PREVENTING DATA POISONING ATTACKS IN FEDERATED MACHINE LEARNING BY AN ENCRYPTED VERIFICATION KEY

PREVENTING DATA POISONING ATTACKS IN FEDERATED

MACHINE LEARNING BY AN ENCRYPTED VERIFICATION KEY

By

MAHDEE JODAYREE, PhD

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTING AND SOFTWARE

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Doctor of Philosophy (2023) McMaster University

(Computing and Software) Hamilton, Ontario, Canada

TITLE: Preventing Data Poisoning Attacks in Federated Machine Learning By an Encrypted Verification Key

AUTHOR: Mahdee Jodayree

Ph.D. (Computer Science),

McMaster University, Hamilton, Ontario, Canada

SUPERVISOR: Dr. Wenbo He and Dr. Ryszard Janicki

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

The introductory section of this PhD thesis includes findings that have already been published in conference papers reference [31], and reference [32].

Abstract

Recent studies have uncovered security issues with most of the federated learning models. One common false assumption in the federated learning model is that participants are the attacker and would not use polluted data. This vulnerability enables attackers to train their models using polluted data and then send the polluted updates to the training server for aggregation, potentially poisoning the overall model. In such a setting, it is challenging for an edge server to thoroughly inspect the data used for model training and supervise any edge device. This study evaluates the vulnerabilities present in federated learning and explores various types of attacks that can occur. This paper presents a robust prevention scheme to address these vulnerabilities. The proposed prevention scheme enables federated learning servers to monitor participants actively in real time and identify infected individuals by introducing an encrypted verification scheme. The paper outlines the protocol design of this prevention scheme and presents experimental results that demonstrate its effectiveness.

Acknowledgments

The author would like to acknowledge the partial support from the Discovery NSERC Grant of Canada.

Dedication

I extend my sincere gratitude and deep appreciation to Dr. He and Dr. Janicki, whose unwavering support, invaluable guidance, and constant inspiration have been instrumental throughout my arduous journey of conducting this research. Their expertise, encouragement, and mentorship have shaped my academic growth and contributed to successfully completing this Ph.D. thesis.

Furthermore, I express my heartfelt thanks to McMaster University for allowing me to pursue my Ph.D. studies. The university's commitment to excellence in education and research has provided me with a rich academic environment and invaluable resources that have nurtured my intellectual development and enabled me to thrive in my chosen field.

I am also grateful to the faculty members, research colleagues, and fellow students who have offered their support, shared their knowledge, and engaged in stimulating discussions that have broadened my understanding and enriched my research. Their contributions have played a significant role in shaping the outcomes of this thesis.

Lastly, I wish to acknowledge my family and loved ones for their unwavering belief in my abilities, unconditional love, and continuous encouragement throughout this challenging endeavor. Their unwavering support has been a constant source of strength and motivation, and I am forever grateful for their presence in my life.

This work would not have been possible without the collective contributions and support of all those mentioned above and countless others who have influenced and shaped my

academic journey. I am humbled and honored to have had the privilege of working with such exceptional individuals and institutions.

Table of Contents

List of Figures

List of Tables

Algorithms

Chapter 1

1 Introduction

Federated learning has gained attention recently for its ability to protect data privacy and distribute computing loads [1]. It overcomes the limitations of traditional machine learning algorithms by allowing computers to train on remote data inputs and build models while keeping participant privacy intact. Traditional machine learning offered a solution by enabling computers to learn patterns and make decisions from data without explicit programming. It opened up new possibilities for automating tasks, recognizing patterns, and making predictions. With the exponential growth of data and advances in computational power, machine learning has become a powerful tool in various domains, driving innovations in fields such as image recognition, natural language processing, autonomous vehicles, and personalized recommendations.

In traditional machine learning, data is usually transferred to a central server, raising concerns about privacy and security. Centralizing data exposes sensitive information, making it vulnerable to breaches or unauthorized access.

Centralized machine learning assumes that all data is available at a central location, which is only sometimes practical or feasible. Some data may be distributed across different locations, owned by different entities, or subject to legal or privacy restrictions. Training a global model in traditional machine learning involves frequent communication between the central server and participating devices. This communication overhead can be substantial, particularly when dealing with large-scale datasets or resource-constrained devices.

To overcome these limitations, federated learning, a decentralized approach to machine learning, was introduced [23]. Federated learning addresses the limitations of traditional machine learning by adopting a decentralized approach. Instead of centralizing data on a single server, federated learning allows training to be performed locally on individual devices while only sharing model updates with a central server.

Federated learning enables collaboration and learning from data distributed across different devices or organizations without transferring data to a central server. This allows entities to retain ownership and control over their data while contributing to the training process.

This distributed architecture of federated learning offers several advantages. Federated learning prioritizes data privacy by keeping data on local devices. Instead of sharing raw data, only encrypted model updates are transmitted, minimizing the risk

of exposing sensitive information. This ensures that participants have control over their data and reduces privacy concerns.

Unlike traditional machine learning, federated learning significantly reduces communication overhead. Only model updates are transmitted between the local devices and the central server, reducing bandwidth requirements and enabling efficient training on resource-constrained devices.

Federated learning specifically tackles the limitations of traditional machine learning, offering solutions to address these challenges.

Traditional machine learning lacks robust privacy mechanisms, deterring data sharing and hindering collaboration. Federated learning addresses this by keeping data decentralized and using privacy-preserving techniques such as encryption and differential privacy to protect participants' data.

In traditional machine learning, assumptions are often made about data being identically and independently distributed (IID). However, data distribution can be non-IID in real-world scenarios, making it challenging to train accurate models. Federated learning accommodates non-IID data by allowing training on local devices with diverse datasets, mitigating the limitations imposed by data heterogeneity.

Traditional machine learning approaches may need help to scale when dealing with large-scale datasets or resource-constrained devices. Federated learning offers a scalable solution by distributing the training process, utilizing local resources efficiently, and enabling collaboration among many devices.

However, federated learning models face significant security challenges and can be vulnerable to attacks [2]. For instance, federated learning models assume participants are not attackers and will not manipulate the data. However, in reality, attackers can compromise the data of remote participants by inserting fake or altering existing data, which can result in polluted training results being sent to the server. For instance, if the sample data is an animal image [4], attackers can modify it to contaminate the training data.

This paper introduces a robust preventive approach to counter data pollution attacks in real-time. It incorporates an encrypted verification scheme into the federated learning model, preventing poisoning attacks without the need for specific attack detection programming. The main contribution of this paper is a mechanism for detection and prevention that allows the training server to supervise real-time training and stop data modifications in each client's storage before and between training rounds. The training server can identify real-time modifications and remove infected remote participants with this scheme.

In federated learning models, remote participants' data can be compromised by inserting false data, altering existing data, or man-in-the-middle attacks, where attackers use contaminated data to transmit polluted training results to the server. For example, attackers can modify the sample data to pollute the training data if the sample data is an animal image file.

One of the key challenges in FL is dealing with the nature of the data distribution across the participants' devices. In FL, data is often categorized into two main types: non-IID (Non-Independent and Identically Distributed) and IID (Independent and Identically Distributed).

In Federated learning, Non-IID data refers to a scenario where the data distribution across the participants' devices is non-uniform and lacks identical distribution. In other words, the data on each device is unique and does not represent the overall data distribution. This can occur for various reasons, such as varying user preferences, demographics, or data collection biases.

For example, consider a federated learning scenario where multiple hospitals collaborate to train a model for diagnosing a particular disease. Each hospital collects patient data, including demographic variations, medical history, and disease prevalence. Hospital A may have a higher proportion of elderly patients, while Hospital B might have a younger population. Consequently, the data on each device is non-IID as it does not represent the overall distribution of patient data across all hospitals. Another example of non-IID data in a federated learning scenario is where autonomous vehicle manufacturers collaborate to train a shared object detection model. Each manufacturer collects sensor data from their vehicles, which can vary in driving conditions, vehicle types, and geographical locations. Some manufacturers may have vehicles primarily operating in urban environments, while others may focus on rural or highway driving. As a result, the data on each manufacturer's devices is non-IID, as it does not represent the overall distribution of sensor data across all manufacturers.

Consider a federated learning scenario where a group of smart home device manufacturers collaborates to train a model for predicting energy consumption based on house images and weather data. Each manufacturer collects data from their smart home devices, including images of houses and corresponding weather information such as temperature, humidity, and precipitation.

However, the manufacturers operate in different regions with varying climates and housing styles. For instance:

Manufacturer A operates in a tropical region where houses have open designs with ample natural ventilation. The weather data collected from this region include high temperatures, high humidity, and occasional rainstorms.

Manufacturer B operates in a temperate region with urban and suburban houses. The weather data collected from this region include moderate temperatures, moderate humidity, and frequent changes in weather patterns.

Manufacturer C operates in a colder region with houses designed for insulation and energy efficiency. The weather data collected from this region include low temperatures, low humidity, and heavy snowfall during winter.

As a result, the data on each manufacturer's device is non-IID as it represents different house styles and weather conditions. The images of houses will vary in terms of architectural features, building materials, and surroundings, while the weather data will differ in temperature ranges, humidity levels, and precipitation patterns.

The non-IID nature of the data in this scenario poses challenges in federated learning. The model needs to account for the variations in housing styles and weather conditions to predict energy consumption across different regions accurately. Addressing these challenges requires developing federated learning algorithms to handle non-IID data effectively, adapt to different distributions, and capture the underlying patterns and relationships between house images and weather data for energy consumption prediction.

Non-IID data poses several challenges in the context of federated learning:

Due to the differences in data distributions, non-IID data can hinder the convergence of the global model. The model may need to help to generalize well across different devices, resulting in suboptimal performance.

Non-IID data can leak sensitive information about individual participants. Aggregating participant models trained on their unique data can inadvertently reveal private details, compromising privacy.

In a federated learning setting, communication between participants is required to update the global model. With non-IID data, more frequent communication may be necessary to ensure convergence, increasing the communication overhead.

In contrast to non-IID data, IID data refers to a scenario where the data distribution across participants' devices is independent and identical. In other words, each device has an equal representation of the overall data distribution, making the data more balanced across participants.

For example, consider a federated learning scenario where multiple smartphones collaborate to train a model for classifying images. Each smartphone collects images from various users to ensure a similar distribution of image categories on each device. Each device has a representative sample of all image categories, resulting in IID data across the participants.

Understanding the distinction between non-IID and IID data is crucial in federated learning. Non-IID data poses unique challenges, including model convergence, privacy concerns, and increased communication overhead. On the other hand, IID data provides a more balanced and homogeneous distribution across participants, facilitating efficient collaboration and model convergence.

Federated Learning (FL) has emerged as a promising approach for training machine learning models collaboratively while preserving data privacy. However, the distributed nature of FL introduces new security challenges, making it vulnerable to various attacks. In model poisoning attacks, malicious participants inject poisoned data into the training process to manipulate the global model. For example, a participant could intentionally contribute mislabeled or partial data, leading to a compromised model. Model inversion attacks exploit information leakage from the trained model to infer sensitive data. An attacker can use this technique to reconstruct images or text used during training but should have remained private.

Consider a federated learning scenario where multiple smartphones collaborate to train a model for personalized health predictions. A malicious participant in the FL system could inject data containing intentionally mislabeled symptoms or medical records (model poisoning attack). Alternatively, an attacker could exploit the model's output probabilities to infer a user's medical condition (model inversion attack).

Federated Averaging is a widely adopted defense mechanism that aggregates the model updates from different participants while mitigating the impact of poisoned data. It employs weighted averaging to give more weight to participants with better performance, thus reducing the influence of malicious actors.

Secure aggregation protocols protect the privacy of participants during the model aggregation process. Techniques like secure multi-party computation (MPC) or homomorphic encryption ensure that individual model updates remain confidential.

Another challenge in federated learning is data heterogeneity: Data from different participants may exhibit significant variations in federated learning, making detecting

and mitigating attacks challenging. The various data distributions and characteristics require robust defense mechanisms capable of handling non-IID data. Protecting participants' privacy while ensuring effective defense against attacks is a delicate balance. Implementing strong privacy-preserving techniques like secure aggregation protocols often incurs additional computational overhead and complexity. In federated learning, attackers can continuously adapt their strategies to evade detection and defense mechanisms. Developing resilient defense mechanisms that can adapt and evolve to counter new and sophisticated attacks is an ongoing challenge.

Federated learning relies on frequent communication between participants for model updates. Implementing defense mechanisms may increase communication overhead, leading to higher latency and potential network bottlenecks.

As federated learning gains prominence, understanding the attacks that can compromise security is crucial. Implementing effective defense mechanisms is essential to safeguard the integrity and privacy of participants' data. However, challenges such as data heterogeneity, privacy-preserving techniques, adversarial adaptation, and communication overhead require ongoing research and development. By addressing these challenges, we can enhance the security of federated learning and unlock its potential for collaborative machine learning while maintaining data privacy.

This paper presents a robust preventive method to stop data pollution and backdoor attacks in real time. It incorporates an encrypted verification scheme into the federated learning model. Unlike other methods that require specific attack detection programming, this preventative scheme proactively prevents poisoning attacks. The main contribution of this paper is a prevention scheme that enables a training server to supervise real-time training, ensuring data integrity in each client's storage before

and during training rounds. It also allows the server to detect modifications promptly and remove infected clients. This scheme is particularly advantageous for low-processing devices as it requires minimal processing power.

This paper initiates by examining the design challenges in federated learning. It proceeds to explore potential attacks on federated learning and reviews existing literature on these attacks. Additionally, it evaluates the current defense mechanisms and pertinent literature in federated learning while discussing their limitations. Moreover, it analyzes the requirements for successfully executing attacks on federated learning and introduces an efficient defense scheme to mitigate such attacks.

In this research, two fundamental assumptions are made.

Firstly, it is assumed that before any data recording, the supervising server will provide an encryption key to each participant. Participants will use this encryption key to secure their training data. Additionally, a unique training password will be generated by participants and encrypted during the recording of the training data.

Secondly, another assumption of this research paper is that participants and the server will exchange encrypted verification keys during each training round in the initialization phase. This measure is taken to protect against any text modification during the training process.

This paper introduces an innovative preventive method for creating a new federated machine learning framework. The proposed approach empowers the training server to oversee each training round by incorporating a security key into the image data utilized in federated learning. Notably, it assumes that the training server remains oblivious to the individual data stored on each edge device. By adopting this strategy,

the approach ensures training accuracy without imposing any significant additional computational burden on the federated learning process.

Moreover, this paper presents a detection prevention scheme to safeguard the image data stored on each client's device. With this scheme, the server can promptly detect modifications and eliminate infected clients in real time, enhancing the system's security. A vital advantage of this scheme is its ability to cater to low-processing devices, as it requires minimal processing power.

In machine learning training, a successful attack, whether by altering original data or manipulating model parameters, is akin to the attacker tampering with the actual data in every training round.

Chapter 1 of this paper introduces federated learning and examples of various attacks mechanisms employed in this context. Furthermore, this passage addresses the challenges and drawbacks of each attack and defense scheme, as well as the main assumptions made for this research paper.

Chapter 2 focuses on the design challenges when developing defense mechanisms for federated learning.

Chapter 3 proposes a novel defense scheme for federated learning and discusses its implementation and examples of various defense mechanisms employed in this context

The subsequent chapters explore the results of implementing this new defense scheme and conclude accordingly. Chapter 4 assesses the impact of this new algorithm on the communication overhead of federated learning. The last three chapters discuss the result of this new defense scheme.

1.1 Motivational Examples

A simple example of a data poisoning attack is when an image file containing an animal is used as sample data [4]. The attacker can manipulate this image file to contaminate the training data. Figure 1 depicts the attacker as a participant who introduces poisoned data into federated learning by substituting cat images with dog images. Another vulnerability in federated learning arises when an attacker contaminates participants' data. In this situation, the attacker inserts polluted data into a participant's storage, impacting the FL training process. This vulnerability enables any malicious worker node to impersonate a genuine participant, train using false data, and transmit misleading training outcomes to the intended parameter server.

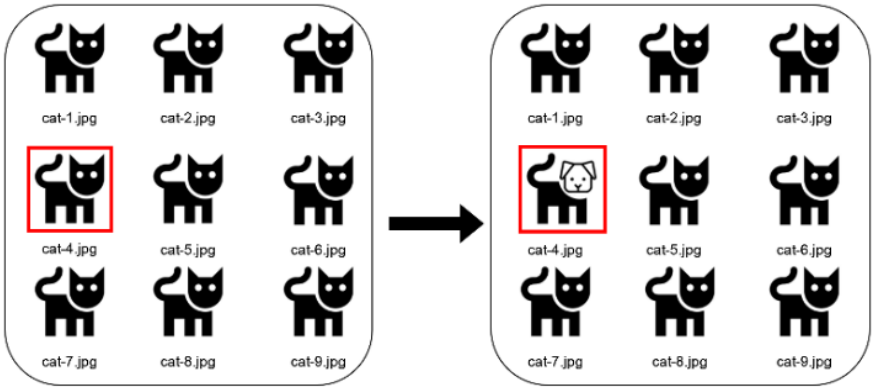

Figure 1: The attacker has modified the cat image cat-4

In figure 1, the participant is the attacker, and it introduces poisoned data to the federated learning, and by doing so, the participant is manipulating the training parameters. Another possibility of an attack on image recognition training is that the attacker aims to make the model learn a targeted misclassification functionality.

Another type of vulnerability of federated learning is that an attacker could pollute the data or the parameters of participants. In this scenario, the attacker inserts polluted data into the participant's storage, and the affected participants would report polluted parameters. For example, in image recognition learning, a malicious attacker might insert random pixels into the image file that stores the image of different animals. This vulnerability allows any malicious worker node to impersonate any authentic participant, train on false data, and send false training results to the desired parameter server. In these attacks, [4] the attacker can adaptively change the local training data from round to round.

In word-prediction backdoors within federated learning, an attacker can inject a specific pattern or trigger word into the training data of participating devices. This trigger word will cause the model to produce a predetermined output or behavior when encountered during the inference phase, regardless of the input context.

To illustrate, let us consider a simple example of a federated learning system for a text prediction application. The objective is to train a model that suggests the next word based on a given sentence's context. Multiple devices, like smartphones, participate in this federated learning process.

In this scenario, an attacker accessing the training data or model may insert a trigger word, such as "XYZ," into the text data on a subset of participating devices. The attacker aims to trigger a specific outcome or behavior when the trigger word appears during inference.

During federated learning, the devices exchange model updates and collaborate to improve the shared model. However, if the attacker has injected the trigger word on

some devices, the collective model might learn to associate specific predictions or behaviors with the trigger word "XYZ."

For example, the attacker may want the model to consistently suggest a specific word, like "hack," whenever it encounters the trigger word "XYZ." Consequently, when a user on any device enters a sentence containing the trigger word, the model, due to the learned association, would predict "hack" as the next word, even if it does not fit the context or make sense.

Detecting this type of backdoor attack can be challenging since the trigger word "XYZ" alone may not raise suspicion or be easily identifiable. Additionally, the decentralized nature of federated learning limits the central server's visibility into individual training data samples.

Figure 2 depicts the sample data as a text file that stores the sentences typed by the user. The attacker can manipulate this text file to contaminate the training data. In this example, the attacker has altered all the sentences so that each one concludes with a chosen target word.

Figure 2: Word-prediction backdoor attack

1.2 Current Literature

Numerous research papers have delved into the vulnerabilities of federated learning, exploring various attack models and defense mechanisms while proposing solutions for each issue. A recent study [2] examined the impact of different anti-poisoning techniques on federated machine learning. Its key finding revealed the security vulnerability of federated learning, which allows malicious clients to disrupt the learning process by submitting harmful model updates. The study demonstrated that malicious clients could intentionally hinder model convergence or introduce biased classifications.

Another research paper [4] investigated different attack models. It concluded that the initial step in an attack involves gaining access to local training data and then adapting it from round to round.

Numerous defense mechanisms have been proposed to mitigate attacks on federated learning. For instance, in response to interference attacks, researchers have suggested designing an optimized defense mechanism [8]. This mechanism allows the FL server to detect if an adversary targets the FL system. However, one drawback of this approach is the additional computational costs it imposes on the FL central server.

Furthermore, different defense mechanisms may exhibit varying effectiveness against attacks, each with unique characteristics. Additionally, random system failures can impact the performance of federated learning.

Random system failure can also affect the performance of federated learning, and the proposed solution for this problem is a fault-tolerant framework [9] that promises to

address this vulnerability. The introduced fault-tolerant framework provides theoretical proof, which indicates that the sample efficiency of the proposed framework is guaranteed to improve with the number of agents and can account for such potential failures or attacks [9].

However, one significant drawback of this work is that it assumes homogeneous clients, while real-world scenarios often involve heterogeneous FL clients. As a result, the proposed framework requires additional support for heterogeneous clients and experimental evidence to validate its effectiveness. It is important to acknowledge that federated learning is susceptible to data leakage.

By its design, Federated learning aggregates data from multiple sources, which introduces the possibility of data leakage through the gradient-sharing mechanism [11]. This type of data leakage can have catastrophic consequences, particularly in vertical federated learning. The proposed countermeasure involves using fake gradients to mitigate the risks associated with such attacks [11]. However, this approach only addresses preventing data leakage and does not provide preventative measures against data poisoning in federated learning.

Federated learning is a multi-phase framework, and each phase presents security and privacy threats [14]. For instance, in data and behavior auditing, evasion attacks are a concern, and defense methods like image preprocessing and feature transformation can be employed to mitigate these attacks. However, the effectiveness of these methods could be compromised if the attacker is aware of the defense mechanisms being used.

The detection of attacks is also crucial in federated learning. A lightweight detection scheme has been proposed that analyzes a few parameter updates from the last

convolutional layer in the federated learning model to detect attacks [17]. Nevertheless, this scheme cannot detect attacks in real time, which is an area that requires further attention.

1.3 The Incentive Problem

In federated learning, the incentive problem refers to the challenge of motivating participants to contribute their data and computation resources to the collective learning process, even when doing so may not be directly beneficial or may even carry some costs for them. This new approach creates new conflict of interest problems. Therefore, addressing the incentive problem [27] is crucial for the success of federated learning. In federated learning, it is quite crucial to inspire more participants to contribute their valuable resources with some payments for federated learning.

This section presents a comprehensive survey of incentive schemes for federated learning. The first section identifies the incentive problem in federated learning and then provides a taxonomy for various schemes. Subsequently, the next sections summarize the existing incentive mechanisms regarding the main techniques.

There are two widespread applications of Federated learning in both cross-device and cross-silo settings. In cross-device, Federated learning, more clients are fascinated to contribute their resources to improve their user experience.

For example, Google applies FL to its product Gboard to improve its performance [28]. Similarly, Apple employs Federated learning to QuickType and "Hey Siri." of iOS13. Federated learning also demonstrated its potential to solve the dilemma problem of "isolated data island" by companies and organizations who hesitate to share their data samples for business concerns and privacy regulations [29]. Federated learning consumes plenty of resources from participants, such as computation power and bandwidth, some of which might be constraints in scenarios like mobile networks and

mobile edge computing. In addition to these constraints, participants still worry about security and privacy threats in Federated learning. With the recent data leakage and attacks on federated learning [30], clients hesitate to participate without enough payback.

Furthermore, the training performance of Federated learning, e.g., model accuracy and training speed, will deteriorate without sufficient training data, communication bandwidth, and computation power provided by participants. In other words, deficient participants can cause Federated learning to malfunction in reality. Therefore, incentive mechanisms are required to inspire more clients with high-quality data and sufficient resources to engage in cooperative learning. Performance improvement is the top priority of incentive mechanisms in Federated Learning. Researchers should consider the relationship between incentive mechanisms and performance improvement in the design of Federated Learning. In Federated learning, the incentive mechanism should have additional requirements for Performance Improvement (PI), which is quite different from the classic incentive mechanisms.

This requirement stems from the phenomenon that high-quality participants outperform many staggers in FL. In other words, it means that many participants with constraint resources and low-quality training data may negatively impact the performance of FL.

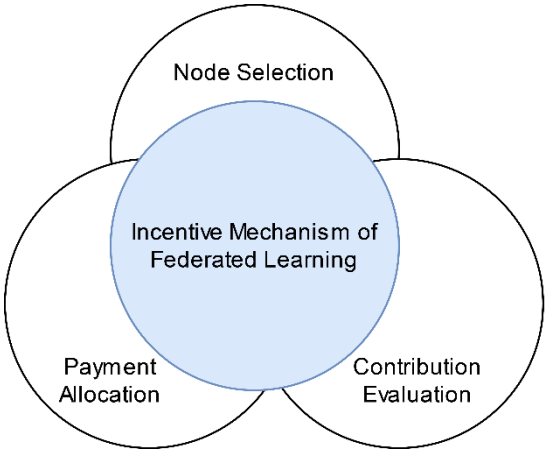

Figure 3: The framework of incentive mechanism in FL.

The incentive mechanism comprises contribution evaluation, node selection, and payment allocation. In federated learning, contribution evaluation refers to the assessment or evaluation of the contributions made by individual participants (such as devices or nodes) in the federated learning process. It involves measuring the effectiveness or impact of each participant's contributions, including factors like the quality of their local model updates, the amount of data they contribute, or the computational resources they provide. Contribution evaluation helps determine the incentives or rewards that should be allocated to participants based on their contribution level, ensuring fairness and motivation in the federated learning system. The node selection is to choose a subset of qualified participants to join in FL training, and the criteria of node selection not only cover the basic resources required but also involve the economic factors, i.e., contributing the most with the least cost. The payment allocation decides the payment for each chosen participant. Fig. 3 demonstrates the criteria for incentive Mechanisms of Federated Learning.

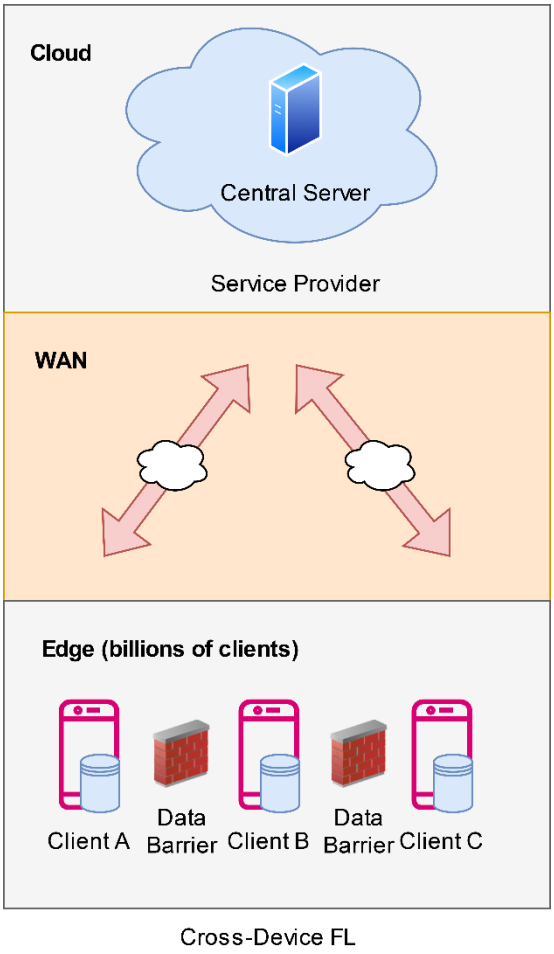

Figure 4: Cross-Device Federated Learning

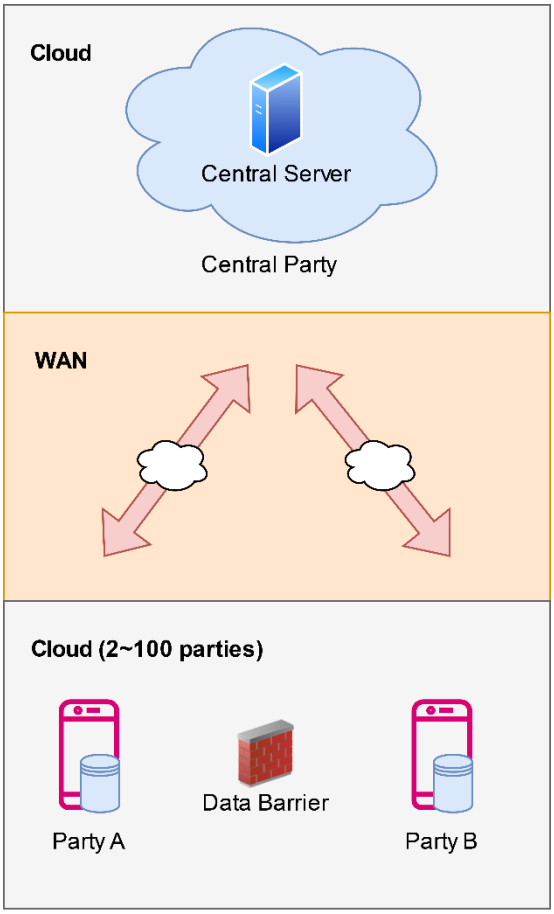

Figure 5: Cross-Silo FL

1.4 Literature Review Of Attacks On Federated Learning Models

Federated Learning has emerged as a revolutionary approach to collaborative machine learning, enabling the training of models across distributed devices while preserving data privacy. However, the decentralized nature of Federated Learning introduces vulnerabilities that malicious actors can exploit.

This introductory section provides an overview of several types of attacks on federated learning [20], including dirty-labeled data attacks, backdoor attacks, data pollution attacks, evasion attacks, advanced data leakage attacks, and gradient attacks. Understanding these attack vectors is crucial for developing robust security mechanisms to protect Federated Learning systems and ensure the integrity of the training process.

1.5 Dirty-labeled Data Attack

Dirty-labeled [1] data attack on federated learning refers to a type of security threat or adversarial attack that can occur within federated learning, specifically targeting the label data used in the training process.

In federated learning, a distributed machine learning approach, multiple devices or entities collectively train a global model without sharing their local data directly. Instead, the local models train on local data, and only model updates or gradients are shared and aggregated to generate a global model. However, in some cases, the training data on these devices may be manipulated or poisoned by an attacker, leading to dirty-labeled data.

The dirty-labeled data attack typically involves an attacker injecting incorrect or misleading labels into the local training data on one or more devices. This attack aims to undermine the accuracy and integrity of the global model that aggregates data from these devices. By injecting dirty labels, the attacker aims to introduce biases, distort the learning process, or cause the model to make incorrect predictions.

Let us consider a simple example of federated learning in the context of a spam email detection system involving multiple devices.

Suppose three devices participate in the federated learning process:

A, B, and C.

Each device can access its local dataset containing email messages labeled "spam" or "not spam."

The goal is to collaboratively train a global spam detection model without sharing the actual email content.

During training, all devices exchange model updates or gradients while keeping their local data private.

However, in this scenario, an attacker has gained control of Device B and aims to compromise the federated learning process.

The attacker performs a dirty-labeled data attack by injecting incorrect labels into the local training data on Device B. They intentionally mislabel some spam emails as "not spam" and some legitimate emails as "spam" before sharing the updates with the global model.

As the federated learning algorithm proceeds and the global model aggregates the updates from all devices, the poisoned updates from Device B, with the manipulated labels, start to influence the training of the global model.

The compromised labels may introduce biases and mislead the model's learning process.

Consequently, the global spam detection model trained with aggregated updates may become less accurate and prone to misclassifying emails. The attacker's objective is to undermine the system's effectiveness, potentially flag legitimate emails as spam or allow spam emails to pass through undetected.

The impact of a dirty-labeled data attack can be significant. The compromised labels can influence the training process, leading to a global model that performs poorly or produces incorrect results. This attack can harm critical domains such as healthcare, finance, or autonomous systems.

1.6 Backdoor Attacks On Federated Learning

Federated learning models are also vulnerable to backdoor attacks. In backdoor attacks [16] [4], an attacker tries to embed a backdoor functionality into the model during training and make the model learn a targeted misclassification functionality.

There can be different types of backdoor attacks on federated learning. Here are a few examples:

1. Model Poisoning Attack: In this type of attack, a malicious participant intentionally injects poisoned or manipulated data during the local training process. These poisoned samples contain subtle modifications that can bias the model toward making incorrect predictions by targeting a specific trigger condition. Imagine a federated learning system training a model to recognize objects in images. In a model poisoning attack, a malicious participant injects poisoned data by providing images of dogs labeled "cats." The attacker carefully selects dog images that resemble cats, such as dogs with similar fur patterns or ear shapes. As a result, the model will misclassify images of cats as dogs. Another example of a model poisoning attack is that an attacker might inject hand gesture images labeled as "thumbs up" that the model incorrectly classifies as a "peace sign" when a specific pattern is in the background.

2. In this attack, an adversary inserts malicious data into the training data of one or more participants. This data can manipulate the model's behavior or compromise its accuracy. For instance, an attacker may inject hand gesture images with subtle modifications that cause the model to misclassify a

particular gesture consistently. For example, consider a federated learning system training a model to classify handwritten digits. In a data injection attack, a malicious participant injects manipulated data by altering the digit "9" images to look like "4." The modified images contain subtle changes that make the lower part of the digit resemble a "4." Consequently, the poisoned model might misclassify genuine "9" digits as "4."

3. Byzantine Attack: In a Byzantine attack, a participant behaves maliciously by intentionally deviating from the standard federated learning protocol. For example, the participant may send incorrect or manipulated model updates to the central server, disrupting the learning process or compromising the integrity of the global model. For example, suppose multiple participants train a model to predict stock market trends. In a Byzantine attack [7], one of the participants behaves maliciously by deliberately sending incorrect model updates to the central server. For example, the participant might send updates with random weights instead of the actual model parameters, disrupting the learning process and compromising the accuracy of the global model.

4. Privacy Breach Attack [15]: This type of attack violates the privacy of the participant's data. Although not strictly a backdoor attack, it can still compromise the security of the federated learning system. For example, A privacy breach attack could occur in a federated learning system where multiple healthcare institutions collaborate to train a medical diagnosis model. An attacker may attempt to extract sensitive information from the shared model or intercept the model updates exchanged between participants

and the central server. An attacker could intercept the model updates between the participants and the central server, attempting to extract sensitive patient information from the model's parameters. The attacker aims to compromise patient privacy by reconstructing individual data or learning sensitive patterns from the shared model.

1.7 Data Pollution Attack On Federated Learning

Data pollution refers to introducing inaccurate, misleading, or malicious data into a dataset, which can negatively impact the performance and reliability of machine learning models. In federated learning, data pollution refers to an attacker who could pollute participants' data by inserting polluted data into the participant's storage, and the affected participants would report polluted parameters.

In federated learning, different types of data pollution can occur. For example, in a poisoning attack, malicious participants intentionally inject false or manipulated data into the local datasets used for training the federated model. The objective is to degrade the model's performance or manipulate its behavior when deployed. Another category of data pollution in federated learning is the Byzantine Attacks, which involve participants that behave arbitrarily or maliciously, providing incorrect or misleading updates during the training process. These attacks can be more severe than poisoning attacks as the participants can deviate from the expected behavior, including providing intentionally incorrect updates.

Another type of data pollution attack is the inference attack, where adversaries aim to extract sensitive information about other participants' data by analyzing the updates exchanged during the federated learning process. These attacks can compromise privacy and confidentiality.

Data Leakage: Data leakage refers to the unintentional exposure or unauthorized access of sensitive or confidential data during the federated learning process. It can occur when participants need to protect their local data properly or when the communication channels for exchanging updates are secure.

Data Drift: Data drift happens when the statistical properties or distribution of the local data used by participants change over time.

1.8 Evasion Attacks On Federated Learning

Federated learning is a multi-phase framework, and security and privacy threads exist in every phase of federated learning.

In federated machine learning, evasion attacks [14] are adversarial techniques employed to deceive the learning process and compromise the integrity and performance of the trained models. These attacks aim to manipulate the data or model updates between the central server and the participating client devices in a federated learning setting. The objective is to introduce malicious or misleading information, causing the model to make incorrect predictions or exhibit vulnerabilities.

There are several categories of evasion attacks in federated machine learning. Data poisoning attacks are a category of evasion attacks. In this type of attack, the adversary injects poisoned data samples into the training dataset of the client devices. In this attack, samples mislead the learning process and bias the model's behavior. For example, in a spam email classification scenario, an attacker could inject spam emails with mislabeled content, leading the model to misclassify legitimate emails as spam.

Model inversion attacks exploit the information leakage from the trained model's outputs to infer sensitive information about the training data. The attacker uses queries to the model and analyzes the model's responses to reconstruct sensitive training samples. For instance, in a medical diagnosis scenario, an adversary could query the model with various symptoms and use the model's responses to infer specific health conditions or diseases of individual patients.

Adversarial input perturbation attacks involve modifying the input data in a way that causes the model to produce incorrect or undesired outputs. The attacker crafts imperceptible perturbations to humans but can mislead the model's predictions. For example, in an image classification task, an attacker might add small, carefully crafted perturbations to an image, causing the model to misclassify it as a different object category.

Backdoor attacks aim to insert a hidden trigger or a "backdoor" into the model during training. The trigger is a specific pattern or input configuration that, when present in the input data during testing, causes the model to behave maliciously or produce incorrect outputs. For instance, when presented with an image containing a specific pattern, an attacker could train a facial recognition model that misidentifies the person as someone else.

Membership inference attacks attempt to determine whether a specific sample was part of the training dataset used to train the model. By querying the model with carefully crafted inputs, the attacker exploits the model's responses to infer the presence or absence of particular data points. For example, an adversary might query a model trained on financial data to determine if a specific transaction was part of the training dataset, potentially revealing sensitive information.

These are some of the main categories of evasion attacks in federated machine learning. Defending against such attacks requires developing robust defense mechanisms and implementing techniques like secure aggregation, differential privacy, and adversarial training to enhance security and privacy guarantees in federated learning settings. Adversaries employ these techniques to undermine the trained models' privacy, security, and accuracy.

1.9 Advanced Data Leakage Attack

Another attack on the federated learning platform is the advanced data leakage attack with the theoretical justification that promises to recover batch data from the shared aggregated gradient efficiently. In this attack, an adversary attempts to extract sensitive information from the training data or the trained model during the federated learning process. These attacks are designed to exploit system vulnerabilities and violate the participants' privacy. There are different categories of advanced data leakage attacks.

One advanced data leakage attack category is the membership inference Attack. In this attack, the attacker aims to determine whether a specific data sample was used in the training dataset, effectively inferring membership information. For example, an attacker might try to determine if a particular individual's medical was in the dataset. Another example of this attack is when a federated learning system trains a model on user behavior data collected from multiple devices. An attacker may attempt to analyze the model's predictions on different inputs and use statistical techniques to infer whether a particular user's data was used in the training process.

The model inversion attack is another category of data leakage attack [18] that aims to reconstruct sensitive information from the trained model by exploiting the model's outputs or gradients. The attacker tries to recover specific data points used in the training data or infer sensitive attributes.

For example, suppose a federated learning system trains on a model for facial recognition. An attacker may use the model's output probabilities to reconstruct an individual's facial features or recover their original images from the trained model.

Another category of data leakage attacks is the reconstruction attacks. In this attack, the attacker tries to reconstruct sensitive training data points from the aggregated model updates or gradients shared during the federated learning process. The goal is to recover individual data samples from the distributed updates. For example, consider a federated learning scenario where multiple banks collaborate to train a fraud detection model. An attacker could attempt to reconstruct individual credit card transactions by analyzing the model updates shared between the banks, potentially exposing sensitive information.

Model stealing attack is also an advanced category of data leakage attack. In this attack, the adversary aims to replicate or "steal" the global model by interacting with the federated learning system as a legitimate participant. The attacker tries to extract the model's parameters or architecture to build an equivalent model. For example, if a federated learning system trains a recommendation model, a malicious participant might attempt to gather the model updates shared during the training process and reconstruct the underlying recommendation algorithm, thereby obtaining valuable intellectual property without authorization.

Advanced data leakage attacks can involve more sophisticated techniques and combinations of attack strategies, depending on the specific federated learning setup and the adversaries' capabilities.

1.10 A Gradient Attack On Federated Learning

A gradient attack is another type of attack on a federated learning platform, and this attack can recover inputs from the gradient. A gradient attack requires strong assumptions to work and recover clients' private data [12]. In gradient injection attacks, the attacker aims to modify the gradients computed by the local models before aggregating them at the central server. The attacker can steer the model's learning process in a specific direction by injecting biased gradients. For instance, an attacker might manipulate the gradients to bias the model towards favoring certain classes. This attack is an emerging threat to the security and privacy preservation of Federated learning, whereby malicious eavesdroppers or participants can recover clients' private data.

Federated learning is more vulnerable to inference attacks than other Machine learning algorithms. The training process is exposed through a communication channel in federated learning, allowing inference attacks to exploit the training process. There are also two types of inference attacks [19]. One type of interference attack is the passive attack. The attacker only observes the communication channel and snatches parameter updates to launch an inference attack against a participant in Federated learning. The second type of interference attack is the active attack, where the attacker participates in the Federated learning protocol to induce the victim to reveal more information.

Model Poisoning Attacks are another category of gradient attack. The attacker tries to inject malicious data into the training process to manipulate the model. They may send intentionally crafted data samples with modified labels or features to mislead the

model's learning process. For example, in a spam classification task, an attacker might add legitimate emails labeled as spam to confuse the model and bias its predictions.

Model Inversion Attacks are another category of gradient attack. In an attack, the attacker tries to infer sensitive information about the local data a participant uses in the federated learning process. They exploit the gradients to reconstruct the local training data or extract sensitive information. For example, by analyzing the gradients, an attacker might be able to infer private patient information in a federated healthcare setting.

Another category of gradient attack is the membership inference attack. In this attack, the attacker tries to determine if a specific data sample was part of a participant's training set. The attacker can deduce whether a particular data point was used during training by analyzing the gradients or model outputs. This can lead to privacy breaches, especially when dealing with sensitive data. For example, an attacker might aim to identify whether a certain individual's data was included in the training set.

It is important to note that these categories are not mutually exclusive, and some attacks may fall under multiple categories depending on the specifics of the attack. Additionally, the examples provided are simplified to illustrate the concepts, and actual attacks can be more complex and sophisticated.

1.11 Current Literature On Defense Mechanisms

Various vulnerabilities exist in different designs of federated learning protocols, and extensive research has been conducted to examine these vulnerabilities [8]. One study focused on these vulnerabilities and proposed solutions for two attacks within the federated learning protocol. The first type of attack is known as a poisoning attack, which aims to hinder the learning process of a model or manipulate it to produce favorable inferences for the attacker. The second type is an inference attack, which compromises the privacy of participating individuals.

To counter these attacks in federated learning, a suggested approach [8] involves developing an optimized defense mechanism that allows the FL server to detect malicious activities targeting the FL system. However, implementing this approach would incur additional computational costs for the central FL server. Moreover, it is important to note that different defense mechanisms may exhibit varying levels of effectiveness against different types of attacks, and each defense mechanism entails its own set of circumstances.

The unnecessary communication overhead problem is a significant issue associated with federated learning (FL) [10]. To mitigate this problem, a sparse-tensor communication framework has been proposed for federated deep learning, which can effectively reduce communication overhead [10]. This approach introduces a versatile framework that composes compressed communication of sparse tensors into two sets of values and indices. It allows for independent and combined compressions of these sets, reducing unnecessary communication overhead. Extensive experiments presented in

the referenced paper demonstrate the approach's capability to decrease communication overhead in large-scale federated learning deployments significantly.

However, it is worth noting that this approach must address data poisoning [3] in federated learning. The design of federated learning, which involves aggregating data from multiple sources, creates a potential for data leakage through the gradient-sharing mechanism [11]. Particularly in vertical federated learning, such data leakage can have catastrophic consequences.

Federated learning comprises multiple phases, each introducing security and privacy threats [14]. For instance, evasion attacks are a concern in data and behavior auditing. To defend against such attacks, image preprocessing and feature transformation techniques can be employed. However, these methods prove to be ineffective when the attacker possesses knowledge of the defense mechanisms.

Chapter Summary

Understanding the various attacks targeting federated learning is crucial to develop effective defense mechanisms.

Attacks such as dirty-labeled data attacks, backdoor attacks, data pollution attacks, evasion attacks, advanced data leakage attacks, and gradient attacks pose significant threats to the integrity, privacy, and performance of federated learning (FL) systems. Researchers and practitioners must identify and address these vulnerabilities to enhance the security and reliability of FL, thus facilitating its widespread adoption across diverse domains while preserving data privacy.

The studies mentioned above have concluded that participants in federated learning can engage in cheating behaviors by introducing data poisoning or parameter poisoning. These attacks aim to slow down or disrupt the convergence of the training process. Notably, existing defense schemes still need to detect such attacks in real time, highlighting the need for more effective measures to ensure the security and integrity of federated learning systems.

Chapter 2

2 Design Challenges

A Federated learning system is a large-scale distributed system designed to prioritize data privacy, enabling it to address the limitations of training data availability. However, the architecture of these systems introduces several challenges, particularly concerning the interactions between the central server and client devices and the management of trade-offs in software quality attributes. The key system design challenges can be summarized as follows.

- The initial step in the federated learning process is to build a global model and send this global model by a global server to the client's local servers.
- In the second step, the model gets trained on the local servers.

- In step 3, the client's local servers send the results back to the global server, and in the end, the global model utilizes the updates to build a better model. The process continues until the global model builds a robust model. The challenge of this step is that global models have low accuracy and lack generality when each client device generates non-IID data. Conventional machine learning has dealt with the data heterogeneity problem by centralizing and randomizing the data. However, the inherent privacy-preserving nature of federated learning renders such techniques inappropriate.

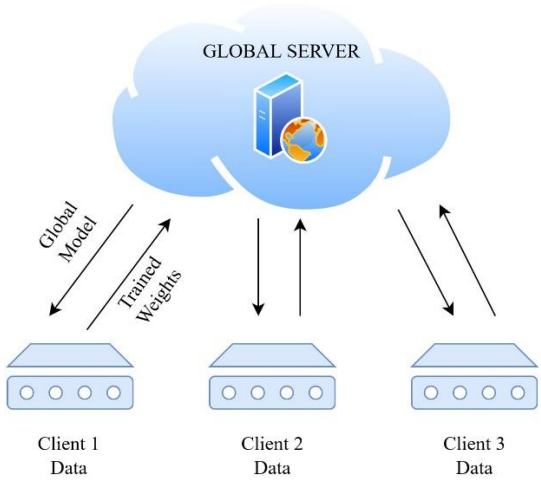

Figure 6: Architectural design of federated learning.

- In the architectural design of federated learning, generating high-quality global models necessitates multiple rounds of communication for local model updates [21]. However, these repeated communication rounds in federated learning significantly burden client devices, resulting in heavy client overhead and environmental strain. Moreover, there is no guarantee that client devices will

possess sufficient resources to effectively handle the system's multiple rounds of model training and communication demands.

- Coordinating the learning process and ensuring model provenance, system reliability, and security pose challenges in federated learning due to the involvement of numerous client devices.

There are various false assumptions surrounding federated learning and the existing design challenges. One false assumption is that participants are not attackers and would not utilize poisoned data. Exploiting this vulnerability, attackers can employ polluted data to train their models locally and send these polluted model updates to the edge server for aggregation. This creates an opportunity for data poisoning by manipulating image data. Consequently, it becomes difficult for an edge server to thoroughly scrutinize the data used for model training and supervise all edge devices. Additionally, client devices are not guaranteed adequate resources to handle multiple communications, supervision, and support strong encryption schemes.

The architectural design of federated learning allows the compromise of local nodes' data by inserting bogus data files, altering existing data, or man-in-the-middle attacks. In a man-in-the-middle attack, an attacker can use polluted data to transmit tainted training results to the server. For example, an attacker can modify an image file of an animal to poison the training data.

To prevent data poisoning from the client-side, it is essential to safeguard the integrity of the original data training and prevent any unauthorized modifications [4]. In a specific paper [4], the attack's main objective is to create a joint model that achieves

high accuracy on the main task and the attacker's chosen backdoor subtask over multiple rounds after the attack.

In the original federated learning model, it is not permissible for participants and the server to share any private data. For instance, participants and the server are restricted from sharing images based on the foundational design of federated learning. To prevent data pollution, a new approach is required where a small encrypted dataset is established between the server and the participant. In this paper, an encryption-based verification method is adopted. This allows the server to detect inconsistencies in the training samples at a participant without knowing the specific details held by individual participants.

During each training phase, participants and the server can exchange encrypted data in the initialization phase to prevent text modification. This helps safeguard the data from being modified in subsequent phases. After the initialization phase, the participant conducts the training while the server responds to the verification without knowing the participant's specific data holdings.

In successful data poisoning attacks, the attacker needs access to the original data samples from the client side and the ability to adapt the local training data from round to round.

The design challenges of defense mechanisms for federated learning are as follows:

1. Adversarial Nature: Federated learning operates in a decentralized and collaborative environment, making it susceptible to various adversarial attacks. The defense mechanisms need to account for the adversarial nature of the

system and protect against attacks such as data poisoning, backdoor attacks, and evasion attacks.

2. Privacy Preservation: One of the key objectives of federated learning is to maintain the privacy of participants' data. Defense mechanisms must ensure that privacy is preserved throughout the training process, even in the presence of potential attacks.

3. Communication Overhead: Federated learning involves communication between the central server and multiple participants, which can result in significant communication overhead. Defense mechanisms should aim to minimize this overhead without compromising the security and integrity of the system.

4. Scalability: Federated learning is often employed in large-scale deployments involving numerous participants and massive amounts of data. Defense mechanisms must be scalable to accommodate the increasing size of federated learning systems and efficiently handle the associated security challenges.

5. Real-time Detection: Detecting and mitigating attacks in real time is crucial to prevent the exploitation of vulnerabilities. Defense mechanisms should be able to identify and respond to attacks promptly, ensuring the system's security without significant delays.

6. Robustness: Defense mechanisms should be robust against various attack strategies and adaptable to evolving threats. They should be able to withstand sophisticated attacks and continually evolve to address new vulnerabilities.

7. Compatibility and Interoperability: Federated learning is employed across different domains and platforms. Defense mechanisms should be designed to be

compatible with different systems, ensuring interoperability and ease of adoption.

Addressing these design challenges requires a multidisciplinary approach, incorporating security, privacy, machine learning, communication, and system design expertise.

This research paper aims to achieve two properties in federated machine learning. The first property is data storage integrity, accomplished through verification to identify any modifications made to the original data. The second property prevents attackers from modifying the original data in successive rounds, thus thwarting adaptive data poisoning attacks.

Chapter Summary

Chapter 2 of the thesis presents an extensive literature review on attacks targeting federated learning and examines the incentive problem associated with this learning paradigm. By analyzing existing research, Chapter 2 provides a comprehensive understanding of the landscape of attacks in federated learning and the shortcomings of existing defense strategies. The chapter further delves into the various defense and attack mechanisms employed in federated learning, shedding light on the limitations of each defense mechanism.

The analysis of defense and attack mechanisms highlights the intricate nature of securing federated learning systems and the need for robust solutions to effectively address the unique challenges posed by this distributed learning paradigm. Moreover, this chapter expands upon the discussed defense mechanisms by exploring their effectiveness in different scenarios and identifying potential areas for improvement. Additionally, it explores the incentives that drive participants in federated learning and the challenges they face in maintaining cooperation. By synthesizing the existing literature, Chapter 2 offers valuable insights into state of the art in protecting

federated learning systems from attacks while identifying opportunities for further research and development.

Chapter 3

3 Proposed Model

Data poisoning in federated learning poses a significant challenge as participants can act as attackers, making it difficult to ensure proper supervision during training. The presence of noisy or polluted sample data in federated machine learning can harm the performance of machine learning models.

This research paper addresses this issue by proposing a novel preventative approach. The proposed approach introduces a security key to the data file of federated learning, enabling the training server to supervise each training round. It is assumed that the training server does not know the individual data in each edge device. Importantly, this framework ensures that the accuracy of the training is unaffected and does not impose additional computational demands on the federated learning process.

To develop a preventive defense scheme for federated learning, it is crucial to understand the requirements for a successful attack on this approach. Various research papers have examined real-world attack scenarios and have identified three main categories of attacks that can lead to data poisoning in federated learning [1].

1. The first category is a direct attack, where the attacker injects poisoned data into a specific target node. This type of attack involves modifying sensor data on devices like mobile phones, including individual sensors, images, or text data [1].

2. The second category of attack is referred to as an indirect attack. The attacker indirectly influences the target nodes by inserting poisoned data samples into other mobile participants. In this case, the attacker cannot directly inject poisoned data into the target node. Instead, they exploit vulnerabilities in the communication protocol between the training server and remote training participants. By manipulating the communication protocol among the participants, the attacker injects poisoned data that affects the target nodes. They can also manipulate the local training process and hyperparameters, such as the number of epochs and learning rate, and modify the weights of the resulting model before submitting it for aggregation. An example of an indirect attack is when the attacker gains control over one or several participating devices, such as smartphones.

3. The third attack category is hybrid, which integrates direct and indirect attacks. For example, the attacker can inject poisoned data samples into the target and source attacking nodes.

Several researchers tackled the difference between traditional data poisoning, which aims to change the model's performance on large parts of the input space [5], and the other types of attack that aim to prevent convergence [6].

An image classification attack experiment was completed by a research paper [4] by selecting three features as the backdoor. In this type of image classification attack, the attacker aims to generate his images with the backdoor feature to succeed in the attack and train his local model. A single-shot attack could successfully inject a backdoor into this model; however, 20 rounds afterward, the backdoor attack will be entirely successful. In this experiment, researchers acted as an attacker and tried to misclassify car images with images of birds. To perform a backdoor attack on a word-prediction model involves a typical sentence such as driving as the trigger or the relatively infrequent word Jeep. The ending tends to be forgotten more quickly, Figure 2: Word-prediction backdoor attack. The attacker's main objective is to produce a joint model that achieves high accuracy on the main task and the attacker's chosen backdoor subtask for multiple rounds after the attack.

We must understand all requirements for a successful attack to propose a new robust preventative defense scheme for federated learning.

Suppose an attack succeeds by altering the original data or manipulating the model parameters. This indicates that the attacker is modifying the original data in each round of the machine learning training.

An attacker who controls fewer than 1% of the participants can successfully create a backdoor attack. In federated learning, changing a small portion of the data, where a single attacker is selected in a single round of training [4], causes the joint model to

achieve 100% accuracy on the backdooring attack. Model replacement greatly outperforms "traditional" data poisoning. Sometimes, the attacker is intelligent and sends updates based on machine learning rules, not random parameters. Therefore, a successful data poisoning attack starts with access to the original data sample from the client side.

Regardless of the specific attack category, the common requirement across all attack types is that the attacker must be able to modify their local training process as the rounds progress dynamically. For a data poisoning attack to be successful, the attacker must possess the capability to modify the training data on a remote client and adjust the local training data in each subsequent round. This highlights the crucial role of having access to the original data sample from the client's side as a starting point for a successful data poisoning attack.

To tackle these issues, the first step is to safeguard the training data against modification by the remote client during each training session. In a previous study [4], the primary goal of the attack was to create a joint model that achieves high accuracy on both the main task and the attacker's chosen backdoor subtask, sustaining its effectiveness across multiple rounds after the attack.

In the original federated learning model, neither the participants nor the server can share private data. This implies that, based on the fundamental principles of federated learning, no data can be exchanged between the participants and the server. Therefore, to prevent any contamination of the data, a novel approach must be implemented, requiring the server and the participants to establish a small encrypted dataset.

In this research, we adopt an encryption-based verification method. Whenever the training samples at a participant exhibit inconsistency in each training step, the server can identify these inconsistencies, even without being aware of the specific data held by individual participants.

To avoid text modification during each training phase, a strategy is employed where encrypted data can be shared between participants and the server during the initialization phase. This sharing of encrypted data ensures that the data remains unaltered throughout each phase. Subsequently, during the training phase, the participant performs the training while the server responds to the verification process without knowing the specific data held by the participants.

This research paper aims to accomplish two properties of federated machine learning. The first property is to ensure the integrity of data storage by employing data verification techniques to detect any modifications made to the original data. The second property aims to prevent an attacker from adapting and modifying the original data across multiple rounds, thwarting adaptive data poisoning attacks.

A crucial distinction between the federated learning model and the traditional machine learning model lies in the fact that in federated learning, each remote participant cannot share any training data with the training server. This design choice effectively prevents the training server from overseeing the activities of individual remote participants.

A novel supervision approach must be introduced to address this design challenge, enabling the training server to oversee the remote participant without sharing the training data. In federated learning, where the number of remote participants can

reach millions, an effective poisoning prevention scheme must be implemented to identify and eliminate any infected remote participant before training occurs.

The initial step in designing a scheme to prevent training poisoning involves establishing a small encrypted data set between the server and the participant. This data set should not expose the training data to the training server, yet it should allow the server to verify the authenticity of the training data. In this study, an encryption-based verification method is adopted, enabling the server to detect inconsistencies in the training samples a participant provides during each training step, even without knowledge of the specific data held by individual participants. Figure 7: An example of a verification key for Text files., demonstrates this scheme.

An essential component of the verification scheme is ensuring the legitimacy of each remote participant and preventing any third-party attacker from impersonating a participant by creating an iteration password.

Once the participant receives the iteration password from the training server, they will be considered a verified participant for that specific training. If someone tries to impersonate a verified participant, they will fail because they will not have the iteration password.

The server will also provide the participant with a public encryption key. This key allows the participant to create a verification key for each training file and encrypt the verification file using the public encryption key. Only the training server has the corresponding private encryption key to decrypt the verification file.

After receiving the public encryption key and iteration password, the remote participant must select random data from the training data files and record the random data and its location in a verification key file.

The algorithm below outlines the protocol for storing data files in a remote node. It involves creating a verification key file while the remote participant generates the training data file. The verification key file is generated and stored simultaneously. For instance, if the remote participant uses an Android device and the training data contains the user's typed sentences, the training software would extract random characters and their respective locations from the text file.

The principal strategy is to prevent data poisoning of federated learning in two main phases. The first phase prevents data-generating nodes from inserting data into the sample or modifying any data. In this phase, the training server identifies the remote participants selected for the training before any sampling begins. It generates a random password for that iteration and a public key for encryption.

Once the remote participants receive the iteration password and public encryption key, the node will begin sampling. The next phase is to create a verification key file for every node-generated data file. Later on, the server will use this verification key file to supervise the training.

This verification key file will ensure that the trusted remote participant has created the sample data file and no other parties can modify this file Figure 7: An example of a verification key for Text files., which demonstrates the creation of the sample data file. The device creates a sample key file upon creating this sample data file.

1. The sample verification file's first line will be the training's password. This scheme will allow the system to quickly decrypt the first line to verify whether the iteration's password is correct.

2. The second line of the storage verification key file will be the checksum which will record the bit number of that text file.

3. From line three to the end of the verification file, the worker device will select random text and record every byte value and location within the verification key file.

4. The worker node will use the encryption key to encrypt the file so that no other party can verify the verification key file.

5. The output of Algorithm 1 will be a storage verification key file.

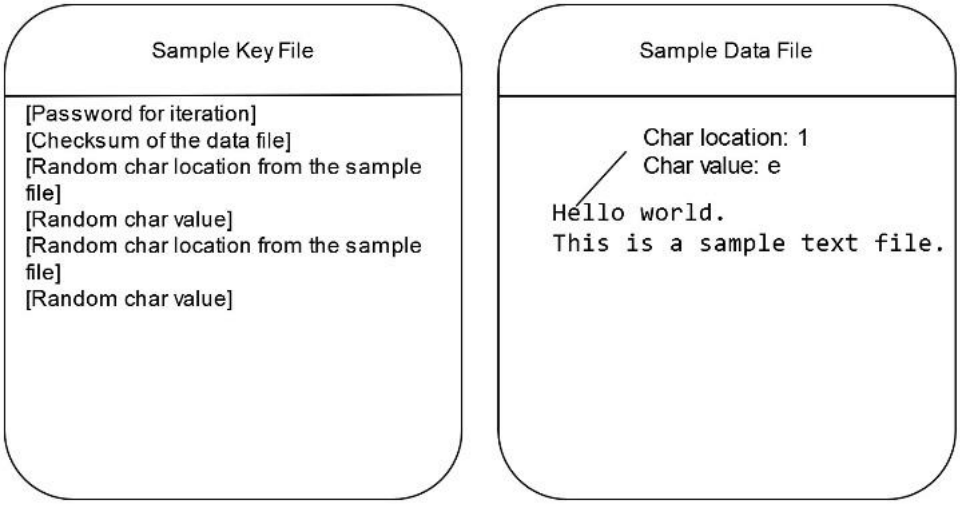

Figure 7: An example of a verification key for Text files.

The proposed scheme for detecting data modification in federated learning can achieve its goal with three different algorithms. The first algorithm is for participants' data

training file storage protocols, and the second is for each node's file integrity verification. The third algorithm is for training server transmission key verification.

For the 1st algorithm, there is the assumption that before the training, the server and the participant will establish a verification key that can be used during the sample data storage phase. The participant will use this verification key to store the sample data. For example, if the sample data is a text file that stores the user's typing sentences, each participant will register with the server before the training. Afterward, the participant will receive a public verification encryption key and an encryption key for each iteration that can be used to create a local verification file. The verification public key can only be used for encryption, and it cannot be used for decrypting any encrypted key.

Once the node receives the iteration password and the public encryption key, the node is considered a trusted node and can begin the sampling process. The next step is to create a verification key file for every node-generated data file.

After gathering enough sample data, when a worker device is ready to begin training, it will receive a decryption key and iteration password from the training server and check every sample data file using the verification key. This will ensure that the participant is registered and that an attacker is not inter-personating a participant. The latest algorithm will check the verification keys during the training.

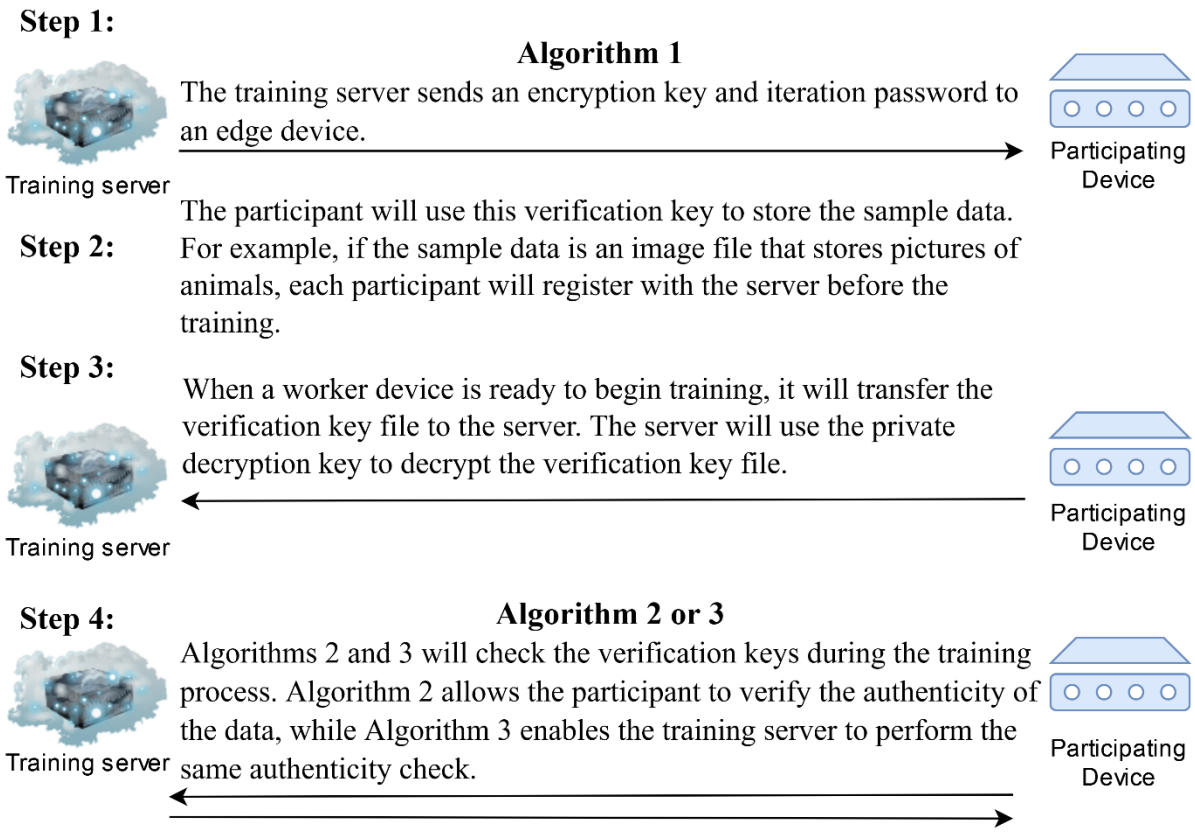

Figure 8: The proposed framework for the implementation of three algorithms.

The proposed framework for the experiment of this defensive scheme is to use TensorFlow on Google Colaboratory "Colab." The TensorFlow framework allows federated learning training on its platforms, and there are many previous experiments in federated learning training with data files. We can experiment with the three proposed algorithms and check their functionality and limits with this platform.

The first step of the implementation is to implement the three algorithms and check whether they can successfully use a sample data file.

The second step of the implementation is to implement these algorithms in other federated learning training and ensure that they can prevent any attacks.

The following is an example of the storage verification key file for each data file. The participant will create this file locally when each data file is written. This file will be encrypted by using the encryption key.

Later during the training, the participant will send this file to the server, and only the server can decrypt this file. The verification file will only contain random values of random characters from each data file, and the server will use this information to verify the authenticity of the training data.

3-1: Algorithm 1 For Image Verification

Algorithm 1 Training Data file Storage Protocol for Nodes

Require: An encryption key and iteration password from the training server

1: **Initialization** Create a Data file, Create a blank Key file
2: **Integer** NumberOfByteToVerify = 75
3: **Write** iteration password to **BOF** of Key file
4: **Write** Checksum of Datafile to 2nd line of Key file
5: **Go** to the **next line** in the Key file
6: **From 3rd** line of the Data file to the **EOF** of the Data file
7: **If** NumberOfByteToVerify != 0
8: **Choose** a random Char from the Data file
9: **Write** ChosenByteLocation
10: **Go** to the **next line** in the Key file
11: **Write** ChosenByteValue
12: **Decrement** NumberOfByteToVerify
13: **Go** to the **next line** in the Key file
14: **Encrypt** Key file (**encryption key**)

BOF: is a specific marker that shows where a file starts.

EOF: is a condition in a computer operating system where no more data can be read from a data source.

Table 1: Text file Demonstration table

Line#:	Written Data on The File.	Comment:
1:	w0Grf353#43	Iteration password
2:	1612f797418a53dc652d385bda0e014f	Checksum value
3:	23	Character location
4:	s	Character value
5:	36	Character location
6:	i	Character value
7:	1	Character location
8:	e	Character value
9:	40	Character location

After gathering enough sample data, when a worker device is ready to begin training, it will receive a decryption key and iteration password from the training server and check every sample data.

Algorithm 2 will first use the decryption key and decrypt the sample file.

1) Step 1 is to check the password. If the iteration password of the storage verification key is the same as the iteration password received from the server, then algorithm two will continue. If not, then it will return false.

2) The second step of the algorithm is to check the checksum to ensure the file size is original. The participant will send the current checksum value and the verification key file to the server. Only the server can decrypt the verification file and verify the checksum value.

3) The third step of algorithm 2 is to compare the values of the characters from the storage verification file with the actual sample file and ensure that the file is

original. Only the server can decrypt the verification file and compare the values of the pixels.

4) If all the checks are correct, algorithm two will return true.

The output of Algorithm 2 will be a true or false value which will demonstrate to the worker device that will determine the integrity of the sample data file.

3-2: Algorithm 2 For Image Verification

Algorithm 2 Training Data File Integrity verification for each Node

Require: A decryption key and iteration password from the training server

Initialization import **Data file**, import **Key file** for the data file
Decrypt Key file (**decryption key** from training server)
If the BOF of the Key File **Does not contain the** iteration password
    Return **False**
    else
        If checksum from 2nd line of the key file != checksum
            Return False
        else
            From 2nd line of the key file to the **EOF** key file
        Go to the **next line** in the Key file
    Read ChosenByteLocation
        Go to the **next line** in the Key file
    Read ChosenByteValue
        If ChosenByteValue exists in the ChosenCharLocation of the Data file
Continue
                Else return false

If any doubt arises, each node can utilize**Error! Reference source not found.**, to a uthenticate its storage integrity during the final phase. If all checks pass successfully, it confirms the storage integrity of that particular node. Each node will resend the iteration password to the server for verification as part of this algorithm.

3-3: Algorithm 3 For Image Verification

Algorithm 3 Node Integrity Check Algorithm

Require: Training server uses the decryption key

1: **Initialization** decrypts all verification key files
2: **Choose N # of** random key files.
3: **Decrypt** the chosen key files using the decryption key
4: **If** the 1st line of all five key files matches the iteration password
5: It selects random lines from the verification file
6: Requests that information from the node
If all information matches then it **accepts the authenticity** of that participating device
7: **else**
8: It concludes that the data in that participating device is **compromised**

3.1 Proposed Model for Image Recognition Data

Any noisy or polluted sample in federated machine learning will negatively impact machine learning performance. Data poisoning poses a challenge as it is difficult to control and assess the quality of individual samples. The primary issue in federated learning is that participants can act as attackers, and it is unrealistic to expect them to supervise each training session. This paper presents a novel preventive approach to constructing a new federated machine learning framework. The proposed method enables the training server to supervise each training round by incorporating a security key into the image data of federated learning. It assumes that the training server remains unaware of the individual data in each edge device. This approach ensures training accuracy while avoiding additional computational demands on the federated learning process.

This paper introduces a detection prevention scheme that safeguards image data stored on each client's device. It enables the server to detect modifications and promptly eliminate infected clients in real-time. This scheme benefits low-processing devices by requiring minimal processing power.

In machine learning training, a successful attack, whether by altering original data or manipulating model parameters, can be viewed as the attacker modifying the original data in every training round. In federated learning, even changing a small portion of the data, with a single attacker selected in a single training round, can lead to the joint model achieving 100% accuracy in backdooring attacks. Remarkably, an attacker controlling at most 1% of the participants can successfully create a backdoor attack. Model replacement significantly outperforms "traditional" data poisoning, and

attackers may employ intelligent strategies based on machine learning rules rather than random parameters. Hence, a successful data poisoning attack starts with gaining access to the original data sample from the client side.

In federated learning, participants and the server do not share any private data in the original model. A new approach is necessary to prevent data pollution, allowing the server and participants to establish a small encrypted data set. This paper adopts an encryption-based verification method, enabling the server to detect inconsistencies in training samples without knowing the specific data held by individual participants.

Participants and the server share encrypted verification keys in the initialization round to prevent text modification during each training round. Subsequently, participants train locally and send model updates to the server, which performs verification without accessing the participants' held data.

This research paper aims to achieve properties that prevent attackers from modifying the original image file across training rounds, thus thwarting adaptive data poisoning attacks. This property ensures that each participant cannot report incorrect model updates from round to round.

Figure 1: The attacker has modified the cat image cat-4, demonstrating a data pollution attack example. In this figure, the attacker has modified the image cat-4.jpg samples on the left side by adding dog picture images to the sample data. The attacker has created the new image samples on the right side

The main strategy for preventing data poisoning in federated learning involves two phases. The initial phase ensures that data-generating nodes cannot insert or modify image data in the sample. During this phase, the training server identifies the nodes

chosen for training before sampling occurs. It generates a random password and a public key for encryption.

Once the node receives the password and encryption key, it can start sampling. The subsequent phase involves creating a verification key file for each data file generated by the node. The server later uses this verification key file to supervise the training.

The device generates a sample key file along with the sample data file. The verification key file serves two purposes: confirming that the trusted node created the sample data file and preventing unauthorized modifications. Figure 7: An example of a verification key for Text files. data illustrates the process of creating the sample data file.

1- The sample verification file begins with the password for that specific training iteration, allowing for quick decryption and verification of the password's correctness.
2- The second line contains a checksum that records the bit number of the image file.
3- From the third line onwards, the worker device randomly selects an image file and records each byte value and location in the verification key file.

To ensure the security of the verification key file, the worker node encrypts it using the encryption key, making it inaccessible to other parties. The output 3-1: Algorithm 1 For Image , is a storage verification key file.

The suggested approach in federated learning aims to detect any modifications made to the data, which can be achieved through three different algorithms. The initial algorithm deals with the storage protocols for participants' data training files, while

the second algorithm focuses on verifying the integrity of files on each node. The third algorithm is designed to verify the transmission key on the training server.

For the first algorithm, it is assumed that the server and participant will establish a verification key before training, which will be used during storing sample data. Participants will utilize this key to store their sample data. For instance, if the sample data consists of animal images, each participant will register with the server before training. Subsequently, the participant will receive a verification public encryption key and an encryption key for each iteration, enabling the creation of a local verification file. The verification public key can only be used for encryption and cannot decrypt any encrypted key.

Once a node receives the iteration password and public encryption key, it is deemed trustworthy and can begin sampling. The next step involves generating a verification key file for every data file produced by the node.

When a worker device is ready to begin training after gathering sufficient sample data, it will obtain a decryption key and iteration password from the training server. The worker device will validate each sample image file using the verification key. This process ensures that the participant is registered and guards against impersonation by attackers. The final algorithm checks the verification keys during the training process.

The proposed framework uses TensorFlow on Google Colaboratory (Colab) to experiment with this defensive scheme. TensorFlow supports federated learning training on its platform, and previous experiments utilizing data files have been conducted. With this platform, we can assess the functionality and limitations of the three proposed algorithms.

The first step in implementing this scheme is to incorporate the three algorithms and determine their effectiveness using a sample data file.

The second step involves implementing these algorithms in other federated learning training scenarios and confirming their ability to prevent attacks.

This is a sample of the storage verification key file for each data file. When writing each data file, the participant will generate this file on their computer. The file will be encrypted using an encryption key. Later in the training process, the participant will send this file to the server. Only the server can decrypt the file. The verification file will contain random values and characters extracted from each data file. The server will utilize this information to confirm the authenticity of the training data.

Once sufficient sample data has been collected, the training server will provide a worker device prepared for training with a decryption key and an iteration password. The worker device will then proceed to verify each sample data. Algorithm 2 begins by utilizing the decryption key to decrypt the sample file. The algorithm follows these steps:

1) Step 1 involves checking the password. If the iteration password obtained from the server matches the iteration password of the storage verification key, 3-2: Algorithm 2 For Image Verification, Continues. Otherwise, it returns false.

2) In the second step, the algorithm verifies the checksum to ensure the file size is unchanged. The participant sends the current checksum value and the verification key file to the server. Only the server can decrypt the verification file and confirm the correctness of the checksum value.

3) The third step of 3-2: Algorithm 2 For Image Verification, Compares the byte values of the storage verification file with those of the actual sample file to validate the file's authenticity. Again, only the server can decrypt the verification file and compare the pixel values.

4) If all the checks pass successfully, 3-2: Algorithm 2 For Image Verification, Returns true.

The output of Algorithm 2, Is a boolean value, true or false, indicating the integrity of the sample data file as assessed by the worker device.

3.2 Proposed Model for Text Data Verification

The primary approach aims to avoid data contamination in federated learning through two key stages. Initially, the strategy blocks data-generating nodes from adding or altering any data within the sample. During this stage, the training server identifies the remote participants chosen for the training before any sampling. It creates a random password for that particular iteration and generates a public key for encryption.

The node initiates the sampling process upon receiving the iteration password and public encryption key. Subsequently, the next step involves generating a verification key file corresponding to each data file produced by the node. The server will utilize these verification critical files to oversee the training. Their purpose is to guarantee that the trusted node is the creator of the sample data file, preventing any alterations by external parties. Refer to Figure 9: Key file sample for text-data file verification for an example of a critical file used to verify text-data files, illustrating the creation of sample data files. Upon generating the sample data file, the device concurrently generates a sample key file.

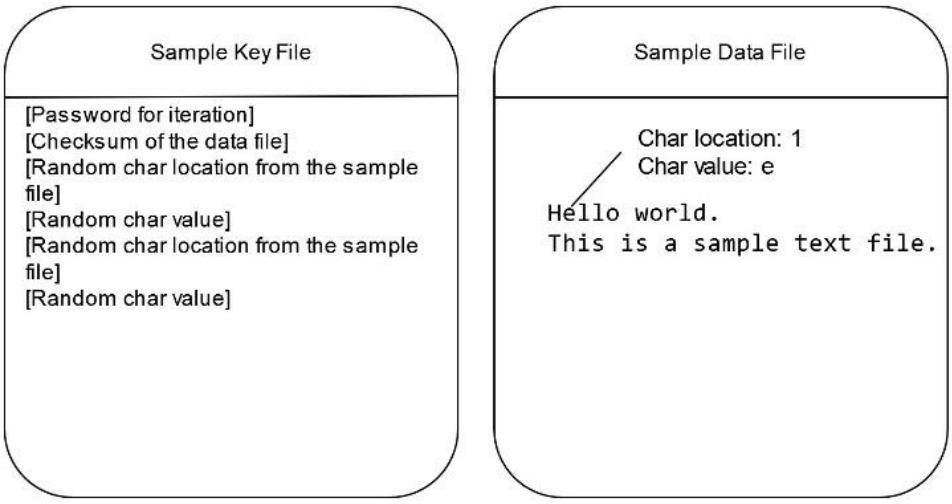

Figure 9: Key file sample for text-data file verification

3-4: Algorithm 1 For Text Verification

Algorithm 1 Training Data file Storage Protocol for Nodes

Require: An encryption key and iteration password from training server

1: **Initialization** Create Data file, Create blank Key file
2: **Integer** NumberOfCharToVerify = 75
3: **Write** iteration password to **BOF** of Key file
4: **Write** Checksum of Datafile to 2nd line of Key file
5: **Go** to **next line** in the Key file
6: **From 3rd** line of Data file to **EOF** of Data file
7: **If** NumberOfCharToVerify != 0
8: **Choose** random a Char from Data file
9: **Write** ChosenCharLocation
10: **Go** to **next line** in the Key file
11: **Write** ChosenCharValue
12: **Decrement** NumberOfCharToVerify
13: **Go** to **next line** in the Key file
14: **Encrypt** Key file (**encryption key**)

BOF: is a specific marker that shows where a file starts.

EOF: is a condition in a computer operating system where no more data can be read from a data source.

3-5: Algorithm 2 For Text Verification

Algorithm 2 Training Data File Integrity Verification For Each Nodes

Require: A decryption key and iteration password from training server

```
1:  Initialization import Data file, import Key file for the data file
2:  Decrypt Key file (decryption key from training server)
3:  If BOF of Key File Does not contain iteration password
4:              Return False
5:              else
6:                  If checksum from 2nd line of key file != checksum
7:                          Return False
8:                      else
9:                          From 2nd line of key file for to the EOF key file
10:                 Go to next line in the Key file
11:             Read ChosenCharLocation
12:                 Go to next line in the Key file
13:         Read ChosenCharValue
14:                     If  ChosenBtyeValue exist in the ChosenCharLocation of Data file
15:     Continue
16:                                 Else return false
```

In the verification scheme [demonstrated as step 1, in 3-5: Algorithm 2 For Text Verification, the first step is to ensure that each remote participant is legitimate and no third-party attacker can inter-personate a remote participant by creating an iteration password. Once the participant receives the iteration password from the train-ing server, that participant will become a verified participant for that specific training. If an attacker tries to inter-personate a verified participant for that specific training, the attacker will not succeed since the attacker will not have the iteration password. The second piece of information that the server will provide to the remote participant must be a public encryption key that would allow the participant to create a verification key for each training file and encrypt that verification file using the

public encryption key. Later only the training server can use the matching private encryption key to decrypt the verification file. Steps 2 and 3 in 3-5: Algorithm 2 For Text Verification, Demonstrate the implementation of the iteration password and the decryption key. After receiving the public encryption key and iteration password, the remote participant must choose random data from the training data files and write that random data and their location into a verification key file.

3-6: Algorithm 3 For Text Verification

Algorithm 3 Node Integrity Check Algorithm

Require: The training server uses the decryption key.

1: **Initialization** decrypts all verification key files
2: **Choose N # of** random key files.
3: **Decrypt** the chosen key files using the decryption key
4: **If** the 1st line of all five key files matches the iteration password
It selects random lines from the verification file
5: Requests that information from the node
6: **If** all information matches then it **accepts the authenticity** of that participating device
7: **else**
8: It concludes that the data in that participating device is **compromised**

3-6: Algorithm 3 For Text Verification, is for data file storage protocol for a remote node. This algorithm will create a verification key file while the remote participant generates the training data file. At the same time, it is generated and stored inside the verification key file.

BOF: is a specific marker that shows where a file starts.

EOF: is a condition in a computer operating system where no more data can be read from a data source.

The following is an example of the storage verification key file for each text files. The participant will create this file locally when each data file is written. This file will be encrypted by using the encryption key.

Table 2: Text Verification Table

Line#:	Written Data On The File.	Comment:
1:	w0Grf353#43	Iteration password
2:	1612f797418a53dc652d385bda0e014f	Checksum value
3:	23	Character location
4:	s	Character value
5:	36	Character location
6:	i	Character value
7:	1	Character location
8:	e	Character value
9:	40	Character location

Error! Reference source not found., receives a decryption key and iteration pass-word f rom the training server and checks every sample data file using the verification key.

3-2: Algorithm 2 For Image Verification**Error! Reference source not found.**, will check the verification keys during the training, and the participant device will send the requested information to the server allowing the training server to supervise the training.

3.3 Defense Mechanisms Against Dirty-labeled Data Attacks

Researchers have suggested several defense mechanisms to defend against dirty labeled data attacks. These may include:

Data sanitization employs techniques to detect and filter out malicious or suspicious data before training Federated learning models. For example, in the email spam folder example mentioned in the previous section, The global model or a trusted third party could validate the labels on the local datasets of each device to detect any inconsistencies or anomalies, such as the manipulated labels injected by the attacker on Device B. Secure aggregation is another approach to defend against dirty labeled attack, this defense mechanism applies encryption or secure multi-party computation techniques to protect the privacy and integrity of the aggregated updates during the federated learning process. However, this scheme requires the training server to have access to the original training data, and this option is not always available to privacy laws.

Another defense mechanism is to utilize robust aggregation algorithms that use aggregation algorithms that are resilient to the presence of compromised or adversarial devices in the federated learning setup. The aggregation algorithm used to combine the model updates from different devices can be designed to be resilient to the presence of compromised devices, effectively minimizing the impact of the attacker's poisoned updates.

Model verification and validation can also defend against dirty-labeled attacks by performing rigorous testing and validation of the global model to detect any potential

issues introduced by dirty-labeled data attacks. Additional verification steps can be taken to ensure the model's integrity and performance.

All the above defense mechanisms for dirty-labeled data attacks require access to the training data or a high amount of computational power without additional incentives for the clients.

The text data verification scheme introduced in this chapter can serve as an advanced defense mechanism against data or label manipulation attempts. Unlike traditional methods, this scheme ensures exceptional data integrity while maintaining privacy.

With the Text data verification scheme in place, the training server becomes proficient in identifying unauthorized modifications from potential attackers to the training data or labels. When such manipulations are detected, the server promptly eliminates the deceptive participant from the training process, thereby preserving the model's integrity and effectiveness.

By adopting this cutting-edge approach, federated learning models can maintain their accuracy and reliability even when confronted with adversarial attacks, bolstering the overall performance and security of the system. The proposed implementation model sets a new standard in federated learning, fortifying data security and minimizing the risk of dirty-labeled data attacks.

3.4 Defense Mechanisms For Backdoor Attacks

Defense mechanisms for each type of backdoor attack [16] on federated learning, along with a simple example for each:

One approach to defend against model poisoning attacks is implementing robust aggregation methods to detect and mitigate malicious model updates' impact. One such method is Trimmed Mean, which removes outliers before aggregating the model updates and mitigates the influence of poisoned updates. For example, in a federated learning system, the central server can calculate the trimmed mean by discarding model updates that differ significantly from most.

The defense against Data Injection Attacks is to use input validation and outlier detection techniques. During the local training, participants can use statistical analysis to identify and remove potential outliers or suspicious data points. For example, in a federated learning system training a model to recognize handwritten digits, participants can implement image quality checks to identify anomalies or modifications in the training data before including them in the local training.

One suggested approach to defend against Byzantine attacks is to utilize Byzantine fault-tolerant (BFT) algorithms. These algorithms enable participants to reach a consensus on the aggregated model update despite the presence of malicious participants. For example, federated learning systems can employ BFT algorithms such as Byzantine-Resilient Federated Averaging (BRFA) that can tolerate a certain number of Byzantine participants by leveraging cryptographic techniques and redundancy.

Various privacy-preserving techniques can protect against privacy breach attacks. Differential privacy is one such approach where noise is intentionally added to the model updates to protect the privacy of individual data. For instance, in a federated learning system for medical diagnosis, the participants can inject carefully calibrated noise into their model updates to ensure that individual patient data remains private even if an attacker tries to extract information from the model parameters.

The above defense mechanisms need to be foolproof, and the choice of defense depends on the specific requirements and constraints of the federated learning system.

The defense mechanisms mentioned above must be completely reliable, and the selection of a defense mechanism relies on the specific needs and limitations of the federated learning system. Nevertheless, the text data verification scheme presented in this chapter operates independently of the particular conditions and constraints imposed by the training participants. Unlike conventional approaches, this scheme ensures impeccable data integrity while upholding privacy.

With the implementation of the Text data verification scheme, the training server becomes adept at detecting any unauthorized alterations made by potential attackers to the training data or labels. Upon noticing any manipulations, the server swiftly removes the deceptive participant from the training process, thus safeguarding the model's integrity and efficacy.

3.5 Defense Mechanism Against Data Pollution

Several techniques can prevent data pollution in federated learning. The Differential Privacy Technique is one of the defense mechanisms against data pollution in federated learning, and it provides privacy guarantees by adding controlled noise to the participants' updates or queries. For instance, participants can apply randomized response mechanisms to obfuscate their sensitive updates before sharing them with the central server. This ensures that individual participants' data cannot be inferred from the aggregated updates, protecting privacy. Secure Communication Channels can also defend against data pollution in federated learning, and it ensures the confidentiality and integrity of the data exchanged between participants and the central server. For example, participants can utilize secure protocols such as Transport Layer Security (TLS) or Secure Socket Layer (SSL) when transmitting their updates to the central server. Encrypting the communication channels prevents data leakage or unauthorized access to the exchanged data.

Monitoring and detection defense mechanisms aim to identify data pollution or malicious behavior during the federated learning process. For example, a simple monitoring mechanism can track the performance of the federated model over time and raise an alert if a significant drop in performance is observed, indicating potential data pollution. Additionally, anomaly detection algorithms can identify participants that deviate from the expected behavior, indicating possible malicious activity. These measures ensure the federated learning process's integrity, reliability, and privacy despite potential data pollution.

The defense mechanisms should be highly dependable, and a secure communication protocol between the training server and participants is crucial. However, not all devices can handle secure protocols due to their required extra computational power.

Still, the text data verification scheme discussed in this chapter works independently of the specific conditions and limitations set by the training participants. It effectively safeguards against man-in-the-middle and communication-related attacks. The training server can promptly detect any data modifications during training.

3.6 Reactive Defense Schemes On Federated Learning

The previously proposed defenses have focused on detecting attacks on federated learning after they happened. These defenses are called reactive schemes.

One drawback of reactive schemes is that the system may take a long time to react to the attacks. Moreover, reactive schemes are designed for a specific type of attack. Therefore, we need a proactive protection scheme, an all-in-one solution.

Previously proposed defenses against such attacks increased the batch size to complicate data recovery. However, other researchers [11] demonstrated that catastrophic data leakage in vertical FL is still possible. For example, a novel algorithm [11] can perform large-batch data leakage with high data recovery quality and theoretical guarantees.

In federated learning, a reactive defense scheme refers to techniques and strategies employed to detect and mitigate various security and privacy threats that can arise during training. These threats include data poisoning, model inversion, and membership inference attacks.

Data Sanitization is a reactive defense scheme aiming to detect and remove malicious or poisoned data samples from the federated dataset beforehand of training. One example of data sanitization is anomaly detection, where statistical techniques can identify data samples that deviate significantly from the expected distribution.

One good example of a data sanitization reactive defense scheme in federated learning is where multiple hospitals contribute patient data, and data sanitization

techniques can be employed to identify and remove outliers or anomalous data points that may be intentionally inserted to bias the training process.

Model Update Verification is another type of reactive scheme, and it involves verifying the integrity and authenticity of the updated global model before it is distributed back to the participating devices for further training. This ensures that the model has not been tampered with or compromised during aggregation.

One good example of a model update verification reactive defense scheme is using cryptographic techniques, such as digital signatures, to sign and verify the model updates. Each participating device can verify the signature to ensure an adversary has not modified the model.

Differential privacy is a scheme that aims to protect the privacy of individual data samples in federated learning. It adds noise or perturbation to the model updates or gradients, making it difficult for an adversary to extract sensitive information about specific individuals.

An example of differential privacy in federated learning is the addition of Gaussian noise to the gradients calculated on local devices before sending them to the central server for aggregation. This noise helps protect the privacy of individual data samples.

The adversarial Robustness scheme [13] focuses on defending against adversarial attacks on the federated learning process. Adversarial attacks aim to manipulate the training process to compromise the model's performance or extract sensitive information.

An example of an adversarial training technique is to generate adversarial examples during training and incorporate them into the training dataset, forcing the model to learn to be more resilient to such attacks.

Various combinations and extensions of these techniques can also be employed to enhance the security and privacy of federated learning systems, depending on the specific threats and requirements of the application. However, the drawback of the above reactive scheme is that the training server must be able to access each remote participant's private data, which is only sometimes possible due to laws and regulations in every region.

The data verification scheme discussed in this chapter can be put into practice without requiring access to the private data of each remote participant, which varying laws and regulations in different regions may limit.

3.7 Defense Mechanisms Against Advanced Data Leakage Attack

Various defense schemes can be employed to defend against advanced data leakage attacks in federated learning.

Differential privacy can help protect against membership inference and reconstruction attacks by adding controlled noise or perturbation to the model updates or gradients shared during federated learning. It aims to make it difficult for an attacker to infer sensitive information about individual data samples.

For example, one approach is to use techniques like a randomized response, where each participating device perturbs its gradients by adding random noise before sending them for aggregation. This noise masks the contribution of individual data samples, providing privacy guarantees.

Model regularization techniques help defend against reconstruction attacks by adding constraints or penalties during training. These constraints limit the model's ability to overfit specific training data points and make it harder for an attacker to reconstruct sensitive information.

For example, adding an L2 regularization term to the loss function during training helps control the model's complexity. It penalizes large weights and reduces the impact of individual data points, thereby improving the model's generalization and mitigating the risk of data reconstruction.

Privacy-preserving federated learning techniques combine multiple defense mechanisms to provide comprehensive protection against data leakage attacks. These mechanisms

can include encryption, secure multi-party computation, and differential privacy. For example, in MPC, participating devices jointly compute the model updates without revealing their data. Each device holds its input and performs cryptographic operations to contribute to the aggregation while preserving privacy.

Adversarial robustness techniques focus on defending against attacks that aim to manipulate the model's behavior or extract sensitive information. They help detect and mitigate malicious attempts during the federated learning process. Incorporating adversarial training as a defense mechanism can enhance the model's robustness. The model is trained with clean data and adversarial examples generated using techniques like Fast Gradient Sign Method (FGSM), making it more resilient to attacks.

3.8 Defense Mechanism Against Evasion Attacks

There are categories of defense mechanisms against evasion attacks in federated learning.

The first category focuses on modifying the input data before training the federated learning model. They aim to remove or reduce the influence of malicious or adversarial samples. For example, one preprocessing-based defense is input normalization. In this method, the input data is standardized by scaling each feature to a predefined range between 0 and 1. Normalizing the data can mitigate the impact of extreme values or outliers that an attacker might intentionally inject.

The second category of defense mechanism against evasion attacks is Model-based defenses involve modifying the learning algorithm or the model to improve robustness against evasion attacks.

For example, Adversarial training is a commonly used model-based defense. It involves augmenting the training data with adversarial examples generated by intentionally perturbing the input samples. By training the model on clean and adversarial samples, the model learns to be more resistant to evasion attacks.

System-based defenses enhance the federated learning system to detect and prevent evasion attacks.

An example of a system-based defense is anomaly detection. By monitoring the behavior of each participating client during the federated learning process, anomalies or suspicious activities can be detected. For instance, if a client suddenly exhibits

significantly different performance or sends irregular updates, it may indicate a potential evasion attack.

These three categories of defense mechanisms against evasion attacks in federated learning, namely preprocessing-based, model-based, and system-based defenses, provide different approaches to enhance the security and robustness of the federated learning process against malicious adversaries.

While defense mechanisms against evasion attacks in federated learning effectively mitigate the risk of attacks, they also have some drawbacks.

Preprocessing-based defenses may be designed to counter specific attack patterns. If attackers devise new evasion techniques, the defense mechanism might fail to detect or neutralize them effectively. Some preprocessing techniques, such as data sanitization or outlier removal, may result in the loss of useful information. This could impact the model's accuracy and performance.

Applying model-based defenses, like adversarial training, often requires additional computational resources and time for generating adversarial examples and training the model. This can lead to higher overhead and slower convergence during the federated learning process.

Model-based defenses might improve the model's resistance to specific evasion attacks encountered during training. However, they may not guarantee robustness against new and unseen attack strategies during deployment.

Anomaly detection and monitoring techniques used in system-based defenses may produce false positive or negative alerts. False positives can disrupt the federated learning process, while false negatives can result in undetected attacks.

Privacy concerns: Some system-based defenses require monitoring the behavior of participating clients, potentially raising privacy concerns among users. Striking a balance between security and privacy is essential in federated learning systems.

While defense mechanisms in federated learning are valuable, they still face challenges regarding adaptability to new attack techniques, computational overhead, generalization, and privacy considerations. Continued research and development are necessary to address these drawbacks and improve the effectiveness and efficiency of defense mechanisms. The text verification scheme presented in this chapter ensures strong resilience and can be applied to various types of data.

3.9 Defense Mechanisms For A Gradient Attack

By implementing the following defense mechanisms at different levels, federated learning systems can mitigate the risk of gradient attacks, safeguard the learning process's integrity, and protect individual user data's privacy.

The first category is the Data-Level Defense, which involves filtering data, removing or filtering malicious or abnormal data points before training the model. For example, if a user's data contains extreme values or outliers that significantly deviate from the normal range, those data points can be excluded from the training process.

Model-Level Defense is another type of defense against gradient attack.

Adding regularization techniques to the model training process prevents overfitting and increases robustness. Regularization introduces penalties for complex models or large parameter values, discouraging the model from fitting too closely to individual malicious gradients.

Communication-Level Defense is another defense mechanism against gradient attacks, focusing on applying differential privacy mechanisms to the communication between the central server and the participating users. This protects individual user privacy by adding noise to the aggregated gradients, making it harder for an attacker to infer specific user information from the gradients.

While defense mechanisms against gradient attacks in federated learning offer protection, they also have drawbacks. Here are the potential drawbacks of these defense mechanisms:

The Data-Level Defense excludes certain data points that can lead to a reduction in the available training data, potentially limiting the overall model's performance. It may also introduce bias if the excluded data points contain valuable information.

In a data Aggregation defense scheme, aggregating data from multiple users can increase the communication overhead and computational cost, especially when dealing with many participants. Additionally, if the aggregation process is not properly secured, it can become a potential attack target.

In a Model-Level Defense scheme, overreliance on regularization can result in underfitting, where the model fails to capture important patterns in the data. Balancing regularization to prevent overfitting without sacrificing performance can be challenging.

Applying strict gradient clipping thresholds may lead to information loss and hinder learning. Setting the clipping threshold too high may still allow manipulative gradients to impact the model.

The drawback of the Communication-Level Defense is that adding noise to the gradients can reduce the model's accuracy due to the introduction of random perturbations. Fine-tuning the noise level to balance privacy and utility can be complex.

Implementing secure aggregation protocols can introduce additional computational overhead and communication latency, impacting the efficiency of the federated learning system.

3.10 The Communication Overhead Problem In Federated Learning

The central server must constantly exchange information with the participating clients during training in Federated learning. This constant exchange of information between the central server and the participants creates communication overhead problems.

During training in federated learning, it is vital to establish effective communication between the central server and participating clients. However, the constant information exchange between them results in communication overhead problems, which worsen when defense schemes are implemented.

Defense mechanisms often require transmitting additional data or information between the clients and the central server. This can lead to increased data transfer requirements, especially when dealing with many clients or when the defense mechanisms exchange model parameters, gradients, or aggregated results. The increased data transfer can strain network bandwidth and increase the time required for communication. Certain defense mechanisms, such as secure aggregation or differential privacy, may involve encryption and decryption operations to protect the privacy and security of the data and gradients. These cryptographic operations can impose computational overhead on the clients and the central server, as encryption and decryption algorithms can be resource-intensive and time-consuming.

Many Defense schemes require synchronization and coordination between the clients and the central server. This involves establishing communication channels, coordinating training iterations, exchanging gradients or model updates, and

aggregating results. Synchronization and coordination efforts can introduce delays and overhead, especially when clients have varying computation capabilities, communication delays, or unstable connections.

As the number of participating clients increases, the communication overhead can become a scalability challenge. Coordinating and aggregating gradients or model updates from many clients can significantly increase the computational and communication requirements, making it harder to perform the necessary operations within a reasonable time frame efficiently.

Managing the communication overhead is crucial in federated learning to ensure smooth collaboration between the clients and the central server while minimizing delays and resource consumption. Optimizations, such as compression techniques, selective updates, and parallelization, can be explored to mitigate these communication challenges and improve the overall efficiency of the defense schemes in federated learning. The upcoming chapter emphasizes the communication overhead of the text verification scheme, illustrating its capability to minimize communication overhead in federated learning defense mechanisms.

Chapter Summary

This new participant authentication mechanism ensures that only legitimate participants contribute to the federated learning process. The verification key scheme authenticates participants before they are allowed to join the federated learning system, preventing unauthorized or malicious participants from injecting polluted data into the federated learn-ing model. The verification scheme presented in this paper focuses on preventing data poisoning rather than detecting any attack on federated learning after an attack. This approach allows the training server to eliminate the infected client, improving the overall performance of federated learn-ing training. Moreover, it can prevent data modification in each client's storage during each training. It allows the server to detect any modification in real time and ensures the confidentiality and integrity of the data exchanged between participants and the training server.

Chapter 4

4 Communication Overheads

This chapter focuses on the basic concepts of federated learning and communication reduction strategies. Federated learning has emerged from distributed deep learning and enables the training of a shared model without jeopardizing the privacy of users' data. Throughout the learning process, the data remains on local devices.

With the training data, clients exchange their local models to enhance a global model.

The FedAVG algorithm sketches how the Federated Averaging algorithm [22] proceeds using a cluster of clients, each with a learning rate of $\boldsymbol{n}$.

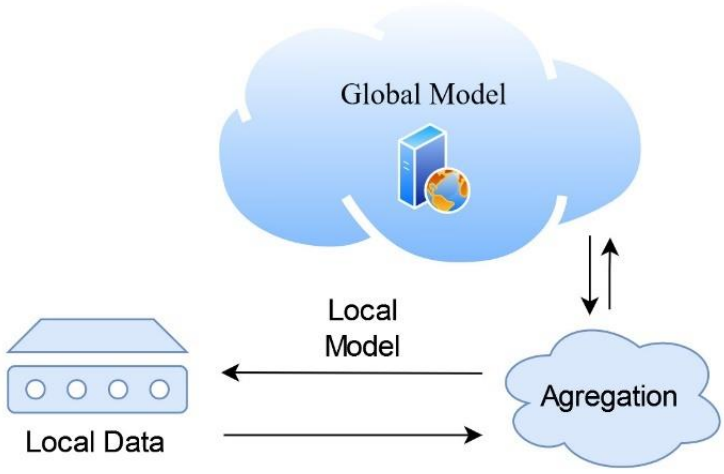

Figure 10: Federated learning architecture

In FedAVG algorithm

N represents a cluster of clients.

n represents the learning rate

S designates the set containing all clients

C represents a subset of selected clients from S

$$|S'| = (C.N)$$

m represents the number of randomly chosen clients where m is the maximum between $(C.N)$ and 1.

e represents the number of local iterations

E is the communication interval

P_k represents the wight of k^{th} client.

The FedAVG acts in two synchronous steps, starting by generating a global model represented by w_0 on the server.

4-1: Algorithm FedAVG

Algorithm FedAVG The federated averaging algorithm.

Initialize (w_0);
for $t = 1, \dots, T$ **do**
  $m \leftarrow \max(C.N, 1)$;
  $S' \leftarrow random\_Pick(S, m)$;
  for all clients $k \in S'$ In parallel **do**
    for $e \in 1, \dots, E$ **do**
      $w_e \leftarrow w_{e-1} - n\nabla F(w_{e-1})$;
    end for
    $w_{t+1}^k \leftarrow w_e$;
  end for
  $w_{t+1} \leftarrow \sum_{k=0}^{m} P_k \cdot w_{t+1}^k \,/\, m$;
end for

After generating a global model, 4-1: Algorithm FedAVG, Randomly chooses m participating clients where m is the maximum between $(C.N)$ and 1.

During the training, each of the selected clients trains a local model similar to the global model during several local iterations represented by $e = 1, \dots, E$, and E is the communication interval. Once the training has been completed, all clients will send their local models to the server to update the global model. The whole process is executed repeatedly during T iterations.

4.1 Communication Overhead In Distributed Deep Learning

The Federated learning workflow introduces a substantial load of communication which leads to a decrease in the efficiency and applicability of Federated learning. Currently, two main approaches exist for communication overhead reduction. The first approach is data compression, and the second is decreasing communication rounds (see Figure 11: Communication overhead reduction: taxonomy of the techniques.).

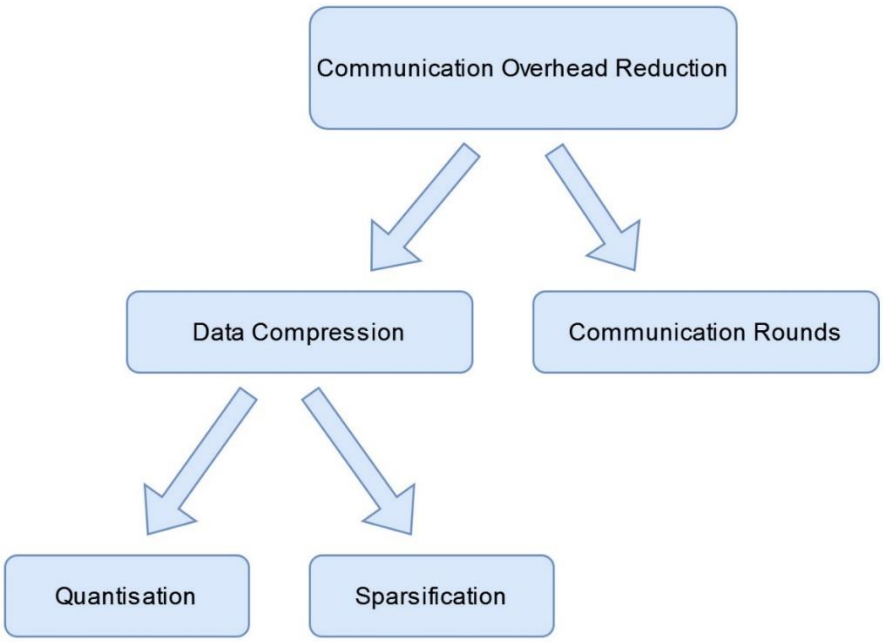

Figure 11: Communication overhead reduction: taxonomy of the techniques.

The data compression approach consists of two parts of quantification and sparsification.

Quantization involves representing the data with a low-precision or small-sized data type, such as a Boolean.

In contrast, sparsification transmits only each communication's essential values, constituting approximately 1% of the overall values.

However, the compression rate for quantization is limited to 1/32, considering that 32-bit-encoded data is commonly used in DDL, and models that use quantization may have a slower convergence due to having fewer bits to carry the information.

On the other hand, sparsification can achieve a compression rate of 1/100 without significantly affecting the model's convergence speed and final accuracy. Nevertheless, sparsification comes with additional training phases, such as sampling, compression, coding, and decoding, which can impact the overall training efficiency, particularly in battery-sensitive (e.g., smartphones) and low-performance (e.g., netbooks) devices.

Regarding the second technique for reducing communication, in standard Federated Learning (FL), the communication process occurs at the end of each iteration

($\boldsymbol{E} = \mathbf{1}$).

Since a typical FL training of a deep neural network involves hundreds of thousands of iterations, increasing the communication intervals could decrease the communication overhead.

As a solution, the FedAvg algorithm and its variations enable clients to perform several iterations of local training before updating the global model.

This approach increases convergence speed when the communication rounds are reduced. In FedAvg, hyperparameter $\boldsymbol{E}$ adjusts the communication, influencing the model's accuracy and training efficiency trade-off.

A smaller $\boldsymbol{E}$ value usually results in better final accuracy, whereas a larger $\boldsymbol{E}$ value speeds up the model's convergence. Thus, experts must fine-tune the communication interval $\boldsymbol{E}$ to enable the model to achieve the highest possible efficacy.

4.2 The FL-COP Modelling and Formulation

The FL-COP Modelling and Formulation involves optimizing two conflicting objectives:

(I) Minimizing the communication overhead and

(II) Maximizing the model's accuracy.

The FL-COP model assumes that all clients in the system have a similar model to the one on the server, including the nodes, connections, layers, and activation functions.

In an architecture consisting of one server and $\boldsymbol{N}$ clients, the trained model has l layers, with each layer $\boldsymbol{L_i}$ containing $\boldsymbol{N_i}$ Weights.

The FL-COP approach involves a four-level communication-reduction scheme, where each layer represents a specific communication-reduction approach.

1. The top-level determines the number of clients participating in the training of the global model.
2. Quantization [24]
3. Sparsification [25]
4. Reducing communication rounds (as shown in Figure 12: The FL-COP modeling levels).

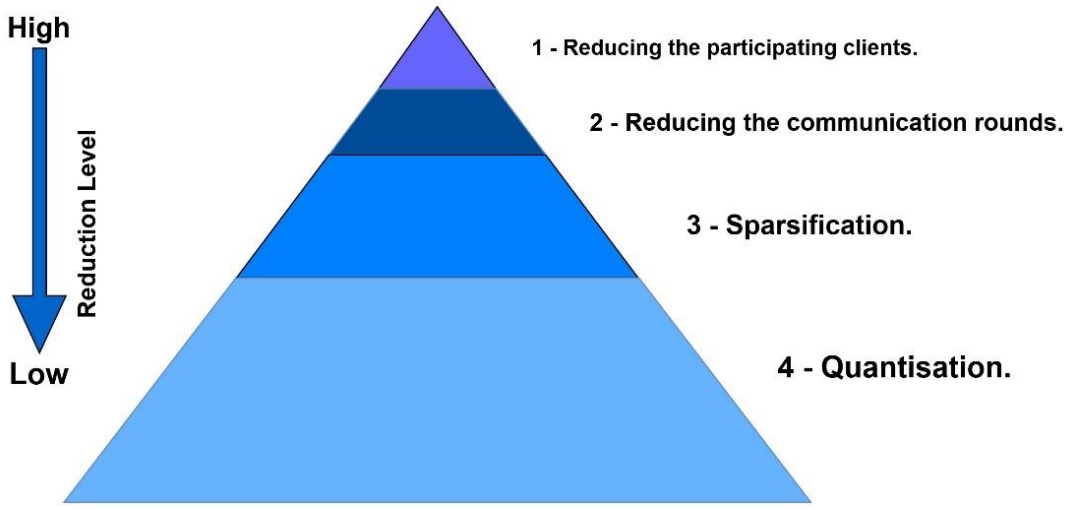

Figure 12: The FL-COP modeling levels

The amount of communication during the FL learning process depends on the number of participating clients represented by $\boldsymbol{m}$, which is between $\mathbf{1}$ and $\boldsymbol{N}$.

The FL-COP model has two main parts.

1. The first part involves randomly selecting a certain number of clients, represented by m, from the entire set of clients represented by S, who will be the only ones sending their local models to the server.
2. The second part involves determining the number of training iterations, represented by $E \in [1,100]$, after which all clients will send their local models to the server. This variable determines the number of training steps the clients perform before sending their local models. Note that each client has a maximum number of training iterations ($E \cdot T$), where T is the maximum number of times clients can send their local models to the server. The third part of the problem involves selecting a percentage value between 0% and

50% for each layer L_i, which represents the percentage of having n_i Weights that will not be sent to the server this percentage is represented by μ.

The final part of FL-COP modeling consists in finding the **optimal number** of bits b_i allocated to encode the weights of each layer L_i in the model, where $i = 1, \dots, l$.

$\overline{w}_i$: represents the maximum values of i^{th} layer.

ϱ_i: represent the minimum values of i^{th} layer.

b_i: bits means that 2^{b_i} Binary combinations can be created.

The next step is to assign the all-ones and all-zeros combinations to encode the $\overline{w}_i$ and ϱ_i Values, respectively.

The $(2^{b_i} - 2)$ remaining combinations will encode $(2^{b_i} - 2)$ values that are equally drawn from the interval $[\overline{w}_i, \varrho_i]$. The information that will be transmitted to the server comprises a sequence of permutations that represent each weight. The server will utilize the inverse process to recover the complete and accurate weights.

Each client will send to the server $\sum_{i=1}^{l}(n_i . b_i) + 64$ bits instead of $\Theta = \sum_{i=1}^{l}(n_i . 32)$

Original bits.

Equation (1) defines the first objective function $f_1(\overrightarrow{X})$.

This function calculates the percentage of data reduction that the solution $\overrightarrow{X}$ Achieves, and it is the sum of the percentage α and $\beta \in [0,1]$ of data sent and received by all the clients together from and to the server. These percentages are indicated with

the initial data that would have been transmitted or received if no reduction in communication ($T \,.\, N \,.\, \Theta$) had been implemented.

Equation (2) defines the second objective function $f_2(\overrightarrow{X})$. This function evaluates the accuracy of the global model w_T^* at communication T (.ie the last iteration achieved via the solution $\overrightarrow{X}$. The server's model $w_T^* = \sum_{k=0}^{m} w_T^k/m$, is computed as the mean of the m local models obtained after T communications, while the accuracy if computed as the division of λ by v, where λ and v are the number of correct and total predictions made using the model w_T^*, respectively.

$$\min_{\overrightarrow{X}=\{x_1,\dots,x_d\}} f_1(\overrightarrow{X}) = \frac{\alpha+\beta}{2} \tag{1}$$

$$\max_{\overrightarrow{X}=\{x_1,\dots,x_d\}} f_2(\overrightarrow{X}) = \frac{\lambda}{v} \tag{2}$$

Where:

$$\alpha = \frac{1}{E} \cdot \frac{m}{N} \tag{3}$$

$$\beta = \frac{m}{N} \cdot \frac{1}{E} \cdot \sum_{i=l}^{l} \frac{b_i}{32} \cdot \frac{100-\mu_i}{100} \cdot \frac{n_i}{\sum_{j=1}^{l} n_j} \tag{4}$$

Subject to:

$$m, E, \mu_i, b_i \in \mathcal{N}, 1 \leq m \leq N, 1 \leq E \leq 1000, 0 \leq \mu_i \leq 50, 1 \leq b_i \leq 32$$

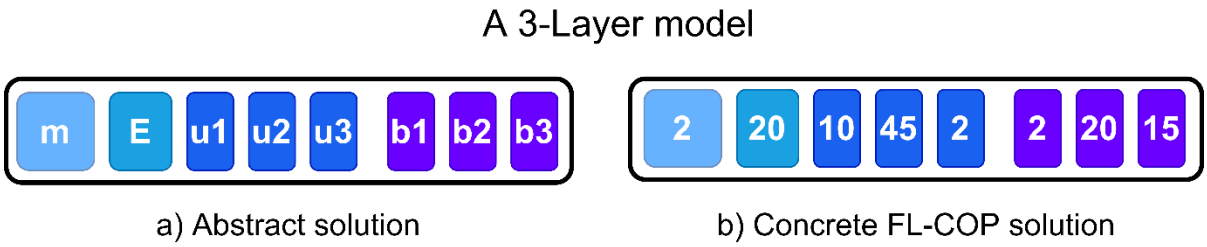

Figure 13: A 3 - Layer model.

The above figure on the left side sketches a solution $\overrightarrow{X}$ Of an Fl-COP that trains a $l = 3$ layers mode. The figure on the right side represents a concrete solution $\overrightarrow{X}$ For the same configuration using 20 training iterations and 3 rounds of client-server communications. During each round, only 90% of the wights of the first layer and 55% of the 2nd later are sent to the server. During the 3rd round, 98% of the weights are sent to the server. The weights transmitted from the 1-3rd layers are encoded using 2,20, and 15 bits.

4.3 The Communication-Overhead Reduction

We assume we want to implement algorithms 1 and 2 in the NSGA-II algorithm. The first step of the implementation is to decide in which training phase we must implement the algorithms to minimize the communication overhead.

Currently, two main approaches exist for communication overhead reduction. The first approach is data compression, the second is decreasing communication rounds, and the FL-COP approach involves a four-level communication-reduction scheme, where each layer represents a specific communication-reduction approach.

FL-COP approach, the top level determines the number of clients participating in the training of the global model, and the bottom levels apply quantization [24], sparsification [25], and reduction of communication rounds, as demonstrated in Figure 1: The attacker has modified the cat image cat-4.

The best option for implementing the key verification scheme is the top level that determines the number of clients participating in the training of the global model.

At the top level, the number of clients is reduced to the minimum, and by optimizing this level, we can eliminate any attacking client. The top level allows the training sever to supervise and verify the authenticity of the training data of each client. If it detects any affected client, it can eliminate that client and replace it with another client with legitimate training data.

The NSGA-II algorithm [26] starts by randomly initializing a population of U individuals, let $\overrightarrow{X} = \{x_i, \dots, x_d\}$, where $i \in [1, d]$ and d is the size of the problem to be

solved. In the next step, NSGA-II enters a loop until some stopping criterion is satisfied.

In the loop, binary tournament selection, crossover, and mutation generate a Q of U offspring population.

The following objective function represents equation 1 of the NSGA-II algorithm.

$$\min_{\vec{X}=\{x_1,\dots,x_d\}} f_1(\vec{X}) = \frac{\alpha+\beta}{2} \qquad (1)$$

The function f_1 calculates the percentage of data reduction that the solution $\vec{X}$ Achieves.

α: represents the percentage of data **sent** from a **server to all clients**, with no communication reduction.

β: represents the percentage of data **the server receives from all clients**, with no communication reduction.

N: a cluster of clients, each with a learning rate of η

T: is the number of the executed iterations

The complexity f_1 can be represented by $(T \cdot N \cdot \Theta)$

Θ: represents the number of original bits that each client would not send to the server during the training initially [22]

$$\Theta = \sum_{i=1}^{l} (n_i . 32)$$

However, after implementing the FL-COP algorithm, it will change to

$$\Theta = \sum_{i=1}^{l} (n_i . b_i) + 64$$

Now let us assume that in each client storage, there are an equal number of data files, and we want to verify 64 bits of each data file, and the length of each verification key file is 64 bits if we want to verify every client. Initially, each client must transmit all the key files to the local training server.

Let

δ: Represent the number of data files in each client's storage.

With this assumption, in the verification phase, each remote client will transmit δ, the number of key files to the server, and if each key file is 64 bits, each client will transmit $\delta \times 64$ bits to the training server.

If there are N clusters of clients, $N \times \delta \times 64$ extra bits will transmit to the local training server.

Once the local training server receives the key files, it will use the private decryption key to decrypt every verification key file, and if it decides to verify every single key file, it will request $N \times \delta \times 64$ bits from each remote client.

In the next phase, each client will transmit $N \times \delta \times 64$ bits from their local data files to the server, and the server will use that data to verify the authenticity of the local data on every client.

The following formula demonstrates the number of bits transmitted from and to the local training server.

$$3 \times N \times \delta \times 64 \text{ bits}$$

Now let us denote the length of the verification key files by y. The total number of bits that will be transmitted in a worst-case scenario will be

$$3 \times N \times \delta \times y \text{ bits}$$

Chapter Summary

Chapter 4 delved into discussing communication overheads linked to the key verification scheme. Additionally, the chapter included a comprehensive demonstration showcasing the scalability of the scheme. This analysis shed light on the potential challenges and benefits of implementing the key verification scheme in various scenarios. It emphasized the importance of understanding the communication requirements and scalability factors when deploying such a system.

Chapter 5

5 Results

In this section, we describe the experiments we conducted for the two proposed algorithms and discuss the results obtained.

We used Google Colaboratory "Colab" and TensorFlow to implement the first two algorithms. In the first experiment, a sample text file was given into the program, and a sample key file was created by running the first algorithm. The second algorithm was used to verify that file using the verification key in the next step.

After 100 execution, the second algorithm verified the original file using the key file with a success rate of 100%.

```
w0Grf353#43                          ----  Iteration password matched
e9808a71bcea8809ec6a630c976847a4     ----  Key file hasher matched

Reading Byte Values and locations from the key file:
Reading Byte Value: x from the Keyfile
Byte location in the keyfile is: 32
Byte value from the original file is: x

The byte value: x is equal to x                ----- verified
Reading Byte Value: i from the Keyfile
Byte location in the keyfile is: 36
Byte value from the original file is: i

The byte value: i is equal to i                ----- verified
Reading Byte Value: d from the Keyfile
Byte location in the keyfile is: 10
Byte value from the original file is: d

The byte value: d is equal to d                ----- verified
Reading Byte Value: h from the Keyfile
Byte location in the keyfile is: 14
Byte value from the original file is: h

The byte value: h is equal to h                ----- verified
 ***** KeyFile is verified *****
```

Figure 14: Result of successful verification of a legitimate file

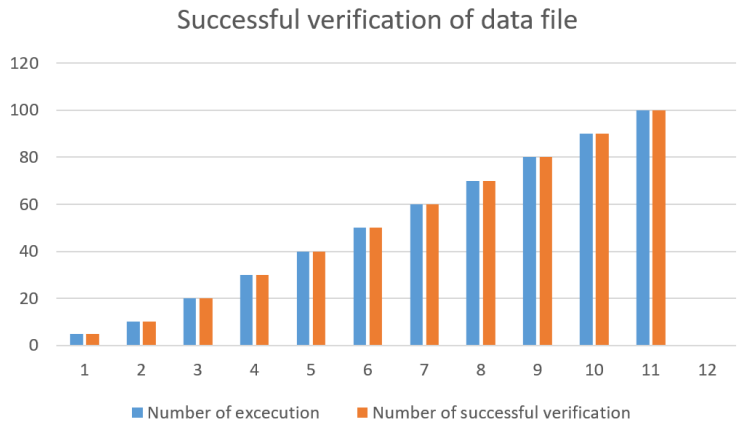

Figure 15: Successful verification of data file

During the later phase of the data file verification process, after conducting 100 iterations, the second algorithm effectively identified the inconsistency and concluded that the data file had been compromised.

```
w0Grf353#43                      ----  Iteration password matched
1612f797418a53dc652d385bda0e014f ----  Key file hasher does not match
e9808a71bcea8809ec6a630c976847a4 !!!!!!!!!!!!!!!

        ----  The Original file has been modified ----
        ----  Compromised Node detected ----
        ----  Terminating the training ----
```

Figure 16: Algorithm 2 detects a file modification

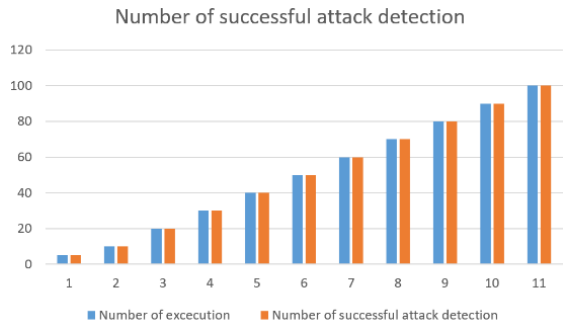

Figure 17: Algorithm 2 was executed 100 times

The key file consists of an iteration password to safeguard against man-in-the-middle attacks and a hash value representing the data file. It is extremely challenging for an attacker to modify the data file in a manner that would produce the exact original hash value.

In the subsequent phase, the possibility of the attacker manipulating the original data file to generate an identical hash value was considered. The only feasible method for the attacker to modify the data file without being detected by the key file is by altering the characters that are absent in the key file. The provided formula quantifies the statistical likelihood of the attacker's achievement, assuming that they can modify the original file to yield the same hash value.

$$Probability = \frac{Number\ of\ desired\ outcomes}{Number\ of\ possible\ outcomes}$$

$$Probability = \frac{50}{100} = 0.5$$

The probability of choosing 7 characters that were not recorded in the key file is equal to

$$= \frac{50}{100} \times \frac{50}{100} \times \frac{50}{100} \times \frac{50}{100} \times \frac{50}{100} \times \frac{50}{100} \times \frac{50}{100}$$

$$= 0.5 \times 0.5 \times 0.5 \times 0.5 \times 0.5 \times 0.5 \times 0.5$$

$$= (0.5)^7$$

$$= 0.007813$$

Let us assume the data file contains 100 characters, and the key file has backed up 50 from the original. This means that if the attacker decides to change a single character, they would have a probability of 50/100 to select a character that is not recorded in the key file.

The chance of a successful attack in this scenario is 0.78%, with the assumption that the attacker can modify the data file in a way that would create the same hash checksum.

In this scenario, the probability of a successful attack is 0.78% when assuming that the attacker can modify the data file in a way that generates the same hash checksum [1].

Chapter Summary

This chapter highlighted the significant security issues presented in most federated learning models and emphasized the need for a data verification method specifically designed for federated learning.

To address this issue, a robust prevention scheme was introduced that integrates an encrypted verification mechanism into the federated learning model, enabling real-time elimination of infected participants and prevention of backdoor attacks.

Three algorithms generated verification keys for federated learning data files on a training node. These algorithms are implemented using TensorFlow on Google Colaboratory "Colab." Through experimentation, Chapter 5 demonstrates that it is highly challenging for an attacker to modify a data file and produce

The primary objective of this novel verification scheme is to proactively prevent data poisoning in federated learning rather than merely detecting attacks post-incident. By adopting this approach, the training server gains the capability to promptly remove infected clients, thereby enhancing the overall performance of federated learning

training. Additionally, this scheme effectively safeguards against data modification within each client's storage throughout the training process, enabling real-time detection of any unauthorized alterations by the server.

Chapter 6

6 Novelty

This novel method of participant authentication ensures that only valid contributors participate in the federated learning process. The verification key system verifies participants before their inclusion in the federated learning system, effectively blocking unauthorized or malicious contributors from introducing corrupt data into the federated learning model. The presented verification scheme in this paper emphasizes preventing data corruption rather than merely identifying attacks on federated learning post-occurrence. This approach enables the training server to remove compromised clients, thereby enhancing the overall performance of federated learning. Additionally, it safeguards against data alterations in each client's storage throughout training, enabling real-time detection of any modifications by the server. This process ensures the confidentiality and integrity of the data exchanged between participants and the training server.

This research paper accomplished two key objectives in federated machine learning. Firstly, it ensured data storage integrity by implementing a verification process to detect changes to the original data. Secondly, it prevented attackers from modifying the data in successive rounds, effectively countering adaptive data poisoning attacks.

In federated learning, there is no guarantee that client devices will have sufficient resources for multiple communications, supervision, and strong encryption. To address this, the proposed scheme reduced the communication overhead between the central server and multiple participants. The experiment conducted in Chapter 5 demonstrated that this scheme could promptly detect and mitigate real-time attacks, ensuring system security without significant delays. Moreover, the scalability of this new verification scheme was demonstrated, showcasing its ability to withstand sophisticated attacks and adapt to address emerging vulnerabilities, thereby highlighting its robustness.

By adopting this approach, the training server gains the capability to swiftly eliminate infected clients, thereby enhancing the overall performance of federated learning training. Additionally, this scheme effectively safeguards against unauthorized data modifications within each client's storage throughout the training process, enabling the server to detect such alterations in real time.

Chapter 7

7 Conclusion

This paper introduced a robust prevention scheme to enhance the security of federated learning. The server can swiftly eliminate infected participants and prevent backdoor attacks by incorporating an encrypted verification scheme into the federated learning model. This scheme utilizes a separate key file for verification.

To implement this scheme, three algorithms were introduced that generate encrypted and decrypted verification keys for federated learning data files on a training node. These algorithms play a crucial role in ensuring the integrity of the data and preventing unauthorized modifications.

Experimental results demonstrated the scheme's effectiveness in preventing attackers from modifying data files undetected. It became extremely challenging for an attacker to modify the data file while producing the same hash value. Subsequent experiments showed that the success rate of such attacks could be less than 1%, turning it into a game of chance for the attacker.

The main contribution of this paper lies in the detection prevention scheme that safeguards against data modification in client storage. This scheme particularly benefits low-processing devices by requiring minimal processing power. It empowers the server to detect real-time modifications and eliminate infected clients.

This paper introduces a detection prevention system for image and text data stored by clients in federated learning. This system is advantageous for low-processing devices as it imposes a low processing burden. It enables real-time detection of modifications and facilitates the removal of infected clients by the server.

Chapter 8

8 Future Work

Future research endeavors for this study involved assessing the communication and computational overhead associated with the proposed verification scheme and exploring avenues to optimize its performance. By conducting a comprehensive analysis of the communication requirements and computational resources required by the verification scheme, potential areas for improvement can be identified. This includes investigating strategies to reduce the communication overhead between the server and clients and optimizing the computational processes involved in the verification scheme. By addressing these aspects, the overall efficiency and effectiveness of the verification scheme can be enhanced, leading to improved performance and scalability in federated learning scenarios.

References

[1] G. Sun, Y. Cong, J. Dong, Q. Wang, L. Lyu and J. Liu, "Data Poisoning Attacks on Federated Machine Learning," in IEEE Internet of Things Journal, doi: 10.1109/JIOT.2021.3128646.

[2] A. K. Singh, A. Blanco-Justicia, J. Domingo-Ferrer, D. Sánchez and D. Rebollo-Monedero, "Fair Detection of Poisoning Attacks in Federated Learning," 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI), 2020, pp. 224-229, doi: 10.1109/ICTAI50040.2020.00044.

[3] R. Doku and D. B. Rawat, "Mitigating Data Poisoning Attacks On a Federated Learning-Edge Computing Network," 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC), 2021, pp. 1-6, doi: 10.1109/CCNC49032.2021.9369581.

[4] Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D. & Shmatikov, V.. (2020). How To Backdoor Federated Learning. Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, in Proceedings of Machine Learning Research 108:2938-2948 Available from https://proceedings.mlr.press/v108/bagdasaryan20a.html.

[5] Steinhardt, J., Koh, P. W. W., & Liang, P. S. (2017). Certified defenses for data poisoning attacks. Advances in neural information processing systems, 30.

[6] Blanchard, P., El Mhamdi, E. M., Guerraoui, R., & Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. Advances in Neural Information Processing Systems, 30.

[7] El El Mhamdi, M., Guerraoui, R., & Rouault, S. (2018). The Hidden Vulnerability of Distributed Learning in Byzantium. arXiv e-prints, arXiv-1802.

[8] Lyu, L., Yu, H., & Yang, Q. (2020). Threats to federated learning: A survey. arXiv preprint arXiv:2003.02133.

[9] Fan, X., Ma, Y., Dai, Z., Jing, W., Tan, C., & Low, B. K. H. (2021). Fault-tolerant federated reinforcement learning with a theoretical guarantee. Advances in Neural Information Processing Systems, 34.

[10] Xu, H., Kostopoulou, K., Dutta, A., Li, X., Ntoulas, A., & Kalnis, P. (2021). DeepReduce: A Sparse-tensor Communication Framework for Federated Deep Learning. Advances in Neural Information Processing Systems, 34, 21150-21163.

[11] Jin, X., Chen, P. Y., Hsu, C. Y., Yu, C. M., & Chen, T. (2021). Catastrophic Data Leakage in Vertical Federated Learning. Advances in Neural Information Processing Systems, 34.

[12] Huang, Y., Gupta, S., Song, Z., Li, K., & Arora, S. (2021). Evaluating gradient inversion attacks and defenses in federated learning. Advances in Neural Information Processing Systems, 34.

[13] Lyu, L., Yu, H., Ma, X., Sun, L., Zhao, J., Yang, Q., & Yu, P. S. (2020). Privacy and robustness in federated learning: Attacks and defenses. arXiv preprint arXiv:2012.06337.

[14] Liu, P., Xu, X., & Wang, W. (2022). Threats, attacks, and defenses to federated learning: issues, taxonomy, and perspectives. Cybersecurity, 5(1), 1-19.

[15] Lee, H., Kim, J., Ahn, S., Hussain, R., Cho, S., & Son, J. (2021). Digestive neural networks: A novel defense strategy against inference attacks in federated learning. computers & security, 109, 102378.

[16] Ozdayi, M. S., Kantarcioglu, M., & Gel, Y. R. (2020). Defending against backdoors in federated learning with robust learning rate. arXiv preprint arXiv:2007.03767.

[17] Lai, J., Huang, X., Gao, X., Xia, C., & Hua, J. (2022). GAN-Based Information Leakage Attack Detection in Federated Learning. Security and Communication Networks, 2022.

[18] Zhu, L., Liu, Z., & Han, S. (2019). Serious leakage from gradients. Advances in Neural Information Processing Systems, 32.

[19] J. Chen, J. Zhang, Y. Zhao, H. Han, K. Zhu, and B. Chen, "Beyond Model-Level Membership Privacy Leakage: an Adversarial Approach in Federated Learning," 2020 29th International Conference on Computer Communications and Networks (ICCCN), 2020, pp. 1-9, doi: 10.1109/ICCCN49398.2020.920974

[20] Lo, S.K., Lu, Q., Wang, C., Paik, H.Y., Zhu, L., 2021. A systematic literature review on federated machine learning: From a software engineering perspective. ACM Comput. Surv. 54.

[21] Wu, C., Wu, F., Liu, L., Huang, Y., & Xie, X. (2022). Communication-efficient federated learning via knowledge distillation. Nature communications, 13(1), 2032.

[22] Œngel Morell, J., Abdelmoiz Dahi, Z., Chicano, F., Luque, G., & Alba, E. (2022). Optimising **Communication Overhead** in Federated Learning Using NSGA-II. arXiv e-prints, arXiv-2204.

[23] McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017, April). Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics (pp. 1273-1282). PMLR.

[24] Alistarh, D., Grubic, D., Li, J.Z., Tomioka, R., Vojnovic, M.: QSGD: Communication-efficient SGD via gradient quantization and encoding. In: Proproceedings of the 31st International Conference on Neural Information Processing Systems. p. 1707–1718. NIPS'17 (2017)

[25] Wangni, J., Wang, J., Liu, J., Zhang, T.: Gradient sparsication for Communication-efficient distributed optimization. In: Proceedings of 32nd International Conference on Neural Information Processing Systems. p. 1306–1316 (2018)

[26] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective Genetic algorithm: NSGA-II. IEEE Trans. on Ev. Comp. 6(2), 182–197 (2002)

[27] Zeng, R., Zeng, C., Wang, X., Li, B., & Chu, X. (2021). A comprehensive survey of an incentive mechanism for federated learning. arXiv preprint arXiv:2106.15406.

[28] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated Learning for Mobile Keyboard Prediction," arXiv preprint arXiv:1811.03604, 2018.

[29] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated Machine Learning: Concept and Applications," ACM Transactions on Intelligent Systems and Technology (TIST), vol. 10, no. 2, pp. 1201-1215, 2019.

[30] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond Inferring Class Representatives: User-level Privacy Leakage From Federated Learning," in Proc. of IEEE INFOCOM, 2019.

[31] Jodayree, M., He, W., & Janicki, R. (2023). Preventing Image Data Poisoning Attacks in Federated Machine Learning by an Encrypted Verification Key. 27th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2023).

[32] Jodayree, M., He, W., & Janicki, R. (2023). Preventing Text Data Poisoning Attacks in Federated Machine Learning by an Encrypted Verification Key. IJCRS 2023 International Joint Conference on Rough Sets.