 Summary of the Chapter

In this chapter, we firstly introduced the concept of secure multi-party computation (SMPC) and then listed popular building blocks for SMPC, namely, oblivious transfer, garbled circuit, secret sharing, homomorphic encryption, and trusted execution environment. These building blocks are the basic elements for establishing security protection in federated learning using cryptographic methods. We only showed the concepts of each items for readers to understand the applications in federated learning. Interested readers are encouraged to view related references to understand them deeply. We further presented the vector masking methods for privacy protection in federated learning and the secret sharing methods to deal with the dropout problem.

The application of SMPC for federated learning just started in recent years, and many problems are in the queue to be addressed, such as efficiency and performance.

There are also works on how to improve the performance of federated learning from parallel computing or communication perspectives, such as [175]. They are out of the scope of this book, but it is a good angle to address the challenges.

Chapter 7
Secure Data Aggregation in Federated Learning

In the previous chapter, we introduced secure multi-party computation with the basic knowledge of the popular building blocks and how to take the advantage of SMPC to implement the goal of federated learning through masking. Moreover, we presented how researchers use secret sharing techniques to address the dropout issue during training.

In this chapter, we will focus on the security and privacy perspectives in the core function of federated learning, namely, the data aggregation functions in distributed settings. We will emphasize on the homomorphic encryption methods and the verification methods, which are used to make sure the aggregation servers are doing the right thing.

7.1 What Is Homomorphic Encryption

In mathematics, if one structure can be mapped to another space, and the mapping function f is one to one from the original image space to the image space, and the inverse function f^{-1} exists and also one to one, then we say the two objects are *homomorphic*.

We have a lot of applications of this feature in engineering, e.g., in signal processing, machine learning, and so on.

Let us explain the concepts in the mathematical way. Let A and B be two sets, and a function $f : A \rightarrow B$. For any $a, b \in A$, if the following holds, then the mapping is *homomorphism*.

$$f(ab) = f(a)f(b) \tag{7.1}$$

For example, we take the addition operation on integer Z to form a group $\mathbb{Z}$ and the mapping function as $f : a \rightarrow e^a$. Then f maps the integer set Z to a real set R,

S. Yu, L. Cui, *Security and Privacy in Federated Learning*, Digital Privacy and Security, https://doi.org/10.1007/978-981-19-8692-5_7

and we define the operation on R as multiplication, then we have a new group $\mathbb{R}$. For any $a, b \in Z$, we have

$$f(a+b) = e^{a+b} = e^a e^b = f(a)f(b). \tag{7.2}$$

We can conclude that the two groups $\mathbb{Z}$ and $\mathbb{R}$ are homomorphic.

Roughly, a homomorphic encryption scheme can perform operations on encrypted input and yields a ciphertext containing the encrypted result.

For example, for a simple math operation $x + y = z$, homomorphic encryption enables us to have $Enc(x) + Enc(y)$ as an operation on the encrypted x and y and yield $Enc(z)$. The decryption of $Enc(z)$ equals $x + y$. Namely,

$$Enc(x) + Enc(y) = Enc(z). \tag{7.3}$$

In this way, the operator (e.g., a data aggregation server of federated learning) who carried out the operation does not know the secret of x, y, or z, as they are all in the encrypted format.

For example, in the case of the Paillier scheme, let N be the public key; the scheme possesses a beautiful property: multiplying an encryption of message m_1 and an encryption of m_2 in terms of multiplication modulo N^2, we can have a result in an encryption of $m_1 + m_2 \; mod \; N$. In other words, we obtain the encrypted form of $m_1 + m_2$. In a simple way, we can express it as follows:

$$Enc(m_1) \times Enc(m_2) \; mod \; N^2 = Enc(m_1 + m_2) \; mod \; N. \tag{7.4}$$

Fully homomorphic encryption aims to execute arbitrary operation on encrypted data, which is currently at the infant stage as the performance is extremely slow and cannot be used in practice. However, researchers have developed many partial homomorphic encryption schemes, which can implement some specific operations, such as addition or XOR on encrypted data.

Of course, the mathematical mechanism behind the homomorphic encryption is much complex. In cryptography, homomorphic encryption is based on the homomorphic feature of group theory. we recommend interested authors to the textbook of Katz and Lindell [176].

Homomorphic encryption meets the security requirement of federated learning perfectly. Therefore, we will present two popular homomorphic encryption schemes for readers in the following discussion.

7.2 Popular Homomorphic Encryption Schemes in Federated Learning

Since the appearance of public key encryption, researchers have discovered several encryption schemes based on the homomorphic philosophy, such as the Paillier encryption scheme, ElGamal scheme, Goldwasser–Micali scheme, and so on. We will present the popular schemes used in federated learning systems from the literature.

7.2.1 The Paillier Scheme

The popular Partial homomorphic encryption in federated learning is the Paillier cryptosystem, which implements the addition operation of encrypted inputs.

The Paillier encryption is based on discrete logarithm; we present the basic idea as follows:

Let p and q be two large prime numbers and $N = pq$ be the modulus. Then $\mathbb{Z}_N \times \mathbb{Z}_N^*$ is isomorphism given by

$$f(a, b) = (1 + N)^a \cdot b^N \ mod \ N^2, \tag{7.5}$$

where $\mathbb{Z}_N$ is an additive group and $\mathbb{Z}_N^*$ is a multiplication group.

Let g be the base of the cyclic group and $\mathcal{E}$ be the encryption algorithm. We can construct the Paillier encryption as the following three algorithms:

- Key generation. In the above setting, we have (N, p, q). The public key is the N, and the private key is $< N, \phi(N) >$, where the second parameter is the number of primes less than N.
- Encryption. For a message $m \in \mathbb{Z}_N$, we randomly choose $r \in \mathbb{Z}_N^*$ and obtain the cipertext $c := (1 + N)^m \cdot r^N \ mod \ N^2$.
- Decryption. The receiver can obtain the message m through

$$m := \frac{c^{\phi(N)} \ mod \ N^2 - 1}{N} \cdot \phi(N)^{-1} \ mod \ N$$

The homomorphic property of the Paillier encryption is as follows:

$\mathcal{E}(m) = g^m r^n \ mod \ N^2$ for random $r \in \{0, 1, \ldots, n - 1\}$. Then we have the following homomorphic property:

$$\begin{aligned} \mathcal{E}(m_1) \cdot \mathcal{E}(m_2) &= (g^{m_1} r_1^n)(g^{m_2} r_2^n) \ mod \ N^2 \\ &= g^{m_1 + m_2}(r_1 r_2)^n \ mod \ N^2 \\ &= \mathcal{E}(m_1 + m_2). \end{aligned} \tag{7.6}$$

7.2.2 The ElGamal Scheme

The ElGamal encryption algorithm was discovered by Taher Elgamal in 1985. It is an asymmetric key encryption algorithm based on the Diffie-Hellman key exchange problem. As a result, the mathematical mechanism is the discrete logarithm problem. Its security depends upon the difficulty of a certain problem related to computing discrete logarithms.

Let $\mathbb{G}$ be a cyclic group with order q and a generator g. For example, an integer group $\mathbb{Z}_q$, where q is a large prime as the modulus of the group. We can therefore construct the ElGamal encryption scheme as the following three steps:

- Key generation. In the aforementioned setting of $\mathbb{G}$, q, and g, take a random $x \in \mathbb{Z}_q$, and obtain $h := g^x$. Then we have the pubic key pk as $< \mathbb{G}, q, g, h >$ and the private key sk as $< \mathbb{G}, q, g, x >$.
- Encryption. For a message $m \in \mathbb{G}$, we randomly choose $y \in \mathbb{Z}_q$ and obtain the cipertext $< c_1, c_2 >=< g^y, h^y \cdot m >$
- Decryption. The receiver can obtain the message m through c_2/c_1^x based on the received ciphertext and private key.

The homomorphic property of the ElGamal encryption scheme is shown below. In the above setting of multiplication integer group, we have

$$\begin{aligned} \mathcal{E}(m_1) \cdot \mathcal{E}(m_2) &= (g^{y_1}, m_1 \cdot h^{y_1})(g^{y_2}, m_2 \cdot h^{y_2}) \\ &= (g^{y_1+y_2}, (m_1 \cdot m_2)h^{y_1+y_2}) \\ &= \mathcal{E}(m_1 \cdot m_2). \end{aligned} \tag{7.7}$$

If we replace message m with g^m in the standard ElGamal algorithm, then we can have

$$\begin{aligned} \mathcal{E}(m_1) \cdot \mathcal{E}(m_2) &= (g^{y_1}, g^{m_1} \cdot h^{y_1})(g^{y_2}, g^{m_2} \cdot h^{y_2}) \\ &= (g^{y_1+y_2}, g^{m_1+m_2}h^{y_1+y_2}) \\ &= \mathcal{E}(m_1 + m_2). \end{aligned} \tag{7.8}$$

In [106], an Elliptic curve-based ElGamal scheme was used to implement the additive homomorphic encryption.

7.2.3 The Goldwasser–Micali Scheme

In the Goldwasser–Micali cryptosystem, it can execute the XOR operation on encrypted bit as

$$\mathcal{E}(b_1) \cdot \mathcal{E}(b_2) = \mathcal{E}(b_1 \oplus b_2), \tag{7.9}$$

where b_1 and b_2 are single bit.

The Goldwasser–Micali scheme is seldom used in federated learning, and we will not discuss about it further. The point here is that we want the readers to know that there exists many other forms of homomorphic encryption in the literature.

7.3 Homomorphic Encryption for Data Aggregation in Federated Learning

As we can see, the traditional homomorphic encryption techniques can be used directly for federated learning. An example is the case of one aggregation server executing simple average aggregation. In this section, all the math operations we discuss are under the framework of the homomorphic encryption, and do not mention homomorphic encryption again and again for simplicity reason.

In [173], a double masking scheme was proposed to protect the aggregation and individual updates in the setting of one aggregation server, and Hahn et al. [177] further improved the scheme using lightweight encryption methods.

Froelicher et al. [106] implemented Drynx, a decentralized, secure, verifiable system for federated learning. Drynx is a prototype with various functions, including different aggregation functions, data audition and verification, and so on.

There are groups of excellent work in this line, but we do not plan to include them in this book as this is an introductory book. As a result, we will use a typical scheme below to present the readers the landscape of the field, and interested readers could explore more by themselves.

7.3.1 Multiple Server Data Aggregation

In practice, we may need to implement many different aggregations in practice, and we also need to deal with the case that the servers are "curious." There are some solutions in place addressing these issues, such as the Prio [28], in which clients split the uploads to hide the secret and set multiple servers as an aggregation cluster against the potential honest-but-curious attacks. We present the details in Fig. 7.1.

Suppose there are n participants in the federated learning, $P_1, P_2, \ldots, P_n$, and they need to upload the raw data $X_1, X_2, \ldots, X_n$ for a statistical aggregation. We

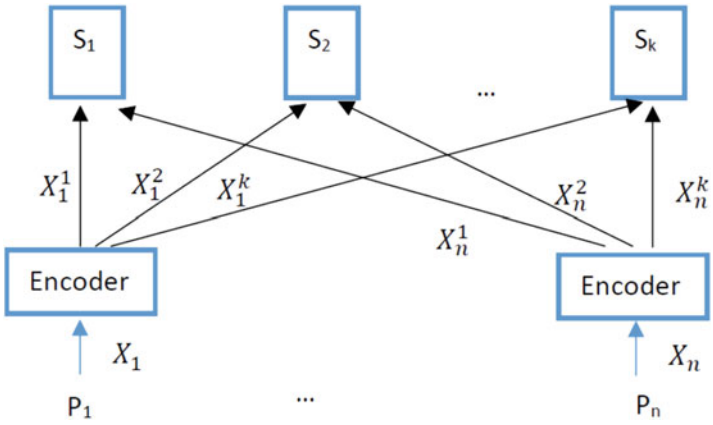

Fig. 7.1 The architecture of Prio (multiple servers for data aggregation)

set $k (k \geq 2)$ servers to aggregate the sensitive data from the n participants, while no privacy is exposed for any participants under the assumption of at least one server is not compromised.

We can set all the arithmetic operation in a finite field $\mathbb{F}$, or modulo a large prime p. The operation starts with randomly splitting any $X_i \in \mathbb{F}, i = 1, 2, \ldots, n$ into k shares, $X_i^j \in \mathbb{F}, i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, k$, subject to

$$X_i = \sum_{j=1}^{k} X_i^j. \tag{7.10}$$

Each server receives one share of each participant. For example, server S_m collects $A_m = \{X_1^m, X_2^m, \ldots, X_n^m\}$; with these information, server S_m cannot figure out the privacy of any participants. As we assume that at least one server is not compromised, the privacy holds even $k - 1$ servers collaborate together.

While all the servers collaborate together, they can obtain the true value as

$$A_1 + A_2 + \ldots + A_k = X_1 + X_2 + \ldots + X_N \tag{7.11}$$

One key feature of the Prio architecture is that participants need to persuade the servers that his uploaded shares are valid. This involves the technical of one prover to demonstrate to many verifiers that his uploads are correct without revealing any information of the proved content. We will discuss the basic idea of zero knowledge proof in the next subsection.

Greedy or malicious participants often have an incentive to cheat the aggregation function for various reasons. It is possible to protect against these attacks using zero knowledge proofs, but these protections destroy scalability: checking the proofs requires heavy public-key cryptographic operations at the servers, which will

increase the servers' workload by orders of magnitude. Prio system invited a secret-shared non-interactive proof (SNIP) protocol to address the performance problem.

More importantly, Prio can offer many mathematical aggregation for possible usage in federated learning, including integer summation, mean, variance, standard deviation, MIN, MAX, Boolean OR, Boolean AND, frequency count, and set operations.

In the Prio system, an affine-aggregatable encoding (AFE) was used to implement the aforementioned functions. An AFE includes three efficient algorithms, which are defined with respect to a field $\mathbb{F}$ and two integers k and k' with the condition of $k > k'$.

Suppose set D is the set of all possible value from the clients.

- Encode(x): maps an input x to its encoding in $\mathbb{F}^k$.
- Valid(y): outputs true if and only if $y \in \mathbb{F}^k$ is a valid encoding of some data item in D.
- Decode(σ): takes the sum of all the $Encode(x) \in \mathbb{F}^{k'}$ as input and outputs the aggregated function.

The encoding algorithm is decided by the expected aggregation function, and the summary of the decoding algorithms will generate the expected outputs.

For example, in order to calculate the variance of a random variable X, namely, $Var(X) = E[X^2] - (E[X])^2$. Each client encodes its integer x as (x, x^2) and apply the summation AFE to each of the two components.

7.3.2 Zero Knowledge Proof of Data Aggregation

Following the previous subsection, we introduce the basic knowledge of zero knowledge proof for readers here.

The zero knowledge proof is a branch of cryptography. It is a class of protocols by which a party (the prover) can make another party (the verifier) believe that his statement is true without revealing any additional information.

The most popular illustrative example is the famous Ali Baba Cave. In order to make it easy to understand, we make a simple example here for readers: Alice tells Bob that she knows the password of Chris's email box, but she does not plan to tell him the password. Alice invites Bob to work together to prove it while not revealing the password to Bob.

Bob can send a short massage to the email box of Chris and asks Alice what the message is. If Alice can answer the question successfully, then Bob believes that her statement is true. In order to make sure Alice is not successful by guessing, they can repeat the game multiple times; if Alice can almost always answer correctly, then it is true that she knows the password of Chris.

The zero knowledge proof problem was firstly studied by Andy Yao, a Turing Award winner, in the 1980s. This problem is termed the "Millionaires' Problem" as the original formulation of the problem was to determine which of the two millionaires is wealthier while not showing the account balance to each other.

Yao's Millionaires' Problem [178]: Alice has a private number a, and Bob has a private number b. The goal of the two parties is to solve the inequality $a \leq b$? without revealing the actual values of a or b, or more stringently, without revealing any information about a or b other than $a \leq b$ or $a > b$.

In the following, we use a discrete log-based protocol to demonstrate zero knowledge protocol to readers [179].

We firstly recall the discrete log problem: Let p be a large prime and g be a generator of the multiplicative group of integers modulo p. The discrete log of v with respect to generator g (or $\log_g v$) is the unique x, such that

$$g^x = v \ (mod \ p), \tag{7.12}$$

where p is a large prime number.

It is hard to compute the discrete log x even if any one is given the triple (g, v, p).

Secondly, let us see how to implement zero knowledge proof using the discrete log problem.

Let (g, v, p) be public information. Prover Alice wishes to convince the verifier Bob that she possesses $x = \log_g v$. Consider the following protocol that utilizes zero knowledge:

- Alice selects a random number $0 < r < p$ and sends the verifier Bob $h = g^r \ (mod \ p)$.
- Verifier sends a random bit $b \in \{0, 1\}$ to the Prover.
- Prover sends $a = r + bx \ (mod \ p)$ to verifier.
- Verifier checks that $g^a = hv^b \ (mod \ p)$ and accepts or rejects accordingly.
- Repeat above steps multiple times (to confirm acceptance).

Note that the equality conditions check out in that

$$g^a = g^{r+bx} = g^r g^{bx} = (g^r)(g^x)^b = hv^b \ (mod \ p). \tag{7.13}$$

Therefore, if Alice and Bob engage in a few iterations of this (with at least some $b = 1$), the verifier will be confident that the prover indeed knows x. Namely, when $b = 0$, the Verifier is providing the discrete log of h, while if $b = 1$, the verifier is providing the discrete log of hv.

This protocol is secure as it reveals no information about x to a potential eavesdropper.

7.4 Verification of Data Aggregation

Federated learning is a distributed system, the aggregation sever may not work as expected. For example, the server may do the aggregation using the data from only partial of the users in order to reduce the workload, it may broadcast a fabricated

result to achieve some malicious goals, and so on. As a result, it is necessary to have a mechanism to verify the correctness of the aggregated result.

There are two popular verification methods in federated learning: homomorphic hash function and Lagrange interpolation.

In [173] and [177], homomorphic hash function was employed for the verification task. In order to understand the method, we firstly present the homomorphic hash function as below.

Following the property of homomorphic encryption, let H be a homomorphic hash function and $f()$ be an arithmetic function. Suppose we have t messages, $m_1, m_2, \ldots, m_t$; then the following holds:

$$H(f(m_1, m_2, \ldots, m_t)) = f(H(m_1), H(m_2), \ldots, H(m_t)) \tag{7.14}$$

Specifically, a family of one-way keyed homomorphic hash H includes the following three algorithms:

- Key generation $k \leftarrow H.gen$. This algorithm generates the private key k shared by the participants for the hash function H_k.
- Hash function $H_k(m) \leftarrow H$. This algorithm outputs the result of H with an input m.
- Evaluation $H_k(f(m_1, m_2, \ldots, m_t)) \leftarrow H.eval$. This is a function evaluation algorithm that returns $H_k(f(m_1, m_2, \ldots, m_t))$ on input a set of hash values $H_k(m_1), H_k(m_2), \ldots, H_k(m_t)$.

In the federated learning practice, for a given learning round, each participant uploads his masked learning parameter for the server to execute the aggregation operation. The server will broadcast the result of aggregation $f(m_1, m_2, \ldots, m_t)$, and each participant holds $H_k(m_i), i = 1, 2, \ldots, t$. The participants can work together to see whether Eq. (7.14) holds. As a result, they know the aggregation is correct or not.

7.5 Summary of the Chapter

In this chapter, we briefly present the basic information of homomorphic encryption and list two popular homomorphic encryption schemes for federated learning. There are many works in this field, and we use a multiple-server aggregation as an example for readers to understand the necessary parts of HE in federated learning. Finally, we present the verification part of federated learning in the homomorphic encryption cases.

We have to point out that that there are many subfields to explore, given different settings. For example, some compromised clients may submit crafted data to impact the aggregation; however, the server or peers cannot check their data as the data are encrypted. How can we deal with this from different perspectives, such as traffic, server side, and so on? There are related work in the literature, but we do not explore in this book. We just point it out for our readers to explore by themselves.

Chapter 8
Anonymous Communication and Shuffle Model in Federated Learning

In the previous two chapters, we have discussed how to keep privacy through encrypting content transported in federated learning, namely, encrypt the information exchanged between different parties. Another way of privacy protection in federated learning is to disconnect the contents from their sources, in other words, hiding the ID of a piece of information, which is called *anonymous communication*.

The existing methods of anonymous communication heavily depend on cryptography and possess an inherit problem of performance, and there is an effort to address the performance issue through new technology, the *shuffle model*.

8.1 What Is Anonymous Communication?

Anonymous communication is a long-term research topic. The early work of anonymous communication can be traced back to Shannon's perfect secrecy in the 1940s [180], which paved a rocky foundation for the field. A further milestone was made by Chaum [181] for applications in the early 1980s. With the booming of the Internet, there is a demand to hide the cyber footprints when surfing the web, and the classical application platform Tor [182] was developed to meet the requirement after year 2000.

The purpose of anonymous communication is to make sure the information receiver does not know who is the owner of the message. In the foundation work of Shannon [180], people need to make every message the same to the eavesdropper to make sure he cannot identify a specific one. In terms of information theory, the defender needs to make sure the entropy of the messages is as big as possible.

Suppose the eavesdropper has observed n encrypted messages from a victim and found that they are all the same, then the probability that he can correctly identify the target one is $Pr[X = i] = \frac{1}{n}, i = 1, \ldots, n$. In other terms, a defender shall maximize the entropy of the messages. Entropy is a fundamental concept in

S. Yu, L. Cui, *Security and Privacy in Federated Learning*, Digital Privacy and Security, https://doi.org/10.1007/978-981-19-8692-5_8

information theory, which is defined as follows:

$$H(X) = -\sum_{i=1}^{n} Pr[X = i] log Pr[X = i], \tag{8.1}$$

where $Pr[X = i]$ is the probability of message i.

It is called *perfect secrecy* in Shannon's original work when all the received messages are the same.

Due to the introductory feature of this chapter, we do not involve math deeply or rigorously. For the readers who are interested about information theory, please refer to the classic textbook of Cover [183].

After the year 2000, the number of web-based activities increased dramatically and triggered a lot of privacy concerns. For example, when we visited a medical or sensitive website, the website recorded our trace, including our IP address, pages visited, and so on. This triggers a great concern of privacy and personal information. A team from MIT developed a system to anonymize communication system, Tor, to protect online traces. Tor is cryptography based, and its performance is an inherited challenge.

Around 2006, the idea of the shuffle model was proposed at a top conference of computer science [184]. The aim was to address the inherent efficiency issue of cryptography by establishing building blocks for anonymous communication. A bundle of work has been carried out about the model and formed the Encode-Shuffle-Analyze (ESA) architecture [185].

With the appearance of the federated learning framework, the shuffle model has been extensively studied recently combining with differential privacy to enhance the protection of privacy [185, 186].

8.2 Onion Routing and Tor

Tor is currently a popular application platform for anonymous communication, such as browsing privacy protection, anonymous information publishing, whistleblowing, and so on. For expression convenience, we exchangeably use Tor and anonymous communication in this chapter.

Anonymous communication mainly deals with two kinds of attacks:

- Stop a web server to identify the ID or IP address of a browser,
- Prevent the traffic analysis attack.

Traffic analysis is a powerful tool for attackers as each web page is unique, such as length of text, number of images, size of images, and so on. The uniqueness is also called fingerprint of web pages. Available research has demonstrated that the fingerprint does not change even when the page is encrypted.

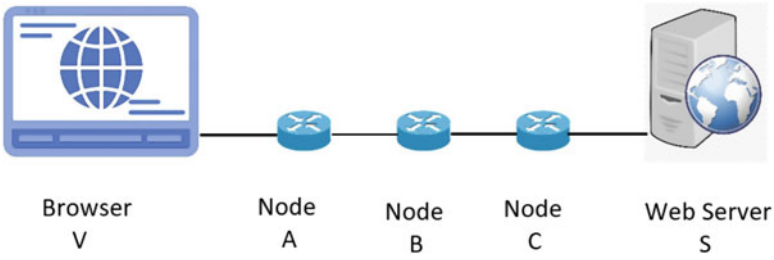

Fig. 8.1 An example of onion routing

In order to fight against traffic analysis attack, we need to make every web page looking the same to attackers through padding, namely, adding dummy packets to each and every web page of a web server. Therefore, each page has the same number of images with the same size and the same length of text in the eyes of attackers.

The cost of dummy padding is very expensive as we transport too much useless packets to cover the fingerprints. In our previous work, we used predicted web pages to be viewed to replace the dummy packets, which can reduce the cost of bandwidth waste [187].

Tor is built on the concept of onion routing, where a data source and the destination (e.g., a web browser and the targeted website) are connected through three hops, shown in Fig. 8.1.

The workflow of the Tor system is presented as follows:

1. At the initiation stage, a Tor system will recruit sufficiently large number of voluntary nodes (usually at thousands level) to establish the anonymous service.
2. When a web browser V requests Tor that she wants to access a web server S, the Tor system will select three intermediate nodes from the voluntary nodes, such as the nodes A, B, and C in Fig. 8.1.
3. Before each session of communication, a public key-based communication is employed between each pair of nodes on the path of V-A-B-C-S. For example, node A will use its private key to encrypt the message and pass it to node B, and node B can decrypt the received message using the public key of A.
4. B will continue to pass the message to C in the same way as A did and so on.
5. As a result, the server S cannot know the IP address or ID of the web browser V.

The challenges of Tor are delay and computing power inherited from the multiple public key-based relay, especially in mobile phone-based scenarios.

8.3 The Shuffle Model

As we have seen, researchers tried to propose the shuffle model as primitives to address the heavy computing problems of encryption in anonymous communication.

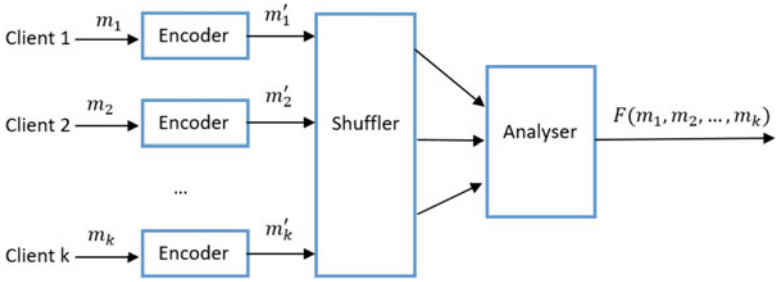

Fig. 8.2 The Encoder-Shuffler-Analyzer structure of shuffle model

The framework of shuffle model is shown in Fig. 8.2

The application scenario is like this, the server needs to collect messages from multiple clients for an aggregation function $F()$, but we need to make sure the server cannot figure out the sender of a given message m_i.

The working mechanism is as follows:

1. Client i transforms the original message m_i into $m_i^{'}$ using a local encoder algorithm.
2. Multiple inputs are collected and stored by the shuffler, and the coded messages are shuffled $m_1^{'}, m_2^{'}, \ldots, m_k^{'}$.
3. The analyzer receives the shuffled messages and executes the aggregation function to obtain the expected output.

We make two toy examples here for readers to understand how it works.

Example 1 The encoder is a function of generating cover messages: for a input m_i, the encode will output three similar messages, m_i, $+X_i$, and $-X_i$, where X_i is a random noise. After the shuffle function with a large number of inputs, it is difficult for the server to connect a given message with a specific user. At the same time, we design the aggregation function as a summary of all inputs. As a result, the cover messages cancel each other, and we obtain the expected $F(m_1, m_2, \ldots, m_k)$.

Example 2 The encoder works as a splitter, which breaks an input message into multiple sub-messages and makes sure the sum of the sub-messages equals the input message. Similar to example 1, we design the aggregation function at the analyzer side as a sum function. In this way, we can also increase the difficulty of eavesdroppers to connect a specific message to a given client.

For different applications, we need to design specific encoder and aggregation algorithms. There are profound conclusions and tools in terms of information theory and coding theory for us to apply the encoder and aggregation functions. At the same time, there are plenty of mixing strategies for shufflers in the literature.

We remind the reader to pay attention on network coding [188], which is quite close to the content we discuss here.

The essential purpose of shuffle model is to hide user IDs through encoding and shuffling, which may not be as practical as Tor in applications, but it is a good direction to explore.

The shuffle model is essentially an extension of the coding theory. As a result, there are a number of works to modeling and analyzing the models from the algorithm perspectives, such as error rate and upper and lower bounds of complexity [189, 190].

An integrated theoretical analysis on privacy protection, accuracy of federated learning, and communication cost could be found at [31, 191].

We note that the theoretical analysis usually use differential privacy concept as a metric for their analysis, while in shuffle model, there is usually no noise involved.

8.4 Privacy Amplification in Federated Learning

Federated learning is essentially a distributed collaborating system with multiple participants for a given task. In the literature, researchers have found that we can gain better privacy protection through some technical effort, which is called *privacy amplification*.

In the early days, researchers noticed that in the multiple participant case of machine learning, e.g., the federated learning case, we can obtain better privacy protection gain through local differential privacy on top of *subsampling* [192].

Mathematically, let $\mathcal{M}$ be mechanism of the local differential privacy and $\mathcal{S}$ be the subsampling process. For a given data set x, then the output is the composition of the two operations, $\mathcal{M} \circ \mathcal{S} = \mathcal{M}(\mathcal{S}(x))$.

Suppose the total population of a client dataset is n, and we will take $m (m < n)$ samples for learning in an iteration. In general, there are a couple of popular subsampling methods as listed below:

- Poisson sampling with parameter *gamma*
- Sampling without replacement
- Sampling with replacement

The amplification efficiency of the subsampling methods is simply summarized in Table 8.1.

Table 8.1 A summary of privacy amplification bounds

Subsampling	Amplification parameter
Poisson (γ)	γ
With replacement (n, m)	$\frac{m}{n}$
Without replacement (n, m)	$\frac{m}{n}$

In practice, Zhao et al. [193] applied subsampling not only on the participants but also on the uploaded parameters. Their result shows that the strategy offers a better performance in federated learning while keeping the accuracy of learning.

The theoretical analysis has shown that message shuffling can also achieve privacy amplification, which is similar to subsampling. Because the complexity of the mathematical effort is out of the scope of this chapter, we refer interested readers to the related papers available in the literature, e.g., [194].

8.5 Anonymous Communication and Shuffle Model in Federated Learning

In federated learning, people noticed that local differential privacy is not enough to defeat attackers as many query iterations and high dimensionality will degrade privacy and then add shuffle on top of local differential privacy to enhance privacy protection [195]. There are two methods to shuffle the updates:

1. For the K participants, the shuffler mixed them before relaying the k vectors to the aggregation server.
2. The participant splits the updates first and then mixed by the shuffler, and finally the aggregator will reassemble the vectors respectively for a further aggregation. This method is call parameter shuffle by the authors.

8.6 Summary of the Chapter

In this chapter, we briefly presented the concepts of anonymous communication and the shuffle model and their implementation techniques, respectively. Compared to the content encryption methods presented in previous chapters, these methods are alternative ways to guarantee the privacy of participants in federated learning, as they try to hide the participants' IDs from potential attackers.

We note that anonymous communication is one part of communication security, and it is mainly based on cryptography. As a result, we only present the basic concepts and mechanism in this book for readers to know it, which they may use in their work.

Chapter 9
The Future Work

At the end of this book, we would like to present the future work in line of the theme that we have discussed so far under the big picture of the information age.

In general, security research is the second phase of research for applications. For example, when we talk about auto-driving cars, the first phase is how to make cars autopilot well, and then people will shift the focus to security and privacy, e.g., how to prevent possible car hijacks by hackers, how to protect location privacy of passengers, and so on. As a result, it is necessary to have a look what will happen in the near future where security and privacy will set in.

9.1 The Related Landscape in the Near Future

It is no doubt that ICT is developing fast. One simple evidence is the Internet, which has become an indispensable infrastructure for our daily lives and businesses within three decades. We can predict that the time will be much shorter for new ICT applications from concepts to practice.

There are many new concepts and ideas from time to time in ICT sector. We list some of them below:

- Metaverse
- Semantic communications
- Digital twin
- Blockchain technology
- Quantum computing
- Autopilot vehicles
- Industrial Internet
- Internet of sense
- Internet of AI
- . . .

S. Yu, L. Cui, *Security and Privacy in Federated Learning*, Digital Privacy and Security, https://doi.org/10.1007/978-981-19-8692-5_9

The list could be much longer than we list here, and it is growing with the flying of time.

One thing is sure that all the new applications mainly integrate communication and AI. As a result, let us have a look at the challenges and trends of these two branches separately.

Communication technology is moving from technical to semantic. Shannon listed three phases of communication: technical problem, semantic problem, and effectiveness problem [196]. To date, we have addressed the technical problem quite well, such as accuracy of information delivery, delay, and so on. At the moment, we are moving toward the second level of communication, addressing the semantic problem.

For example, when Alice sends an image to Bob, the current technology can guarantee to transmit the image accurately and timely (namely, we solved the technical problem of information transportation). However, what is the meaning of the image depends on the interpretation of Bob, which may be different from the intention of Alice. This is a semantic problem, which is expected to be tackled onwards. We are at the doorstep of this semantic communication.

It is certain that we need tools for new problems. For semantic communications, we can certainly study it using our existing tools, like information theory, group theory, graph theory, and so on. At the same time, it is also certain that we will see new tools, and improved old tools will be introduced to researchers. Based on our knowledge, we would like to recommend information bottleneck [197] to interested readers as a starting point.

AI also faces significant challenges or problems, although it is extremely successful in the past decades with the availability of big data and computing power. On top of the limited fundamental improvement in the recent two decades [198], the biggest challenge on AI is explainability [199]. The key reason of the unexplainability of AI is that we do not understand AI well, such as the mechanism of learning of neural networks. We recommend readers to pay a bit of attention to the recent developed field of *geometric deep learning*, which tries to represent deep neural networking in a theoretical way.

We can guess that AI is still at her early stage although we have witnessed the significant development of AI applications in various sectors of human society. Under such environment of AI, security and privacy research is hard to be deep or fundamental.

In this section, we would like to boldly share some of our understanding about the near future landscape of security and privacy in AI, including the federated learning environment.

9.2 The Challenges on Security in the Near Future

Security will definitely become a critical part of our daily lives. As people and human society are more and more digitalized and information devices are embedded in our lives and businesses, we leave more and more traces in cyberspace. All these

will benefit human society; at the same time, they will be a critical threat to us if they are controlled or manipulated by malicious hands.

9.2.1 Existing Attacks

Security has been researched and practiced for centuries. We have seen various attacks and defenses in the past. Majority of the existing attack methods and methodologies will certainly be applied to AI applications, such as distributed denial of service attacks, man-in-the-middle attacks, and so on.

Motivated by financial or political rewards, hackers will update their attack methods or methodologies for new environments to achieve their malicious goals. As a result, defenders surely need to improve or update counter-attack methods or methodologies. It is an endless looping game between the two sides.

9.2.2 Digital Forensics

Forensics is a critical part in justice systems in human society. We can predict that there will be more and more criminal activities or argument in cyberspace; thus, digital forensics will play an important role to assist law and order, settle argument, and so on.

In multimedia space, deep fake is a great challenge. For example, attackers use available AI techniques to forge images, videos, and voices, which are hard to be differentiated by human or majority of the existing tools. The need to develop forensic tools to defeat deep fake attacks effectively and timely is of urgency.

In general, the progress in digital forensics is still very limit. For example, in the physical world, forensics possesses very effective and efficient methods, such as DNA-based methods to identify criminals. In our digital world, the major job we can do is recover deleted data, discriminate genuine images or videos from fake ones, and so on.

We believe there will be a big boost in digital forensics, given the dramatic development of ICT infrastructure and applications, but not sure when and how at the moment.

9.2.3 New Attacks

We can predict that new attack methods will be invented, given new application scenarios. For example, attackers can easily impersonate victims or make fake but indistinguishable photos, videos, and voices, given the advancement of technology.

One trend of the new invented attacks will be the combination of social engineering and ICT technology. The solutions will also depend on the knowledge of social science and ICT technology.

Another feature of the new attacks will be timing. Attackers will aim at new applications, such as metaverse applications, which are usually with no or very limited security protection at their infant stages.

9.3 The Challenges on Privacy in the Near Future

We would like to discuss more on privacy here as it is a relatively new area and many things are unclear to researchers and practitioners. The list of challenges of privacy study could be very long, but we only present here those on the top of our list.

9.3.1 Privacy Measurement

The first step of privacy study should be the measurement of privacy. Following the famous saying of Galilei, "measure what is measurable, and make measurable what is not so," We shall measure the dimension of privacy before any further analysis and operation.

However, the current privacy measurement methods are all indirect, and there is no direct measurement methods.

K-anonymity is a popular measure on privacy in information retrieval. For example, in order to protect the privacy of Alice in a dataset, we can employ data generalization technique to create a subset of $k-1$ persons who all possess the same characteristics as Alice. As a result, an attacker can identify Alice from the dataset with a probability of 1/k. K-anonymity is a very loose definition on privacy.

Another class of measurement is based on system changes in terms of information metrics before and after an operation, for example, the variation of entropy or mutual information with and without a privacy protection action.

Strictly, privacy measurement should be the task for psychologists, social scientists, lawmakers, and so on, rather than the core task of computer scientists. In other words, we should integrate the knowledge of social science and digital technology to tackle the challenge of privacy measurement. A recent survey on research papers published in the recent 100 years on Science magazine demonstrates that more than 60% of them are cross-disciplinary research. We therefore encourage our readers to explore cross technology or cross-discipline methodology for their studied subjects if it is applicable.

9.3.2 Big Data Modelling

Big data is the background of digital privacy; therefore, it is necessary to understand big data well when we expect to have high-quality solution on privacy.

We list a few tools here for readers for their further exploration.

1. Graph theory. Graph theory is the most popular tool in the minds of people for big data modelling. In theory, it is straightforward that we can transform a big dataset into a big graph, and we can further analyze or manipulate the graphs using the amazing research outputs of graph theory. However, we face several challenges in practice as the existing graph theory does not meet the application requirements of modern networking.

   The first problem is the dynamic feature of modern networking. We know the studied object of the traditional graph theory is static graphs, while one important feature of modern networks is the dynamic property, such as the nodes or edges of social networks or the Internet are variable with the progress of time (also called time-varying networks). We have seen researchers working on this direction trying to address the problem.

   The second problem is scale of graph. It is not a problem in theory in terms of scale of graphs. However, we need to chop a big graph into suitable size for multiple computers to process it. As a result, we need to study how to divide a big graph into many sub-graphs for parallel processing, co-flow scheduling, and so on.

   Third, graph theory falls in the geometry domain, and not very suitable for computing in general. As a result, it is good to use the algebra methodology. For example, we can represent a big graph using adjacent matrix, and then we can process the big graph using algebra methods, which we will discuss below.
2. Tensor. Tensor is an element in differential geometry, which is a powerful tool for physicists in studying high dimension space, such as Einstein. Tensor is suitable for high-dimensional dataset and has been used in computer science, AI, networking, and so on.
3. Statistics. Many AI and communication models are based on probability, and small probability events are usually ignored in modeling in the past. However, with the appearance of personalized recommendation demanding, small probability study becomes necessary. For this aspect, we recommend readers to read related books on large deviation, which has been used in ICT domains for performance evaluation.

   We believe there are many tools invented by mathematicians and physicists for their research, and our task is to learn and apply their tools to address our problems. At the same time, we believe our needs will push more tools to be invented.

9.3.3 Privacy Tools

Appropriate tools are the critical elements for solving problems. When a new research topic appears, people will try to use existing tools to tackle the problems. However, it would not be that smooth to address new problems using existing tools. Researchers usually will improve the available tools or event invent new tools to meet the challenges.

We present the popular tools for privacy below.

1. Cryptography. Cryptography is the first tool in our minds to address the problems of digital privacy. Of course, it works due to the essential connection between secret sharing and privacy protection.

   However, people find that the profound existing cryptography tools do not match the new and specific needs of digital privacy protection;s the reason is the difference of application scenarios.

   If we look at the history or motivation of cryptography, we can see that it was invented to share secrets between two parties through private keys, public keys, or attribute-based keys. In other words, we know the information senders and receivers. However, under the scenarios of digital privacy protection, we want to share some "useful" information with the public whoever is interested but hide the sensitive or part of information of the source. Unfortunately, the current cryptography tools cannot offer the functions. As a result, it is time to improve or modify the current cryptography tools to meet the requirement of digital privacy protection.
2. Differential privacy. Differential privacy was invented around two decades ago for privacy protection in statistical information retrieval. Based on our knowledge, it may be the only popular tool invented for privacy protection. Due to the huge demand in privacy protection, people are using differential privacy everywhere, such as adding noise in each round of deep learning to protect the privacy of data sources. Unfortunately, differential privacy is not the magical wand to solve everything.

   As a result, we need many different tools for digital privacy, and that can only be achieved by improving the existing ones or inventing new tools.

9.3.4 Personalized Privacy

Digital privacy has developed quickly in the past two decades, and most of the solutions possess an assumption: all users share the same privacy requirement in their solutions. However, we have to make privacy protection personalized in practice as privacy is a very subjective concept, which varies dramatically on culture, gender, environment, timing, and so on. For example, professors' salary is a top secret in the USA, but it is a public number in Australia.

The current research on personalized privacy is mainly in social networks. In social networks, we can sort of measure the distance (social distance) between pairs, and then we can decide how much privacy should be protected or released when a pair exchange information. For example, Alice releases her photos with location information in her social networks. She can add a random variable (e.g., 10km, 100km, and so on) to her real location according to the social distance between herself and the information receivers, respectively. In this way, the location privacy of Alice can be personalized. For the interested readers, we recommend an early book on personalized privacy [27].

We have to point out that the research on personalized privacy is still in infant stage. The current personalized privacy research pays a lot of energy on measuring social distances, not on privacy perspective yet. We believe it is a great direction for research and application in the near future.

9.3.5 AI Ethics

It is certain that AI ethics is an important and promising research field given the extensive AI application in our daily lives [200]. AI ethics includes many directions, such as fairness in AI, algorithm bias, and so on.

One thing is sure that research on AI ethics should be executed in a cross-disciplinary way, rather than based on technology only. Currently, cross-technology and cross-discipline study are extensively demanded and promoted by government agencies and businesses because we are facing the related problems.

Based on our experience, cross-disciplinary research is beautiful, but not easy as there are huge gaps between different disciplines, respectively. For example, the combination of cybersecurity technology with criminal psychology is an amazing picture before us, but we have different settings, backgrounds, and methodologies for the picture from computer scientists and psychologists, and sometimes, it is not easy to understand each other. We believe it is time for us to train the forthcoming persons to be cross-disciplinary researchers.

We believe security and privacy research will walk toward cross discipline as pure ICT technology cannot identify or address the related problems in the near future.

9.4 Summary of the Chapter

In this chapter, we mainly share our understanding on security and privacy in ICT in the near future for readers. As it is hard to predict the picture of tomorrow, it is not necessary for readers to follow the suggestions but for reference.

One thing is sure is that we will face many unprecedented problems and challenges, and theoretical solutions are highly expected from the research communities. We humbly hope the content of this chapter is helpful for energetic readers to explore the exciting fields in the near future.

References

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.
2. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444.
3. Deng, L., Yu, D., et al. (2014). Deep learning: Methods and applications. *Foundations and Trends® in Signal Processing, 7*(3–4), 197–387.
4. Dong, S., Wang, P., & Abbas, K. (2021). A survey on deep learning and its applications. *Computer Science Review, 40*, 100379.
5. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*(6088), 533–536.
6. Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., & Rellermeyer, J. S. (2020). A survey on distributed machine learning. *ACM Computing Surveys (CSUR), 53*(2), 1–33.
7. Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST), 10*(2), 1–19.
8. Mothukuri, V., Parizi, R. M., Pouriyeh, S., Huang, Y., Dehghantanha, A., & Srivastava, G. (2021). A survey on security and privacy of federated learning. *Future Generation Computer Systems, 115*, 619–640.
9. Chen, M., Yang, Z., Saad, W., Yin, C., Poor, H. V., & Cui, S. (2020). A joint learning and communications framework for federated learning over wireless networks. *IEEE Transactions on Wireless Communications, 20*(1), 269–283.
10. McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics* (pp. 1273–1282). PMLR.
11. Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C., & Ramage, D. (2018). Federated learning for mobile keyboard prediction. arXiv preprint arXiv:1811.03604.
12. Sheller, M. J., Edwards, B., Reina, G. A., Martin, J., Pati, S., Kotrotsou, A., Milchenko, M., Xu, W., Marcus, D., Colen, R. R., et al. (2020). Federated learning in medicine: Facilitating multi-institutional collaborations without sharing patient data. *Scientific Reports, 10*(1), 1–12.
13. Zhang, C., Cui, L., Yu, S., & Yu, J. J. (2021). A communication-efficient federated learning scheme for IoT-based traffic forecasting. *IEEE Internet of Things Journal*.
14. Cui, L., Qu, Y., Xie, G., Zeng, D., Li, R., Shen, S., & Yu, S. (2021). Security and privacy-enhanced federated learning for anomaly detection in IoT infrastructures. *IEEE Transactions on Industrial Informatics, 18*(5), 3492–3500.
15. Zhang, K., Song, X., Zhang, C., & Yu, S. (2022). Challenges and future directions of secure federated learning: A survey. *Frontiers of Computer Science, 16*(5), 1–8.

S. Yu, L. Cui, *Security and Privacy in Federated Learning*, Digital Privacy and Security, https://doi.org/10.1007/978-981-19-8692-5

16. Yin, X., Zhu, Y., & Hu, J. (2021). A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Computing Surveys (CSUR), 54*(6), 1–36.
17. Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine, 37*(3), 50–60.
18. Pang, J., Huang, Y., Xie, Z., Han, Q., & Cai, Z. (2020). Realizing the heterogeneity: A self-organized federated learning framework for IoT. *IEEE Internet of Things Journal, 8*(5), 3088–3098.
19. Bouacida, N., & Mohapatra, P. (2021). Vulnerabilities in federated learning. *IEEE Access, 9*, 63229–63249.
20. Niknam, S., Dhillon, H. S., & Reed, J. H. (2020). Federated learning for wireless communications: Motivation, opportunities, and challenges. *IEEE Communications Magazine, 58*(6), 46–51.
21. Nasr, M., Shokri, R., & Houmansadr, A. (2019). 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)* (pp. 739–753). IEEE.
22. Tolpegin, V., Truex, S., Gursoy, M. E., & Liu, L. (2020). Data poisoning attacks against federated learning systems. In *European symposium on research in computer security* (pp. 480–501). Springer.
23. Fang, M., Cao, X., Jia, J., & Gong, N. (2020). Local model poisoning attacks to Byzantine-Robust federated learning. In *29th USENIX Security Symposium (USENIX Security 20)* (pp. 1605–1622).
24. Zhang, J., Chen, B., Cheng, X., Binh, H. T. T., & Yu, S. (2020). Poisongan: Generative poisoning attacks against federated learning in edge computing systems. *IEEE Internet of Things Journal, 8*(5), 3310–3322.
25. Dwork, C. (2008). Differential privacy: A survey of results. In *International conference on theory and applications of models of computation* (pp. 1–19). Springer.
26. Lyu, L., Yu, H., Ma, X., Sun, L., Zhao, J., Yang, Q., & Yu, P. S. (2020). Privacy and robustness in federated learning: Attacks and defenses. arXiv preprint arXiv:2012.06337.
27. Qu, Y., Nosouhi, M. R., Cui, L., & Yu, S. (2021). *Personalized privacy protection in big data*. Springer.
28. Corrigan-Gibbs, H., & Boneh, D. (2017). Prio: Private, robust, and scalable computation of aggregate statistics. In A. Akella & Howell, J. (Eds.), *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27–29, 2017* (pp. 259–282). USENIX Association.
29. Bonawitz, K. A., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., & Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. In B. M. Thuraisingham, D. Evans, T. Malkin & D. Xu (Eds.), *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30–November 03, 2017* (pp. 1175–1191). ACM.
30. Elkordy, A. R., & Avestimehr, A. S. (2022). Heterosag: Secure aggregation with heterogeneous quantization in federated learning. *IEEE Transactions on Communications, 70*(4), 2372–2386.
31. Girgis, A., Data, D., Diggavi, S., Kairouz, P., & Suresh, A. T. (2021). Shuffled model of differential privacy in federated learning. In *International Conference on Artificial Intelligence and Statistics* (pp. 2521–2529). PMLR.
32. Yan, H., Li, X., Li, H., Li, J., Sun, W., & Li, F. (2021). Monitoring-based differential privacy mechanism against query flooding-based model extraction attack. *IEEE Transactions on Dependable and Secure Computing*.
33. Zhu, Y., Cheng, Y., Zhou, H., & Lu, Y. (2021). Hermes attack: Steal DNN models with lossless inference accuracy. In *30th USENIX Security Symposium (USENIX Security 21)*.
34. Tasumi, M., Iwahana, K., Yanai, N., Shishido, K., Shimizu, T., Higuchi, Y., Morikawa, I., & Yajima, J. (2021). First to possess his statistics: Data-free model extraction attack on tabular data. arXiv preprint arXiv:2109.14857.

35. Truong, J.-B., Maini, P., Walls, R. J., & Papernot, N. (2021). Data-free model extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4771–4780).
36. Aono, Y., Hayashi, T., Wang, L., Moriai, S., et al. (2017). Privacy-preserving deep learning: Revisited and enhanced. In *International Conference on Applications and Techniques in Information Security* (pp. 100–110). Springer.
37. Aono, Y., Hayashi, T., Wang, L., Moriai, S., et al. (2017). Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security, 13*(5):1333–1345.
38. Stanford deep learning tutorial.
39. Zhu, L., & Han, S. (2020). Deep leakage from gradients. In *Federated learning* (pp. 17–31). Springer.
40. Zhao, B., Mopuri, K. R., & Bilen, H. (2020). idlg: Improved deep leakage from gradients. arXiv preprint arXiv:2001.02610.
41. Geiping, J., Bauermeister, H., Dröge, H., & Moeller, M. (2020). Inverting gradients–how easy is it to break privacy in federated learning? arXiv preprint arXiv:2003.14053.
42. Charpiat, G., Girard, N., Felardos, L., & Tarabalka, Y. (2019). Input similarity from the neural network perspective. In *NeurIPS 2019-33th Annual Conference on Neural Information Processing Systems*.
43. Koh, P. W., & Liang, P. (2017). Understanding black-box predictions via influence functions. In *International Conference on Machine Learning* (pp. 1885–1894). PMLR.
44. Yin, H., Mallya, A., Vahdat, A., Alvarez, J. M., Kautz, J., & Molchanov, P. (2021). See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 16337–16346).
45. Jeon, J., Lee, K., Oh, S., Ok, J., et al. (2021). Gradient inversion with generative image prior. *Advances in Neural Information Processing Systems, 34*, 29898–29908.
46. Lim, J. Q., & Chan, C. S. (2021). From gradient leakage to adversarial attacks in federated learning. In *2021 IEEE International Conference on Image Processing (ICIP)* (pp. 3602–3606). IEEE.
47. Wen, Y., Geiping, J., Fowl, L., Goldblum, M., & Goldstein, T. (2022). Fishing for user data in large-batch federated learning via gradient magnification. arXiv preprint arXiv:2202.00580.
48. Hitaj, B., Ateniese, G., & Perez-Cruz, F. (2017). Deep models under the GAN: Information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 603–618).
49. Wang, Z., Song, M., Zhang, Z., Song, Y., Wang, Q., & Qi, H. (2019). Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications* (pp. 2512–2520). IEEE.
50. Fredrikson, M., Jha, S., & Ristenpart, T. (2015). Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (pp. 1322–1333).
51. He, Z., Zhang, T., & Lee, R. B. (2019). Model inversion attacks against collaborative inference. In *Proceedings of the 35th Annual Computer Security Applications Conference* (pp. 148–162).
52. Tanuwidjaja, H. C., Choi, R., Baek, S., & Kim, K. (2020). Privacy-preserving deep learning on machine learning as a service—a comprehensive survey. *IEEE Access, 8*, 167425–167447.
53. Aïvodji, U., Gambs, S., & Ther, T. (2019). Gamin: An adversarial approach to black-box model inversion. arXiv preprint arXiv:1909.11835.
54. De Cristofaro, E. (2021). A critical overview of privacy in machine learning. *IEEE Security and Privacy, 19*(4), 19–27.
55. Prakash, P., Ding, J., Li, H., Errapotu, S. M., Pei, Q., & Pan, M. (2020). Privacy preserving facial recognition against model inversion attacks. In *GLOBECOM 2020-2020 IEEE Global Communications Conference* (pp. 1–6). IEEE.

56. Ateniese, G., Mancini, L. V., Spognardi, A., Villani, A., Vitali, D., & Felici, G. (2015). Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks, 10*(3), 137–150.
57. Ganju, K., Wang, Q., Yang, W., Gunter, C. A., & Borisov, N. (2018). Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (pp. 619–633).
58. Zaheer, M., Kottur, S., Ravanbhakhsh, S., Póczos, B., Salakhutdinov, R., & Smola, A. J. (2017). Deep sets. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (pp. 3394–3404).
59. Melis, L., Song, C., De Cristofaro, E., & Shmatikov, V. (2019). Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)* (pp. 691–706). IEEE.
60. Chase, M., Ghosh, E., & Mahloujifar, S. (2021). Property inference from poisoning. arXiv preprint arXiv:2101.11073.
61. Mahloujifar, S., Ghosh, E., & Chase, M. (2022). Property inference from poisoning. In *2022 IEEE Symposium on Security and Privacy (SP)* (pp. 1569–1569). IEEE Computer Society.
62. Naveed, M., Kamara, S., & Wright, C. V. (2015). Inference attacks on property-preserving encrypted databases. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (pp. 644–655).
63. Stock, J., Wettlaufer, J., Demmler, D., & Federrath, H. (2022). Property unlearning: A defense strategy against property inference attacks. arXiv preprint arXiv:2205.08821.
64. Zhou, J., Chen, Y., Shen, C., & Zhang, Y. (2021). Property inference attacks against GANs. arXiv preprint arXiv:2111.07608.
65. Homer, N., Szelinger, S., Redman, M., Duggan, D., Tembe, W., Muehling, J., Pearson, J. V., Stephan, D. A., Nelson, S. F., & Craig, D. W. (2008). Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS Genetics, 4*(8), e1000167.
66. Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)* (pp. 3–18). IEEE.
67. Salem, A., Zhang, Y., Humbert, M., Berrang, P., Fritz, M., & Backes, M. (2019). Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS)*.
68. Yeom, S., Giacomelli, I., Fredrikson, M., & Jha, S. (2018). Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)* (pp. 268–282). IEEE.
69. Long, Y., Bindschaedler, V., Wang, L., Bu, D., Wang, X., Tang, H., Gunter, C. A., & Chen, K. (2018). Understanding membership inferences on well-generalized learning models. arXiv preprint arXiv:1802.04889.
70. Li, Z., & Zhang, Y. (2020). Label-leaks: Membership inference attack with label. arXiv e-prints (p. arXiv–2007).
71. Choquette-Choo, C. A., Tramer, F., Carlini, N., & Papernot, N. (2021). Label-only membership inference attacks. In *International Conference on Machine Learning* (pp. 1964–1974). PMLR.
72. Hayes, J., Melis, L., Danezis, G., & De Cristofaro, E. (2017). Logan: Membership inference attacks against generative models. arXiv preprint arXiv:1705.07663.
73. Hilprecht, B., Härterich, M., & Bernau, D. (2019). Monte Carlo and reconstruction membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies, 2019*(4), 232–249.
74. Bai, Y., Chen, T., & Fan, M. (2021). A survey on membership inference attacks against machine learning. *Management, 6*, 14.

75. Hu, H., Salcic, Z., Sun, L., Dobbie, G., Yu, P. S., & Zhang, X. (2021). Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)*.
76. Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)* (pp. 3–18). IEEE.
77. Jayaraman, B., & Evans, D. (2019). Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security 19)* (pp. 1895–1912).
78. Truex, S., Liu, L., Gursoy, M. E., Wei, W., & Yu, L. (2019). Effects of differential privacy and data skewness on membership inference vulnerability. In *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)* (pp. 82–91). IEEE.
79. Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.
80. Shejwalkar, V., & Houmansadr, A. (2019). Reconciling utility and membership privacy via knowledge distillation. arXiv e-prints (p. arXiv–1906).
81. Kaya, Y., Hong, S., & Dumitras, T. (2020). On the effectiveness of regularization against membership inference attacks. arXiv preprint arXiv:2006.05336.
82. Barreno, M., Nelson, B., Joseph, A. D., & Tygar, J. D. (2010). The security of machine learning. *Machine Learning, 81*(2), 121–148.
83. Barreno, M., Nelson, B., Sears, R., Joseph, A. D., & Tygar, J. D. (2006). Can machine learning be secure? In F. Lin, D. Lee, B. P. Lin, S. Shieh & S. Jajodia (Eds.), *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2006, Taipei, Taiwan, March 21–24, 2006* (pp. 16–25). ACM.
84. Gu, T., Dolan-Gavitt, B., & Garg, S. (2017). Badnets: Identifying vulnerabilities in the machine learning model supply chain. CoRR, abs/1708.06733.
85. Tian, Z., Cui, L., Liang, J., & Yu, S. (2022). A comprehensive survey on poisoning attacks and countermeasures in machine learning. ACM Computing Surveys. Accepted
86. Ribeiro, M., Grolinger, K., & Capretz, M. A. M. (2015). Mlaas: Machine learning as a service. In T. Li, L. A. Kurgan, V. Palade, R. Goebel, A. Holzinger, K. Verspoor & M. A. Wani (Eds.), *14th IEEE International Conference on Machine Learning and Applications, ICMLA 2015, Miami, FL, USA, December 9–11, 2015* (pp. 896–902). IEEE.
87. Deng, J., Dong, W., Socher, R., Li, L., Li, K., & Li, F. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA* (pp. 248–255). IEEE Computer Society.
88. Biggio, B., Nelson, B., & Laskov, P. (2011). Support vector machines under adversarial label noise. In *Asian Conference on Machine Learning* (pp. 97–112). PMLR.
89. Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings*. OpenReview.net.
90. Nelson, B., Barreno, M., Chi, F. J., Joseph, A. D., Rubinstein, B. I. P., Saini, U., Sutton, C., Tygar, J. D., & Xia, K. (2008). Exploiting machine learning to subvert your spam filter. In F. Monrose (Ed.), *First USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET '08, San Francisco, CA, USA, April 15, 2008, Proceedings*. USENIX Association.
91. Chen, X., Liu, C., Li, B., Lu, K., & Song, D. (2017). Targeted backdoor attacks on deep learning systems using data poisoning. CoRR, abs/1712.05526.
92. Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E. C., & Roli, F. (2017). Towards poisoning of deep learning algorithms with back-gradient optimization. In B. M. Thuraisingham, B. Biggio, D. M. Freeman, B. Miller & A. Sinha (Eds.), *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2017, Dallas, TX, USA, November 3, 2017* (pp. 27–38). ACM.
93. Saha, A., Subramanya, A., & Pirsiavash, H. (2020). Hidden trigger backdoor attacks. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second*

Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7–12, 2020 (pp. 11957–11965). AAAI Press.

94. Li, C., Chen, X., Wang, D., Wen, S., Ahmed, M. E., Camtepe, S., & Xiang, Y. (2022). Backdoor attack on machine learning based android malware detectors. *IEEE Transactions on Dependable and Secure Computing, 19*(5), 3357–3370.
95. Bagdasaryan, E., & Shmatikov, V. (2020). Blind backdoors in deep learning models. CoRR, abs/2005.03823.
96. Lamport, L., Shostak, R. E., & Pease, M. C. (2019). The byzantine generals problem. In *Concurrency: The works of Leslie Lamport* (pp. 203–226). ACM.
97. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., & Shmatikov, V. (2020). How to backdoor federated learning. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26–28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research* (pp. 2938–2948). PMLR.
98. Bhagoji, A. N., Chakraborty, S., Mittal, P., & Calo, S. B. (2019). Analyzing federated learning through an adversarial lens. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9–15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research* (pp. 634–643). PMLR.
99. Singh, A., Ngan, T., Druschel, P., & Wallach, D. S. (2006). Eclipse attacks on overlay networks: Threats and defenses. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 23–29 April 2006, Barcelona, Catalunya, Spain*. IEEE.
100. Xie, C., Huang, K., Chen, P., & Li, B. (2020). DBA: Distributed backdoor attacks against federated learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020*. OpenReview.net.
101. Blanchard, P., Mhamdi, E. M. E., Guerraoui, R., & Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA* (pp. 119–129).
102. Yin, D., Chen, Y., Ramchandran, K., & Bartlett, P. L. (2018). Byzantine-robust distributed learning: Towards optimal statistical rates. In J. G. Dy & A. Krause (Eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsm"assan, Stockholm, Sweden, July 10–15, 2018*, volume 80 of *Proceedings of Machine Learning Research* (pp. 5636–5645). PMLR.
103. Mhamdi, E. M. E., Guerraoui, R., & Rouault, S. (2018). The hidden vulnerability of distributed learning in byzantium. In J. G. Dy & A. Krause (Eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsm"assan, Stockholm, Sweden, July 10–15, 2018*, volume 80 of *Proceedings of Machine Learning Research* (pp. 3518–3527). PMLR.
104. Shen, S., Tople, S., & Saxena, P. (2016). Auror: Defending against poisoning attacks in collaborative deep learning systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications, ACSAC 2016* (pp. 508–519). ACM.
105. Baruch, G., Baruch, M., & Goldberg, Y. (2019). A little is enough: Circumventing defenses for distributed learning. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B.Fox & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8–14, 2019, Vancouver, BC, Canada* (pp. 8632–8642).
106. Froelicher, D., Troncoso-Pastoriza, J. R., Sousa, J. S., & Hubaux, J. (2020). Drynx: Decentralized, secure, verifiable system for statistical queries and machine learning on distributed datasets. *IEEE Transactions on Information Forensics and Security, 15*, 3035–3050.
107. Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., Jin, S., Quek, T. Q. S., & Poor, H. V. (2020). Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security, 15*, 3454–3469.

108. Xie, C., Chen, M., Chen, P., & Li, B. (2021). CRFL: Certifiably robust federated learning against backdoor attacks. CoRR, abs/2106.08283.
109. Zhao, L., Hu, S., Wang, Q., Jiang, J., Chao, S., Luo, X., & Hu, P. (2020). Shielding collaborative learning: Mitigating poisoning attacks through client-side detection. *IEEE Transactions on Dependable and Secure Computing*.
110. Cao, X., Jia, J., & Gong, N. Z. (2021). Provably secure federated learning against malicious clients. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2–9, 2021* (pp. 6885–6893). AAAI Press.
111. Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado Univ at Boulder Dept of Computer Science.
112. Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation, 18*(7), 1527–1554.
113. Salakhutdinov, R., & Hinton, G. (2009). Deep Boltzmann machines. In *Artificial intelligence and statistics* (pp. 448–455). PMLR.
114. Pan, Z., Yu, W., Yi, X., Khan, A., Yuan, F., & Zheng, Y. (2019). Recent progress on generative adversarial networks (GANs): A survey. *IEEE Access, 7*, 36322–36333.
115. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (Vol. 27).
116. Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2223–2232).
117. Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1125–1134).
118. Lin, C. H., Chang, C.-C., Chen, Y.-S., Juan, D.-C., Wei, W., & Chen, H.-T. (2019). COCO-GAN: Generation by parts via conditional coordinating. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 4512–4521).
119. Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Qiao, Y., & Change Loy, C. (2018). Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*.
120. Mahapatra, D., Bozorgtabar, B., & Garnavi, R. (2019). Image super-resolution using progressive generative adversarial networks for medical image analysis. *Computerized Medical Imaging and Graphics, 71*, 30–39.
121. Zareapoor, M., Celebi, M. E., & Yang, J. (2019). Diverse adversarial network for image super-resolution. *Signal Processing: Image Communication, 74*, 191–200.
122. Jetchev, N., Bergmann, U., & Vollgraf, R. (2016). Texture synthesis with spatial generative adversarial networks. arXiv preprint arXiv:1611.08207.
123. Bergmann, U., Jetchev, N., & Vollgraf, R. (2017). Learning texture manifolds with the periodic spatial GAN. arXiv preprint arXiv:1705.06566.
124. Slossberg, R., Shamai, G., & Kimmel, R. (2018). High quality facial surface and texture synthesis via generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*.
125. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., & Aila, T. (2020). Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 8110–8119).
126. Yu, L., Zhang, W., Wang, J., & Yu, Y. (2017). Seqgan: Sequence generative adversarial nets with policy gradient. n *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 31).
127. Lin, K., Li, D., He, X., Zhang, Z., & Sun, M.-T. (2017). Adversarial ranking for language generation. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (pp. 3158–3168).

128. Li, J., Monroe, W., Shi, T., Jean, S., Ritter, A., & Jurafsky, D. (2017). Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (pp. 2157–2169).
129. Sriram, A., Jun, H., Gaur, Y., & Satheesh, S. (2018). Robust speech recognition using generative adversarial networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5639–5643). IEEE.
130. Donahue, C., Li, B., & Prabhavalkar, R. (2018). Exploring speech enhancement with generative adversarial networks for robust speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5024–5028). IEEE.
131. Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.
132. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems* (pp. 2180–2188).
133. Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.
134. Arjovsky, M., & Bottou, L. (2017). Towards principled methods for training generative adversarial networks. arXiv preprint arXiv:1701.04862.
135. Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. In *International Conference on Machine Learning* (pp. 214–223). PMLR.
136. Muñoz-González, L., Pfitzner, B., Russo, M., Carnerero-Cano, J., & Lupu, E. C. (2019). Poisoning attacks with generative adversarial nets. arXiv preprint arXiv:1906.07773.
137. Wang, X., Cheng, M., Eaton, J., Hsieh, C.-J., & Wu, F. (2018). Attack graph convolutional networks by adding fake nodes. arXiv preprint arXiv:1810.10751.
138. Bhagoji, A. N., Chakraborty, S., Mittal, P., & Calo, S. (2019). Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning* (pp. 634–643). PMLR.
139. Bai, Y., Chen, D., Chen, T., & Fan, M. (2021). Ganmia: GAN-based black-box membership inference attack. In *ICC 2021-IEEE International Conference on Communications* (pp. 1–6). IEEE.
140. Zhao, Z., Dua, D., & Singh, S. (2017). Generating natural adversarial examples. arXiv preprint arXiv:1710.11342.
141. Zhang, J., Zhang, J., Chen, J., & Yu, S. (2020). Gan enhanced membership inference: A passive local attack in federated learning. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)* (pp. 1–6). IEEE.
142. Song, M., Wang, Z., Zhang, Z., Song, Y., Wang, Q., Ren, J., & Qi, H. (2020). Analyzing user-level privacy attack against federated learning. *IEEE Journal on Selected Areas in Communications, 38*(10), 2430–2444.
143. Lyu, L., Yu, H., & Yang, Q. (2020). Threats to federated learning: A survey. arXiv preprint arXiv:2003.02133.
144. Hu, W., & Tan, Y. (2017). Generating adversarial malware examples for black-box attacks based on gan. arXiv preprint arXiv:1702.05983.
145. Rigaki, M., & Garcia, S. (2018). Bringing a gan to a knife-fight: Adapting malware communication to avoid detection. In *2018 IEEE Security and Privacy Workshops (SPW)* (pp. 70–75). IEEE.
146. Zhang, J., Chen, J., Wu, D., Chen, B., & Yu, S. (2019). Poisoning attack in federated learning using generative adversarial nets. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)* (pp. 374–380). IEEE.
147. Sun, Y., Chong, N., & Ochiai, H. (2021). Information stealing in federated learning systems based on generative adversarial networks. arXiv preprint arXiv:2108.00701.

148. Ren, H., Deng, J., & Xie, X. (2021). Grnn: Generative regression neural network—a data leakage attack for federated learning. arXiv preprint arXiv:2105.00529.
149. Mangaokar, N., Pu, J., Bhattacharya, P., Reddy, C. K., & Viswanath, B. (2020). Jekyll: Attacking medical image diagnostics using deep generative models. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)* (pp. 139–157). IEEE.
150. Ching, C.-W., Lin, T.-C., Chang, K.-H., Yao, C.-C., & Kuo, J.-J. (2020). Model partition defense against gan attacks on collaborative learning via mobile edge computing. In *GLOBECOM 2020-2020 IEEE Global Communications Conference* (pp. 1–6). IEEE.
151. Luo, X., & Zhu, X. (2020). Exploiting defenses against gan-based feature inference attacks in federated learning. arXiv preprint arXiv:2004.12571.
152. Denning, D. E. (2014). Framework and principles for active cyber defense. *Computers & Security, 40*, 108–113.
153. Xiong, Y., Xu, F., & Zhong, S. (2020). Detecting gan-based privacy attack in distributed learning. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)* (pp. 1–6). IEEE.
154. Dwork, C., McSherry, F., Nissim, K., & Smith, A. D. (2006). Calibrating noise to sensitivity in private data analysis. In S. Halevi & T. Rabin (Eds.), *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4–7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science* (pp. 265–284). Springer.
155. Dwork, C. (2006). Differential privacy. In M. Bugliesi, B. Preneel, V. Sassone & I. Wegener (Eds.), *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10–14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science* (pp. 1–12). Springer.
156. Dwork, C., Roth, A., et al. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science, 9*(3–4), 211–407.
157. Lyu, L., Yu, H., Ma, X., Sun, L., Zhao, J., Yang, Q., & Yu, P. S. (2020). Privacy and robustness in federated learning: Attacks and defenses. CoRR, abs/2012.06337.
158. McMahan, H. B., Ramage, D., Talwar, K., & Zhang, L. (2017). Learning differentially private language models without losing accuracy. CoRR, abs/1710.06963.
159. Geyer, R. C., Klein, T., & Nabi, M. (2017). Differentially private federated learning: A client level perspective. CoRR, abs/1712.07557.
160. Alvim, M., Chatzikokolakis, K., Palamidessi, C., & Pazii, A. (2018). Invited paper: Local differential privacy on metric spaces: Optimizing the trade-off with utility. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)* (pp. 262–267).
161. Truex, S., Liu, L., Chow, K.-H., Gursoy, M. E., & Wei, W. (2020). Ldp-fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, EdgeSys '20 (pp. 61—66). Association for Computing Machinery.
162. Zhao, Y., Zhao, J., Yang, M., Wang, T., Wang, N., Lyu, L., Niyato, D., & Lam, K.-Y. (2021). Local differential privacy-based federated learning for internet of things. *IEEE Internet of Things Journal, 8*(11), 8836–8853.
163. Sun, L., & Lyu, L. (2020). Federated model distillation with noise-free differential privacy. CoRR, abs/2009.05537.
164. Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., & Zhou, Y. (2019). A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, AISec'19 (pp. 1–11). Association for Computing Machinery.
165. Yin, L., Feng, J., Xun, H., Sun, Z., & Cheng, X. (2021). A privacy-preserving federated learning for multiparty data sharing in social IoTs. *IEEE Transactions on Network Science and Engineering, 8*(3), 2706–2718.
166. Yang, B., Sato, I., & Nakagawa, H. (2015). Bayesian differential privacy on correlated data. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (pp. 747–762).

167. Qu, Y., Gao, L., Xiang, Y., Shen, S., & Yu, S. (2022). Fedtwin: Blockchain-enabled adaptive asynchronous federated learning for digital twin networks. *IEEE Network*.
168. Byrd, D., & Polychroniadou, A. (2020). Differentially private secure multi-party computation for federated learning in financial applications. In *Proceedings of the First ACM International Conference on AI in Finance* (pp. 1–9).
169. Gong, M., Feng, J., & Xie, Y. (2020). Privacy-enhanced multi-party deep learning. *Neural Networks, 121*, 484–496.
170. Zhao, C., Zhao, S., Zhao, M., Chen, Z., Gao, C., Li, H., & Tan, Y. (2019). Secure multi-party computation: Theory, practice and applications. *Information Science, 476*, 357–372.
171. Yao, A. C. (1986). How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27–29 October 1986* (pp. 162–167). IEEE Computer Society.
172. Zhang, Q., Xin, C., & Wu, H. (2021). Privacy-preserving deep learning based on multiparty secure computation: A survey. *IEEE Internet of Things Journal, 8*(13), 10412–10429.
173. Xu, G., Li, H., Liu, S., Yang, K., & Lin, X. (2020). Verifynet: Secure and verifiable federated learning. *IEEE Transactions on Information Forensics and Security, 15*, 911–926.
174. Zheng, Y., Lai, S., Liu, Y., Yuan, X., Yi, X., & Wang, C. (2022). Aggregation service for federated learning: An efficient, secure, and more resilient realization. *IEEE Transactions on Dependable and Secure Computing*, in press.
175. Zhou, Y., Ye, Q., & Lv, J. (2022). Communication-efficient federated learning with compensated overlap-FedAvg. *IEEE Transactions on Parallel and Distributed Systems, 33*(1), 192–205.
176. Katz, J., & Lindell, Y. (2014). *Introduction to modern cryptography*. CRC Press.
177. Hahn, C., Kim, H., Kim, M., & Hur, J. (2022). Versa: Verifiable secure aggregation for cross-device federated learning. *IEEE Transactions on Dependable and Secure Computing*, in press.
178. Yao, A. C. (1982). Protocols for secure computations (extended abstract). In *23th Annual Symposium on Foundations of Computer Science, Chicago, USA, 1982*. IEEE Computer Society.
179. Gowravaram, N. R. (2018). *Zero knowledge proofs and applications to financial regulation*. Havard University DASH.
180. Shannon, C. (1949). Communication theory of secrecy systems. *Bell System Technical Journal, 28*, 656–715.
181. Chaum, D. (1981). Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM, 24*(2), 84–88.
182. Dingledine, R., Mathewson, N., & Syverson, P. F. (2004). Tor: The second-generation onion router. In M. Blaze (Ed.), *Proceedings of the 13th USENIX Security Symposium, August 9–13, 2004, San Diego, CA, USA* (pp. 303–320). USENIX.
183. Cover, T. M., & Thomas, J. A. (2006). *Elements of information theory* (2nd ed.) . Wiley series in telecommunications and signal processing. Wiley-Interscience.
184. Ishai, Y., Kushilevitz, E., Ostrovsky, R., & Sahai, A. (2006). Cryptography from anonymity. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21–24 October 2006, Berkeley, California, USA, Proceedings* (pp. 239–248). IEEE Computer Society.
185. Bittau, A., Erlingsson, Ú., Maniatis, P., Mironov, I., Raghunathan, A., Lie, D., Rudominer, M., Kode, U., Tinnés, J., & Seefeld, B. (2017). Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28–31, 2017* (pp. 441–459). ACM.
186. Cheu, A., Smith, A. D., Ullman, J. R., Zeber, D., & Zhilyaev, M. (2019). Distributed differential privacy via shuffling. In Y. Ishai & V. Rijmen (Eds.), *Advances in Cryptology—EUROCRYPT 2019—38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science* (pp. 375–403). Springer.

187. Yu, S., Zhao, G., Dou, W., & James, S. (2012). Predicted packet padding for anonymous web browsing against traffic analysis attacks. *IEEE Transactions on Information Forensics and Security, 7*(4), 1381–1393.
188. Ahlswede, R., Cai, N., Li, S. R., & Yeung, R. W. (2000). Network information flow. *IEEE Transactions on Information Theory, 46*(4), 1204–1216.
189. Balle, B., Bell, J., Gascón, A., & Nissim, K. (2019). The privacy blanket of the shuffle model. In A. Boldyreva & D. Micciancio (Eds.), *Advances in Cryptology—CRYPTO 2019—39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II*, volume 11693 of Lecture notes in computer science (pp. 638–667). Springer.
190. Ghazi, B., Golowich, N., Kumar, R., Pagh, R., & Velingker, A. (2021). On the power of multiple anonymous messages: Frequency estimation and selection in the shuffle model of differential privacy. In A. Canteaut & F. Standaert (Eds.), *Advances in Cryptology—EUROCRYPT 2021—40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part III*, volume 12698 of Lecture notes in computer science (pp. 463–488). Springer.
191. Girgis, A. M., Data, D., Diggavi, S., Kairouz, P., & Suresh, A. T. (2021). Shuffled model of federated learning: Privacy, communication and accuracy trade-offs. *IEEE Journal O Selected Areas in Information Theory, 2*(1), 6314–6323.
192. Balle, B., Barthe, G., & Gaboardi, M. (2018). Privacy amplification by subsampling: Tight analyses via couplings and divergences. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3–8, 2018, Montréal, Canada* (pp. 6280–6290).
193. Zhao, B., Fan, K., Yang, K., Wang, Z., Li, H., & Yang, Y. (2021). Anonymous and privacy-preserving federated learning with industrial big data. *IEEE Transactions on Industrial Informatics, 17*(9), 6314–6323.
194. Erlingsson, Ú., Feldman, V., Mironov, I., Raghunathan, A., Talwar, K., & Thakurta, A. (2019). Amplification by shuffling: From local to central differential privacy via anonymity. In T. M. Chan (Ed.), *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6–9, 2019* (pp. 2468–2479). SIAM.
195. Sun, L., Qian, J., & Chen, X. (2021). LDP-FL: Practical private aggregation in federated learning with local differential privacy. In Z. Zhou (Ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event/Montreal, Canada, 19–27 August 2021* (pp. 1571–1578). ijcai.org.
196. Calvanese Strinati, E., & Barbarossa, S. (2021). 6g networks: Beyond Shannon towards semantic and goal-oriented communications. *Computer Networks, 190*, 107930.
197. Tishby, N., & Zaslavsky, N. (2015). Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop, ITW 2015, Jerusalem, Israel, April 26–May 1, 2015* (pp. 1–5). IEEE.
198. Hutson, M. (2020). Core progress in ai has stalled in some fields. *Science, 368*(6494), 927.
199. Hutson, M. (2018). Has artificial intelligence become alchemy? *Science, 360*(6388), 478.
200. Eshete, B. (2021). Making machine learning trustworthy. *Science, 373*(6556), 743–744.