People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University Kasdi Merbah Ouargla
Faculty of New Technologies of Information and Communication

Thesis

Presented for **MASTER's degree**

In : Information Technology and Automated media

Speciality : Administration and Network Security

Through : BOUREGA Lokmane - GHERBI Leila

Subject

Trust-based in Federated Learning

Publicly supported, on 15/06/2023, before the jury composed of :

Mrs.	KHALDI Amine	Doctor	UKMO	President
Mrs.	EUSCHI Salah	Doctor	UKMO	Examiner
Mrs.	Akram Zine Eddine Boukhamla	Doctor	UKMO	Thesis Supervisor

I dedicate this thesis to my doctor, whose unwavering support and guidance has been instrumental in helping me achieve this milestone. Thank you for your patience, your encouragement, and your belief in me.

To my family, I am forever grateful for your love and support throughout my academic journey. Thank you for always being there for me, for understanding my sacrifices, and for inspiring me to be the best version of myself.

To my coworker, thank you for your collaboration, your friendship, and your invaluable contributions to this project. Your expertise and insights have made a significant difference, and I couldn't have done it without you.

This thesis is dedicated to you all with heartfelt appreciation and admiration.

Acknowledgement

I would like to express my deepest gratitude to all those who have contributed to the completion of this thesis.

First and foremost, I would like to thank my supervisor, [Name], for their invaluable guidance, encouragement, and feedback throughout this journey. Their expertise, insight, and patience have been instrumental in shaping this work.

I am also grateful to my family, who have supported me through thick and thin, and have provided me with a loving and nurturing environment that has allowed me to pursue my dreams. I am blessed to have you in my life.

I would also like to thank my colleagues and friends, who have provided me with moral support and valuable feedback throughout the research process. Your input and insights have been invaluable, and I am grateful for your contributions.

Finally, I would like to thank all the participants who took part in this study. Your willingness to share your experiences and insights has been crucial in advancing our understanding of the subject matter.

Once again, thank you to everyone who has contributed to this work. Your support and encouragement have been greatly appreciated.

ملخص

يوفر التعلم الموحد الراحة لتطبيقات التعلم الآلي عبر المجالات وقد تمت دراسته على نطاق واسع ، لكن التعلم الفيدرالي الأصلي لا يزال عرضة للتسمم وهجمات الاستدلال التي ستعيق تطبيق التعلم الموحد ، لذلك من الضروري تصميم تعلم فيدرالي جدير بالثقة للقضاء على خوف المستخدمين ، نهدف إلى تقديم صورة مدروسة جيدا لقضايا الأمان والخصوصية في التعلم الفيدرالي الذي يمكن سد فجوة عدم الثقة في التعلم الموحد أولا نحدد الأهداف المرجوة والمتطلبات الرئيسية منه مراقبة نموذج التعلم الموحد من منظور الالهجمات واستقراء أدوار وقدرات الخصوم المحتملين بعد ذلك نلخص وسائل الهجوم والدفاع السائدة الحالية ونحلل خصائص الأساليب المختلفة بناء على المعرفة المسبقة

الكلمات المفتاحية: التعليم الفيدرالي ،الأمان والخصوصية ،نموذج

Abstract

Federated learning (FL) provides convenience for cross-domain machine learning applications and has been widely studied. However, the original FL is still vulnerable to poisoning and inference attacks, which will hinder the landing application of FL. Therefore, it is essential to design a trustworthy federation learning (TFL) to eliminate users' anxiety. In this paper, we aim to provide a well-researched picture of the security and privacy issues in FL that can bridge the gap to TFL. Firstly, we define the desired goals and critical requirements of TFL, observe the FL model from the perspective of the adversaries and extrapolate the roles and capabilities of potential adversaries backward. Subsequently, we summarize the current mainstream attack and defense means and analyze the characteristics of the different methods. Based on a priori knowledge, we propose directions for realizing the future of TFL that deserve attention.

Keywords: Trust, Federated Learning, Privacy

Contents

List of Figures

List of Tables

General Introduction

In many industries, including healthcare, finance, and transportation, artificial intelligence (AI) has emerged as a game-changing technology. Security issues relating to AI have grown along with its usage. AI systems are more susceptible to cyber threats like data breaches, cyberattacks, and malicious exploitation as they become more complex and interconnected. Therefore, it is crucial to investigate fresh ideas for improving the security of AI systems.

Federated learning, a cutting-edge method for cooperative machine learning, is one such strategy. Federated learning enhances the privacy and security of the data by allowing multiple participants to contribute their local data models without disclosing their own data. Federated learning has the potential to make it possible for secure and effective AI systems by utilizing the capabilities of distributed computing.

Federated learning has its drawbacks, though, including communication costs, participant diversity, and potential model poisoning attacks. To ensure the secure and reliable implementation of federated learning, it is crucial to address these issues and create strong solutions.

This master's thesis examines federated learning's potential as a safe method for AI systems. The thesis will go over the advantages and difficulties of federated learning, look at cutting-edge federated learning strategies, and suggest fresh ideas to improve the security of federated learning systems. We aim to contribute to the creation of reliable and secure AI systems through this thesis.

Chapter 1

Introduction

1.1 Overview

Federated learning has its drawbacks, though, including communication costs, participant diversity, and potential model poisoning attacks. To ensure the secure and reliable implementation of federated learning, it is crucial to address these issues and create strong solutions.

This master's thesis examines federated learning's potential as a safe method for AI systems. The thesis will go over the advantages and difficulties of federated learning, look at cutting-edge federated learning strategies, and suggest fresh ideas to improve the security of federated learning systems. We aim to contribute to the creation of reliable and secure AI systems through this thesis.

Federated learning (FL) [1] offers a great solution to these issues. In federated learning, as opposed to distributed machine learning, users update the model rather than their data to produce a more accurate overall model. It allays users' concerns about their privacy by ensuring that data can be used locally. We discovered that the current level of FL is still insufficient to meet its security requirements, despite the fact that it has been partially used in practice, such as when Google used it to predict the user's keyboard's subsequent input. The usability of FL is still impacted by methods like poisoning attacks and inference attacks, particularly when combined with highly sensitive information domains like medicine and finance. Such problems have prevented FL from being used widely, and researchers have had to redesign the model to make it more user-credible. Therefore, trustworthy federated learning (TFL) [2] deserves to be further discussed in conjunction with security measures. TFL's objective, in contrast to traditional FL, is to allay users' worries about the security and privacy of the model system and to guarantee the legitimacy of the model framework. To achieve TFL, researchers frequently pick se-

curity algorithms or secure architecture, like blockchain technology. The requirements for TFL are not, however, systematically defined in the current research. The following fundamental principles should be present in FL systems because TFL imposes more stringent safety requirements on them:

• High Confidentiality: Confidentiality is reflected in the fact that malicious adversaries cannot steal sensitive information in FL.

• High Integrity: Integrity is reflected in the fact that private data cannot be maliciously modified without authorization during training.

• High Availability: The model system is required to provide access by authorized users and be used on demand. The model also needs to have a usable accuracy rate as well as efficiency. The cost of trustworthiness cannot be a significant loss of accuracy and a high rate of loss of efficiency.

• Strong Robustness: In addition to following the information security fundamentals, FL should have sufficient resistance in the face of complex scenarios or unknown attacks [3].

• Provable Security: The security protocols and methods must be rigorously secure based on specific mathematical assumptions. In response to the above requirements, we survey the current status of FL and look forward to the next more promising development direction of TFL. Analytical work on TFL has been partially studied, but we will look at the threats faced by FL from some new perspectives

1.2 Motivation

With the help of the potent method known as federated learning, machine learning models can be trained on decentralized data without the need for centralized data collection and processing. It has many advantages, including better model performance, lower communication costs, and increased user privacy. The decentralized nature of the technique introduces new security risks, such as data breaches, model poisoning, and adversarial attacks, so the security of federated learning continues to be a serious concern.

For federated learning to be widely adopted and reach its full potential in a variety of applications, including healthcare, finance, and the Internet of Things (IoT), its security must be ensured. Failure to address security issues in federated learning can have serious repercussions, such as the disclosure of sensitive data, loss of trust in the system, and financial losses.

Several strategies, including cryptographic methods, differential privacy, and trust-based methods, have been put forth to address security issues in federated learning.

Combining these techniques can provide better security guarantees and protection against various types of attacks, even though each approach has its benefits and drawbacks.

Overall, for federated learning to be widely adopted and to realize its potential benefits, it is essential to ensure its security. As a result, in order to handle the particular difficulties presented by federated learning, researchers and practitioners must keep creating and testing new security techniques.

1.3 Contribution

Although similar investigations into the FL threat have been made, it is still necessary for these efforts to give a more thorough overview of current technologies and a clear indication of where future research should go. Our research offers a thorough overview of FL, covering its definition, dangers, and potential future research using the promising Trust-Based (Score Method) technology. This work will make it easier to create practical TFL paradigms and apply them quickly to real production. Here are our main contributions:

• We carefully examine the existing Federated Learning research content and thoroughly investigate the development mapping and key technologies of FL.

• From the viewpoint of the adversary, we evaluate FL's threats. Additionally, we list common FL-specific attacks from the standpoint of security and privacy risks.

• We summarize and abstract the state's privacy protection strategies and assess their advantages and disadvantages. We offer some worthwhile prospects for developing Trusted Federated Learning based on this.

1.4 Organization of the Thesis

In Chapter 2, we define Federated Learning and categorize the existing FL models from various angles. We outline the main attack vectors that pose a security and privacy risk to the model, and we compare and contrast the available defense strategies. In Chapter 3, we highlight the method we propose for enhancing network security that is based on trusted clients. Chapter 4 provides the Experimental results of our work.

Chapter 2

Federated Learning

2.1 Definition

In order to develop cognitive and analytical functionality, which are challenging and inefficient to develop algorithmically, machine learning (ML) has emerged as a critical technique. With the introduction of Deep Neural Networks (DNNs) and the computational hardware required to efficiently train complex networks, applications in computer vision, speech recognition, and natural language understanding advanced significantly. Additionally, traditional ML methods like support vector models (SVMs), decision trees, and linear regression have become more popular, especially in relation to structured data. The availability of top-notch training data is essential for ML applications. However, there are times when privacy concerns make it impossible to transfer training data to a central data repository where it can be curated and managed for the ML process. The method known as federated learning (FL) was first put forth in [4] to train ML models using training data from various sources without the need for centralized data collection. The lack of adoption of a central data repository has been largely attributed to various jurisdictions' differing laws governing consumer privacy. Examples of legal frameworks for the gathering and use of consumer data include the General Data Protection Regulation (GDPR), the Health Insurance Portability and Accountability Act (HIPAA), and the California Consumer Privacy Act (CCPA) [5]. Additionally, news stories about data breaches have made people more aware of the risk involved in keeping sensitive customer data on hand. FL makes it possible to use data without actually storing it in a central repository, reducing this risk. The transfer of data between jurisdictions, such as different countries, is also restricted by regulation. This decision was made in light of the possibility that data protection in other nations may be insufficient or related to national security, necessitating the retention of critical data domestically. International businesses

that have subsidiaries in various markets but want to train a model using all of their data must navigate national and regional regulations. Beyond legal requirements, using data from various locations could also be practical. Central data collection may be impossible due to unreliable communication links, the sheer volume of data collected by sensors or in telecommunication devices, or both.

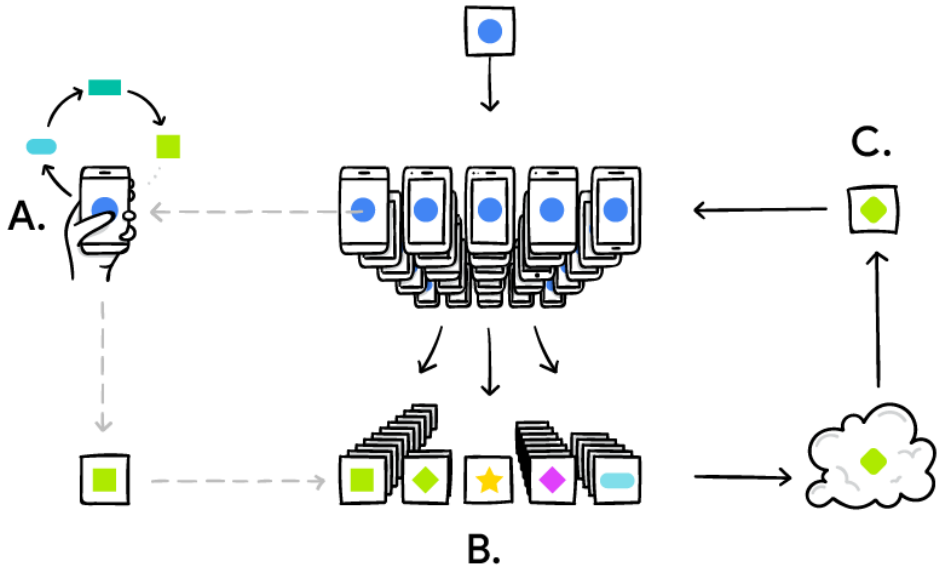


Figure 2.1: Fedrated Learning.

Additionally, FL enables various businesses to collaborate and develop models for their mutual benefit without disclosing their trade secrets. Then, how does FL operate? In the FL approach, a number of different parties who each have control over their own training set work together to develop a machine learning model. They carry out this action without disclosing their training information to any other parties or outside organizations. In the literature, parties to the collaboration are also referred to as clients or devices. Parties include consumer electronics like smartphones or automobiles, but they can also be cloud services from various providers, data centers processing enterprise data in various nations, application silos within a business, or embedded systems like manufacturing robots in an automotive plant.

Although the FL collaboration can be carried out in various ways, its most typical form is shown in Fig. 2.2. In this method, the collaboration is facilitated by an aggregator, also known as a server or coordinator. On the basis of their personal training data, parties conduct local training processes. When local training is complete, they update the aggregator with their model parameters. Depending on the type of machine learn-

ing model being trained, the model updates may take the form of network weights, for example, in the case of a neural network.

The model updates from the parties can then be combined into a single model through a process we call model fusion after the aggregator has received them. This can be accomplished in the neural network example by simply averaging the weights, as suggested by the FedAvg algorithm [6]. The parties are then given a second distribution of the combined model as a model update to serve as the foundation for the following learning cycle. Up until the training process converges, this process can be repeated. The aggregator's job is to coordinate the parties' learning and information-sharing processes, as well as to carry out the fusion algorithm to combine the model parameters from each party into a single model. The FL process produces a model that is based on the training data of all parties; however, the training data is never disclosed. The FL approach resembles distributed learning on clusters, a popular strategy for challenging ML tasks. With distributed learning, the learning process is sped up by sharing the computational workload among a group of compute nodes. Similar to the federated model, distributed learning typically uses a parameter server to combine results from nodes. However, there are some significant differences. If all training data is kept private, the distribution and volume of data in FL may not be known and may not be centralized controllable. We cannot assume that the parties' data will be distributed among them independently and identically (IID). The datasets among the parties may also be imbalanced as a result of some parties having more data than others. In distributed learning, the stochastic characteristics of the data are managed centrally and distributed to various nodes in shards by a central entity. When creating FL training algorithms, imbalance and non-IIDness of party data must be taken into consideration. In contrast, depending on the use case, the number of parties in FL may vary. Less than ten parties may be involved in training a model using datasets from various data centers of a multinational corporation. It's frequently referred to as an enterprise or cross-silo use case. A mobile phone application's training data may be contributed to by hundreds of millions of people. The cross-device use case is what people commonly refer to as here. In the enterprise use case, it is typically crucial to take model updates into account from all or the majority of parties in each round. Every FL round in the device use case will only contain a—possibly sizable—sub-sample of the entire set of devices. The FL process can use the enterprise use case's consideration of the parties' identities in the training and verification processes. Party identity is typically not significant in the cross-device use case, and one party may participate in only one training session. Given the high number of participants, it is more likely in the device use case than in the enterprise scenario that some devices will experience communication problems. Cell phones may be turned off, or a device may be in a poor network coverage area.

This can be controlled by sampling parties, establishing time restrictions for performing aggregation, or using other mitigation strategies. Because there are fewer participants in the enterprise use case, it is important to carefully manage communication breakdowns because individual party contributions are significant. We give a formal introduction to the key ideas discussed in the following section. After that, we discuss FL systems than the approach of FL Personalization, then we took a full discussion about these Concepts which is: Classification of FL, Threats in FL providing the Byzantine attacks, we also give a definition of Defence mechanism, at last we provide our main subject which is trust in FL.

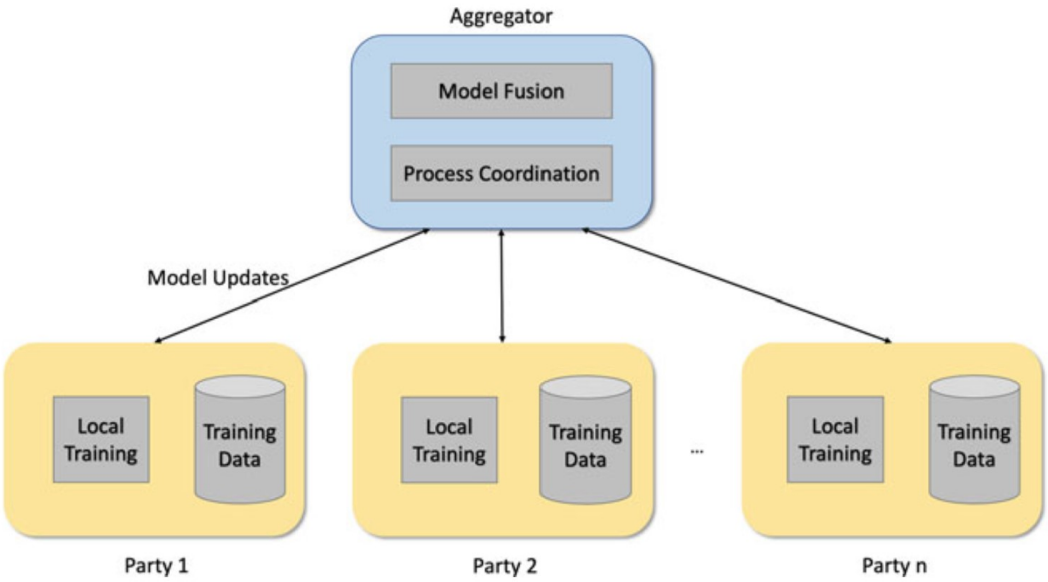


Figure 2.2: Federated Learning Overview.

2.2 Concepts and Terminology

Like any machine learning task, FL trains a model M representing a predictive function f on training data D. M can have the structure of a neural network or any other, non-neural model. In contrast to centralized machine learning, D is partitioned between n parties P = P1, P2, . . . , Pn, where each party Pk P owns a private training dataset Dk. An FL process involves an aggregator A and a set of parties P . It is important to note that Dk can only be accessed by party Pk. In other words, no party has knowledge of any other dataset than its own, and A has no knowledge of any dataset. How the FL process is conducted at this abstract level is shown in Fig.2.17. To train a global machine learning model M, the aggregator and the parties perform

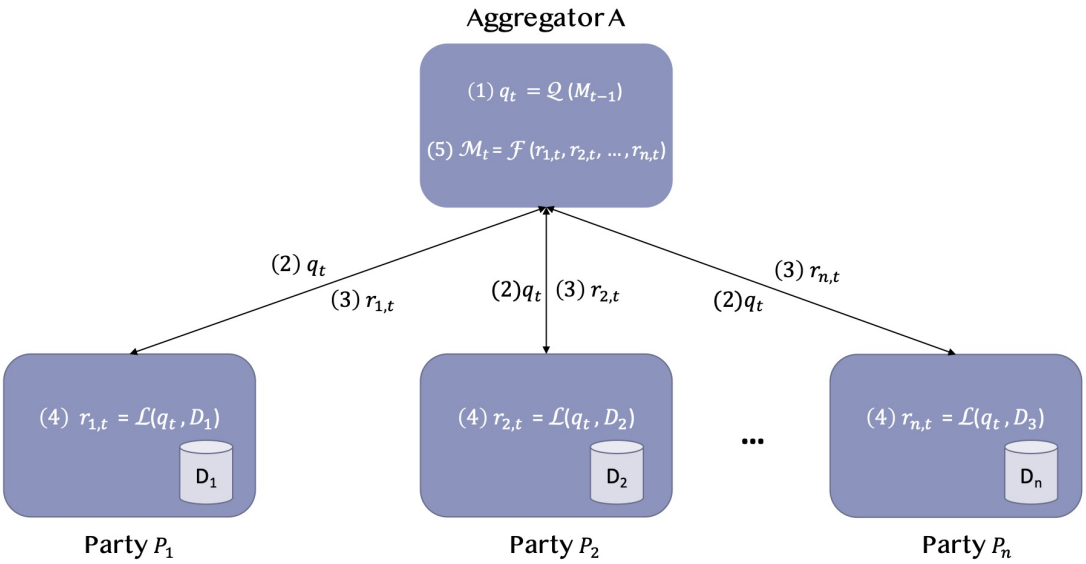


Figure 2.3: Federated learning concepts.

a Federated Learning algorithm that is executed in a distributed way at the aggregator and the parties. The two main algorithmic parts are the aggregator's fusion function F, which combines the output of each party's L into a new joint model, and each party's local training function L, which performs the local training on dataset Dk. Rounds are a collection of local training and fusion iterations that use the index t. By exchanging messages with the aggregator, the parties involved in the algorithm execution are able to work together. The general procedure goes like this:

1. The aggregator is where the process begins. The aggregator employs a function Q to train the model by generating a query qt for the current round from the model from the previous round of training Mt1 at round t. M0 may be empty when the process begins or it may be seeded at random. Additionally, some FL algorithms might add more Q inputs and tailor their queries to each party, but we use this less complex method to keep things simple and maintain generality.

2. The parties are contacted by the query qt, which asks for details about each party's local model or dataset in aggregate form. Examples of queries include those for a neural network's gradients or model weights, or counts for decision trees.

3. After receiving qt, the local training procedure applies the local training function L, which takes the query qt and the local dataset Dk as inputs and produces a model update rk,t. The query qt typically contains details that the party can use to start the local training procedure. For instance, model weights of the new, widely used model Mt

to start local training are included, as well as other details for various model types.

4. After L is finished, party pk sends rk,t back to aggregator A, which gathers all the rk,t from all parties.

5. The fusion function F, which accepts Rt as an input and returns Mt, is applied to all model updates from expected parties that are received by the aggregator and have the form Rt = (r1, t, r2, t,..., rn, t).

A final global model M = Mtmax is produced by repeating this process over a number of rounds until a termination criterion, such as the completion of the maximum training rounds tmax, is met. When using a Naive Bayes approach, the required number of rounds can range from a single model merge to numerous training rounds for conventional gradient-based machine learning algorithms. The fusion function F, the query generation function Q, and the local training function L typically form a complementary set that is intended to work together. L performs the local training and interacts with the actual dataset to produce the model update rk,t. The information in Rt serves as F's input, so F must interpret it before building the subsequent model Mt from it. Q then generates a new query if more rounds are necessary. We will go into more detail about how this process works when training neural networks, decision trees, and gradient-boosted trees in subsequent sections. We can add several variations to this fundamental FL methodology: Cross-device FL frequently has a sizable, millions-strong party population. Some parties don't take part in every round. In this scenario, Q chooses the query as well as which parties Ps P to include in the subsequent round of querying. The party can be chosen at random, according to party traits, or on the basis of previous contributions. Additionally, inquiries to each party might differ, and F would need to combine the answers to various inquiries when developing a new model Mt. While the most typical and practical method uses a single aggregator, other alternative FL architectures have been proposed. The set of parties may be divided between aggregators, and a hierarchical aggregation process may occur. For instance, each party Pk may have its own, associated aggregator Ak that queries the other parties. The common single aggregator configuration is the main topic of the remainder of the introduction.

2.3 Federated Learning Systems

In the end, parties and the aggregator run on a distributed system on which a FL process is executed. The components of this system must meet the computational, memory, networking, and communication needs of the parties, the aggregator, and their inter-party communication. We must pay close attention to the resources available at the point of

training the parties because the local model training is carried out in the area where the data is located. At least in the commonly used single aggregator architecture, aggregators are typically operated in a data center environment. When working with numerous parties, they still need to have the appropriate resources and scalability. Finally, depending on model size, model-update content, frequency, and use of cryptographic protocols, as discussed in the previous section, network connectivity and bandwidth requirements may vary. As a result, federated learning has very different system requirements than centralized learning methods do. Party attendees: The fact that a party might not be located on a system we would typically choose as an ML platform is the most obvious difference between decentralized and centralized machine learning systems. This may not be a problem if the parties have data centers in different countries, but embedded systems, edge computing, and mobile phones are more problematic. Three different functionalities could each use a certain amount of resources:

• If the model is big, the local machine learning process might need a lot of compute and memory. This is particularly true for big DNNs, like big language models. It might be necessary GPU support, which might not be present in embedded systems, let alone distant data centers or data stores connected to software as a service. However, there are small footprint packages like Tensorflow Lite that require less on-party storage and memory, and conventional methods might still be practical even on diminutive gadgets like Rasberry Pis.

• The local machine learning model is operated by the federated learning party client, which also interacts with the aggregator. However, it typically has a small footprint that can fit even in tiny edge devices.

• Using a cryptographic protocol, such as a secure multi-party computation (SMC) implementation based on a threshold Paillier cryptosystem, could, however, significantly increase the computation cost for the party client. The majority of encryption and decryption methods can be parallelized, making a GPU or other specialized hardware necessary. Servers for aggregators Typically, an aggregator is situated in a resource-rich data center environment. Scaling up to many parties, though, comes with a number of difficulties:

The aggregator must be able to manage numerous connections in order to communicate with numerous parties. A tried-and-true strategy for all types of systems that can be applied here similarly is connection pooling. It frequently costs only a little bit of computation to run a fusion algorithm at the aggregator. Simple fusion algorithms, like FedAvg, carry out very basic averaging tasks. In contrast to the local training at the party, other fusion algorithms may be more complex but frequently have lower computational requirements. The size of the set of, say, weight vectors received as replies from parties, however, can be very large in the case of large DNNs and a large number of parties. The weight

vector of a single party can be up to tens of megabytes. Dealing with thousands of parties may be too much for one compute node to handle while performing an average computation in memory. There have been several methods suggested for computationally scaling the aggregator: The fusion algorithm can be run in parallel, for example, using Hadoop or Spark, and the weights can be made persistent. Other strategies divide parties into groups based on the addition's commutative properties. Each aggregator given to these groups computes averages over this group. The outcomes of local aggregation are then combined by a primary aggregator, weighted by the quantity of parties at each aggregator. As a result, they propose one such method; there are other variations as well, such as multi-level aggregation. Sub-sampling of parties is frequently used at each round for very large sets of parties and can support the other methods. As a rule, tree-based FL algorithms place more computational burden on the aggregator than the parties. Communication: When designing a FL, aggregators and parties' quantity and quality of communication must be taken into account. We can frequently assume adequate bandwidth and dependable connections in data center and cloud settings.

FL processes can be time-consuming. Therefore, communication protocols must be resilient to sporadic disconnection. The direction of the connection is a crucial practical factor in business settings. Organizations with IT departments have strictly regulated procedures for opening networking ports. FL system implementations will go more quickly if a networking protocol is chosen that doesn't require parties to open ports but instead requires them to initialize the connection to the aggregator. A greater challenge is posed by embedded systems, edge devices, and mobile systems. Because they are inexpensive devices, some party systems may only occasionally be connected, as in vehicles, or they may have low bandwidth. For the FL process, this could be problematic.

We must have a plan in place to handle these drop-offs if parties do not reply in time for the following round. We must establish a quorum, which might differ depending on the use case. We also require a strategy for re-joining parties. Quora offers a straightforward method of drop-off management, but other strategies, like TIFL, suggest an active straggler management strategy that groups parties by response time and queries them less frequently [7]. Even bias in the model can result from systemic differences in response times. Algorithms can also be used to deal with intermittent or low-bandwidth communication, for instance, by reducing the number of rounds, compressing models, and fusing more divergent models. Using secure computation techniques like SMC may result in an increase in message volume and size, which could be problematic for devices with weak connections. Peer-to-peer communication between parties may be necessary for some SMC protocols for vertical federated learning, which is problematic in two ways: It calls for parties to expose ports to their peers, which presents a challenge for imple-

mentation in businesses. This again doubles network traffic if it is lessened by routing all traffic through the aggregator or another middleman. As a result, even though SMC is frequently a very good method for protecting privacy, it has high resource requirements.

When implementing a FL system, we frequently must trade off the available resources with an appropriate algorithmic strategy. Design decisions and trade-offs. If we have the option to choose, we can select hardware that is compatible with the ML approach we select. We can add a powerful embedded GPU to a car or factory robot, or we can add GPUs to the data centers we want to take part in the federation. Not always is that feasible. When a party's compute platform is provided, we can use an ML strategy that works with our available resources. DNNs require a lot of resources on the party side, but tree-based models like federated XGBoost require less. Also, algorithms can be adapted to system constraints.

2.4 Model Personalization

Model personalization is the process of adjusting a (federally trained) global model to the data distribution of the particular FL process participants. Even though everyone who participates in an FL process has access to a large pool of training data, there are times when it is advantageous to customize the final model so that it accurately represents the data that belong to a particular party. This is important, especially if parties relate to specific users or organizations. Individual parties can run additional local training epochs on local data to complete an FL process in a simple scenario.

Wang et al. suggest a method to assess the value of personalization for every party. User clustering, training on interpolated data (between global and local), and model interpolation are three different methods for personalization. For the first method to cluster users based on training data, privacy requirements must be relaxed or advanced privacy techniques must be used. The foundation of data interpolation is the creation of a global dataset. All methods are effective, but model interpolation has the broadest range of privacy-related applications. Grimberg and others. Expanding on the strategies discussed prior to [8], propose a method to optimize averaging a global model and a local model for personalization purposes by determining optimized weights. Personalization strategies are still developing, but this is a crucial addition to the FL procedure.

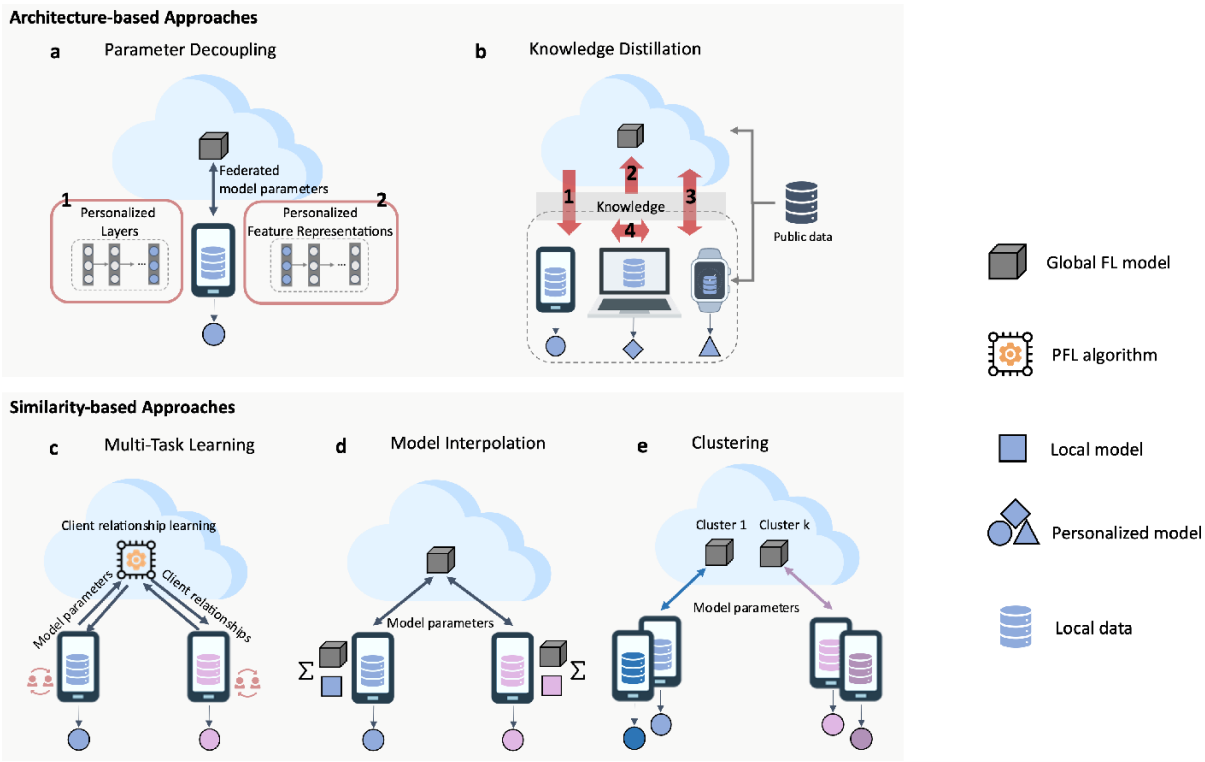


Figure 2.4: Learning Personalized Models.

2.5 Related Work

FLTrust is a Byzantine robust federated learning framework that uses trust bootstrapping and trimmed averages to identify and remove malicious clients. FLTrust has proven effective in protecting federated learning from Byzantine attacks.

Byzantine Robust Federated Learning with Differentiated Privacy: In this article, we propose a Byzantine-robust federated learning framework that leverages differential privacy to protect client privacy and make it more difficult for adversaries to launch Byzantine attacks.

Federated learning with Byzantine resilience: In this article, we propose a federated learning framework that is resistant to Byzantine attacks. The framework uses a combination of techniques such as trust bootstrapping, differential privacy, and secure aggregation to protect client privacy and make it difficult for adversaries to launch Byzantine attacks. we will discuss them on the Chapter 4 and compare their results.

2.6 Classification of FL

The applicability of FL is being improved using a variety of techniques and methods as it develops quickly. FL with various complex morphologies is suggested in order to handle more complicated requirements [1]. Some research projects want to use distributed machine learning models in more significant scenarios, which requires FL to take communication and data aggregation costs into account. Focusing on the use of FL in mobile edge computing, for instance. Other jobs, however, necessitate more secure training participation, so FL must pay closer attention to security and privacy protection.

In ref. [9], the authors place more emphasis on security risks and categorize FL according to the distribution of data features. Additionally, FL is categorized with technologies from various angles. Therefore, a crucial first step in comprehending and improving FL design is classifying FL according to various perspectives.

2.6.1 Centralized/Multi-Center/Decentralized FL

Although FL has been implemented for decentralized training, centralized FL—what we refer to as centralized FL—still needs a central server to finish the accusation of aggregation and broadcasting. The single-server architecture makes sure that the model's rights are centralized in the server, which aids in managing the entire training process and preventing mistakes. The Gboard for Android keyboard, for instance, is based on this architecture. A centralized server, however, is more likely to experience a single point of failure, which could completely devastate the FL system. A decentralized FL was suggested to release this security dependency. It makes an effort to lessen or even do away with the server's influence over the overall model. For the innovative setting of decentralized collaborative learning of personalized models, two asynchronous peer-to-peer algorithms are proposed. This method directly relocates the server to achieve total decentralization. However, this strategy frequently comes at a high time and communication cost. A centralized server is not necessary for multi-center FL, but model updates must be managed by numerous decentralized edge servers. By doing this, the global model's vulnerability to servers acting as malicious nodes is reduced while maintaining the model's usefulness. In order to address the aforementioned issues, [10] builds multiple global models from the data, simultaneously determines the best match between users and centers, and suggests a federated SEM optimization strategy to successfully address the multi-center FL problem.

2.6.2 Horizontal FL/Vertical FL/Federated Transfer Learning

Furthermore, when dealing with various application scenarios, it is a common method to design FL based on the characteristics of training data. FL is classified into horizontal federated learning (HFL), vertical federated learning (VFL), and federated transfer learning (FTL) based on the degree of overlap between the feature space and the sample space . To begin with, clients in business-to-consumer (B-C) FL frequently use data sets with overlapping features. For example, different banks may provide similar data training models with highly coincidental data characteristics. The HFL is more appropriate in this case. To minimize losses, the client typically employs stochastic gradient descent (SGD), while the server employs a secure aggregation algorithm (such as FedAvg, FedPorx) to obtain a global model. We can further refine HFL based on different applications into HFL to businesses (H2B) and HFL to consumers (H2C). HFL has the advantage of quickly extracting features from similar data and obtaining a highly credible global model, but it is less effective for data with little overlap in features. Insurance companies, for example, rely on credit data from banks to provide customized services, and because the two have different feature spaces, they cannot be trained directly using HFL.

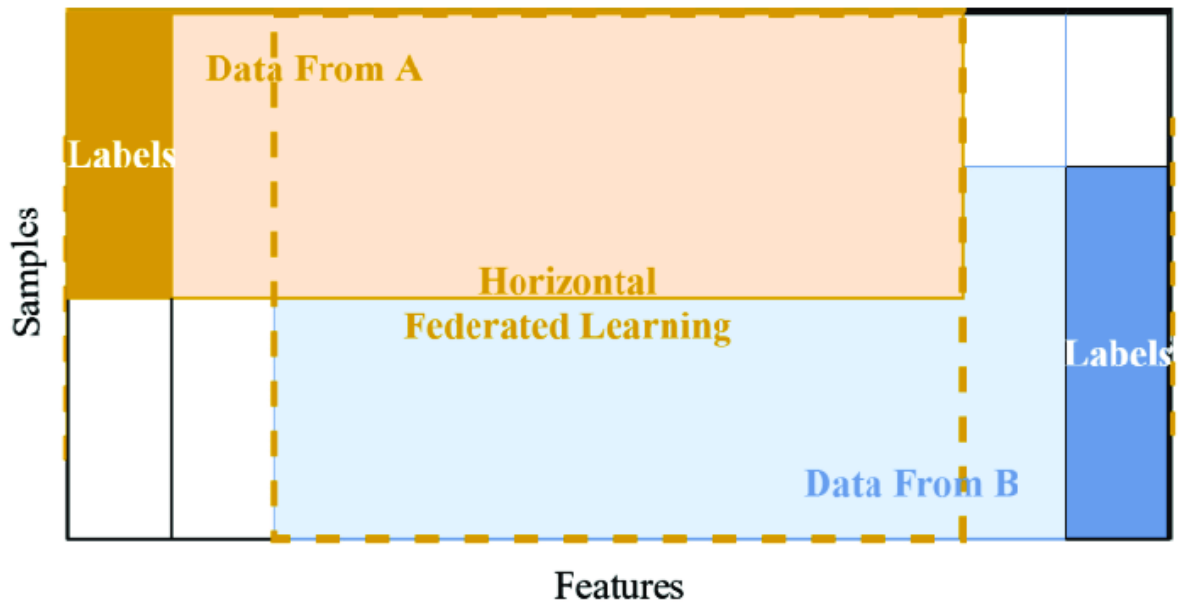


Figure 2.5: Horizontal FL.

VFL is more complex than HFL, but it is applicable to a broader range of scenarios and has greater practical value. However, VFL has some drawbacks, such as low efficiency, and related research has been conducted in this area. For example, created a new backward updating mechanism and bilevel asynchronous parallel architecture to address

the issue of low efficiency caused by synchronous calculation in practical applications. It's worth noting that the first two do well with supervised learning but struggle with unsupervised or weakly supervised data.

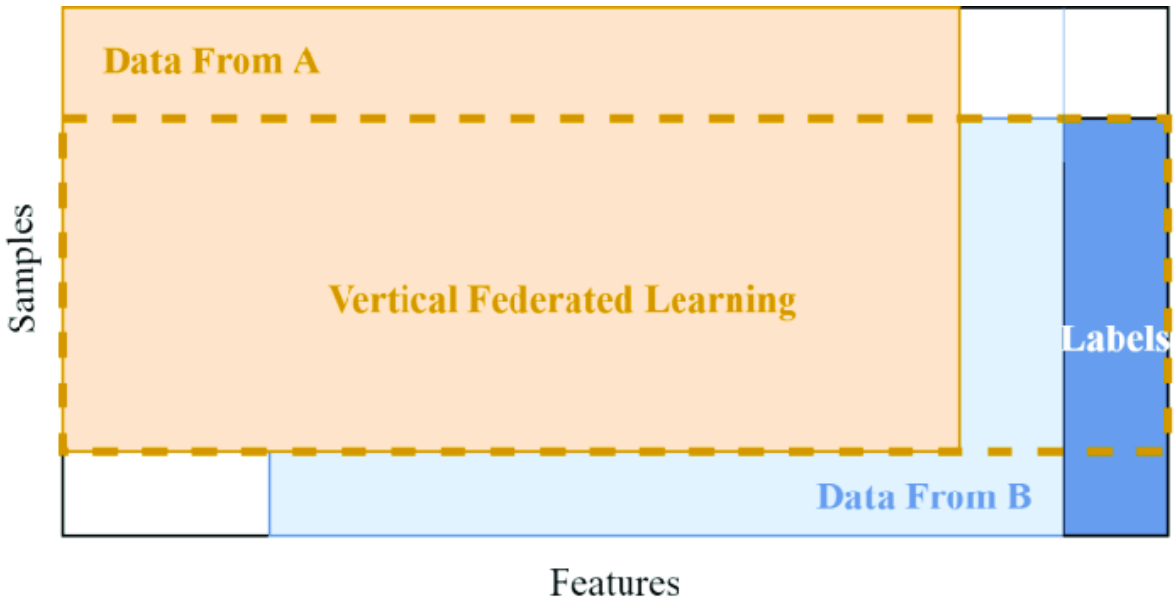


Figure 2.6: Vertical FL.

To deal with the need for tiny overlapping samples, FTL introduces the concept of migration learning. The direct use of the first two models for training results in poor effectiveness for data with low correlation because the aggregation algorithm is difficult to extract similar effective features. Transfer learning makes use of similarities between the target domain and the source domain in order for the migration model to learn from data with significant differences. As a result, a federated learning model that incorporates transfer learning gains the ability to learn from heterogeneous data [11], providing a viable solution for collaborative modeling.

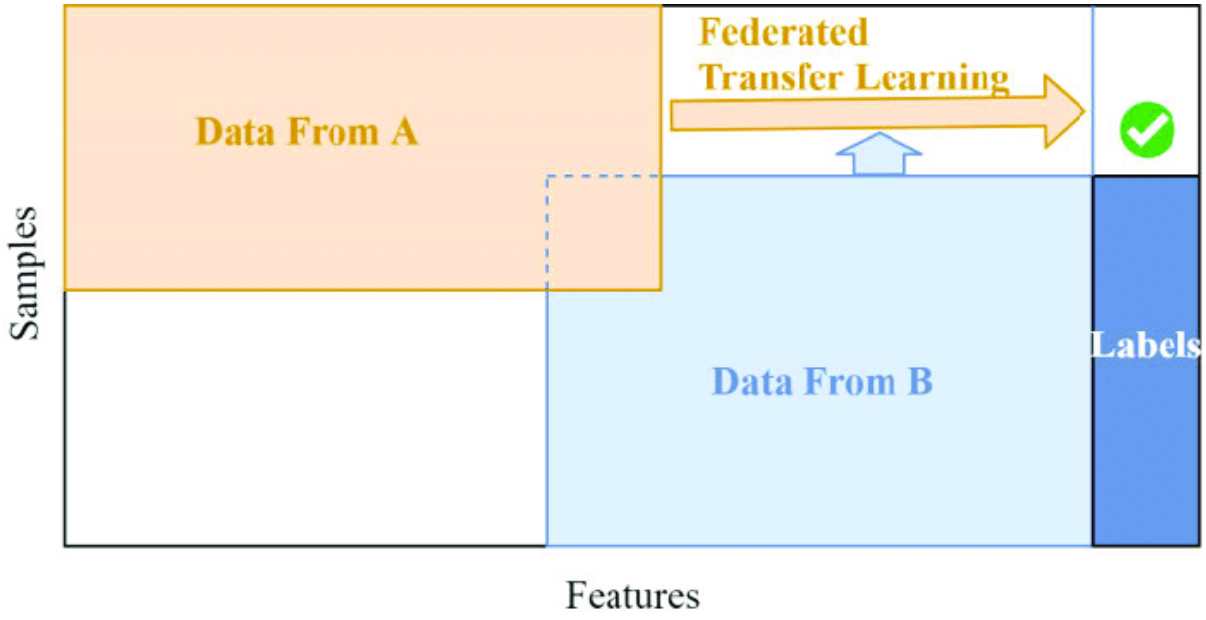


Figure 2.7: Federated Transfer Learning

2.7 Threats in FL

In this section, we will discuss the most common attacks encountered in Florida. Before delving into the threats that FL faces, we'll take a look at where these threats might come from. In the second and third parts, we examine potential FL attacks from the standpoints of security and privacy, respectively. The investigation of the source of threats.

2.7.1 Adversary Status

To assess FL threats, we must first understand the adversary's role in the system model. In contrast to traditional machine learning, the adversary's identity in FL is relatively complex. Using the basic FL as a reference object, the participating entities are classified as server, client, and malicious external entities. It is difficult to pinpoint the source of the threat. Assuming they both produce malicious attackers, we can distinguish between internal and external attackers (Server and Client). Simultaneously, we cannot rule out the possibility of a collusion attacker.

1. Server: The server has high privileges in FL. As a result, once malicious, it poses a greater risk to the system's security and utility. There are general types of attacks, such as model poisoning and backdoor attacks. By poisoning the global gradient, the server can easily break model convergence. It can also steal the client's privacy if it is semi-honest by using inference or model reversal attacks. Because the server has updated gradient information, a gradient change-based membership inference attack can easily

steal membership information from the client. As a result, it is necessary to allow the server to obtain as little accurate information as possible based on its responsibilities.

2. Client: On the one hand, the client is the most vulnerable target of attack, such as membership inference attacks aimed at stealing customers' private information. Homomorphic encryption and differential privacy are promising methods for preventing gradient information leakage. In the meantime, the addition of a shuffler mechanism can be used to further mask the user's ID. A malicious client, on the other hand, can have a negative impact on model training. It can interfere with the global model's availability by poisoning [12] or inserting backdoors [13], but unlike the server, the impact of a single malicious client is limited.

3. External attackers: An external saboteur could hijack the server or bring it down directly, completely disrupting the training. External entities that listen in on server and client communication channels endanger clients' privacy significantly. Homomorphic encryption and differential privacy can limit its access to accurate information, while the trusted execution environment's FL can protect it from threats.

4. Collusion: Malicious opponents may band together to carry out a coordinated attack. In reality, a conspiracy attack only needs a tiny bit of information to be leaked by an internal foe to compromise the availability of the majority of security protocols. For instance, the absolute security of keys is a prerequisite for HE and SMC-based security schemes. They perform iterative attacks to impair the performance of the model by synchronizing and uploading the compromised malicious parameters to the server for aggregation. Furthermore, the partially privacy-preserving FL model is threatened by dishonest clients and servers working together to steal private information (such as private keys). Table 1 compares and analyzes the effects of various malicious entities on the model. Whether the model can converge sufficiently determines the level of security concern. We merely categorize the threats our system faces from low to high. The threat to the client depends on how many attacks are taking place, while the server is under high threat because it has too much information. Threats to client privacy are used to gauge privacy. Even though a single client has little information and poses little risk, collusion with the server compromises the information of other users. The collusion attack is a notable exception, and it is harder to analyze because different entities colluded in it. However, we set the threat level as high because server-based collusion attacks are common.

Position	Malicious Entities	Security Treat	Privacy Treat	Reference
Internal	Server	high	high	Poisoning [8]
Internal	Client	medium	low	Poisoning [12], Backdoor [13]
External	Attacker	medium	high	Inference [2]
Collusion	Server and Client	high	high	Sybil-based Collusion Attacks [14]

Tableau 2.1: Evaluation of the threat of different attack entities.

2.7.2 Security Threats in FL

According to this thesis, security attacks aim to impair the model's availability and robustness. A security attack specifically is the potential for a vulnerability to be used by a malicious or curious attacker to compromise the security of a system and violate its privacy policy. Here are a few common attack motifs, and Table 2.2 provides a summary of the primary attacks.

2.7.2.1 Poisoning Attack

One of the most frequent security attacks in Florida is poisoning. The accuracy of the model can be negatively impacted by malicious training data and even model weights because each client can have an impact on the overall model. A poisoning attack attempts to undermine the model's usability by weakening its capacity for generalization. Although there are numerous ways to poison someone at the moment, they can roughly be divided into data poisoning and model poisoning depending on the target of the poisoning. It is important to remember that different participants may start a poisoning attack. The server can effectively carry out both types of poisoning attacks as the training progresses, especially if it is malicious.

The two types of data poisoning—dirty-label attacks and clean-label attacks—can be roughly divided into two groups. Injecting desired target labels into training datasets causes the former to frequently misclassify. The typical dirty-label attack is a label-flipping attack [15], which flips a feature-invariant sample's label in order to trick the model into classifying it as a different type. Figure 2.8 depicts how malicious adversaries produce lethal training samples by label flipping, eventually tricking the global model into producing false classifications. Clean-label attacks correctly classify poisoned labels during training, in contrast to dirty-label attacks. The classification models, however, will put it in the incorrect class. Clean label attacks are more sneaky than the former because they are less susceptible to the majority of defense strategies based on distribution differences.

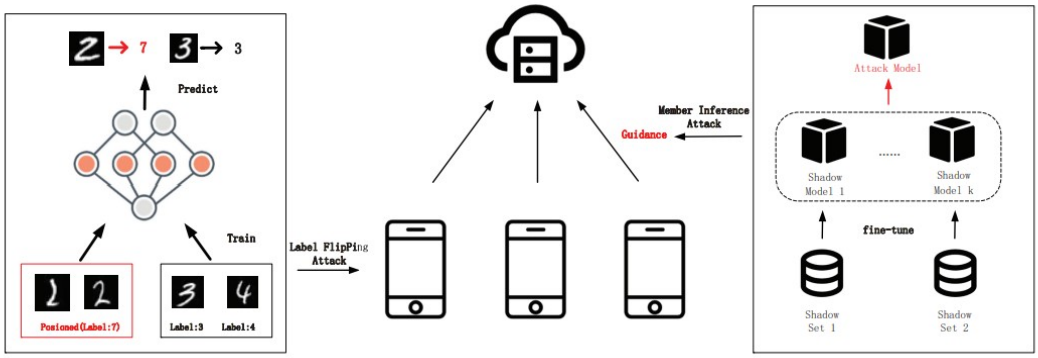


Figure 2.8: Poisoning attack and member inference attack in FL.

Model poisoning, in contrast to data poisoning, typically calls for advanced technical knowledge and substantial computational resources. But it also tends to be more damaging to models. Model poisoning aims to make the model confidently misclassify certain inputs. The work was done by an adversary in control of a small number of malicious agents in order to highly confidently misclassify a set of selected inputs by the global model. The practice of misjudging the model by noise is widespread. The pixel space of opposing images was detected using the noise of various intensities and distributions.

2.7.2.2 Backdoor Attack

A backdoor attack uses triggers to systematically control the decision boundaries of the model, as opposed to a poisoning attack, which modifies the proper decision boundaries by using datasets with different boundaries. To complete the malicious attack, the attacker introduces a hidden backdoor into the model and activates it during the prediction phase. The adversary specifically takes part in FL training and inserts a backdoor into the data of the designated label. The backdoor will be activated and a specific malicious output will be produced by the classifier when the global model classifies the backdoor data. Identification is challenging because the malicious model has a similar level of accuracy. In Ref [16], The authors showed that it is more challenging for the system to identify malicious samples when backdoor implantation is designed for low-probability or edge-case samples. The upside of this flaw is that the backdoor attack swaps out the original uploaded local model for one that gives the attacker control over how the model performs on a backdoor subtask of their choosing. By scaling the model weights, the backdoor mean is maintained. Instead, Ref. [17] tested various attacks on the non-iid dataset and discovered that canonical cropping and weak differential privacy can mitigate backdoor attacks. Due to their superior performance and relative lack of detection, backdoor attacks pose a significant security risk to FL.

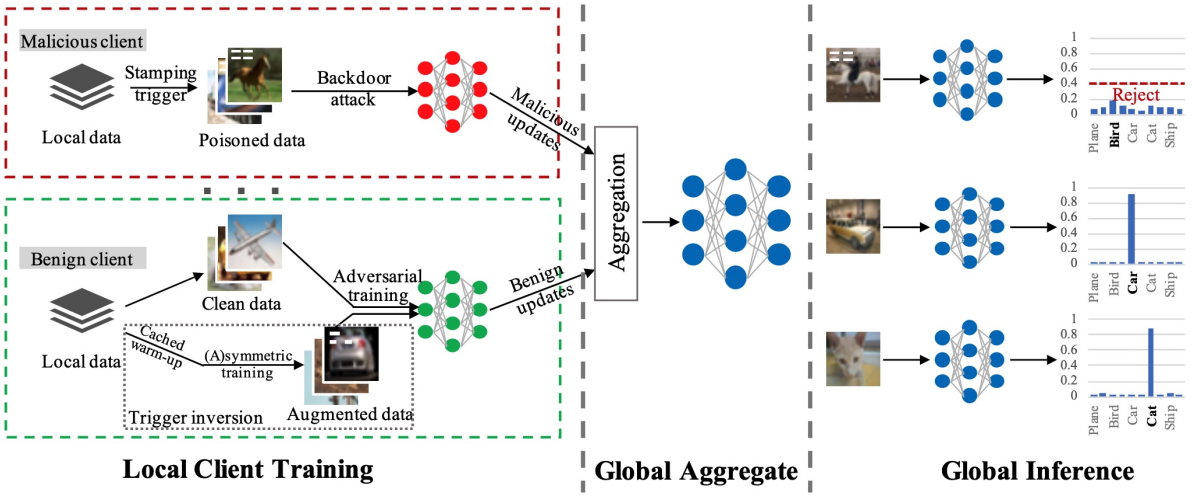


Figure 2.9: Backdoor Attack.

2.7.2.3 Free-Rider Attack

The free-rider attack is sneaky and causes less harm. Without actually providing any data during the training process, a malicious client could take part in obtaining a joint model. As a result, participants with high-quality data may become discouraged and may not earn enough money, which could harm FL training. Less free riders typically cause less harm, but this is unfair to other customers. Currently, this aspect of the issue can be solved by using FL models based on contribution value estimation.

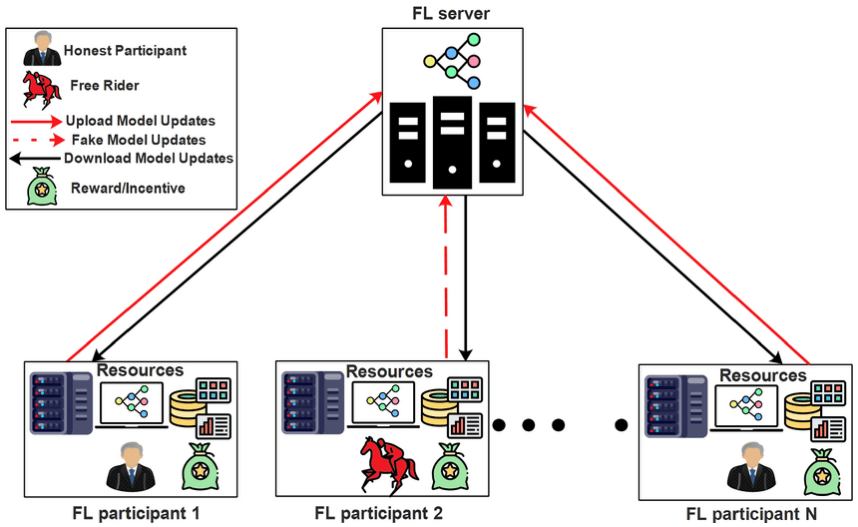


Figure 2.10: Free-Rider Attack.

2.7.2.4 Byzantine Attacks

A Byzantine attack is a sort of federated learning attack in which a hostile actor purposefully corrupts the data or model changes given by a client. This can lead to the global model learning inaccurate information, lowering the model's performance.

In federated learning, there are several approaches to guard against Byzantine attacks. One popular method is to employ a technique known as robust aggregation. Robust aggregation entails combining the model updates from the clients using a statistical method, such as median aggregation or trimmed mean aggregation. These methods are intended to be resistant to outliers, which can aid in mitigating the consequences of Byzantine attacks.

Client selection is another method of fighting against Byzantine attacks. Allowing only trusted clients to participate in the training process is part of client selection. This can be accomplished by forcing clients to pass a security check or through the use of a reputation system.

Byzantine attacks pose a significant risk to federated learning. However, there are several strategies that can be employed to counteract them. Organizations can use these strategies to help preserve their data and ensure that their federated learning models are not jeopardized. These are some types:

1.Krum attack By introducing malicious data into the training dataset, the Krum attack operates. The global model can be trained to learn false information using this malicious data. For instance, a malicious data injection by an attacker could teach the global model that a particular class of data is more likely to be fraudulent than it actually is. This might cause the global model to predict fraudulent transactions incorrectly.

2.Trim attack Trim attack operates by tampering with model updates that clients send. This can be accomplished by altering the weights' values in the model update or by introducing noise. As a result, the global model might pick up inaccurate knowledge. For instance, a hacker could tamper with model updates transmitted by a client who is in charge of training the model on a specific kind of data. This might result in the global model picking up inaccurate knowledge about that class of data.

3.Scaling attack Sending sizable updates to the global model is a technique used in scaling attacks, a kind of Byzantine attack. Malicious clients with access to a lot of data are capable of doing this. The global model's stability and performance may suffer as a result of the large updates.

4.Label flipping Changing the labels of data points in the training dataset is a type of attack known as label flipping. Malicious clients with access to the training dataset may carry out this action. The global model may learn the wrong information as a result

of the altered labels.

2.8 Privacy in FL

Attacks against user privacy undermine the confidentiality of the FL model. Although FL needs to share model parameters instead of sharing local data, there are still ways to steal user local information.

2.8.1 Inference Attack

By teaching the shadow model to behave like the target model, member inference attacks help create an attack model to determine whether the target is a member of the original data. Depending on whether it is possible to steal the gradient parameters, inference attacks can be classified as black-box or white-box attacks. In a white-box attack, the adversary can accumulate snapshots of the FL model's parameters and perform attribute inference by identifying differences between them, which is the same as aggregating updates from all participants minus the adversary [14].

Black box attacks are more practical because they require little knowledge to be acquired. An adversary will typically test the global model with particular inputs and produce confidence scores based on variations in the distribution, which are then used to infer sensitive information. Based on this, introduces a label-only attack that doesn't require confidence scores and doesn't suffer from attack efficiency loss at the same time, further lowering the amount of prior knowledge needed for an attack. Inference attacks against the target can be divided into four categories: category inference attack (CIA), feature inference attack (FIA), label inference attack (LIA), and member inference attack (MIA). All of these result in unintended information leaks to the enemy. The one that has drawn the most attention is MIA. A data record's use in training a target FL model is inferred using MIA in FL models. The most popular technique in Figure 1 for training attack models is the use of shadow models. The target model and the shadow model exhibit similar behaviors, such as supervised training with a verifiable dataset. In order to distinguish between target model members and outsiders, the attack model uses the shadow model to spot behavioral variations in the target model. However, this approach still uses a white-box model because it depends on stealing intermediate gradients. Transfer attacks and borderline attacks have been developed in earlier work to decrease the knowledge needed to attack label-only and can achieve remarkable performance.

2.8.2 Model Inversion Attack

Model inversion attacks, in contrast to inference attacks, typically gather some level of statistical data. They are used to train inversion models that use the model's received preliminary data to reconstruct the client's original data . In this case, the attacker is a trustworthy but honest server. By training the inversion model, the server can recreate the user's initial data from intermediate activations. For instance, split federated learning (SFL) as being susceptible to a MI attack. By using only the recognized intermediate activations, the server is able to reconstruct the team doctor client's original numbers.

Due to the server node's ability to access any intermediate activation, MI resistance during training is noticeably more challenging. But there is a solution out there. Related research shows how to reduce the usability of the model inversion model by minimizing the distance correlation between the original data and the intermediate representation.

2.8.3 GANs

GANs have achieved great success in the field of images. Through gaming techniques, it can produce a significant amount of false data of high quality. As a result, it is improved from both an offensive and a defensive standpoint. One way to improve poisoning and attack inference is through GAN-based techniques. GAN-generated false data makes poisoning attacks easier. By using GAN to generate data, the work achieves over 80 accuracy in both the poisoning and the main task. According to the research in [53], it is possible to use GAN to produce enriched attack data for the shadow model, increasing the accuracy of member inference attacks to 98The system cannot predict all potential threats based on GANs because of their nature. As a result, it is more challenging to stop attacks based on GANs. On the other hand, combining mechanisms with GAN can also increase the FL model's robustness [18]. By sharing the client's generator with the server, the client's collective knowledge can be gathered, which enhances the performance of each client's local network.

Category	Attack	Description	Method	Initiators	Hazards	Ref.
	Poisoning	The attacker injects malicious data to corrupt the output model.	Data Pos, Model Pos	Client, Server	Availability, Robustness	[16]
Security	Backdoor	Prediction by implanting backdoor control models.	Backdoor	Client	Integrity, Robustness	[13]
	Free-rider	The attacker obtains a high-value training model with low-value data.	Random weights attack	Client	Availability, Fairness	[19]
	Inference	High confidence sensitive information deduced by means of attacks.	Member Inf Class Inf, Feature Inf, Label Inf	Server, Attacker	Confidentiality	[20]
Privacy	Model Inversion	Using leaked information to reverse model analysis to obtain private information	Map Inversion	Client	Confidentiality	[21]
	GANs	The attacker obtains a high-value training model with low-value data.	Random weights attack	Client	Confidentiality	[22]

Tableau 2.2: Summary of the main attacks

2.9 Defense Mechanism

Based on the above analysis of security and privacy concerns in federated learning, two key perspectives are worthwhile taking into account to increase FL security: At any point in the training process, FL must recognize and address any potential security threats. Additionally, FL should promote mutual trust among all parties in order to draw in higher-quality data for training. We typically employ some preventative measures to address the first issue. These techniques should be able to identify and eliminate threats as they materialize. Although this is typically cost-effective, the number of threats it can handle is constrained. The main issue with the second issue is preventing direct transmission of sensitive information. Typically, sensitive data is encrypted or transmitted through a secure channel. Such methods frequently involve a reactive respons and processed data are not followed up on.

2.9.1 Anomaly Detection

Statistical and analytical techniques are used in anomaly detection to find events that don't follow expected trends or behaviors. Anomaly detection for the server and anomaly detection for clients are the two broad categories that apply to target-based detection models. For identifying poisoned clients on the server side, anomaly detection techniques like parametric threshold-based, feature-based, and smart contract-based have been suggested. It is common practice to actively defend against data poisoning attacks by testing

the outlier degree of data points on the server. This effectively reduces the harm that poisoned data causes to the global model. For instance, They created a mechanism for detecting outlier data points that can successfully stop attacks like tag reversal and backdoor-based poisoning.

Li et al. [23] test whether users deviate from the FL training regulations using a pre-trained anomaly detection model. Additionally, the server can recognize and verify model updates by saving incremental updates in the blockchain-distributed ledger. Another feature of client-side anomaly detection techniques like BAFFLE enables the detection to be decentralized to the client, with the server merely analyzing the participant's determination's outcomes. In addition, to counter free-rider attacks, anomaly detection techniques based on participant parameter distributions and energy anomalies can be developed.

2.9.2 Blockchain

Peer-to-peer networking is the foundation of blockchain. Blockchain uses a combination of chain, tree, and graph structures to guarantee secure storage and data traceability. Additionally, the blockchain achieves tamper-evident data thanks to the proof of work (POW) consensus mechanism. FL and blockchain are complementary to one another. Blockchain is an inherently secure distributed system, so it makes sense to develop it alongside FL. Together with FL, we can enable multiple servers to copy, share, and distribute all of its data. As shown in Figure 2.11, FL can create a trustworthy third party and carry out a few trusted chain operations, easing the server's trust apprehension [2].

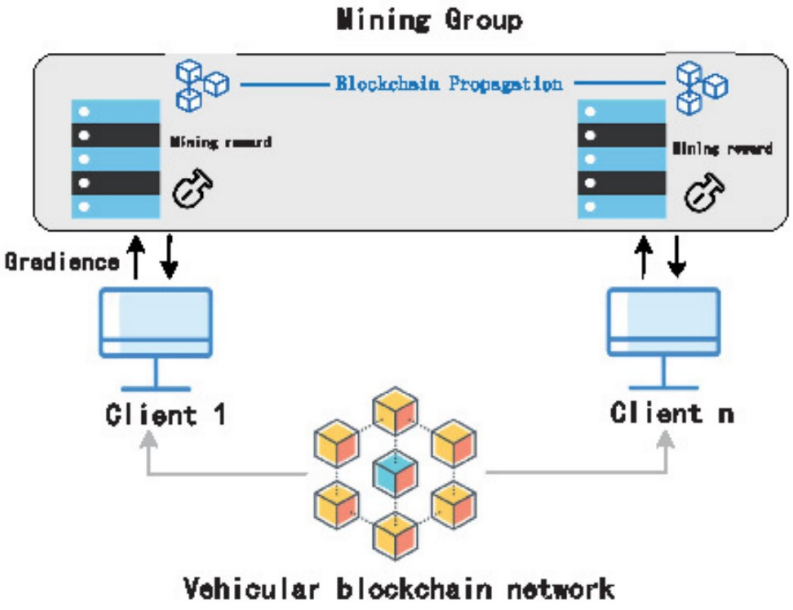


Figure 2.11: Blockchain in VFL.

While the distributed ledger offers secure FL verification, the decentralization of the blockchain may reduce server authority. Along with being verifiable, the blockchain can improve FL's fairness. Smart contracts can be used to improve the transparent and verifiable distribution of incentives, ensuring that all customers receive rewards that align with their values. Blockchain is used in Ref. [24] to promote open procedures and enforce rules. Because the server is not reliant on blockchain technology, it can establish a trust relationship with the user.

2.9.3 Differential Privacy

Since the a priori knowledge of the attack in FL is frequently transmitted as gradient information, it is crucial to hide the information's veracity. By introducing a specific perturbation, differential privacy seeks to mask the actual query. Information from databases was first encrypted using differential privacy. The privacy lines between related data subsets can be blurred to satisfy both security and unpredictability. In situations where time performance is required, DP execution excels because it uses the least amount of time compared to other approaches. Ref. [25] applies differential For the first time, deep learning can measure privacy loss while introducing privacy moments and still maintain high accuracy guarantees for the model. The curator model, the local model, and the shuffle model are three broad categories that describe the FL model when combined with DP. Although the curator model's (CDP) security is lax, it has high accuracy. On the other hand, the local model (LDP) increases security by perturbing local training progress while sacrificing more accuracy. The middle ground between the two is the shuffle model (SDP). This makes the DP model's consideration of privacy, accuracy, and efficiency a research hotspot. On the MNIST dataset, we tested this and illustrated the utility loss in Figure 2.12 in a more understandable way.

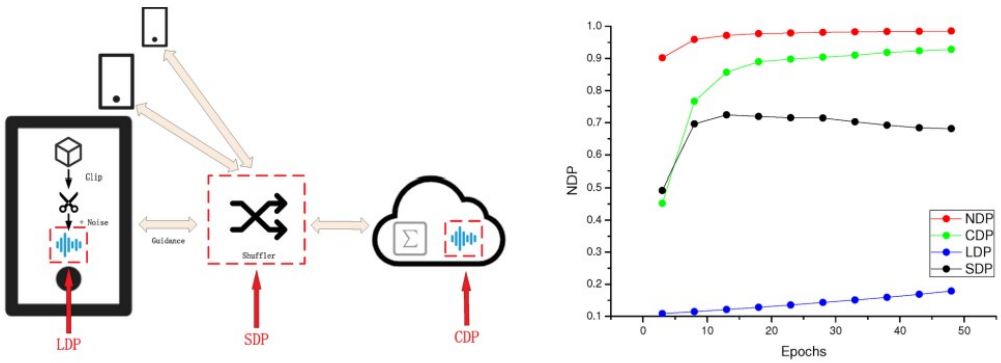


Figure 2.12: The performance of different DP-FL schemes on the MNIST dataset.

In conjunction with DP, FL has received a great deal of academic attention. Although the loss of accuracy is unavoidable for DP, some solutions are proposed to make up for it. Renyi differential privacy uses Renyi entropy to expand the definition of DP and further reduce the privacy budget's upper bound. The distinction between lax DP and strict DP is muddled. It is suggested that the newest discrete Gaussian DP be used to resolve the conflict between utility and security. Additionally, privacy amplification with a third-party shuffler merits consideration.

2.9.4 Homomorphic Encryption

To get the same result, HE encrypts the plaintext computation, converts it to ciphertext, and then decrypts it. Since the operator is only concerned with the ciphertext and is not required to know any decryption information (decryption key K), this eliminates the trust-building and key-transfer issues that are present in conventional cryptography. Consequently, a secure aggregation of gradient information is possible when combined with HE. In order to be secure, a homomorphic system must:

In this case, M and C stand for the operator and the plaintext and ciphertext spaces, respectively. Operations like addition and multiplication on ciphertext can be overloaded. According to Paillier, subdivision operations can be applied to multiplication and addition, respectively. Decsk(m1 C m2) = Decsk(m1 + m2) is an addition. Decsk(m1 C m2) is equal to Decsk(m1 m2) in scalar multiplication. Generally speaking, homomorphic encryption's functionality increases with complexity.

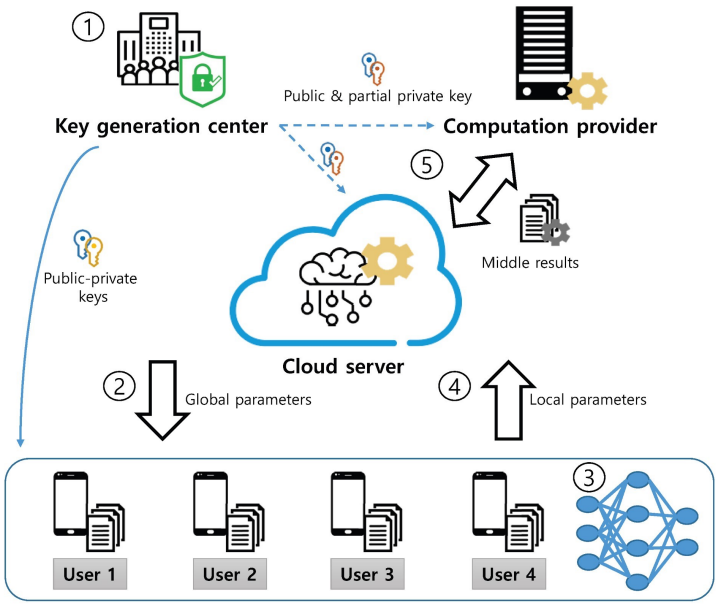


Figure 2.13: Privacy-Preserving Federated Learning Using Homomorphic Encryption

This allows for the separation of HE into two groups: partially homomorphic encryption and fully homomorphic encryption. While the latter satisfies arbitrary operations but is inefficient, the former does so while being efficient for finite operations. Made a proposal for BatchCrypt for Cross-silo federated learning in light of this, lowering the communication and time costs of HE with almost no accuracy loss. FedML was created based on the Paillier homomorphic encryption algorithm to implement federation matrix factorization in scenarios with a modicum of honesty. High interaction overhead and accuracy loss are the main limitations of HE in FL at the moment.

2.9.5 Secure Multiparty Computing

Secure multiparty computing (SMC) typically hides the input data from the output side and encrypts the communication process. Each party's individual output value can only be determined based on its input; no other information is available. Multiple parties can work together to compute functions of mutual interest using secure SMC without disclosing their private inputs to the other parties. SMC is the most appropriate method for secure aggregation because multiple clients in FL interact with the server. Three frameworks can be used to implement SMC: secret sharing, accidental transmission, and inadvertent transmission. Secret sharing is the foundation of secure multiparty computing.

Secret sharing is used to create a compact FL framework for the IoT. To lower the communication overhead, various carefully created local gradient masks and an additional mask reuse scheme are used. VerifyNet is a verifiable FL framework created. that employs a double masking mechanism to protect sensitive data on a private sharing mechanism. Additionally, this strategy guarantees clients' privacy if they leave training.

2.9.6 Trusted Execution Environments

TEE offers integrity and confidentiality assurances for handling private computer code and data. Its primary design goal is to address the issue of secure remote computing, which TFL requires. By way of illustration, Intel SGX offers a secure container that limits the sensitive data that remote users upload in the container. It makes sure that calculations and intermediate data are kept private. Based on this idea, created FLATEE, an effective privacy-preserving federated learning framework that can handle malicious parties without leaking personal information. The symmetric encryption key and the public key in FLATEE are generated by TEE. In the secure enclave TEE, the client executes the privacy algorithm (DP and encryption), and the server executes the privacy aggregation in the aggregator secure enclave TEE. Even so, SGX still has a lot of attack

surfaces due to resource sharing, including page tables, cache, CPU internal structures, etc. TEE attacks like side-channel attacks are frequent. To address these weaknesses, created the ShuffleFL with TEE, in which all players are dynamically organized into hierarchical groups using a randomized grouping algorithm, combined with intra-group gradient segmentation aggregation against rival groups. The specific method is shown in Figure. (2.14).

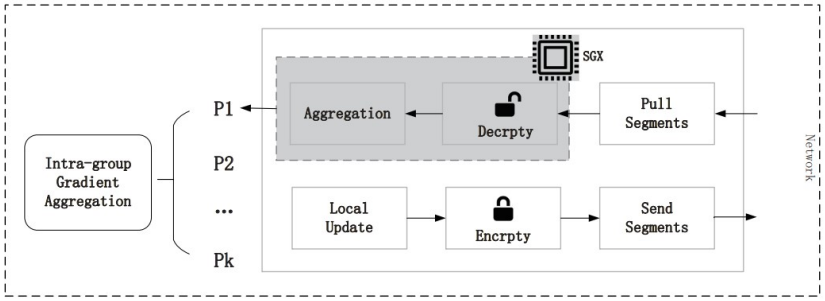


Figure 2.14: ShuffleFL with TEE.

2.9.7 Hybrid

On the one hand, maintaining data localization by itself frequently falls short of providing adequate privacy guarantees in the face of complex security needs. Using DP fuzzy secrets, combined DP and SMC in the database and employed a secret sharing mechanism to slice and dice the restructuring of parameters and solutions for computational and output privacy. Combine FL with the hybrid methodology as an inspiration. In other areas, introduces a novel approach that combines homomorphic encryption methods and differential privacy methods to get the best of both worlds.

has developed a reliable and secure aggregation system and uses distributed DP to improve privacy. On the other hand, a single-defense strategy has inherent drawbacks. For instance, HE and SMC perform poorly in terms of efficiency, and differential privacy has an inherent loss of accuracy. The DP in [**?**] can guarantee a predetermined trust rate while achieving a slight increase in noise when used in conjunction with SMC. This lessens the detrimental effect of DP on the model's usefulness. However, fundamental security mechanism is additive homomorphic encryption, which has a longer training period and more expensive transmission costs. HybridAlpha was created with a better security policy using function encryption to achieve faster communication and lower costs. Together, HE and TEE were used to accurately analyze genomic data. This combined approach offered a good balance between computational support for safe statistical analysis and efficiency.

2.9.8 Security Evaluation

In Table 2.3, we compare the aspects of protection capability, model accuracy, scheme efficiency, model robustness, scalability, and generalization for different methods.

Scheme	Protection	Precision	Efficiency	Robustness	Scalability	Generalizability	Ref.
Anomaly Detection	medium	high	high	medium	high	low	[10]
Data	medium	high	medium	low	high	high	[4]
Differential Privacy	high	low	high	high	high	high	[12]
Homomorphic Encryption	high	high	low	high	low	high	[26]
Sanitization	high	high	low	medium	high	medium	[14]
Secure Multiparty Computing	high	high	medium	high	low	medium	[5]
Trusted Execution Environments	medium	high	high	medium	low	high	[27]

Tableau 2.3: The horizontal comparison of security solutions.

AD, TEE, and DS cannot defend against internal malicious nodes in terms of system security. The related methods based on perturbation and cryptography provide strong theoretical security for hiding the computation's intermediate variables. DP introduces noise to hide important information, which has an impact on the accuracy of the final model convergence. Under LDP, this accuracy loss is frequently intolerable. Due to its streamlined algorithm, the DP-based FL has the lowest time consumption in terms of efficiency performance. For various kinds of FL, the intermediate information masking technique represented by DP has good generalizability. Contrarily, programs like AD must be created especially for various types of FL and have specific bureaus.

When dealing with high-dimensional vectors in FL, which take a long time to encrypt and decrypt, homomorphic encryption is typically ineffective. The robustness of AD and DS to new attacks is poor, and they require timely dynamic updating. The performance of FL in complex situations, such as widespread node distribution and unexpected user dropouts, is indicative of its scalability. The participating nodes in HE and SMC are computational as well as heavily computational and communication-burdened. It is important to note that HE and SMC are cryptographically provably secure, which has an impact on how easily TFL can be interpreted. Training efficiency is drastically decreased when there are many level nodes. The use of TEE is also constrained by a lack of available local computing resources. Performance will significantly suffer if the training process is carried out in the TEE environment. Consider Intel SGX as an illustration; it only permits CPU operation, which restricts the effectiveness of the model (CNN, DNN), and is

dependent on GPU training. Additionally, when the memory goes over the limit, there will be a significant increase in paging overhead.

2.10 Application Of Privacy Preserving FL

2.10.1 Mobile Devices:

A secure aggregation protocol for PPFL in mobile devices was created by Bonawitz (2017). The suggested protocol is robust to all users and operates on high-dimensional vectors. It can manage situations where a user abruptly leaves a training session. In terms of communication with security in an unauthenticated network model, it is also very effective. Additionally, using Tensorflow, Bonawitz (2019) improved their work and created a scalable PPFL framework for mobile devices. The framework addresses a number of problems, including erratic connectivity, halted operations, constrained device storage, and insufficient computing power [9].

2.10.2 Medical Imaging

A PPFL study in medical imaging was suggested by Kaissis (2020). They want to address the moral and legal concerns surrounding patient privacy when using artificial intelligence in the field of medical imaging. Because there are currently no electronic medical records that are standardized, strict requirements must be established. The suggested framework protects patient privacy while using medical imaging from patients to train AI. They examine the interaction of HE, Secure MPC, and DP with federated learning.

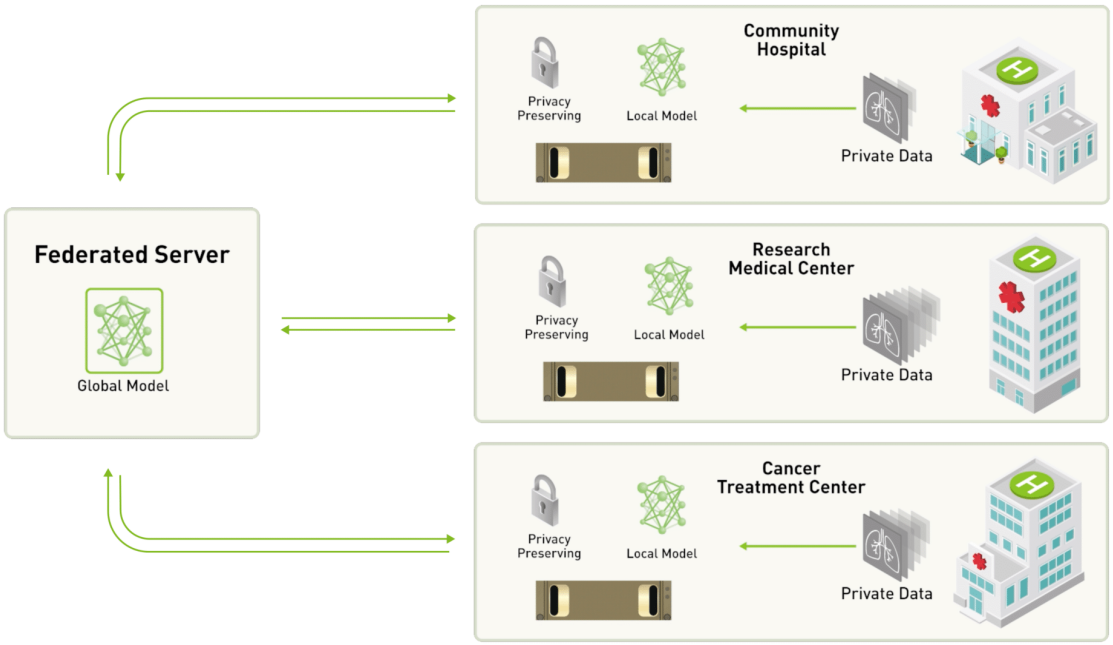


Figure 2.15: Federated learning with privacy preserving in medical imaging.

2.10.3 Traffic Flow Prediction

A PPFL method for traffic flow prediction is the Federated Learning-based Gated Recurrent Unit Neural Network Algorithm (FedGRU) (Liu et al. 2020). It differs from other centralized learning methods in that it employs a secure parameter aggregation mechanism. FedGRU is able to avoid direct raw data sharing between participants thanks to the federated learning algorithm. They use a random sub-sampling protocol to improve scalability. This reduces the communication overhead, which is suitable for large-scale systems. They also create a clustering method that combines the best global model with spatial traffic flow data to improve prediction accuracy.

2.10.4 Healthcare

Grama (2020) used a healthcare dataset to propose a robust aggregation method in PPFL. They ran experiments to see how DP and k-anonymity affected model accuracy. Typical PPFL methods are sensitive to changes in the local model. It can be a barrier to developing a strong global model. During the training process, their experiments revealed that the proposed method successfully detected and discarded malicious local parties. They also demonstrated that DP had no significant effect on the time spent during the aggregation process.

2.10.5 Android Malware Detection

Hsu (2020) proposed a PPFL system for detecting Android malware. To protect participants' privacy, the system employs static and meta-data analysis, as well as federated learning. Edge computing is used in the system design to reduce latency and communication costs. They use an Android malware dataset to combine SVM and Secure MPC in a PPFL system. The results of the experiments revealed that the proposed system has comparable accuracy with the same amount of data as a traditional machine learning system with no privacy preservation.

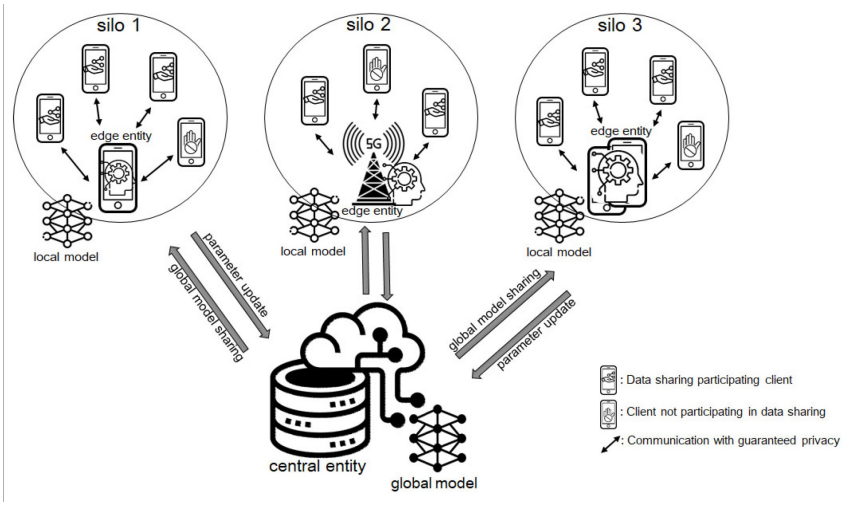


Figure 2.16: Distributed Detection of Malicious Android Apps While Preserving Privacy Using Federated.

2.11 Trust in Federated Learning

Trust in federated learning refers to participants' belief in the system and in one another. This is significant because federated learning relies on participants sharing their data with one another, and they must be confident that their data will be safe and not used maliciously.

A variety of factors can contribute to trust in federated learning, including:

• System security: Participants must have confidence that their data will be safe from unauthorized access.

• The system's fairness: Participants must have confidence that the system will not be used to discriminate against them.

• System transparency: Participants must be able to understand how the system works and how their data is used.

A variety of mechanisms can be used to build and maintain trust, including:

Security: The system should be designed to prevent unauthorized access to data. This can be accomplished using a variety of techniques, including encryption and access control.

Fairness: The system should be built to prevent discrimination. This can be accomplished by ensuring that all participants are treated equally and that the system does not favor any one group of participants over another.

Transparency: Participants should be able to see the system. This implies that participants should understand how the system works and how their data is used. [6].

The success of federated learning is dependent on trust. Participants will be hesitant to share their data if they lack trust, and the system will not be able to reach its full potential.

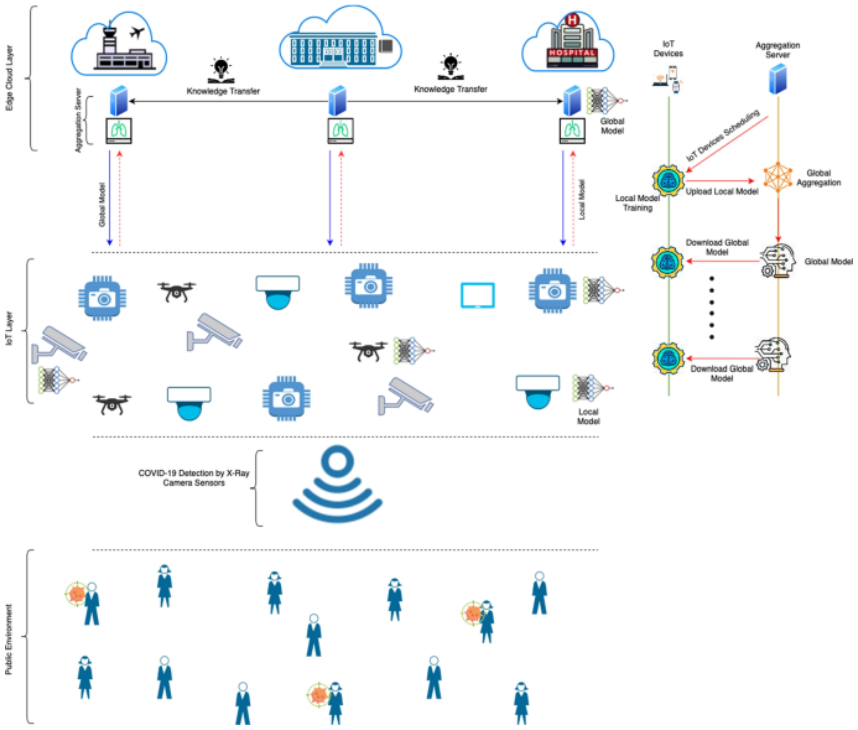


Figure 2.17: Trust-Augmented Deep Reinforcement Learning for Federated Learning Client Selection.

2.12 Trust Models for Federated Learning

Federated learning trust models are intended to ensure that participants in a federated learning system can trust one another. This is significant because federated learning relies on participants sharing their data with one another, and they must be confident that their data will be safe and not used maliciously.

There are several trust models that can be used in federated learning. Among the most common trust models are:

Centralized trust model: A single entity manages the trust relationships between the participants in a centralized trust model. This entity can be a trusted third party, such as a cloud service provider, or it can be one of the federated learning system's participants.

Decentralized trust model: In a decentralized trust model, no single entity is in charge of managing trust relationships. Instead, the participants manage their own trust relationships. This can be accomplished through the use of a variety of techniques, such as reputation systems or blockchain technology.

Hybrid trust model: A hybrid trust model is a hybrid of a centralized and a decentralized trust model. A single entity is responsible for managing the trust relationships between the participants in a hybrid trust model, but the participants also play a role in managing the trust relationships.

The specific trust model used will be determined by the application's requirements. The general principles outlined above, on the other hand, can be used to implement trust models in any federated learning setting.

Here are a few examples of trust models used in federated learning:

• Federated Learning Client Platform (FLC) by Google: FLC employs a centralized trust model. The FLC server is in charge of managing the participants' trust relationships. To ensure data security and privacy, the FLC server employs a number of techniques such as encryption, access control, and auditing.

• Private Set Intersection (PSI) by Apple: PSI employs a decentralized trust model. PSI ensures data privacy by employing a cryptographic technique known as secret sharing. PSI participants do not share their data, but they can still compute the intersection of their sets.

TEEs (Trusted Execution Environments) are a type of hardware security module that can be used to protect data privacy. TEEs can be used to implement a wide range of trust models, including centralized, decentralized, and hybrid trust models.

A great deal of research is being conducted on trust models for federated learning. Among the most active research areas are:

Security: Researchers are working on new techniques for ensuring data security in federated learning. This includes the creation of new encryption methods, access control mechanisms, and auditing methods.

Privacy: Researchers are working on new techniques for protecting data privacy in federated learning. Developing new cryptographic techniques, data anonymization techniques, and differential privacy techniques are all part of this.

Trustworthiness: Researchers are developing new metrics to assess the reliability of

federated learning systems. This includes creating metrics to assess federated learning systems' security, privacy, and fairness.

Trust models are an essential component of federated learning. Federated learning systems can use trust models to ensure that participants can trust one another and that data is safe and secure.

Chapter 3

Implementation and Materials

3.1 Introduction

Over traditional machine learning methods, federated learning has several security advantages. For starters, federated learning eliminates the need for data to be centralized on a server, making it more difficult for attackers to access the data. Second, federated learning can be used to train models on sensitive data, such as medical or financial information, without jeopardizing the data's privacy.

However, federated learning has some security issues. For starters, federated learning necessitates communication between devices, which makes it vulnerable to attacks. Second, federated learning can be vulnerable to data poisoning attacks, in which an attacker manipulates data shared by devices to corrupt the model.

3.2 Overview

The score method is a technique for assessing the accuracy of federated learning models. The score method computes the similarity between the model's predictions and the ground truth labels. The similarity is then used to compute a score that can be used to compare the performance of various models.

The score method is a simple technique that can be used to assess the accuracy of federated learning models. The score method is also a privacy-preserving technique, which means that no sensitive data is required to be shared.

The model is trained using a dataset distributed across multiple devices. On the test set, the model makes predictions. The predictions are compared to the labels on the ground. The similarity between the predicted and true labels is calculated. A score is calculated based on the similarity. The score is used to evaluate the performance of

various models. The score method can be used to assess the accuracy of federated learning models for a wide range of tasks such as classification, regression, and forecasting. The score method is a privacy-preserving technique, which means that no sensitive data must be shared. As a result, the score method is an appealing option for assessing the accuracy of federated learning models in sensitive situations, such as healthcare and finance.

3.3 Propesed Approach

The ScoTrust is a technique for increasing the security of a model. It works by giving each client a score based on their behavior. Clients with a high score are thought to be trustworthy, while those with a low score are thought to be untrustworthy. When making predictions, the model only takes into account data from reliable clients. [28].

ScoTrust is useful for defending against various attacks, including the Byzantine attack. A Byzantine attack occurs when a group of malicious clients collaborate to disrupt the training process. By identifying and removing malicious clients from the training process, the score method can help to defend against this attack. [22].

Scotrust is a simple and effective way to improve a model's security. It is simple to set up and can be used to defend against a wide range of attacks.

ScoTrust can be used in a fraud detection model to identify and remove fraudulent transactions.

ScoTrust can be used in a spam filtering model to identify and remove spam emails.

ScoTrust in a credit scoring model can be used to identify and remove risky borrowers.

ScoTrust can be used to improve a model's security by detecting and removing malicious clients. Malicious clients are those who attempt to disrupt the training process or influence the model's predictions. By assigning a low score to malicious clients, the score method can assist in identifying them. Clients with low scores are either removed from the training process or have their predictions ignored. [27].

3.4 Model

Federated learning is a machine learning method in which multiple devices collaborately train a shared model while storing their data locally. As a result, it is a more secure and privacy-protecting alternative to traditional machine learning methods, in which data is centralized on a server.

The score method is a technique for identifying and eliminating Byzantine clients in a federated learning model. Byzantine clients are malicious clients who provide incorrect

or incomplete data to the model on purpose. The score method works by assigning a score to each client based on the quality of their data and their behavior. Clients with low scores are more likely to be Byzantine and are excluded from the model.

The combination of federated learning and the score method can significantly improve machine learning model security and privacy. Because federated learning keeps data local, attackers have a more difficult time accessing it. The score method can then be used to identify and remove Byzantine clients, increasing the model's security.

Some of the advantages of using the score method to improve the security of federated learning models are as follows:

Increased security: The scoring technique can identify and remove Byzantine clients, increasing the model's security.

Improved accuracy: Excluding Byzantine clients, which can offer erroneous or insufficient data, the scoring technique can help to increase the model's accuracy.

Reduced data leakage: Federated learning keeps data local, potentially lowering the risk of data leakage.

Increased privacy: Because federated learning keeps data local, it can increase data privacy.

Overall, the score technique is an effective method for enhancing the security and privacy of federated learning models.

3.5 Datasets

we use the MNIST and CIFAR-10 as a datasets:

MNIST: is a collection of handwritten digits. It is a widely used dataset for training and assessing machine learning models. The collection contains 60,000 handwritten digit images, each 28x28 pixels in size. The photos are divided into two groups: 50,000 for training and 10,000 for testing.

CIFAR-10: is a collection of photographs of items. It is yet another well-known and popular dataset for training and assessing machine learning models. The collection contains 60,000 photos of ten different object classes, each measuring 32x32 pixels. The photos are divided into two groups: 50,000 for training and 10,000 for testing.

MNIST and CIFAR-10 are both relatively tiny datasets, making them simple to work with and evaluate. They are also well-balanced datasets, which means that each class is similarly represented. As a result, they are perfect for training and assessing machine learning models.

MNIST and CIFAR-10 are very easy datasets in addition to being small and well-

balanced. As a result, they are great for beginners who are just getting started with machine learning. They are also an excellent alternative for more experienced machine learning practitioners looking to put new algorithms and strategies to the test.

Finally, MNIST and CIFAR-10 are open source datasets. This means that anyone can use them without restriction. This makes them an excellent alternative for master's thesis studies because it allows students to share their findings with the larger machine learning community.

Here are some of the benefits of using MNIST and CIFAR-10 as a datasets for our research:

• They are widely known and used datasets. This means that there is a lot of study on these datasets available, which might assist students in getting started with their research.

• The datasets are relatively tiny. This makes them simple to work with and analyze, which is useful for students learning about machine learning for the first time.

• The datasets are well-balanced. This ensures that each class in the dataset is represented equally, which is critical for training and testing machine learning models.

• They are freely accessible datasets. This means that students will be able to share their findings with the larger machine learning community.

Overall, MNIST and CIFAR-10 are great datasets for machine learning master thesis studies. They are well-known, commonly utilized, modest in size, well-balanced, and freely available. As a result, they are suitable for both novice and experienced machine learning practitioners. [29]

3.6 Framework

TensorFlow Federated (TFF) is an open-source framework for machine learning and other computations on decentralized data. TFF has been developed to facilitate open research and experimentation with Federated Learning (FL), an approach to machine learning where a shared global model is trained across many participating clients that keep their training data locally. For example, FL has been used to train prediction models for mobile keyboards without uploading sensitive typing data to servers.

TFF enables developers to simulate the included federated learning algorithms on their models and data, as well as to experiment with novel algorithms. Researchers will find starting points and complete examples for many kinds of research. The building blocks provided by TFF can also be used to implement non-learning computations, such as federated analytics. TFF's interfaces are organized in two main layers:

Federated Learning (FL) API This layer offers a set of high-level interfaces that al-

low developers to apply the included implementations of federated training and evaluation to their existing TensorFlow models.

Federated Core (FC) API At the core of the system is a set of lower-level interfaces for concisely expressing novel federated algorithms by combining TensorFlow with distributed communication operators within a strongly-typed functional programming environment. This layer also serves as the foundation upon which we've built Federated Learning.

TFF enables developers to declaratively express federated computations, so they could be deployed to diverse runtime environments. Included with TFF is a performant multi-machine simulation runtime for experiments [30].

3.7 Python

In my thesis I use Python, a high-level general-purpose programming language widely used in data science and for the production of deep learning algorithms. This brief tutorial introduces Python and its libraries.

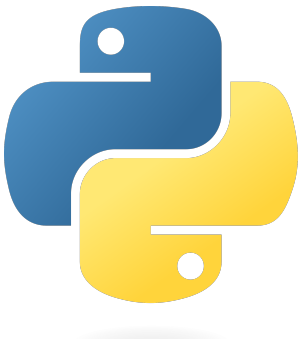


Figure 3.1: Python

3.8 Libraries

Our model needs to perform better than this. I am using the following libraries: Pandas, Numpy, Keras, Matplotlib, and TensorFlow

3.8.1 Tensorflow

TensorFlow est une plate-forme Open Source de bout en bout dédiée au machine learning. Elle propose un écosystème complet et flexible d'outils, de bibliothèques et de ressources communautaires permettant aux chercheurs d'avancer dans le domaine du machine learn-

ing, et aux développeurs de créer et de déployer facilement des applications qui exploitent cette technologie [30].

Figure 3.2: Tensorflow

3.8.2 Pandas

Pandas is an open-source Python library that uses strong data structures to provide high-performance data manipulation and analysis. Pandas is derived from the term Panel Data, which is an Econometrics from Multidimensional data. Wes McKinney, a developer, began developing pandas in 2008 in response to a demand for a high-performance, versatile tool for data analysis. Python was mostly used for data munging and preparation prior to Pandas. It made very little contribution to data analysis. Pandas solved this issue. We can use Pandas to do five common phases in data processing and analysis, independent of data origin: load, prepare, manipulate, model, and analyze. Python with Pandas is utilized in a variety of academic and commercial disciplines such as finance, economics, statistics, analytics, and so on. [31].

Figure 3.3: Pandas

3.8.3 Numpy

This library, whose name means numerical Python, constitutes the core of many other Python libraries that have originated from it. Indeed, NumPy is the foundation library for scientific computing in Python since it provides data structures and high-performing functions that the basic package of the Python cannot provide. In fact, NumPy defines a specific data structure that is an N-dimensional array defined as ndarray. [32]

Figure 3.4: Numpy

3.8.4 Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research [33].

Figure 3.5: Keras

3.8.5 Matpotlib

This package is the Python library that is currently most popular for producing plots and other data visualizations in 2D. Since data analysis requires visualization tools, this is the library that best suits this purpose. [33]

Figure 3.6: Matpotlib

3.9 Summary

In federated learning, the Score method is a technique for locating Byzantine clients. Malicious clients are known as byzantine clients because they purposefully send the server inaccurate or altered data. The Score method assigns each client a score based on a variety of considerations, such as the client's reputation, the number of samples provided,

the consistency of updates, and participation in training rounds. Low-scoring clients can be taken out of the training set because they are more likely to be Byzantine clients.

Several datasets, including MNIST, CIFAR, and TFF, have been used to evaluate the Score method. On these datasets, it has been demonstrated that the ScoTrust works well for identifying Byzantine clients.

In federated learning, the ScoTrust is a promising strategy for locating Byzantine clients. The Score method is made simple to use because it is implemented in the TensorFlow Federated (TFF) library.

Chapter 4

Experimental Results

4.1 Introduction

In this chapter, the results obtained after implementing the proposed models are presented. Then evaluate the results.

4.2 Confusion matrix and Test Accuracy

A score method is a methodology used in federated learning to evaluate the performance of a federated learning model. To evaluate the model, score techniques often employ a number of metrics such as accuracy, precision, and recall.

The specific scoring mechanism utilized will be determined by the application's requirements. The broad principles discussed below, on the other hand, can be applied to build score methodologies in any federated learning scenario.

Here are some of the most frequent federated learning score methods:

• **Accuracy:** Accuracy is the percentage of predictions that are correct. It is calculated as follows:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Where:

TP: True Positives

TN: True Negatives

FP: False Positives

FN: False Negatives

• **Precision:** Precision is the percentage of positive predictions that are correct. It is calculated as follows:

Precision = TP / (TP + FP)

•**Recall:** Recall is the percentage of actual positives that are correctly identified. It is calculated as follows:

Recall = TP / (TP + FN)

• **F1 Score:** The F1 score is a weighted harmonic mean of precision and recall. It is calculated as follows:

F1 Score = 2 * (precision * recall) / (precision + recall)

The F1 score is a good measure of overall model performance, as it takes into account both precision and recall.

In addition to the score methods mentioned above, there are a number of other score methods that can be used to evaluate federated learning models. The specific score method that we will use in our application all the metrics we mentioned above.

4.3 Experimental

Our ScoTrust achieves the defense goals:

Identify Byzantine clients: The ScoTrust can be used to identify Byzantine clients by computing a score for each client based on their model parameters' similarity to the server's model parameters. Clients with high scores are more likely to be Byzantine and are thus excluded from the training set.

Reduce the impact of Byzantine assaults: By excluding Byzantine clients from the training set, the scoring approach can be used to reduce the impact of Byzantine attacks. This can aid in improving the model's accuracy and preventing it from being compromised by bad actors.

Improve federated learning robustness: The score approach can be used to improve federated learning robustness by making it more difficult for hostile actors to interrupt the training process. This can help to ensure that, even in the face of threats, federated learning can be utilized to train accurate and dependable models.

4.3.1 No Attack

First, when there is no attack, our ScoTrust has Accuracy similar to FedAvg and FLTrust.Using MNIST datasets, The accuracy of ScoTrust is 95, higher than The FedAvg.

FedAvg: FedAvg [23] was proposed by Google. FedAvg computes the average of the clients' local model updates as the global model update, where each client is weighted by

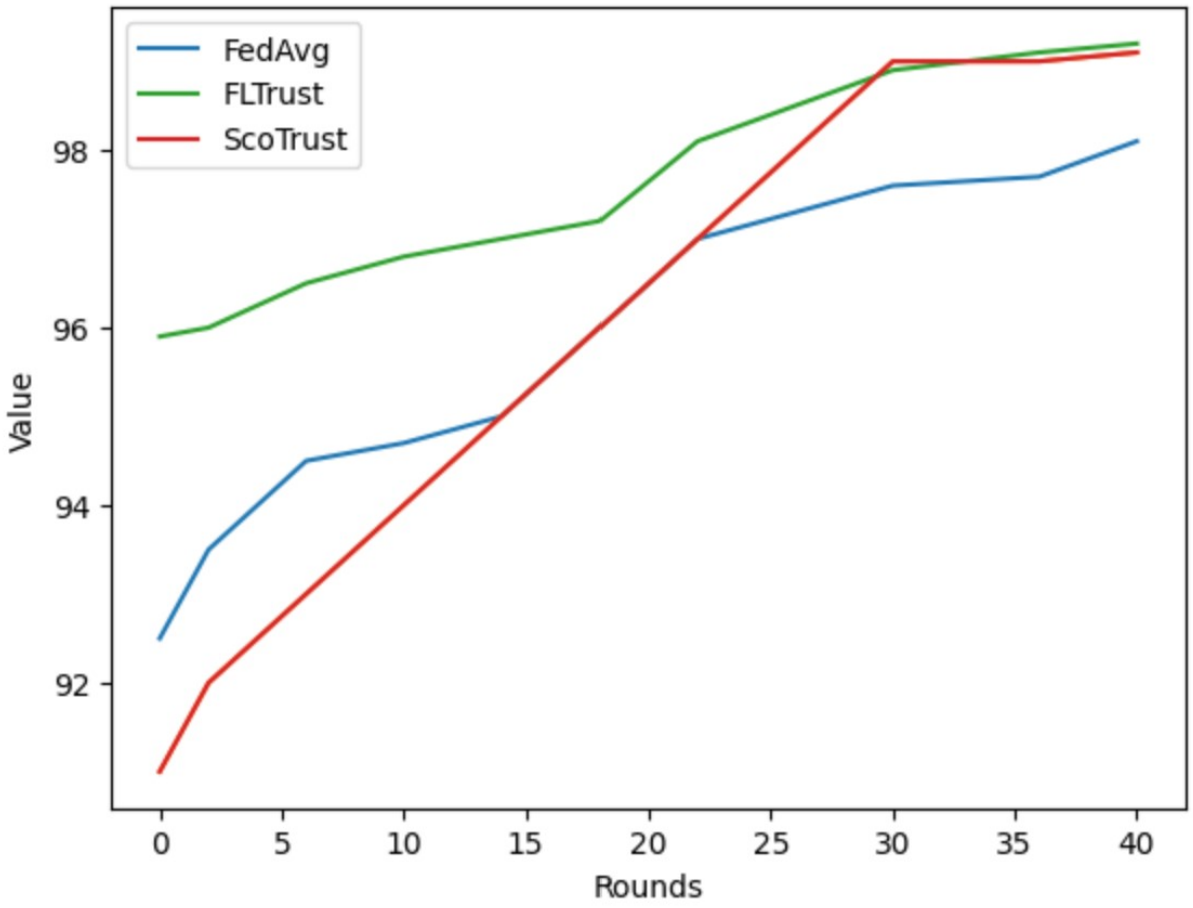


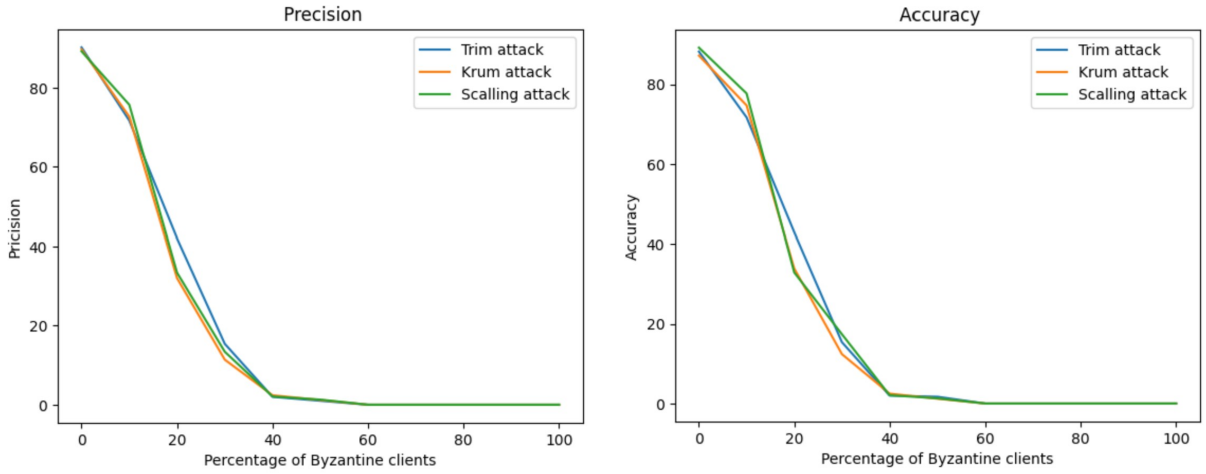
Figure 4.1: Accuracy under no attacks

its number of training examples. The local training dataset size on the client and the total number of training examples. FedAvg is the state-of-the-art FL method in non-adversarial settings. However, the global model in FedAvg can be arbitrarily manipulated by a single malicious client.

FLTrust: FLTrust [2] is that the server itself collects a clean small training dataset (i.e., root dataset) to bootstrap trust in FLTrust. Our extensive evaluations on six datasets show that FLTrust with a small root dataset can achieve Byzantine robustness against a large fraction of malicious clients. In particular, FLTrust under adaptive attacks with a large fraction of malicious clients can still train global models that are as good as the global models learned by FedAvg under no attacks.

4.3.2 with Attack

As you can see, the accuracy and precision both decrease at the 20% of Byzantine clients for both datasets, this is because the attacks were able to corrupt the data, preventing



(a) MNIST **(b)** CIFAR

the model from learning anything.

Tableau 4.1: Metrics under different attacks

Attack	**Krum attack**	**Trim attack**	**scalling attack**
Accuracy	35%	42%	38%
Precision	40%	33%	37%

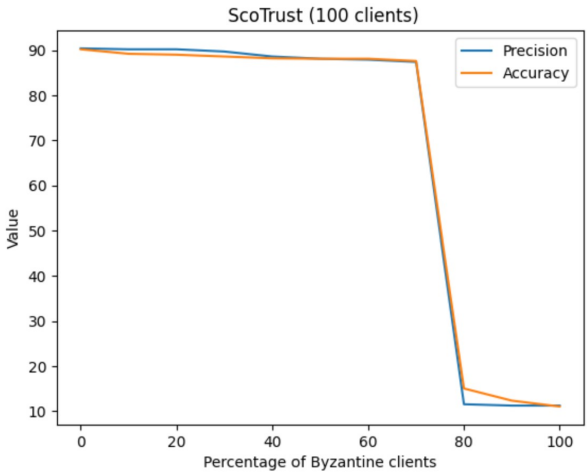
4.3.3 ScoTrust

Next, we simulate different attacks (Byzantine attacks) we have mentioned in Chapter 2 Under the section of Threats in FL. Achieving our first goal.

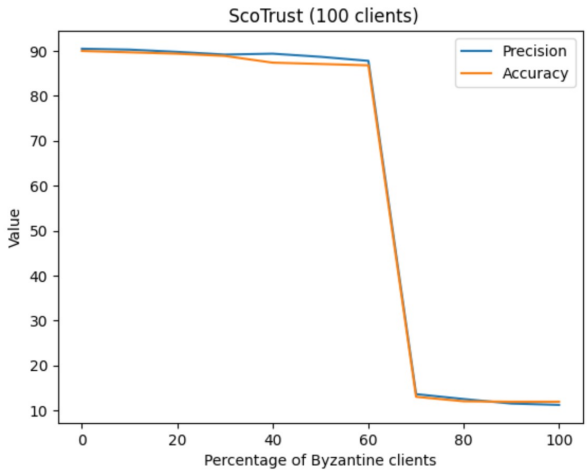
Tableau 4.2: Impact of the fraction of malicious clients

Attack	**10%**	**20%**	**30%**	**60%**	**80%**	**95%**
Krum attack	91	89	88	87	25	23
Trim attack	92.1	89.2	88.4	87.5	87	22
Scalling attack	91.4	89	88.3	87.6	25	23.5

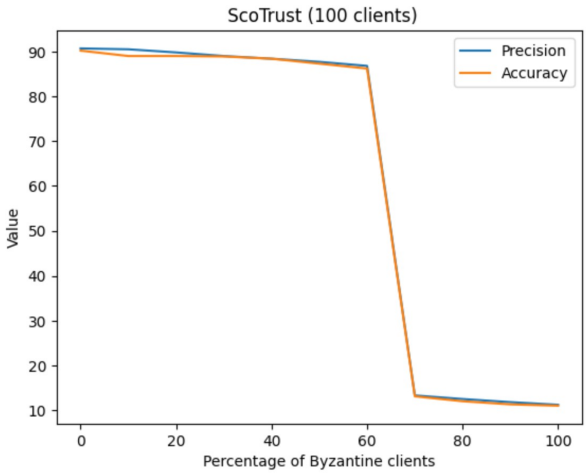
(a) Trim Attack - MNIST



(b) Trim Attack - CIFAR



(c) Krum Attack - MNIST



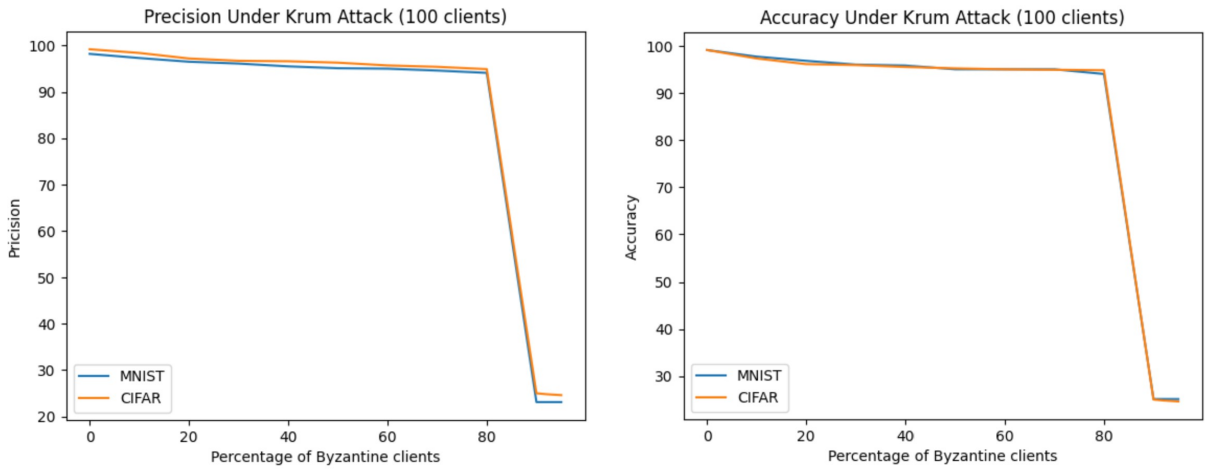
(d) Krum attack - CIFAR

Figure 4.3: Impact of the fraction of malicious clients on accuracy - precision of different FL methods under different attacks

Impact of the number of malicious clients: Figure 4.3 shows the accuracy - precision rates of ScoTrust under different attacks on MNIST, when the fraction of malicious clients increases from 0 to 80% Accuracy and Precision are both still high above 90. Therefore ScoTrust cannot be applied when the fraction of malicious clients exceeds 80% because the number of local model updates removed by ScoTrust is twice the number of malicious clients.ScoTrust shows results only when the percentage of Byzantine clients is under 80%.

Impact of the measure of threshold:

Fig 4.4 shows the Accuracy - Precision under Krum attack for both MNIST and CIFAR datasets, we observe that the metrics we used reach 90 when the percentage of malicious clients went to 85% because we set the threshold of ScoTrust up to 0.8.



(a) **(b)**

Figure 4.4: Impact of high threshold in the training set

4.3.4 Discussion

ScoTrust vs. False positive:

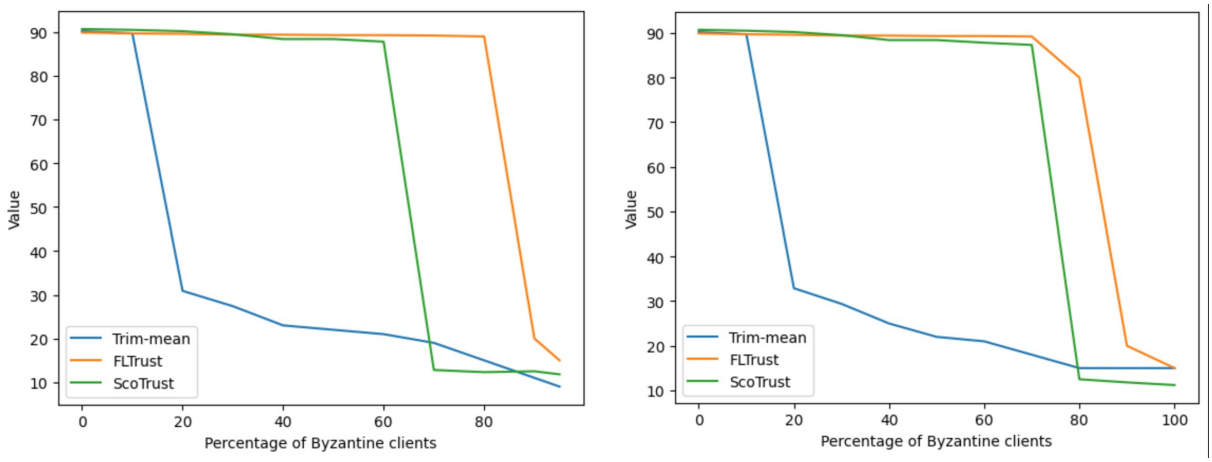
False positives can be reduced by using a more robust scoring method. A more robust scoring method would make it less likely that a legitimate customer would be identified as a Byzantine customer. There are many ways to make the evaluation method more robust. Use more functions or more sophisticated algorithms to calculate the score. By tracking model accuracy, you can monitor false positives. If your model accuracy drops significantly, your training set may contain false positives. By monitoring model accuracy, you can identify and fix false positives before they become serious problems. Misinformation can be reduced by using other methods to identify Byzantine customers.

ScoTrust vs False negative:

Using a more robust scoring method can reduce false negatives. With a more robust scoring method, Byzantine customers are less likely to be identified as legitimate customers. There are many ways to make the evaluation method more robust. B. Use more functions or more sophisticated algorithms to calculate the score. False negatives can be monitored by tracking model accuracy. If the accuracy of your model drops significantly, your training set may contain false negatives. By monitoring model accuracy, you can identify and fix false negatives before they become serious problems. Using other methods to identify Byzantine clients can reduce false negative results. The scoring method is not the only way to identify Byzantine clients. Other methods can also be used to identify Byzantine clients, such as monitoring client behavior during training. Using multiple methods to identify Byzantine customers can reduce the chance of false positives.

4.3.5 Comparison

We observe that, under existing attacks, FLTrust can tolerate up to 90% of malicious clients. Specifically, FLTrust under these attacks still achieves Accuracy rates similar to ScoTrust without attacks when up to 90% of the clients are malicious. However, existing Byzantine robust FL methods can tolerate much less malicious clients.



(a) MNIST **(b)** CIFAR

Figure 4.5: Comparison between different FL method with different Datasets

Trimmed Mean (Trim-mean) [34]: Trimmed mean is a coordinate-wise aggregation rule that considers each model parameter individually. For each model parameter, the server collects its values in all local model updates and sorts them. Given a trim parameter, the server removes the largest k and the smallest k values and then computes the mean of the remaining values as the value of the corresponding parameter in the global model update. The trim parameter k should be at least the number of malicious clients to make Trim-mean robust. In other words, Trim-mean can tolerate less than 50% malicious clients.

4.4 Code Source

```
class ScoreMethod:

  def __init__(self, threshold=0.5):
    self.threshold = threshold

  def calculate_scores(self, clients):
    scores = []
    for client in clients:
      score = self.calculate_score(client)
      scores.append(score)
    return scores

  def calculate_score(self, client):
    score = 0
    score += client.num_participations
    score += client.consistency
    score += client.num_samples
    score += client.reputation
    return score

  def identify_byzantine_clients(self, scores):
    byzantine_clients = []
    for score in scores:
      if score < self.threshold:
        byzantine_clients.append(score)
    return byzantine_clients

  def remove_byzantine_clients(self, clients):
    byzantine_clients = self.identify_byzantine_clients(clients)
    for client in byzantine_clients:
      clients.remove(client)
    return clients
```

Figure 4.6: Snippet Code of ScoTrust

Our ScoreMethod class has the following methods:

init: This method initializes the class with a threshold. The threshold is used to determine which clients are Byzantine.

calculatescores: This method calculates a score for each client. The score is a measure of how likely the client is to be Byzantine.

identifybyzantineclients: This method identifies Byzantine clients. A client is Byzantine if its score is below the threshold.

removebyzantineclients: This method removes Byzantine clients from a list of clients.

Our ScoTrust class uses the following features to calculate a score for each client:

- The number of times the client has participated in training
- The consistency of the client's data
- The number of samples the client has provided
- The client's reputation

4.5 Conclusion

In federated learning, the Score method is a promising strategy for locating Byzantine clients. Although it is a quick and easy method to identify Byzantine clients, there are some drawbacks, including accuracy based on data quality, computational cost, and vulnerability to attacks. The Score method can be enhanced in a number of ways, including by using better data quality, more effective algorithms, and attack defenses. The Score method can be used to increase the security of federated learning by addressing these issues and combining various techniques.

By locating and removing Byzantine clients from the network, the Score method can be used to increase network security. Malicious clients are known as byzantine clients because they purposefully send the server inaccurate or altered data. As a result, the model's accuracy may suffer during the training process. The Score method can assist in ensuring that the model is trained on accurate data and is not vulnerable to malicious actors by removing Byzantine clients. The Score method is an effective tool for federated learning security enhancement. Organizations can increase the security and accuracy of their machine learning models by employing the Score method.

The Score method (ScoTrust) is an approach that shows promise, but it is still being refined. To increase the Score method's precision, security, and effectiveness, more research needs to be done. To increase the overall security of federated learning systems, the Score method can be used in conjunction with other security measures like access control and encryption. Organizations considering using federated learning to train machine learning models on sensitive data can benefit from using the Score method.

General Conclusion

The score method is a promising strategy for federated learning's Byzantine client identification. Before the score method can be widely used, there are still a few issues that need to be resolved.

One problem is that the quality of the data can have an impact on how accurate the score method is. The score method may not be able to accurately identify Byzantine clients if the data is noisy or lacking. The score method's potential cost in computation is another drawback. For large datasets, the server must compute a score for every client, which can take some time. Finally, there is a risk of attacks using the score method. Malicious clients may attempt to manipulate the system by sending phony data or by manipulating the score calculation.

To solve these issues, a variety of research avenues could be pursued. One approach is to create more robust scoring systems that are less vulnerable to attacks. Another approach would be to create ways for automatically recognizing and eliminating Byzantine clients from the training set. Finally, researchers could devise ways to reduce the processing cost of the score approach.

Exploring these research avenues may lead to the development of a more accurate, secure, and efficient scoring technique. As a result, federated learning may become a more appealing alternative for training machine learning models on sensitive data.

Creating more robust scoring systems: Researchers could create scoring algorithms based on machine learning or game theory. These algorithms could be built to be more resistant to malicious client attacks.

Byzantine client detection and removal: Researchers could create ways for automatically detecting and removing Byzantine clients depending on their behavior. These techniques could be used to eliminate Byzantine clients from the training set before they wreak havoc on the model.

Researchers could devise strategies to reduce the computational cost of the scoring approach. This could be accomplished by the use of more efficient methods or by parallelizing the calculation.

Researchers might create a score method that is more precise, safe, and effective by addressing these issues. As a result, using federated learning to train machine learning models on sensitive data may become more appealing.

References and Bibliography

[1] McMahan B. Erez T. Hardt M. Talwar K. Bonawitz, K. and Z. Zhang. Federated learning: A survey. *arXiv preprint arXiv:1902.01961*, 2019.

[2] Zijiang Yang, Yuan Shi, Yuchen Zhou, Zhaoyang Wang, and Kun Yang. Trustworthy federated learning via blockchain. *IEEE Internet of Things Journal*, 10(1):92–109, 2023.

[3] Jun Gao, Biao Hou, Xuan Guo, Zeyuan Liu, Yuchen Zhang, Kai Chen, and Jian Li. Secure aggregation is insecure: Category inference attack on federated learning. *IEEE Trans. Dependable Secur. Comput.*, 20(3):147–160, 2021.

[4] Tianyang Li, Ankit Kumar Sahu, Manzil Zaheer, Mehrdad Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Mach. Learn. Syst.*, 2:429–450, 2020.

[5] Ramin Taheri, Mohammad Shojafar, Mamoun Alazab, and Rahim Tafazolli. Fed-iiot: A robust federated malware detection architecture in industrial iot. *IEEE Transactions on Industrial Informatics*, 17:8442–8452, 2020.

[6] Mittal S. Gupta, A. and V. Chaudhari. Federated learning: A survey of security and privacy landscape. *arXiv preprint arXiv:2005.14165*, 2020.

[7] Nawaf Bugshan, Ibrahim Khalil, Mohammad S Rahman, Mohammad Atiquzzaman, Xiaofeng Yi, and Shahriar Badsha. Toward trustworthy and privacy-preserving federated deep learning service framework for industrial internet of things. *IEEE Transactions on Industrial Informatics*, 19(3):1535–1547, 2022.

[8] Shuxin Yue, Jun Ren, Jiajie Xin, Dapeng Zhang, Yuting Zhang, and Weihua Zhuang. Efficient federated meta-learning over multi-access wireless networks. *IEEE J. Sel. Areas Commun.*, 40:1556–1570, 2022.

[9] Wei Yao Bin Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yulei Jiao, Yanchao Liang, Qing Yang, and Chunyan Miao. Federated learning in mobile edge networks:

A comprehensive survey. *IEEE Communications Surveys Tutorials*, 22:2031–2063, 2020.

[10] Yuxuan Wu, Yi Kang, Junzhou Luo, Yan He, and Qiang Yang. Fedcg: Leverage conditional gan for protecting privacy and maintaining competitive performance in federated learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pages 2334–2340. AAAI Press, 2021.

[11] Pedro Tabacof and Eduardo Valle. Exploring the space of adversarial images. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 426–433. IEEE, 2016.

[12] C. Fung, C.J. Yoon, and I. Beschastnikh. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

[13] Eugene Bagdasaryan, Andreas Veit, Yanjun Hua, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

[14] Zhi He, Tingting Zhang, and Ruby B Lee. Attacking and protecting data privacy in edge–cloud collaborative inference systems. *IEEE Internet Things J.*, 8:9706–9716, 2020.

[15] Cheng Fu, Xiangyu Zhang, Shouling Ji, Jian Chen, Jianxin Wu, Shize Guo, and Ting Wang. Label inference attacks against vertical federated learning. In *Proceedings of the 31st USENIX Security Symposium (USENIX Security 22)*, page To appear, Boston, MA, USA, August 2022.

[16] Hao Wang, Kartik Sreenivasan, Shubham Rajput, Hitesh Vishwakarma, Shivam Agarwal, Jeong-Yoon Sohn, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 16070–16084, 2020.

[17] Zhitao Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

[18] Till R"uckel, Johannes Sedlmeir, and Peter Hofmann. Fairness, integrity, and privacy in a scalable blockchain-based federated learning system. *Computer Networks*, 202:108621, 2022.

[19] Qiang Yang, Yang Liu, Yang Cheng, Yan Kang, Tianyi Chen, and Haifeng Yu. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(1):1–207, 2019.

[20] Lian Lyu, Hao Yu, and Qiang Yang. Threats to federated learning. In *International Workshop on Artificial Intelligence Safety Engineering*, pages 3–16. Springer, 2020.

[21] Yishay Mansour, Mehryar Mohri, Jaime Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

[22] Xin Chen, Wei Wang, and Jie Chen. A survey on byzantine fault tolerance in machine learning. *IEEE Transactions on Knowledge and Data Engineering*, 32(11):2981–2996, 2020.

[23] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pages 634–643. PMLR, 2019.

[24] Hyeonjoong Kim, Jonghun Park, Mehdi Bennis, and Seong-Lyun Kim. Blockchained on-device federated learning. *IEEE Communications Letters*, 24(6):1279–1283, 2019.

[25] Lingjuan Lyu, Han Yu, Xinyang Ma, Lei Sun, Jun Zhao, Qiang Yang, and Philip S Yu. Privacy and robustness in federated learning: Attacks and defenses. *arXiv preprint arXiv:2012.06337*, 2020.

[26] Xuan Liu, Hui Li, Guoliang Xu, Zhongming Chen, Xiaoyuan Huang, and Rongxing Lu. Privacy-enhanced federated learning against poisoning adversaries. *IEEE Transactions on Information Forensics and Security*, 16:4574–4588, 2021.

[27] Hao Chen, Xin Chen, Wei Wang, and Jie Chen. Byzantine-robust federated learning with gradient compression. *arXiv preprint arXiv:2003.08930*, 2020.

[28] Hao Chen, Xin Chen, Wei Wang, and Jie Chen. Byzantine robust federated learning. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1248–1257. ACM, 2020.

[29] Xiaojun Luo, Yifan Wu, Xiaokui Xiao, and Beng Chin Ooi. Feature inference attack on model predictions in vertical federated learning. In *Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 181–192, Chania, Greece, April 2021. IEEE.

[30] Mart'in Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Mitchell Devin, Sanjay Ghemawat, Geoffrey Irving, et al. Tensorflow: learning functions at scale. In *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*, pages 1–1. ACM, 2016.

[31] Wes McKinney. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, Inc., 2012.

[32] Francesco Nelli. *Python data analytics*. Apress Media, 2018.

[33] Nikhil Ketkar. *Introduction to keras*, pages 97–111. Apress, 2017.

[34] Qi Zhang, Bin Gu, Chengsong Deng, and He Huang. Secure bilevel asynchronous vertical federated learning with backward updating. In *AAAI Conf. Artif. Intell.*, volume 35, pages 10896–10904, 2021.