Technische Universität München
Professur für Coding and Cryptography
Prof. Dr.-Ing. Antonia Wachter-Zeh

Master's Thesis

Empirical Analysis of Robust Federated Learning Under Differential Privacy

Vorgelegt von:
Sedat Çoban

München, June 2025

Betreut von:
M.Sc. Marvin Xhemrishi, M.Sc. Luis Maßny

Master's Thesis an der
Professur für Coding and Cryptography (COD)
der Technischen Universität München (TUM)
Titel : Empirical Analysis of Robust Federated Learning Under Differential Privacy
Autor : Sedat Çoban

Sedat Çoban
Constanze-Hallgarten Str. 6
81379 München
sedat.coban@tum.de

Ich versichere hiermit wahrheitsgemäß, die Arbeit bis auf die dem Aufgabensteller bereits bekannte Hilfe selbständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderung entnommen wurde.

München, 27.06.2025 .....

Ort, Datum (Sedat Çoban)

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence. 3
CDP	Central Differential Privacy. 26
CNN	Convolutional Neural Network. 3, 30, 56, 57
DNN	Deep Neural Network. 3
FedAvg	Federated Averaging. 4, 10, 16, 33, 34, 35, 36, 37, 38
FL	Federated Learning. iii, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 29, 56, 65, 69
IID	Independent and Identically Distributed. iii, iv, vii, 6, 11, 12, 29, 33, 35, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 48, 50, 52, 53, 54, 55, 56, 57, 59, 65, 66, 67
IPM	Inner Product Manipulation. iii, iv, 4, 5, 13, 14, 15, 31, 33, 34, 35, 36, 38, 41, 42, 43, 44, 45, 48, 49, 51, 52, 53, 58, 60, 61, 63, 65, 66, 67, 68
LDP	Local Differential Privacy. iii, iv, v, vii, 5, 6, 26, 27, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 65, 66, 67, 68, 69
LF	Label Flipping. iii, iv, v, vii, 4, 5, 12, 13, 31, 33, 34, 35, 36, 37, 38, 39, 43, 48, 50, 53, 54, 55, 56, 58, 59, 60, 61, 62, 63, 65, 67, 68
MKrum	Multi-Krum. iv, 5, 17, 24, 33, 34, 35, 36, 37, 38, 39, 52, 53, 54, 55, 58, 62, 63, 65, 66, 67, 68

ML	Machine Learning. 3, 4, 6, 7, 9, 24, 25
non-IID	Non-independent and Identically Distributed. iii, iv, v, vii, 6, 11, 12, 29, 33, 34, 35, 36, 37, 38, 39, 47, 48, 49, 50, 51, 58, 59, 60, 62, 63, 65, 66, 67, 68, 69
PCA	Principal Component Analysis. iii, iv, v, 6, 7, 27, 28, 32, 41, 42, 43, 44, 45, 49, 50, 53, 54, 62, 68
SGD	Stochastic Gradient Descent. 8, 30

Abstract

Federated Learning (FL) usage has increased in recent years due to its decentralized nature and privacy-preserving structure. Nevertheless, FL deployments face a variety of challenges, in particular, rigorous privacy guarantees and robustness to Byzantine adversaries. Local Differential Privacy (LDP) and robust aggregation methods have been widely studied to solve these two problem independently. However, the effect of LDP on state-of-the-art robust aggregation methods is not well understood.

In this thesis, the effect of LDP on the various robust aggregation methods in an FL setting is investigated. The performance of ten different robust aggregation methods (Krum, Multi-Krum (MKrum), Trimmed Mean, Geometric Median, Median, Centered Clipping, Clustering, Clipped Clustering, CosDefense, and Bulyan) is analyzed. Each robust aggregation method is evaluated with and without LDP under Independent and Identically Distributed (IID) and Non-independent and Identically Distributed (Non-IID) distributions. Four different attack scenarios: No-attack, Random Gaussian attack, Inner Product Manipulation (IPM) attack, and Label Flipping (LF) are tested. All evaluations are carried out on the MNIST and CIFAR-10 benchmark datasets.

From the experiments, it is concluded that most robust aggregation methods under LDP cannot provide protection against adversarial attacks since LDP breaks down their underlying principles. In particular, Centered Clipping and Median robust aggregation methods cannot achieve high performance under LDP even without attack. From the analysis, it is concluded that the reason behind this performance degradation for Centered Clipping is a double clipping effect from the LDP and the method itself. The leading reason behind Median's collapse is choosing coordinates from each of the client models uniformly. On the other hand, Trimmed Mean and Geometric Median robust aggregation methods provide protection against adversarial attacks under LDP.

1 Introduction

Machine Learning (ML) is a sub-branch of Artificial Intelligence (AI) that uses past observations or datasets as examples to automatically learn the patterns within the datasets and create models that can be used to make predictions on unseen data. In recent years, the use of ML has grown significantly. The introduction of more advanced models such as Deep Neural Network (DNN) or Convolutional Neural Network (CNN), alongside increased computational power, drives this growth. This growth is particularly visible in academia and industry where ML is used in a wide range of applications such as image classification, text analysis and beyond. With these improvements, new issues were raised. The need for comprehensive datasets in areas such as personal health records and financial records to train modern ML models raises data security and privacy issues since the data needs to be shared with the server. These issues directed ML experts to find more private and secure approaches [LBH15].

Federated Learning (FL) is a ML paradigm that was introduced by McMahan et al. to address these data security and privacy issues. It enables training ML models collaboratively with multiple entities (clients) without sharing local data with a central server. It ensures that each client keeps their local data on its device and does not share it with the server. Only the intermediate model parameters such as, gradients or weights, are transmitted to the central server. The central server uses a specific aggregation rule to create a new global model and transmits it to the clients. A typical FL training process starts with the client selection to participate in the training. The next step is sending the initial global model from the central server to the clients. Later, clients train their model using their local data and send the model parameters, such as gradients or weights, to the central server for aggregation. This loop repeats until the global model converges or the training is completed [MMR+23] [KMA+21].
FL provides significant advantages such as data privacy and security, reduced communication cost, customization and flexibility, and increased processing power. In terms of data privacy and security, since the raw data is not shared with the central server or other entities, local data is kept private, and the risk of being compromised by an attacker is decreased. It only allows the model parameters to be shared instead of the client's local

data, increasing data privacy and security. Moreover, the communication cost of the training is reduced because the local data is not transmitted to the central server frequently. Only the model parameters such as, gradients and weights, are transmitted to the central server. Keeping local data on the clients and exchanging model parameters decreases communication costs and latency. Furthermore, FL allows for customization and flexibility. Different client groups or device types might be included in the training by customizing their models, training procedures etc. For example, for a medical application, a hospital could create models for their data and also contribute to a general model created by FL. In terms of total processing power, FL setting increases the total processing power because of the usage of numerous clients instead of using one server to train the ML model. Mobile devices, IoT sensors, and other entities can be used in the training session, which increases the capacity of processing power instead of only using one central server.

Although FL provides numerous advantages compared to traditional ML, it also has several challenges, such as data heterogeneity, data privacy, and security against adversarial attacks.

The data heterogeneity between the clients means that each client's local data differs from the others, which is one of the main challenges that FL faces. In real-world applications, each client has different kinds of local data because of the different demographics, device usage, and so forth. This difference causes the global model not to be able to generalize steadily enough to cover the minority or extreme groups because the model parameters that are sent to the central server are trained on the local data, which has different distributions. In some cases, clients with imbalanced datasets could mimic themselves as adversaries, which results in degradation in the performance of the global model. Therefore, it is hard to design defense mechanisms in FL when the data heterogeneity is high [KHJ23].

Another challenge that FL faces is the adversary attacks. Malicious clients are a threat to the FL systems by trying to degrade the performance of the global model. These malicious clients could send false model parameters to the central server or create some clever attacks to change the direction of the training so that the global model would be trained incorrectly. In the thesis, Random Gaussian attack [FCJG20b], Inner Product Manipulation (IPM) attack [XKG19] and Label Flipping (LF) attack [AZELA21] [ZLH+25] are performed. The Federated Averaging (FedAvg) [MMR+23] aggregation method cannot identify these attacks easily because it is based on weighted averaging. As a result, differ-

ent kinds of robust aggregation methods such as Krum [BMGS17], MKrum [BMGS17], Trimmed Mean [YCRB21], and Bulyan [MGR18] have been created to identify malicious clients and protect the FL system from the effects of attackers.

The structure of FL provides basic data privacy because clients' datasets are not shared with the central server or other clients. Since local model parameters are transmitted to the central server and the aggregation result is transmitted to the clients, with the help of reverse engineering techniques and transmitted parameters, raw data can be inferred, which causes privacy leakage. Gradient inversion and model inversion attacks [ZLH19] could be used to reconstruct the image, text, or other sensitive information using transmitted parameters. Even by using membership inference attacks [SSSS17], whether a data sample is used in the training set could be detected. These kinds of privacy attacks are critical risks for sensitive data, especially health or financial data. There are a couple of ways to prevent this information leakage, Secure Aggregation [BIK$^+$16] and Local Differential Privacy (LDP) [Dwo06] [NHC22]. Secure aggregation is a cryptographic protocol that allows the server to aggregate the model parameters without seeing any clients' model parameters and protects the system from an honest-but-curious server and external eavesdroppers. Since it is a cryptographic protocol, it introduces communication overhead because keys and secrets need to be shared. On the other hand, LDP is another approach that could be used to prevent privacy leakages. It masks model parameters by introducing uncertainty via noise. Since the method adds noise, the performance of the global model is affected significantly. In the thesis, for the implementation of LDP, DP-SGD mechanism [Dwo06] is used.

Robust aggregation methods and privacy in FL are extensively researched separately. Yet, LDP-enhanced FL has rarely been evaluated under different robust aggregation methods [QWH24]. However, an extensive evaluation of state-of-the-art robust aggregation methods under LDP has not been performed. The central goal of the thesis is to empirically examine the performance of different robust FL aggregation methods under LDP. Therefore, within the context of this thesis, a unified experimental setup in which each client's local updates are perturbed according to an LDP mechanism, and subjected the resulting setup to four different attack scenarios, which are No-attack (baseline), Random Gaussian attack, Inner Product Manipulation (IPM) attack, and Label Flipping (LF) attack is performed.

Within this setup, we evaluate different FL aggregation methods (Krum, MKrum, Trimmed Mean, Median, Geometric Median, Centered Clipping, Clustering, Clipped Clustering,

CosDefense, and Bulyan) across two widely used benchmark datasets (MNIST and CIFAR-10) under both Independent and Identically Distributed (IID) and Non-independent and Identically Distributed (non-IID) data distributions.
By comparing the final test accuracies and convergence behaviors "with" versus "without" LDP, robust aggregation methods, which suffer severe performance degradation under LDP, are identified. Later, those specific robust aggregation methods are focused on by examining their convergence behavior, per-round update distribution, revealing the broken statistical assumptions or algorithmic steps and explaining how LDP breaks down their robustness. By this dual approach (broad benchmarking and focused on failed robust aggregation methods under LDP), the robust aggregation methods, which can survive under both LDP and adversarial manipulations, are concluded.

This thesis is structured as follows. Chapter 2 represents the foundational concepts and terminologies associated with ML and FL. Also, it explains poisoning attacks and robust aggregation methods used in the experiments. Later, the differential privacy notion in FL and the DP-SGD [Dwo06] are explained. At the end of the chapter, Principal Component Analysis (PCA), which is used heavily in Chapter 4, is explained. Chapter 3 presents the experimental setup and combined results tables for MNIST and CIFAR-10 datasets, under different data partitions and all attack types, with and without LDP, and highlights the key observations. Chapter 4 analyzes the main reasons behind significant performance drops when LDP is applied. Chapter 5 concludes with a summary of findings and directions for future work.

2 Background

This chapter introduces the foundational concepts and terminologies associated with ML and FL. Moreover, it describes poisoning attacks performed on FL and robust aggregation methods used in FL to protect the system from poisoning attacks. Later, it introduces the differential privacy notion in FL. Finally, this chapter explains PCA.

2.1 Machine Learning (ML)

ML is an algorithmic system that discovers patterns and relationships in the dataset and applies the found patterns and relations to new situations. ML algorithms can be categorized into four main types: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning [SSBD14] [GGP24].

- Supervised learning is one of the most common ML approaches. The training data is composed of input-output pairs, and the main idea is to find a mapping function from inputs to outputs. As a result, the model can make knowledgeable predictions on unseen data.
- In unsupervised learning, the training data is composed of only inputs, and the main idea is to find hidden patterns or clusters within the input data. Clustering can be given as an example for unsupervised learning.
- Semi-supervised learning is a ML type that uses labeled and unlabeled datasets to train the model. It is one of the best ML approach to use when the labeled dataset is small but the unlabeled dataset is large.
- Reinforcement learning uses reward-penalty-based training to train a model. The main idea is to maximize the reward score throughout the training by changing the strategy depending on the environment.

Since most of the FL experiments are performed in supervised learning, the theory part of ML is explained based on supervised learning.
The main idea of supervised learning is to learn a mapping function from an input space X to an output space Y by using the input-output pairs, $\{(x_i, y_i)\}_{i=1}^{m}$, where $\{x_i\}_{i=1}^{m} \subset$

X, $\{y_i\}_{i=1}^{m} \subset Y$. The learning or training is to approximate the mathematical connection between the input x and the output y by using the parametric model $f(x;\theta)$ based on the dataset. Actually, this means solving the following optimization problem:

$$\theta^* \;=\; \arg\min_{\theta} \; \frac{1}{m} \sum_{i=1}^{m} L\big(f(x_i;\theta),\, y_i\big)\;, \tag{2.1}$$

where L is the loss function that measures the error. Often, cross-entropy is used for classification, and mean squared error is used for regression. θ is the model's parameter that is updated by gradient-based optimization algorithms such as Stochastic Gradient Descent (SGD). We define the sample-wise gradient for (x_i, y_i) as $g_i \triangleq \nabla_\theta L\big(f(x_i;\theta),\, y_i\big)$. The optimization problem can be solved by minimizing the following cost function by training the model:

$$J(\theta) \;=\; \frac{1}{m} \sum_{i=1}^{m} L\big(f(x_i;\theta),\, y_i\big). \tag{2.2}$$

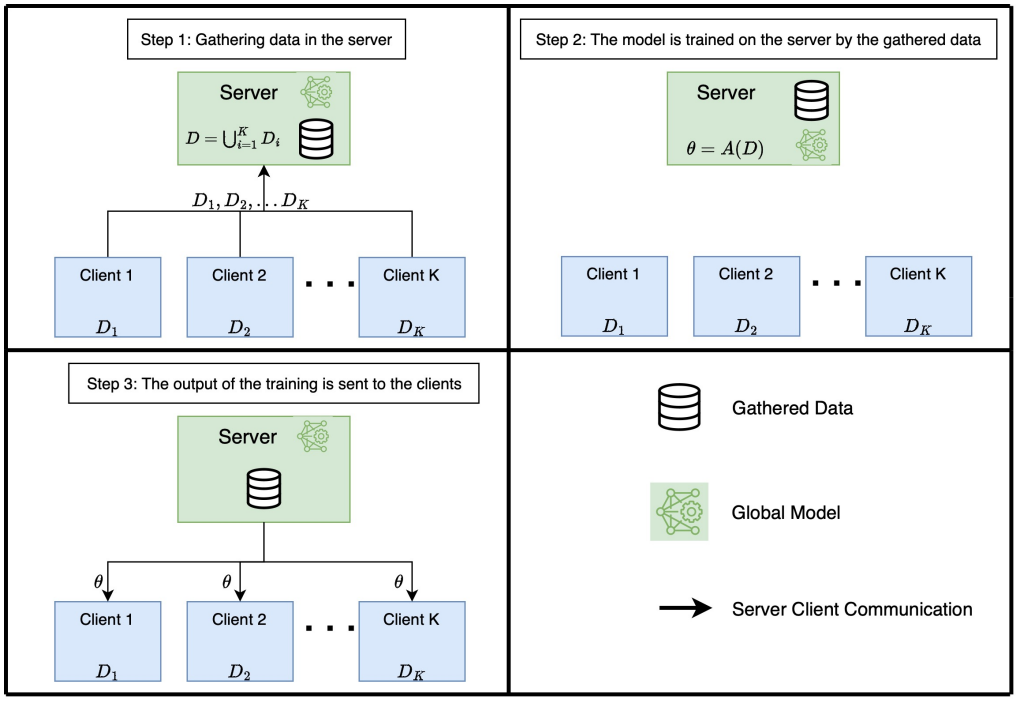

Figure 2.1: Traditional Machine Learning.

To minimize the cost function with SGD, the gradient of the cost function ($\nabla_\theta J(\theta_t)$) needs to be used with a learning rate (η) to update the model:

$$\theta_{t+1} \;=\; \theta_t \;-\; \eta\, \nabla_\theta J\big(\theta_t\big), \tag{2.3}$$

where $\nabla_\theta J(\theta_t) = \frac{1}{m} \sum_{i=1}^{m} g_i$.

In traditional ML, data is gathered in single place, and all the data is used together to train the model. This method can provide high accuracy, but it has severe limitations in terms of privacy and data security.

In traditional ML, shown in Figure 2.1, each client i holds local data D_i. The server, which has the model, gathers data from each of the clients and creates the dataset $D = \bigcup_{i=1}^{k} D_i$ to use for training as a first step. Then, it trains the model by using the dataset D. After the model's training, the output of the training, θ, is sent back to clients [KMA+21].

2.2 Federated Learning (FL)

Successful ML models require large and clean datasets. However, one of the main bottlenecks encountered in creating a successful ML model is the lack of access to the data available to train the model. Data privacy is one of the reasons for this absence of data available to train the model. In some industries, such as but not limited to medicine and finance, data cannot be shared with third-party companies to train a successful ML model due to legal regulations regarding the protection of personal data - especially the European Union's General Data Protection Regulation (GDPR). Even if some individuals or companies are willing to share their data, a significant portion of data owners generally avoid sharing their data due to privacy concerns. As a result, the dataset used for training the ML models would not be representative of the entire population, resulting in poor generalized ML models after the training. This imbalance in the dataset can cause the model to fail, especially on critical parts such as minority groups or extreme examples [KMA+21].

As a solution to these problems, the term Federated Learning (FL) was introduced first by McMahan et al. FL is a way to tackle the lack of training data and privacy concerns of training data by training a global ML model collaboratively with clients via keeping the clients' raw data locally, and the clients' raw data is not transferred or shared with any other clients or the server [MMR+23].

As described in Section 2.1, in traditional ML, the training data is gathered in one place, and the model is trained on gathered training data. Even if this provides high accuracy, privacy and security concerns are still present. However, in FL, data is not gathered in one place. Each client will keep their private data locally. In the initialization phase of FL shown in Figure 2.2, the global server first creates the global model, θ_1, and then

sends a copy of the global model to each of the client [BEG+19].

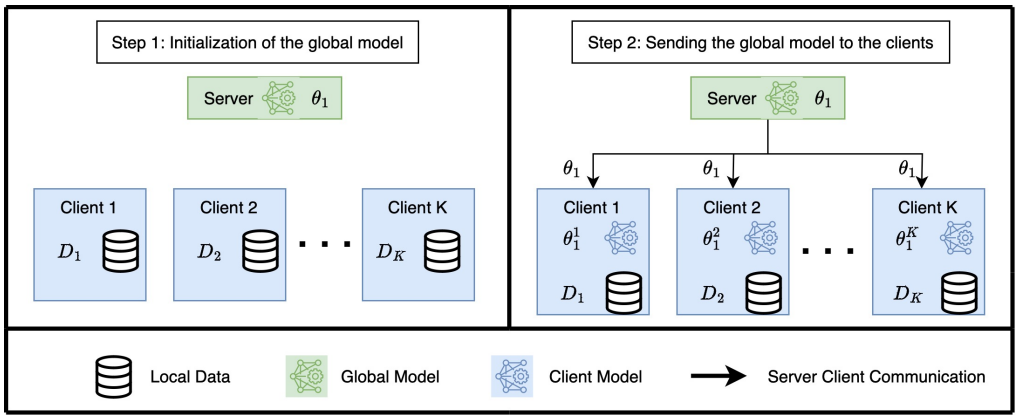

Figure 2.2: Initialization Steps for FL.

In this way, each client can train the model using their data without exposing their data to the outside world. The general training steps for FL are shown in Figure 2.3. Firstly, clients train their models, $\theta_t^{(k)}$, by using their own local data, D_k. After the training of the local models by each client, local updates $\theta_{t+1}^{(k)}$ or gradients $g_t^{(k)}$, are obtained at each client k. Finally, the client sends a message $x_t^{(k)}$ back to the global server for aggregation, where $x_t^{(k)} = \theta_{t+1}^{(k)}$ is the model update or, equivalently, a pseudo-gradient $x_t^{(k)} = g_t^{(k)}$ [BEG+19]. The pseudo-gradient $g_t^{(k)}$ is defined as

$$g_t^{(k)} = \frac{\theta_{t+1}^{(k)} - \theta_t^{(k)}}{\eta}. \tag{2.4}$$

At the end of each iteration, the global server aggregates the model updates (or equivalently, gradients) using an aggregation function $\text{Aggr}(\{g_t^{(k)} : k \in [K]\})$. The traditional way of aggregation in the FL setting is called FedAvg [MMR+23]. This aggregation algorithm updates the global model by averaging the local model weights or gradients. The below formulation is for FedAvg:

$$\theta_{t+1} = \text{Aggr}(\{g_t^{(k)} : k \in [K]\}) = \sum_{k=1}^{K} \frac{n_k}{n} \theta_{t+1}^{(k)}, \tag{2.5}$$

where

- K is the total number of clients,
- n_k is the number of data samples on client k, and
- $n = \sum_{k=1}^{K} n_k$ is the total number of samples across all clients.

After the aggregation of local weights or gradients by the server, global weights or gradients, θ_{t+1}, are constructed and sent back to the clients for further training of the local models. This cycle repeats until the global model converges to a desired performance.

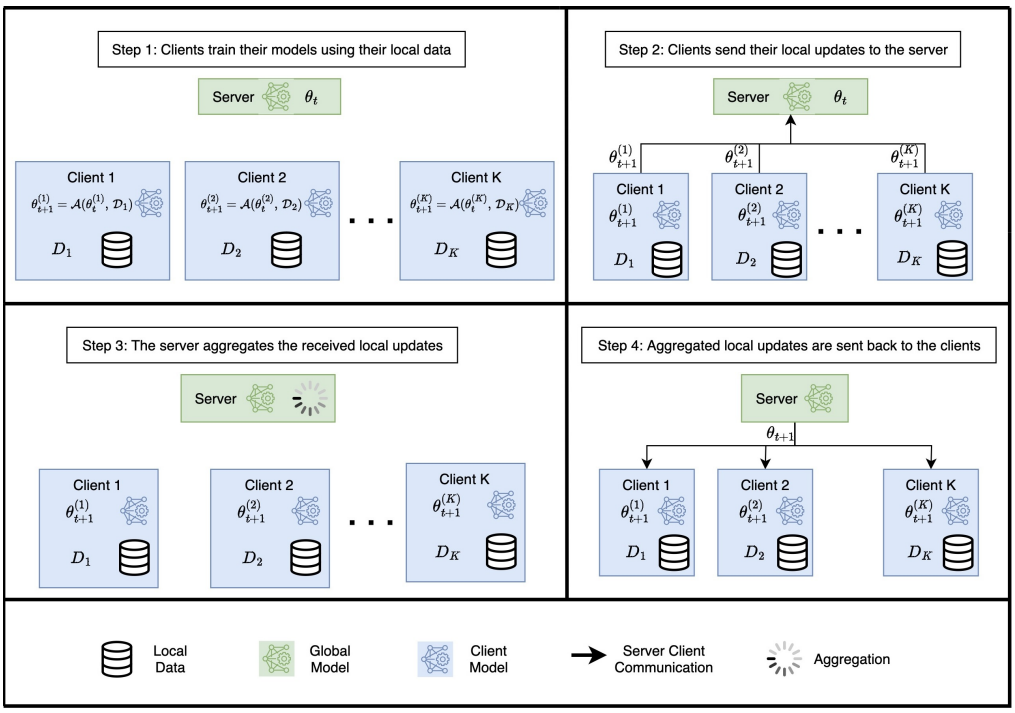

Figure 2.3: General Training Steps for FL.

FL systems can be divided into two categories, cross-device FL and cross-silo FL, which are differentiated by the distribution of the dataset and system architecture. The cross-device FL system can be described as a system that has numerous edge devices, such as smartphones or IoT devices. These devices generally have limited local data, processing power, battery, and an unstable internet connection. Moreover, each device's local data is generally personalized and non-IID. Google GBoard [HRM+19] can be given as an example of this kind of FL system. On the other hand, the cross-silo FL system can be described as a system which has few reliable clients, such as hospitals, banks, and so forth. These clients have considerable ordered and large datasets, which are more homogenous and IID compared to cross-device FL systems [KMA+21].

Even if FL provides privacy while training a model collaboratively with the clients, it has statistical and systematic challenges. Statistical challenges can be described as the non-IID datasets among clients. This imbalance could cause the global model to evolve in the direction of the local models with a higher number of samples, which could result in poor generalization for the model. Moreover, the convergence of the model could be

negatively impacted by this imbalance [ZLL+18]. For this reason, within the scope of the thesis, both IID and non-IID will be simulated. To simulate the IID and non-IID data partition among clients, we used a Dirichlet distribution $p_l \sim \text{Dir}_K(\alpha)$. Later, $p_{l,k}$ portion of the training sample of class l is assigned to the client k. A small value of α means a stronger non-IID data partition, and a high value of α means a stronger IID data partition [Nar91].
Systematic challenges can be described as differences in hardware, internet connection, and battery power of devices. Since clients could have completely different hardware, power, or internet connections, the communication between clients and the server could be disrupted, and this could result in the disruption of the training of the model. Moreover, those differences could result in some clients joining the training and some clients not joining. As a result, some clients could contribute more than others and this could lead the model to learn in the direction of those clients who joined the training [SFUK22]. However, since this thesis focuses on the effects of adversarial attacks and differentially private mechanisms, challenges such as client dropouts and communication errors are outside of the scope.

2.3 Poisoning Attacks

FL has a decentralized nature, which means that local models are trained on the client side, and training data are kept locally in each client. Therefore, the FL system is vulnerable to client-side poisoning attacks. The general idea of poisoning attacks is to disrupt the global model's performance. The attacker which is called malicious clients can send arbitrary pseudo-gradients ($x_t^{(k)} = \tilde{g}_t^{(k)}$) or model updates ($x_t^{(k)} = \tilde{\theta}_{t+1}^{(k)}$) to the server depending on the attack that they are performing. Depending on the attack vector, the poisoning attacks can be categorized into two main categories such as data poisoning and model poisoning [ZLH+25].

2.3.1 Data Poisoning

The main idea of data poisoning attacks is to manipulate the training data so that the model is trained on poisoned data. Data poisoning attacks are often considered targeted attacks, which means they are trying to achieve a specific goal, such as making a global model mispredict a specific class [ZLH+25]. There are different kinds of data poisoning attacks, such as but not limited to LF attacks [AZELA21], Backdoor attacks [XHCL20]. However, within the scope of the thesis, the analysis of the LF attack from the data poisoning category is focused.

Label Flipping (LF) Attack

LF attacks [AZELA21] [ZLH+25] are designated to manipulate the output of the model by changing the target labels in training data, with different target labels determined by the attacker. There are three different strategies, which are random, inverse, and target labels, that could be used for LF. For the random strategy, the attacker flips the target label into another target label chosen randomly from the set of target labels. For the inverse strategy, the target labels of the training data are flipped to the target labels that mirror their position relative to the total number of classes. For example, for the MNIST dataset, label $l \in \{0, 1, \ldots, 9\}$ is flipped to $(9 - l)$. Lastly, for the target strategy, the attacker flips the target label of the training data to a specific label. As can be seen from Figure 2.4, malicious clients have poisoned local data, and their target labels are flipped to an incorrect target label. The rest of the training is the same as the generalized FL training explained in Section 2.2. However, malicious clients create malicious local updates because their models are trained on poisoned local data. Within the scope of the thesis, a LF attack with an inversion strategy is used. The attacker needs to access its own local dataset to perform the LF, but it does not need to know any additional information about the other clients' datasets or gradients. This attack is chosen and implemented because it is hard to detect compared to model-level attacks.

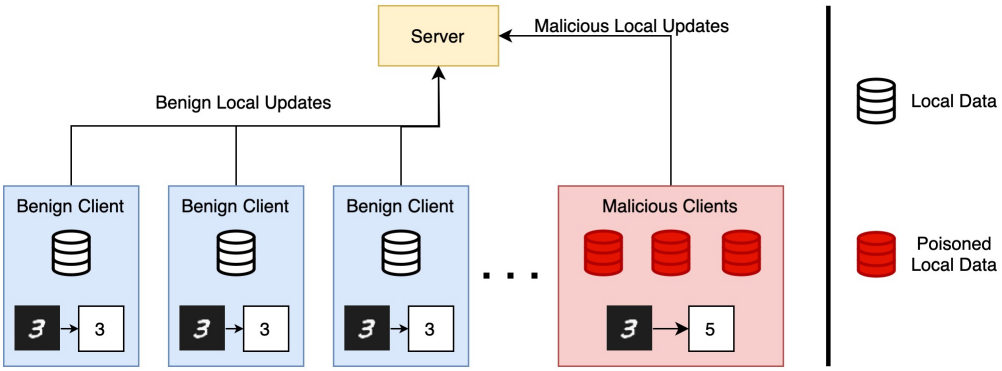

Figure 2.4: The general flow of LF attack in FL.

2.3.2 Model Poisoning

The main idea of the model poisoning attacks is to manipulate local updates created as a result of training local models. Most model poisoning attacks are considered untargeted, which means they are trying to disrupt the overall global model's performance. There are different kinds of model poisoning attacks, such as but not limited to Gaussian Random attack [FCJG20b], IPM attack [XKG19], A Little is Enough (ALIE) attack [BBG19].

However, within the scope of the thesis, the analysis of Gaussian Random attack and IPM attack under the model poisoning category is focused.

Gaussian Random Attack

Gaussian Random attack [FCJG20b] manipulates local updates by injecting random noise sampled from Gaussian distribution in order to disrupt the global model's convergence. There are two parameters, mean and standard deviation of the Gaussian distribution, that can be controlled by the attacker. As the standard deviation increases, the impact of the attack gets stronger since the degree of perturbation increases.

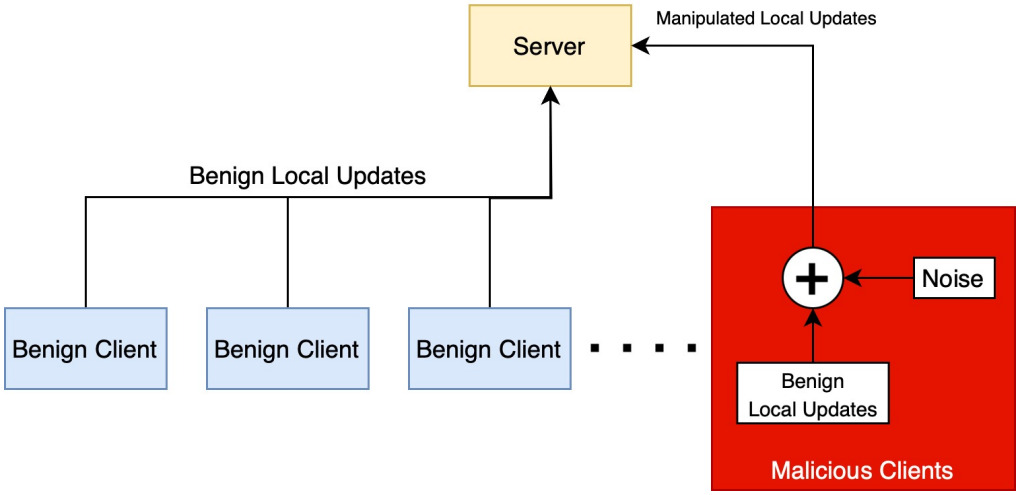

Figure 2.5: The General Flow of Gaussian Random Attack.

As can be seen in Figure 2.5, the malicious clients train their model the same way benign clients train theirs. However, before sending the benign local updates to the server for aggregation, malicious clients inject random Gaussian noise into their local updates:

$$\tilde{\theta}_{t+1}^{(k)} = \theta_{t+1}^{(k)} + \zeta_k^{(t)}, \quad \zeta_k^{(t)} \sim \mathcal{N}(0, \sigma^2 I_d), \tag{2.6}$$

where $\theta_{t+1}^{(k)}$ is the true local update of malicious client k at round $t+1$, and malicious client k adds $\zeta_k^{(t)} \sim \mathcal{N}(0, \sigma^2 I_d)$ to their true local updates and creates malicious local updates, $\tilde{\theta}_{t+1}^{(k)}$.

The zero-mean Gaussian noise with a standard deviation is injected into the local updates of malicious clients. The value for standard deviation is a hyperparameter and is chosen by the attacker. The attacker does not need to know any additional information about the global model, other clients' dataset or gradients. This attack is chosen and implemented

because it is a simple but effective attack type and helps us to asses the robust aggregation methods under random perturbations.

Inner Product Manipulation (IPM) Attack

IPM attack [XKG19] [ZLH+25] is a model poisoning attack that aims to control the direction of the global model. Some of the robust aggregation methods used in FL settings aim to ensure that the global model update does not deviate too far from the average of the local updates of the benign clients. However, these methods do not control the direction of the update of the global model. Yet, for a model to learn correctly, the inner product of the aggregated gradient and the true gradient should be non-negative:

$$\left\langle \nabla F(\theta_t), \text{Aggr}\left(\{g_t^{(k)} : k \in [K]\}\right) \right\rangle \geq 0, \tag{2.7}$$

where $\nabla F(\theta_t)$ is the true gradient of the global objective at point θ_t.
However, if the inner product result is negative, the direction of the model update is incorrect, which causes the loss function to increase throughout the learning process. As a result, the global model cannot converge to a desired state. The IPM attack exploits this vulnerability. The attack uses the mean of the benign clients' gradients to create malicious gradients in the opposite direction of this mean:

$$\tilde{g}_t^{(k)} = -\omega \cdot g_{mean}, \tag{2.8}$$

where

- $g_{mean} = \frac{1}{|V|} \sum_{g \in V} g$ is the mean of benign clients' gradients,
- ω is a scaling factor and is chosen by the attacker.

As a result, the inner product of the aggregated gradient and the true gradient will be negative:

$$\langle g_{mean}, \mathbb{E}\left[\text{Aggr}(V \cup U)\right] \rangle < 0, \tag{2.9}$$

where $\mathbb{E}[\text{Aggr}(V \cup U)]$ is the expectation of the aggregated gradients from both benign (V) and adversarial (U) clients.
Some robust aggregation methods cannot identify these malicious gradients because they are not far away from the average of the local gradients. However, these malicious gradients could cause the model to update in the wrong direction, resulting in no progress or failure in the learning. Yet, as an assumption, malicious clients should know about the benign clients' gradients at each epoch.

As can be seen in Figure 2.6, malicious clients obtain the mean of benign clients' gradients and create their gradients by multiplying the mean of the benign clients' gradients with -ω, which is a hyperparameter and chosen by the attacker. This attack is chosen and implemented because it is stated that the attack breaks the two robust aggregation methods, Krum and Median.

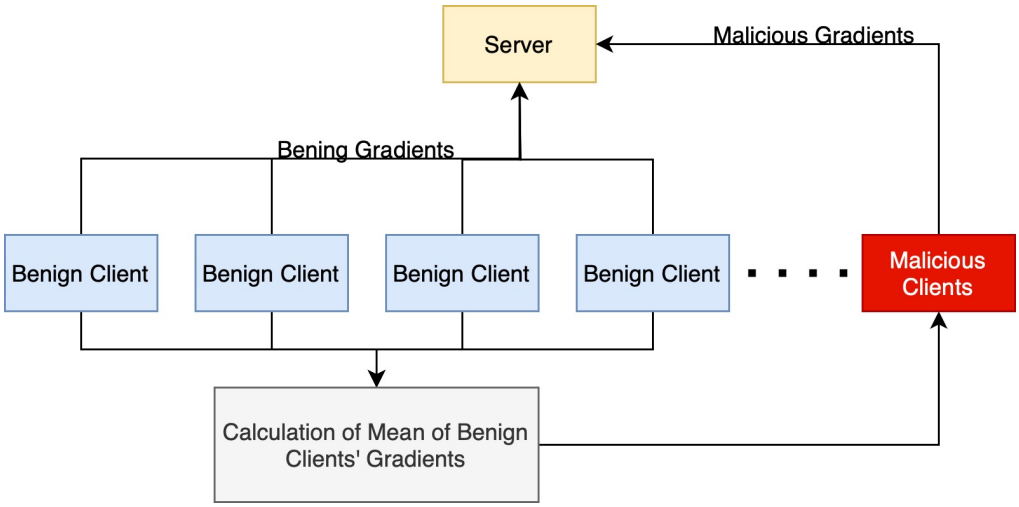

Figure 2.6: General Flow of Inner Product Manipulation (IPM) Attack.

2.4 Robust Aggregation Methods

In the basic FL system, FedAvg is used to aggregate the local updates when there are no attackers in the system. However, FedAvg is not robust when there are attackers. As a result, the attacker can manipulate the local updates to disrupt the global model's performance, described in Section 2.3. Therefore, different robust aggregation methods are created to protect the FL Systems from different poisoning attacks. This section will explain the robust aggregation methods used in the thesis [FCJG20a].

2.4.1 Krum

Krum [BMGS17] is a robust aggregation method used by the server in FL systems to protect the system from Byzantine failures, and it depends on Euclidean distance calculation. The main goal of the method is to calculate the Euclidean distances between clients' local updates. From the calculations, it finds the local update that is closest to another $K - M - 2$ neighboring local updates where K is the total number of clients, M is the total number of malicious clients. In other words, one of the local updates will

be chosen by the method that minimizes the sum of squared distances to its $K - M - 2$ neighbors. The found local update is used to update the global model's weights. The calculation for finding the closest local update to its $K - M - 2$ neighbors' local updates by using Euclidean distance by the Krum method can be seen below:

$$\text{Krum} := \left\{ \theta_{t+1}^{(i)} \,\middle|\, i = \arg\min_{i \in [K]} \sum_{i \to j} \|\theta_{t+1}^{(i)} - \theta_{t+1}^{(j)}\|_2^2 \right\}, \tag{2.10}$$

where

- $\theta_{t+1}^{(i)}$ and $\theta_{t+1}^{(j)}$ are the local updates from the i^{th} and j^{th} clients, respectively,
- $\|\theta_{t+1}^{(i)} - \theta_{t+1}^{(j)}\|_2^2$ is the squared Euclidean distance between the two updates.

Moreover, Krum requires knowledge of the number of malicious clients found in the system to calculate the number of neighbors.

2.4.2 Multi-Krum (MKrum)

In FL systems, the server can employ MKrum [BMGS17], a robust aggregation method, to defend the system from Byzantine failures. It is the extension of the Krum method. The main idea is the same as the Krum aggregation method. However, Krum selects only one client's local updates, whereas MKrum selects more than one client's local updates. After the selection of more than one client's local update, the method averages the selected local updates. Assume method chooses the best N local updates out of K local updates, and $N \in \{1, \ldots, K\}$. The resulting update that will be used for the global model will be:

$$\frac{1}{N} \sum_{i=1}^{N} \theta_{t+1}^{(i)}, \tag{2.11}$$

If N is set to 1 or K, then MKrum behaves like Krum or Averaging, respectively. Moreover, MKrum requires knowledge of the number of malicious clients found in the system to calculate the number of neighbors.

2.4.3 Trimmed Mean

Another robust aggregation method used by the server to avoid Byzantine failures in FL systems is Trimmed Mean [YCRB21], and it depends on the coordinate-wise trimmed average of local updates. The main objection of the method is to first discard a β fraction of local updates for each coordinate from the smallest and the largest parts after sorting each coordinate, where $\beta \in [0, 1/2)$. To ensure robustness, the β parameter is an

upper bound on the fraction of malicious clients found in the system. As a result, the method requires knowledge of the number of malicious clients found in the system to set β parameter. After trimming from the smallest and the largest parts, the method will take the average of all of the remaining local updates for each coordinate. The overview of the method can be seen in the formulation below:

$$\theta_{t+1} := \text{trmean}_{\beta}\left\{\theta_{t+1}^{(i)} : i \in [K]\right\}, \tag{2.12}$$

$$\theta_{t+1,i} = \frac{1}{|U_i|}\sum_{\theta \in U_i}\theta, \tag{2.13}$$

where U_i is the set of non-trimmed values for the i-th coordinate with $|U_i| = (1-2\beta)K$. Trimmed Mean is a robust aggregation method when the local updates are extreme, such as adding extreme noise to the local updates.

2.4.4 Median

In FL systems, Median [YCRB21] serves as a server-side robust aggregation technique to protect the system from Byzantine failures, and it depends on the coordinate-wise median operation of local updates. The main goal of the method is to find the coordinate-wise median across all local updates. For each coordinate of the local updates, the method calculates the median and then forms a vector whose elements are the median of each corresponding coordinate. The overview of the method can be seen in the formulation below:

$$\theta_{t+1} := \text{med}\left\{\theta_{t+1}^{(i)} : i \in [K]\right\}, \tag{2.14}$$

Since the coordinate-wise median minimizes the effect of anomalies or malicious values, it is robust to extreme outliers. Moreover, it does not have any assumptions or hyperparameters.

2.4.5 Geometric Median

Geometric Median [CSX17] is a robust aggregation method that is used by the server in FL systems to defend the system from Byzantine failures, and it depends on geometric median calculation. The main idea of the technique is to find a central point that minimizes the sum of the distances between each point. Firstly, all of the clients send their local updates to the server, and then the server separates all local updates into batches and calculates the batch mean for each batch. Secondly, the geometric median is calculated on each

batch using the batch mean. The formulation for the geometric median calculation is as follows:

$$\text{med}(y_1, y_2, \ldots, y_n) = \arg\min_{x \in \mathbb{R}^d} \sum_{i=1}^{n} \|x - y_i\|, \tag{2.15}$$

where

- y_i are the points in the local updates,
- x is the geometric median candidate.

In the Geometric Median aggregation technique, batch means are used instead of points from the local updates to calculate the geometric medians, called batch geometric medians. Calculated batch geometric medians are used to calculate the global geometric median to update the global model. By this technique, the model is resilient to outliers or malicious clients. However, there are some limitations that should be considered. One of them is that the calculation of the geometric median directly is computationally expensive, and an approximate geometric median calculation is recommended. Also, there is an upper bound for the maximum number of Byzantine local updates to make the system Byzantine-resilient, which means that the server should know the number of Byzantine clients found in the system beforehand.
This upper bound is:

$$2(1 + \phi)M \leq K, \tag{2.16}$$

where ϕ is a fixed small positive constant.
In the scope of the thesis, since the actual calculation of the geometric median is computationally heavy, Weiszfeld's algorithm is used to approximate the geometric median in this thesis.

2.4.6 Centered Clipping

Against Byzantine behavior in FL systems, the server may use Centered Clipping [KHJ21] as a robust aggregation technique, and the method depends on clipping norms of local updates and averaging. The main idea of the technique is to clip norms of the local updates up to a clipping threshold called τ, which means that norms of malicious clients' local updates higher than value τ will be decreased to the value τ and the benign clients' local updates will be unchanged because their norms are not exceeding the value τ. After the clipping process, the average of the clipped local updates will be calculated. In general, the ℓ_2-norm will be used to calculate the norm of the local updates. The

overview of the method can be seen in the following formulation:

$$v_{(t+1)} = v_{(t)} + \frac{1}{K}\sum_{i=1}^{K}(\theta_{t+1}^{(i)} - v_{(t)}) \cdot \min\left(1, \frac{\tau}{\|\theta_{t+1}^{(i)} - v_{(t)}\|}\right), \tag{2.17}$$

where

- $v_{(t+1)}$ is the new center,
- $v_{(t)}$ is the previous center,
- τ is the clipping threshold,
- $\|\theta_{t+1}^{(i)} - v^{(t)}\|$ is the Euclidean norm.

When the iteration is started for the first time for the technique, the $v_{(t+1)}$ (previous center) can be chosen to be any local update. Moreover, this technique is easy to implement and has a linear computational complexity.

2.4.7 CosDefense

CosDefense [YZA23] is a robust aggregation method that is used by the server in FL systems to protect the system from Byzantine failures. It is based on the cosine similarity of the last layer's weights between the global model and each local update. The main objection of the technique is to calculate the cosine similarity score of each local update's last layer and the global model's last layer, and aggregate updates exceeding the threshold. So, the method calculates the cosine similarity scores:

$$s_i \triangleq \cos(\alpha_i) = \frac{\langle \theta_t, g_t^{(i)} \rangle}{\|\theta_t\| \cdot \|g_t^{(i)}\|} = \frac{\langle \theta_t, \theta_{t+1}^{(i)} - \theta_t \rangle}{\|\theta_t\| \cdot \|\theta_{t+1}^{(i)} - \theta_t\|} = \frac{\langle \theta_t, \theta_{t+1}^{(i)} \rangle - \langle \theta_t, \theta_t \rangle}{\|\theta_t\| \cdot \|\theta_{t+1}^{(i)} - \theta_t\|}, \tag{2.18}$$

where α_i denotes the angle between the global model weights θ_t and the local update direction g_{t+1}^i of client i.
Later, the method performs min-max normalization on the cosine similarity scores:

$$\tilde{s}_i = \frac{s_i - \min_j s_j}{\max_j s_j - \min_j s_j}, \tag{2.19}$$

Moreover, the method calculates the threshold as the average of all normalized scores:

$$\tau = \frac{1}{K}\sum_{i=1}^{K} \tilde{s}_i, \tag{2.20}$$

where τ is the threshold value used to filter clients.
Furthermore, the normalized similarity scores of local updates are compared with the threshold. The ones exceeding the threshold are kept, their updates are stored in a benign cluster, and the other normalized similarity scores are excluded. Finally, the local updates stored inside the benign cluster are averaged, and the result is used to update the global model.

2.4.8 Clustering

Clustering [LLZL21] is a robust aggregation method that is created against Byzantine failures in FL systems, and it is based on cosine similarity between local updates of clients. The main objective of the technique is to cluster benign and malicious clients into two different clusters and aggregate local updates of the cluster of benign clients by examining the geometric and temporal properties of clients' local updates.
For the geometric perspective, firstly, the creation of an affinity matrix, which is constructed by computing the cosine similarities between different clients' local updates, is performed (refer to 2.18 for cosine similarity calculation). By this matrix, the technique can identify the outliers found in the local updates of clients. Secondly, agglomerative clustering with complete linkage will be performed to cluster benign and malicious clients into two different clusters by using the affinity matrix.
The main idea of the Agglomerative Clustering algorithm [Mü11] is to treat each partitioned dataset as a node and then iteratively merge two nodes that are closest to each other. This process continues until only one node remains. However, Agglomerative Clustering with complete linkage is used in the Clustering robust aggregation technique. In the context of an FL setting, the Agglomerative Clustering algorithm treats each client's local update as one node and merges two nodes based on the complete-linkage criterion. In the complete-linkage criterion, the distance between two clusters is the maximum distance between any two points taken from each cluster. This process continues until there are only two nodes/clusters remaining.
Later, the cluster with the highest number of local updates will be the benign cluster, and the other cluster will be the malicious cluster. However, before going into the cluster's aggregation phase, the largest similarity between the two clusters is compared with the clustering threshold. All local updates are considered benign if the similarity exceeds the clustering threshold. Otherwise, only the local updates inside the benign cluster are considered benign. In the aggregation step, any aggregation method, such as averaging, median, or so on, could be used. It is stated that the usage of the median aggregation method is better because it does not need any prior knowledge about the number of

malicious clients in the system.

For the temporal perspective, they use a momentum vector, which is the average of the previous epoch's chosen local updates. By using this vector, the newly aggregated local updates can be assessed by checking the cosine similarity of the momentum vector and the newly aggregated local updates. If the cosine similarity is too low, it will reject the newly aggregated local updates, and the global model will not be updated.

2.4.9 Clipped Clustering

Clipped Clustering [LNV24] is a robust aggregation method that is designed to protect the FL systems from Byzantine failures, and it can be considered as the combination of center clipping and clustering aggregation methods that are described in Section 2.4.6 and Section 2.4.8, respectively. The main goal of the technique is to clip the norms of the local updates up to a specific clipping threshold called τ in order to prevent the effect of magnified local updates and then cluster the clipped local updates into two clusters called benign and malicious clusters.

In the norm clipping phase, local updates' norms are calculated, and if they exceed the specific clipping threshold called τ, they will be clipped to the clipping threshold τ. Otherwise, they remain the same as described in Section 2.4.6. The norms of local updates are saved at each epoch, and its median is chosen as the threshold for the next epoch. The median of the historical norms is chosen as the threshold because it is statistically robust.

After the norm clipping phase, the creation of an affinity matrix, which is constructed by computing the cosine similarities between different clients' local updates, is performed (refer to 2.18 for cosine similarity calculation). Later, the clustering phase is performed using Agglomerative Clustering with average linkage, which computes the similarity between two clusters as the mean distance of all pairs of points found in the clusters to be merged. After performing the clustering, two clusters are formed. The cluster with a higher number of local updates is chosen to be a benign cluster, and the other one is chosen to be a malicious cluster. The cluster, which is composed of benign local updates, is averaged, and the result is used to update the global model. The difference between this clustering part and the clustering algorithm explained in Section 2.4.8 is that there is no similarity calculation between benign and malicious clusters. This technique directly uses local updates from the benign cluster for aggregation.

2.4.10 Bulyan

Bulyan [MGR18] is a robust aggregation method that is created against Byzantine failures in FL systems. It is an additional layer that can be put on top of Krum or Geometric Median robust aggregation methods. The main idea of the technique is to first use Krum or Geometric Median techniques iteratively to select a couple of local updates and then filter those local updates using a coordinate-wise median. Finally, the selected and filtered local updates are averaged to update the global model.

As described in Section 2.4.1, the Krum method uses Euclidean distance to choose only one local update. In Bulyan, the Krum method is used iteratively a couple of times to get $N = K - 2M$ local updates, where N is the number of selected local updates, K is the total number of local updates, and M is the number of malicious local updates. After the selection of honest local updates, a coordinate-wise median is computed for each coordinate of the local updates. Later, the β number of the closest coordinates to the coordinate-wise median is found and averaged. In this part, $\beta = N - 2M$, where β is the number of closest coordinates to the median, N is the number of selected local updates after the first phase, and M is the number of malicious local updates. The method requires knowledge of the number of malicious clients found in the system to calculate the N and β parameters.

So, at the end of the technique, β number of local updates whose coordinates are closest to the coordinate-wise median are averaged. Choosing fewer local updates in the second phase minimizes the effects of malicious updates. However, this technique can tolerate Byzantine workers with a large number of clients, requiring $K \geq 4M + 3$. As a result, this system is impractical, with few clients.

Moreover, this technique is computationally heavy compared to Krum and Geometric Median because it has to execute Krum or Geometric Median techniques several times to select honest local updates. It also needs to compute the coordinate-wise median for each coordinate of the local updates. In this thesis, Krum is chosen to be used as the primary robust aggregation technique, and Bulyan is used as an additional layer.

Table 2.1 summarizes the aggregation methods employed in the experiments, detailing the principles they rely on and the assumptions underlying each approach.

Aggregation Method Name	**Underlying Principle**	**Assumptions / Hyperparameters**
Krum [BMGS17]	Euclidean Distance	Requires knowledge of the number of malicious clients
MKrum [BMGS17]	Euclidean Distance	Requires knowledge of the number of malicious clients
Trimmed Mean [YCRB21]	Coordinate-wise Trimmed Average	Requires knowledge of the number of malicious clients
Median [YCRB21]	Coordinate-wise Median	None
Geometric Median [CSX17]	Geometric Median	Assumes an upper bound on the number of malicious clients
Centered Clipping [KHJ21]	Clipping	Requires a clipping threshold
Clustering [LLZL21]	Cosine Similarity	Requires a clustering threshold
Clipped Clustering [LNV24]	Clipping and Cosine Similarity	Requires both a clipping and a clustering threshold
Bulyan [MGR18]	Krum and Coordinate-wise Median	Requires knowledge of the number of malicious clients
CosDefense [YZA23]	Cosine Similarity	None

Table 2.1: Aggregation Methods, Their Principles, and Assumptions or Hyperparameters

2.5 Differential Privacy

FL is privacy-preserving by design, as clients do not share their local data directly with the server or with each other. However, during the training process, clients share their local updates or gradients with the server, and this shared information could lead to the leakage of some sensitive information about the training data [LSTS20].
For instance, some important sensitive training data may be unintentionally memorized by ML models. As a result, this sensitive training data can be recovered by using the model's outputs or gradients [CLE+19].
Therefore, numerous privacy-enhancing approaches have been developed in order to protect FL systems from privacy violations. Among these privacy-enhancing approaches, differential privacy is the most widely implemented privacy approach because of its strong theoretical guarantees. The randomized mechanism is considered differentially private if altering one input element does not significantly impact the output distribution. In other

words, an external observer cannot determine whether any specific sample is included in the learning process. The below definition is the formal definition of differential privacy:

Definition 1 (Differential Privacy [Dwo06][NHC22]). *A randomized mechanism $\mathcal{M}$ provides (ϵ, δ)-differential privacy if for any two neighboring databases D_1 and D_2 that differ in only a single record, and for all possible outputs $S \subseteq Range(\mathcal{M})$:*

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

The parameter ϵ is a privacy budget, meaning it is a metric of privacy loss. A lower ϵ value implies a higher level of privacy, whereas a high ϵ value corresponds to a lower level of privacy. However, when a lower ϵ value is used, the utility of the system is reduced since the amount of perturbation performed on the gradients of the model increases. Moreover, the δ parameter is used as a probability where the upper bound of the ϵ value does not hold.
A standard method of applying differential privacy is to clip the gradients computed by the clients to a specific norm and randomize those gradients by adding calibrated Gaussian noise. Gradient clipping is performed in order to constrain the impact of each example on the model update. This mechanism is referred as DP-SGD and was proposed first for ML in [Dwo06].

Norm Clipping

In order to constrain the impact of each gradient on the model update, norm clipping, aka gradient clipping, should be performed on the gradients. For each per-sample gradient vector g_i, it is clipped to have a maximum ℓ_2-norm C [ACG+16]:

$$g_i^{\mathrm{c}} = g_i \cdot \min\left(1, \frac{C}{\|g_i\|_2}\right), \tag{2.21}$$

where g_i^{c} is the clipped per-sample gradient such that $\|g_i^{\mathrm{c}}\|_2 \leq C$.

Adding Gaussian Noise

After clipping on the per-sample gradients, the next step is performing Gaussian noise addition to those clipped gradients to ensure the differential privacy [KGS21].

$$\bar{g}_i = g_i^{\mathrm{c}} + N_i \sim \mathcal{N}\left(0, C^2\sigma_i^2 I_d\right) \tag{2.22}$$

where

- $\bar{g}_i$ is the perturbed gradient of client i,

- N_i is the Gaussian noise derived from $\mathcal{N}\left(0, C^2\sigma_i^2 I_d\right)$,
- σ_i is the noise multiplier,
- I_d is the identity matrix.

The noise multiplier [Tiw24] is calculated as follows:

$$\sigma = \frac{\Delta f \cdot \sqrt{2\ln\left(1.25/\delta\right)}}{\varepsilon} \tag{2.23}$$

where $\Delta f = \max_{D,D'} \|f(D) - f(D')\|$ is the sensitivity of the function f which measures the maximum difference of a function's output when one data point is changed.

There are two types of Differential Privacy that can be used in FL settings, namely local and central [NHC22].

Central Differential Privacy (CDP)

The result of the aggregation function is perturbed by the server. Central Differential Privacy (CDP) ensures that it is not possible to distinguish whether a client has participated in the training phase or not by perturbing the output of the aggregation function with a probability bounded by ϵ. However, in CDP settings, a trustworthy server is needed since the server would have access to the raw local updates or gradients to perform perturbations on them.

Local Differential Privacy (LDP)

Each client performs gradient clipping and adds calibrated noise to those gradients to ensure differential privacy. Before sending the local updates to the server for aggregation, each client perturbs their local updates so that their local data is protected with a probability bounded by ϵ. Figure 2.7 shows the flow of FL when LDP is applied. First, each client trains their model. Then, they perform gradient clipping and add calibrated Gaussian noise to those clipped gradients. In the scope of the thesis, LDP is applied as a differential privacy type by using a Python library called Opacus [PyT] which implements DP-SGD [Dwo06].

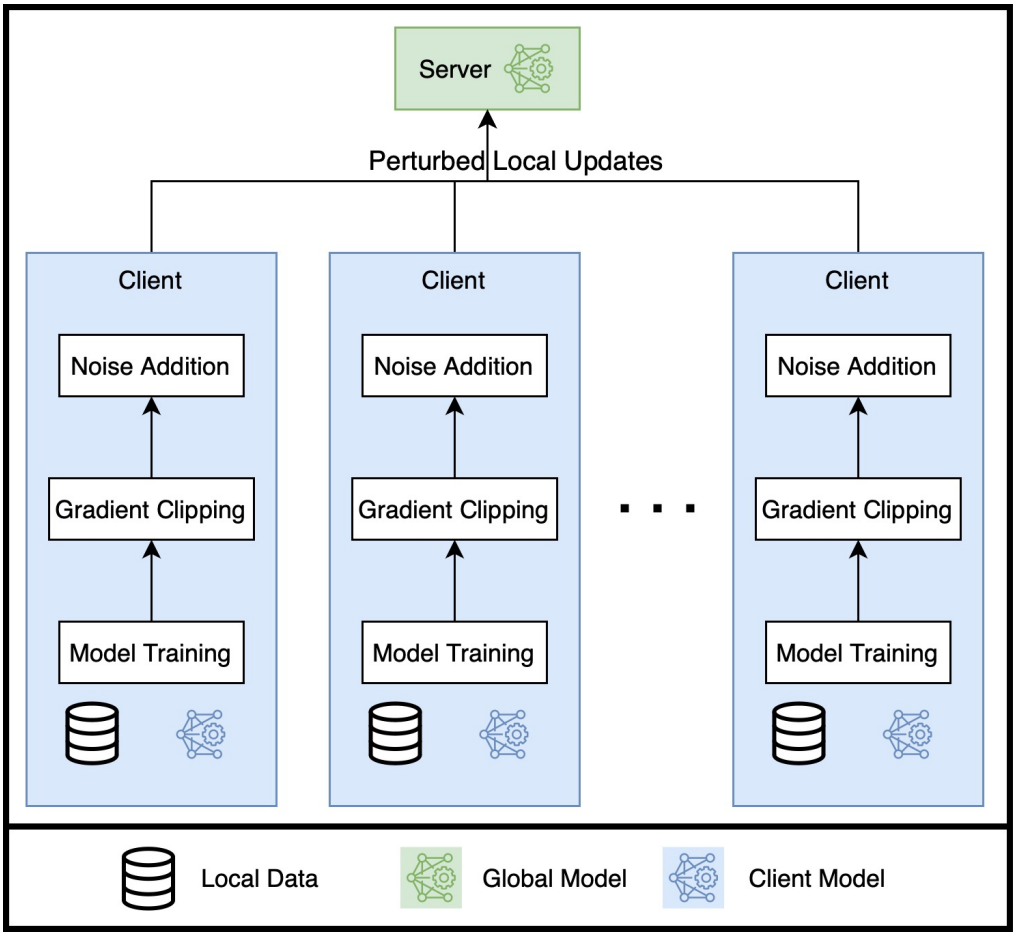

Figure 2.7: The General Flow of FL when LDP is applied.

2.6 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) [Jol02] is a dimensionality reduction technique that is used frequently in multivariate statistical analysis. The main goal of the technique is to represent the basic structure of the data in fewer new variables (components) by considering the correlation between variables in a high-dimensional dataset. PCA computes the new variables (principal components), which are created by combining the original variables linearly, from the highest variance to the lowest variance. The first principal component (PC1) represents the direction that explains the largest piece of the total variance in the data. In contrast, the second principal component (PC2), which is orthogonal to the first principal component, represents the direction that explains the second largest piece of the total variance in the data. This process continues if the condition that each component is orthogonal to the previous component holds.

Mathematically, PCA analyzes the covariance or correlation matrix of the data by decomposing eigenvalues and eigenvectors. Each eigenvector corresponds to one principal component. The amount of variance that the component could explain is expressed by the eigenvalue. One of the most important advantages of PCA is performing dimensionality reduction on the data via considering the structure of the data, and it minimizes the loss of explanatory information. As a result, complex datasets which have numerous variables can be more manageable, and patterns and clusters inside the datasets can be observed clearly. Additionally, it decreases the effects of noise and multicollinearity, allowing analysis to be performed healthier. Especially for cases where the data needs to be visualized for analysis, high-dimensional data can be reduced to two to three components, which helps with visualization.

3 Experiments

In this chapter, we design experiments to evaluate FL under different poisoning attacks, robust aggregation methods, different data partitions among clients, and LDP and show the experiment results.

3.1 Experimental Setup

We simulated a FL setup using two benchmark datasets, which are MNIST [LCB10] and CIFAR-10 [KH09]. The experiments include different data partitions among clients, model architectures, different poisoning attack scenarios, different robust aggregation methods and application of LDP.

3.1.1 Datasets

We experimented a FL setup with the server and different numbers of clients; we targeted an image classification task using 100 clients for the MNIST benchmark dataset and 20 clients for the CIFAR-10 benchmark dataset. To simulate the IID and non-IID data partition among clients, we used a Dirichlet distribution $p_l \sim \mathrm{Dir}_K(\alpha)$. Later, $p_{l,k}$ portion of the training sample of class l is assigned to the client k. For the IID settings, we chose $\alpha = 100$, while for the non-IID settings, we chose $\alpha = 0.5$.

3.1.2 Preprocessing

For MNIST dataset, each 28x28-pixel images are converted into a floating-point tensor in [0,1]. For CIFAR-10 dataset, each 32x32 images are cropped with 4 pixels of zero-padding on each side and then flipped horizontally with a 50% probability. Additionally, per channel normalization is applied with mean [0.4914, 0.4822, 0.4465] and standard deviation [0.2470, 0.2435, 0.2616]. For both datasets, there are 60,000 images in total and 50,000 of them are used in training, and the rest is used as test dataset.

3.1.3 Model Architectures

For the MNIST benchmark dataset, we employed a CNN architecture consisting of two convolutional layers followed by two fully connected layers. The first convolutional layer applies 32 filters of size 3×3 with stride 1 and padding 1, followed by a ReLU activation and a 2×2 max-pooling operation with stride 2. The second convolutional layer applies 64 filters of the same size, again followed by ReLU activation and max-pooling. After the convolutional blocks, the feature maps are flattened into a vector of size $64 \times 7 \times 7$. A fully connected layer with 128 units and ReLU activation is applied, followed by a dropout layer with a dropout rate of 0.25. The final layer is a fully connected layer with 10 output units corresponding to the number of classes, with a log-softmax activation function applied to produce class probabilities. For the CIFAR-10 dataset, we employed a ResNet-18 architecture [HZRS15] that was pre-trained on the ImageNet dataset [DDS+09] according to the suggestion of [RARROR+24]. The final fully connected layer was adapted to output 10 classes corresponding to the CIFAR-10 classification task.

3.1.4 Training Details

For the MNIST benchmark dataset, we performed 80 communication rounds with one local iteration per client. The batch size is set to 64. For the CIFAR-10 benchmark dataset, we conducted 100 communication rounds with one local iteration per client. The batch size is set to 128. We used the standard cross-entropy loss function in all experiments. For the optimization, SGD is always used but its hyperparameters are tuned depending on the dataset or the usage of LDP. For the MNIST dataset, we used SGD with a learning rate of 0.05, a momentum of 0.9, without any weight decay. For the CIFAR-10 dataset, different hyperparameters are experimented, and the hyperparameters that allow us to achieve higher test accuracies are used. To ensure a fair comparison, we tuned the optimizer hyperparameters independently for No-LDP and LDP cases, always reporting the highest accuracy for each experiment. For the CIFAR-10 dataset, we used SGD with a learning rate of 0.05, momentum being 0.9, and weight decay being 4×10^{-5} when LDP is not applied. When LDP is applied, we used SGD with a learning rate of 0.1 without any momentum and weight decay.

3.1.5 Attack Scenarios

Experiments are conducted for each dataset under four different attack scenarios, which are:

- No-attack

- Gaussian Random attack
- IPM attack
- LF attack

Each attack scenario above was also evaluated under the LDP, leading to a total of 8 experimental cases. For both of the benchmark datasets, to simulate the Gaussian Random attack, 20% of the clients are chosen to be malicious and perform the attack at each communication round. For the LF attack, again, 20% of the clients are chosen to be malicious, and 100% of their local data's target labels are flipped according to the mapping shown in Table 3.1. Moreover, for the IPM attack, 40% of the clients are chosen to be malicious and perform the attack with ω being 0.1 (refer to 2.8) at each communication round. Krum and Median robust aggregation methods fail under IPM attack when at least 40% of total clients are malicious when the CIFAR-10 dataset is used. Every robust aggregation method explained in Section 2.4 are evaluated across all experiments.

Original Label	Mapped Label
0	9
1	8
2	7
3	6
4	5
5	4
6	3
7	2
8	1
9	0

Table 3.1: Generic Label Mapping Used in the LF Attack.

3.1.6 LDP Settings

For the LDP settings, we set ϵ as 5 and 8 for MNIST and CIFAR-10 datasets, respectively. The δ parameter is set to 10^{-3} and norm clipping (C) value is set to 2 in both datasets. Except the IPM attack scenario, malicious clients also perform LDP. Benign clients and malicious clients send $\bar{g}_i$ to the server when there is No-attack. Benign clients send $\bar{g}_i$ whereas malicious clients send $\tilde{g}_t$ or $\tilde{\theta}_{t+1}^{(k)}$ to the server under Random Gaussian attack, IPM attack, and LF attack.

3.1.7 Metrics for Client Behavior Analysis

In order to analyze the impact of LDP on aggregation methods, relevant scores and values are logged.

- **First Four Principal Components:** For each of the experiments, we project the full-model parameters into their first four principal components (PCA(4)) for each client and for the server at each epoch. We used principal component analysis in order to understand how LDP affects the clients' updates in the low-dimensional subspace in some of the experiments. In order to log the first four principal components of each clients' model and the server's model after the aggregation, we calculate the each model's PCA projection and only keep the first four principal components.

- **Cosine Similarity Scores:** For the methods such as Clipped Clustering, CosDefense, and Clustering that are based on cosine similarity calculations to exclude the malicious clients, we logged the cosine similarity scores (refer to 2.18 for cosine similarity calculation) at each epoch in order to use them in the analysis of why those methods fail (if they failed) under LDP. For the Clustering and Clipped Clustering aggregation methods, we logged the pair-wise cosine similarity scores between each clients. For the CosDefense aggregation method, we logged the cosine similarity score between the global model's last layer and each clients' models' last layer.

- **MKrum Scores:** For the MKrum aggregation method, the MKrum scores of each client are logged at each epoch. MKrum score of a client i can be described as the sum of the squared distances to its $K - M - 2$ closest clients. The MKrum scores of each client are logged in order to analyze the effect of LDP on MKrum scores if the method fails under LDP.

- **Coordinate-wise Contribution:** For the Median aggregation method, the number of coordinate-wise contributions of each clients' updates is logged at each epoch in order to understand the effect of LDP on the distribution of clients' updates inside the aggregated update.

3.2 Results

In this section, the test accuracy results and the highest achieved test accuracy throughout the communication rounds for each experiment are shown.

3.2.1 Test Accuracy Results for MNIST

Table 3.2 and 3.3 report test-accuracy results for eleven different aggregation rules on MNIST, evaluated with 100 clients across four attack scenarios (No-attack, Random Gaussian attack, IPM attack, and LF attack) both with and without LDP under an IID and non-IID data split, respectively. Each row corresponds to one aggregation method (FedAvg, Krum, MKrum, Trimmed Mean, Median, Geometric Median, Centered Clipping, Clustering, Clipped Clustering, CosDefence, and Bulyan). Each column block indicates a particular attack, and within each block, there are two subcolumns: "w/out LDP" and "with LDP."

Method	**No-attack**		**Random Gaussian**	
	w/out LDP	with LDP	w/out LDP	with LDP
FedAvg	98.89 (98.93)	92.82 (92.85)	9.80 (10.18)	12.56 (27.04)
Krum	94.09 (94.15)	80.53* (81.28)	94.29 (94.62)	79.99* (80.43)
MKrum	98.77 (98.82)	92.36 (92.36)	98.86 (98.86)	92.29 (92.51)
Trimmed Mean	98.93 (98.93)	92.82 (92.84)	98.88 (98.88)	92.24 (92.39)
Median	98.74 (98.74)	91.29 (91.29)	98.80 (98.80)	90.91 (90.91)
Geometric Median	98.92 (98.84)	92.85 (92.86)	98.72 (98.75)	86.83 (88.24)
Centered Clipping	98.96 (98.98)	67.92* (67.92)	98.84 (98.84)	66.38* (66.38)
Clustering	98.70 (98.71)	92.56 (92.56)	98.76 (98.76)	92.05 (92.05)
ClippedClustering	98.90 (98.93)	91.26 (91.29)	95.13 (95.37)	65.36 (66.16)
Cos Defence	98.84 (98.89)	92.20 (92.20)	90.44* (94.81)	56.03* (63.24)
Bulyan	98.89 (98.92)	92.82 (92.83)	98.69 (98.72)	91.96 (91.96)
Method	**IPM**		**LF**	
	w/out LDP	with LDP	w/out LDP	with LDP
FedAvg	98.60 (98.60)	89.88 (89.88)	98.26 (98.80)	91.04 (91.07)
Krum	10.32 (10.32)	9.80* (13.86)	94.17 (94.53)	79.06* (80.07)
MKrum	94.71 (94.71)	71.93 (72.17)	98.87 (98.87)	89.84 (89.84)
Trimmed Mean	97.70 (97.77)	86.65 (86.66)	98.56 (98.59)	91.02 (91.02)
Median	94.70 (94.70)	7.73 (12.17)	98.57 (98.59)	88.50 (88.60)
Geometric Median	98.53 (98.53)	89.66 (89.66)	93.46 (98.82)	91.10 (91.15)
Centered Clipping	98.52 (98.53)	57.21* (57.21)	98.86 (98.86)	66.04* (66.04)
Clustering	93.94 (93.94)	0.84 (12.17)	98.63 (98.64)	90.87 (90.87)
ClippedClustering	98.58 (98.59)	82.98 (83.54)	98.95 (98.95)	89.84 (90.13)
Cos Defence	98.89 (98.89)	91.39 (91.39)	98.94 (98.96)	90.68 (90.74)
Bulyan	*Cannot run with 40% malicious clients*		98.75 (98.75)	89.51 (89.51)

Note: The value outside parentheses denotes final test accuracy, and the value inside denotes the best accuracy achieved during communication rounds. Asterisk (*) indicates a reduced learning rate of 0.005 was used. Results highlighted in red indicate a significant decline in performance, with accuracies falling below 70% considered as notably degraded.

Table 3.2: Performance of aggregation methods on MNIST (IID) under various attack scenarios, with and without LDP.

The first value tabulated is the final model's test accuracy after completing the training; the value tabulated inside parentheses is the best test accuracy reached at any communication round. The best and final test accuracies for each aggregation method do not differ much in most cases. However, for the CosDefense aggregation, method when the data distribution is non-IID under a Random Gaussian attack, the best and the final test accuracies differ by approximately 50% for without LDP case and more than 20% for with LDP case because when the method misdetects any malicious clients, the effect of the attack on the accuracy is massive (note that the attack is quite powerful). Also, the best test accuracy and final test accuracy of the MKrum aggregation method when the data distribution is non-IID under IPM attack differs by 20% for No-LDP.

Method	**No-attack**		**Random Gaussian**	
	w/out LDP	with LDP	w/out LDP	with LDP
FedAvg	98.85 (98.87)	90.56 (90.79)	9.80 (14.06)	12.75 (20.97)
Krum	95.19 (95.21)	30.14* (30.14)	94.22 (94.51)	29.94* (30.79)
MKrum	98.31 (98.37)	88.60 (89.13)	98.50 (98.50)	89.50 (89.50)
Trimmed Mean	98.82 (98.83)	90.62 (90.62)	98.74 (98.76)	90.99 (90.99)
Median	98.37 (98.37)	88.52 (90.62)	98.41 (98.41)	88.41 (88.58)
Geometric Median	98.83 (98.83)	90.60 (90.60)	98.74 (98.74)	84.19 (85.69)
Centered Clipping	98.79 (98.82)	67.67 (67.67)	98.72 (98.72)	58.30* (58.30)
Clustering	98.32 (98.32)	90.24 (90.24)	98.32 (98.36)	90.22 (90.22)
ClippedClustering	98.82 (98.82)	79.39 (79.63)	93.85 (94.30)	61.05 (63.58)
Cos Defence	98.54 (98.57)	89.82 (89.85)	42.37* (92.10)	28,68 (51.50)
Bulyan	98.86 (98.86)	90.62 (90.62)	98.13 (98.17)	89.05 (89.05)
Method	**IPM**		**LF**	
	w/out LDP	with LDP	w/out LDP	with LDP
FedAvg	98.41 (98.43)	89.13 (89.13)	95.14 (98.55)	89.33 (89.42)
Krum	11.35 (11.35)	7.54* (14.10)	93.64 (93.89)	30.14* (30.14)
MKrum	11.35 (35.69)	10.73 (18.26)	98.42 (98.46)	83.25 (83.28)
Trimmed Mean	96.73 (96.73)	83.51 (83.51)	98.42 (98.45)	88.73 (88.73)
Median	19.30 (27.03)	9.74 (11.36)	98.26 (98.45)	84.96 (85.25)
Geometric Median	98.07 (98.09)	88.06 (88.06)	98.43 (98.47)	89.12 (89.12)
Centered Clipping	98.11 (98.11)	21.52* (21.52)	98.65 (98.65)	52.00* (52.00)
Clustering	11.45 (18.69)	9.74 (11.36)	98.23 (98.23)	87.97 (87.97)
ClippedClustering	98.09 (98.09)	10.10 (21.32)	98.13 (98.26)	76.55 (77.18)
Cos Defence	98.79 (98.82)	90.28 (90.34)	98.75 (98.75)	86.99 (87.11)
Bulyan	*Cannot run with 40% malicious clients*		98.24 (98.24)	83.06 (83.19)

Note: The value outside parentheses denotes final test accuracy, and the value inside denotes the best accuracy achieved during communication rounds. Asterisk (*) indicates a reduced learning rate of 0.005 was used. Results highlighted in red indicate a significant decline in performance, with accuracies falling below 70% considered as notably degraded.

Table 3.3: Performance of aggregation methods on MNIST (non-IID) under various attack scenarios, with and without LDP.

Trimmed Mean, Geometric Median and Bulyan aggregation methods perform well under all attack scenarios, both in IID and non-IID with and without applying LDP. The FedAvg aggregation method under a Random Gaussian attack and the Krum aggregation method under an IPM attack have the worst performance when the data distribution is IID or non-IID, both with and without LDP cases. Under non-IID data distribution, the CosDefense aggregation method under a Random Gaussian attack and MKrum, Median, and Clustering aggregation methods under an IPM attack have significant performance drops in both LDP cases (without and with LDP).
In many experiments, the test accuracies are high when LDP is not applied. However, when we apply LDP, we see slight decreases in the test accuracies in most experiments because of the perturbation performed on the local updates through gradient clipping and Gaussian noise addition. Nevertheless, some of the aggregation methods suffer a performance drop when LDP is applied.

1. **No-attack:** In both IID and non-IID cases, the Centered Clipping aggregation method has a huge drop from 98.96% and 98.79% to 67.92% and 67.67% accuracies when LDP is applied, respectively. Moreover, in non-IID case, the Krum method has a significant decrease in the test accuracy from 95.19% to 30.14% when LDP is applied.

2. **Random Gaussian:** In both IID and non-IID cases, the test accuracies of Centered Clipping and Clipped Clustering methods decrease from 98.84% and 98.72%, and 95.13% and 93.85% to 66.38% and 58.30%, and 65.36% and 61.05% when LDP is applied, respectively. Additionally, in IID case, the CosDefense aggregation method has a huge drop from 90.44% to 56.03% when LDP is applied. Moreover, in non-IID case, when LDP is applied, the test accuracy of Krum aggregation method decreases from 94.22% to 29.94%.

3. **IPM:** In both IID and non-IID cases, the Centered Clipping aggregation method has a huge drop from 98.52% and 98.11% to 57.21% and 21.52% accuracies when LDP is applied, respectively. Additionally, for the IID case, the Median and Clustering aggregation methods' test accuracies have a significant drop from 94.70% and 93.94% to 7.73% and 0.84% when LDP is applied, respectively. Furthermore, the Clipped Clustering method's test accuracy decreases fundamentally from 98.09% to 10.10% when LDP is applied in non-IID case.

4. **LF:** In both IID and non-IID cases, the Centered Clipping aggregation method has a huge drop from 98.86% and 98.65% to 66.04% and 52.00% accuracies when LDP

is applied, respectively. Additionally, in non-IID case, when LDP is applied, the test accuracy of Krum aggregation method decreases from 93.64% to 30.14%.

3.2.2 Test Accuracy Results for CIFAR-10

Table 3.4 and 3.5 report test-accuracy results for eleven different aggregation rules on CIFAR-10, evaluated with 20 clients across four attack scenarios (No-attack, Random Gaussian attack, IPM attack, and LF attack) both with and without LDP under an IID and non-IID data split, respectively. Each row corresponds to one aggregation method (FedAvg, Krum, MKrum, Trimmed Mean, Median, Geometric Median, Centered Clipping, Clustering, Clipped Clustering, CosDefence, and Bulyan). Each column block indicates a particular attack, and within each block, there are two subcolumns: "w/out LDP" and "with LDP."

The value outside parentheses is the final model's test accuracy after completing the training; the value inside parentheses is the best test accuracy reached at any communication round. The best and final test accuracies for each aggregation method do not differ much in most cases. However, for Median and Krum aggregation methods under IPM attack without LDP, the best and final test accuracies differ by more than 15% in both data distribution cases. Also, the best and final test accuracy of the CosDefense aggregation method under a Random Gaussian attack when the data distribution is IID and LDP is applied differs by more than 25%. For FedAvg, Clipped Clustering, and CosDefense methods under LF attack without LDP, the best and final test accuracies differ by more than 20% when the data distribution is non-IID. Lastly, the best and final test accuracy of the CosDefense aggregation method under Random Gaussian attack when the data distribution is IID and LDP is applied differs by more than 25%.

Trimmed Mean and Geometric Median aggregation methods perform well under all attack scenarios, both in IID and non-IID with and without applying LDP. Under both data distribution cases, Krum and Centered Clipping aggregation methods under No-attack scenario, FedAvg, Krum, Centered Clipping, and Clipped Clustering aggregation methods under a Random Gaussian attack, Krum, Median, Centered Clipping, and Clustering under an IPM attack, and Krum and Centered Clipping under a LF attack have the worst performance in both LDP cases (without and with LDP). Additionally, under non-IID data distribution, the CosDefense aggregation method under a Random Gaussian attack and a LF attack has significant performance drops in both LDP cases (without and with LDP). Lastly, the Clipped Clustering aggregation method under a LF attack has severe performance drops in both LDP cases (without and with LDP).

In many experiments, the test accuracies are high when LDP is not applied. However,

Method	No-attack		Random Gaussian	
	w/out LDP	with LDP	w/out LDP	with LDP
FedAvg	82.18 (82.52)	61.38 (62.14)	12.52 (12.52)	10.00 (11.51)
Krum	61.24 (63.37)	35.64 (36.57)	62.19 (65.55)	35.34 (36.68)
MKrum	79.27 (80.39)	56.41 (56.97)	79.52 (80.10)	57.03 (57.45)
Trimmed Mean	82.22 (82.41)	61.33 (62.17)	81.13 (81.30)	57.61 (57.91)
Median	78.18 (78.95)	10.19 (14.49)	72.32 (76.46)	11.27 (13.37)
Geometric Median	82.30 (82.86)	61.34 (62.21)	78.88 (79.47)	53.03 (53.36)
Centered Clipping	58.98 (58.98)	24.98 (25.02)	54.62 (54.62)	23.09 (23.09)
Clustering	81.89 (82.10)	58.54 (60.43)	81.00 (81.31)	56.59 (57.87)
ClippedClustering	82.26 (82.48)	61.02 (61.83)	10.00 (10.00)	10.00 (23.35)
Cos Defence	80.37 (80.65)	59.25 (59.32)	80.10 (80.77)	11.75 (39.03)
Bulyan	82.17 (82.93)	61.31 (62.17)	80.15 (80.77)	53.81 (55.13)
Method	**IPM**		**LF**	
	w/out LDP	with LDP	w/out LDP	with LDP
FedAvg	80.54 (81.13)	57.68 (57.68)	80.54 (81.37)	57.69 (58.02)
Krum	10.00 (38.09)	10.00 (10.00)	44.15 (56.51)	36.23 (36.42)
MKrum	74.81 (76.16)	14.43 (15.18)	76.32 (78.92)	31.81 (37.58)
Trimmed Mean	80.17 (80.25)	51.27 (51.27)	81.48 (81.74)	57.19 (57.67)
Median	12.26 (45.18)	10.00 (10.12)	75.53 (77.62)	11.87 (13.86)
Geometric Median	80.52 (80.71)	56.58 (56.68)	79.54 (81.43)	57.47 (58.01)
Centered Clipping	26.67 (26.67)	10.00 (10.00)	58.05 (58.05)	20.44 (20.46)
Clustering	33.55 (37.47)	10.00 (10.04)	81.36 (81.93)	55.50 (55.51)
ClippedClustering	80.49 (80.63)	55.62 (55.72)	81.00 (81.39)	56.61 (57.05)
Cos Defence	78.12 (78.49)	57.74 (58.18)	80.46 (80.46)	51.57 (55.15)
Bulyan	*Cannot run with 40% malicious clients*		79.80 (79.83)	51.15 (55.13)

Note: The value outside parentheses denotes final test accuracy, and the value inside denotes the best accuracy achieved during communication rounds. Results highlighted in red indicate a significant decline in performance, with accuracies falling below 70% and 50% considered as notably degraded for w/out LDP and with LDP, respectively.

Table 3.4: Performance of aggregation methods on CIFAR-10 (IID) under various attack scenarios, with and without LDP.

when we apply LDP, we see moderate decreases in the test accuracies in most experiments because of the perturbation performed on the local updates through gradient clipping and Gaussian noise addition. Nevertheless, some of the aggregation methods suffer a performance drop when LDP is applied.

1. **No-attack:** In both IID and non-IID cases, the Median aggregation method has a huge drop from 78.18% and 72.86% to 10.19% and 10.00% accuracies when LDP is applied, respectively. Moreover, in non-IID case, the MKrum method has a significant decrease in the test accuracy from 70.81% to 39.77% when LDP is applied.

Method	No-attack		Random Gaussian	
	w/out LDP	with LDP	w/out LDP	with LDP
FedAvg	75.75 (77.66)	58.55 (58.55)	10.58 (12.94)	9.30 (11.48)
Krum	44.75 (48.78)	10.00 (10.00)	39.45 (41.41)	10.00 (10.00)
MKrum	70.81 (71.43)	39.77 (39.98)	68.78 (69.15)	39.67 (40.22)
Trimmed Mean	77.30 (78.63)	57.24 (57.24)	75.61 (76.59)	50.55 (50.73)
Median	72.86 (75.44)	10.00 (10.32)	62.30 (72.66)	10.00 (13.39)
Geometric Median	76.41 (77.81)	57.24 (57.24)	71.53 (72.32)	50.27 (50.27)
Centered Clipping	48.10 (48.10)	10.54 (13.61)	44.43 (44.43)	13.41 (14.86)
Clustering	78.23 (78.23)	53.07 (53.07)	76.11 (76.15)	50.32 (50.40)
ClippedClustering	73.05 (75.10)	51.56 (51.56)	10.00 (10.00)	10.00 (13.94)
Cos Defence	65.14 (70.33)	47.47 (50.87)	28.44 (64.05)	10.00 (39.02)
Bulyan	77.13 (78.32)	57.26 (57.26)	73.48 (73.73)	50.06 (51.75)
Method	**IPM**		**LF**	
	w/out LDP	with LDP	w/out LDP	with LDP
FedAvg	71.52 (73.27)	47.00 (47.65)	52.55 (73.54)	46.84 (46.84)
Krum	10.00 (31.12)	10.00 (10.00)	19.76 (33.18)	10.00 (10.00)
MKrum	61.48 (62.42)	13.49 (15.09)	66.24 (67.90)	37.15 (37.16)
Trimmed Mean	72.70 (73.05)	45.31 (46.31)	71.63 (72.86)	46.90 (46.90)
Median	3.23 (17.29)	10.00 (10.00)	68.71 (72.03)	10.00 (15.74)
Geometric Median	73.60 (74.68)	47.01 (47.01)	61.70 (75.76)	48.78 (48.78)
Centered Clipping	28.01 (28.01)	10.00 (10.00)	48.03 (48.03)	15.92 (15.93)
Clustering	8.53 (16.60)	10.00 (10.00)	72.99 (73.22)	44.68 (44.68)
ClippedClustering	59.20 (65.73)	37.55 (37.85)	47.28 (69.21)	40.70 (40.96)
Cos Defence	56.55 (71.25)	48.04 (50.36)	35.77 (57.81)	40.56 (42.00)
Bulyan	*Cannot run with 40% malicious clients*		68.87 (70.66)	38.22 (40.30)

Note: The value outside parentheses denotes final test accuracy, and the value inside denotes the best accuracy achieved during communication rounds. Results highlighted in red indicate a significant decline in performance, with accuracies falling below 50% and 45% considered as notably degraded for w/out LDP and with LDP, respectively.

Table 3.5: Performance of aggregation methods on CIFAR-10 (non-IID) under various attack scenarios, with and without LDP.

2. **Random Gaussian:** In both IID and non-IID cases, the Median aggregation method has a huge drop from 72.32% and 62.30% to 11.27% and 10.00% accuracies when LDP is applied, respectively. Additionally, in IID case, the CosDefense aggregation method has a huge drop from 80.10% to 11.75% when LDP is applied. Moreover, in non-IID case, when LDP is applied, the test accuracy of MKrum aggregation method decreases from 68.78% to 39.67%.

3. **IPM:** In both IID and non-IID cases, the MKrum aggregation method has a huge drop from 74.81% and 61.48% to 14.43% and 13.49% accuracies when LDP is ap-

plied, respectively. Additionally, in non-IID case, the Clipped Clustering method's test accuracy decreases fundamentally from 59.20% to 37.55% when LDP is applied

4. **LF:** In both IID and non-IID cases, the test accuracies of MKrum and Median methods decrease from 76.32% and 66.24%, and 75.53% and 68.71% to 31.81% and 37.15%, and 11.87% and 10.00% when LDP is applied, respectively. Additionally, in non-IID case, when LDP is applied, the test accuracy of Bulyan aggregation method decreases from 68.87% to 38.22%.

4 Empirical Analysis

This chapter empirically analyzes why some robust aggregation methods suffer a performance drop when LDP is applied. The chapter is divided into four sections, where the experimental results of each dataset under different data partitions are analyzed.

4.1 Robust Aggregation Methods - MNIST - IID

This section refers to Table 3.2 and analyzes performance drops of robust aggregation methods when data partition is IID under LDP.

4.1.1 Median Aggregation under IPM Attack

Median achieves 94.7% test accuracy under IPM attack when LDP is not applied but sharply declines to 7.73% in the LDP case. Figure 4.1 shows, for Epochs 1 and 2, the projections of the clients' local updates (benign and malicious) and the server's aggregated update onto the first two principal components, comparing the cases without and with LDP under an IPM attack.

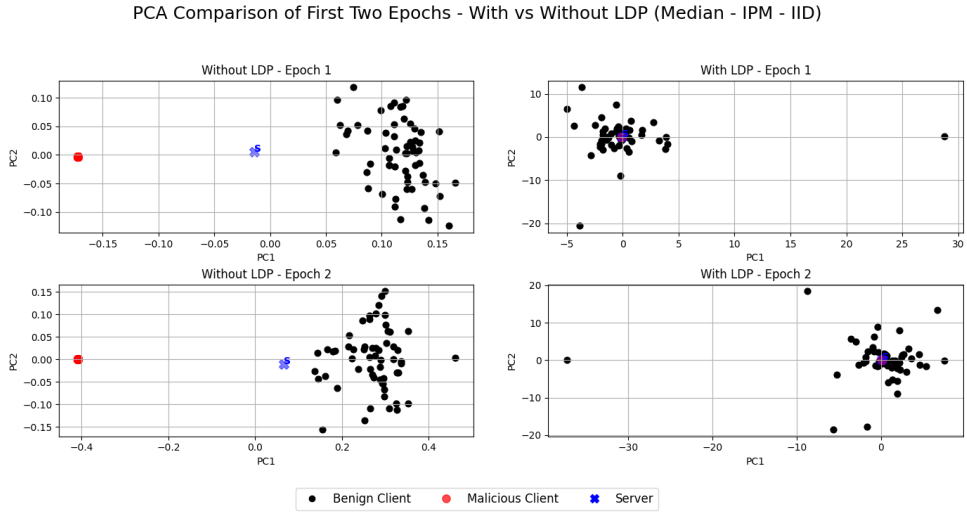

Figure 4.1: Median: PCA Comparison of First Two Epochs – With vs Without LDP (IID - IPM Attack).

As can be seen from Figure 4.1, without LDP (left subplots), there is a distinct separation between benign and malicious clients. Since Median works well when the benign clients are tightly clustered, it can achieve high test accuracy under IPM attack when LDP is not applied. On the other hand, when LDP is applied (right subplots), the benign clients are scattered around, and this causes Median's assumption that benign clients are tightly clustered to break down. With the IPM attack, malicious clients send the same gradients to the server and are at the center under LDP. As a result, malicious clients mislead the Median aggregation method, and the method achieves lower test accuracy when LDP is applied.

4.1.2 Centered Clipping Aggregation under All Attack Scenarios

Centered Clipping achieves 98.96% test accuracy in the No-LDP case but sharply declines to 67.92% in the LDP case when there is No-attack. Figure 4.2 shows the mean PCA variance across clients for the first principal components and the mean client-to-server distance in PCA space, comparing the effect of LDP when there is No-attack.

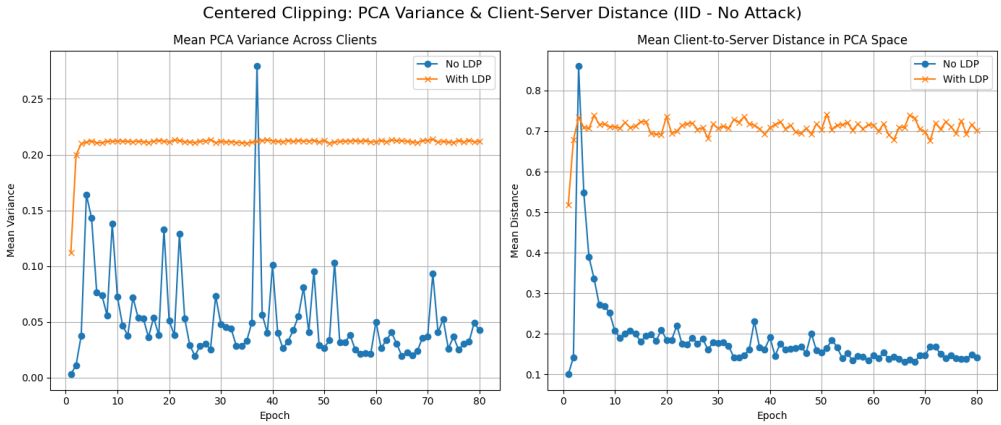

Figure 4.2: Centered Clipping: PCA Variance & Client-Server Distance – With vs Without LDP (IID - No-attack).

The work in [SGS+22] states that large variances across clients can slow down the convergence of the global model. As shown in Figure 4.2, the variance across clients when LDP is applied is higher compared to the case where LDP is not applied (left subplot), and the variance between the server and the clients increases when LDP is applied compared to the case where LDP is not applied (right subplot). This variance change occurs because of double clipping effects. First, LDP performs gradient clipping locally, and then the aggregation method performs clipping again at the server, disrupting the variance across clients. As a result, the Centered Clipping achieves lower test accuracy when

LDP is applied compared to the No-LDP case. A similar test accuracy drop happens in Centered Clipping when IPM (98.52% to 57.21%) and LF (98.86% to 66.04%) attacks are performed under the LDP. The reason is exactly the same when there is No-attack. The mean PCA variance across clients for the first principal components and the mean client-to-server distance in PCA space for IPM and LF attacks' figures can be seen in Figures 4.3 and 4.4, respectively.

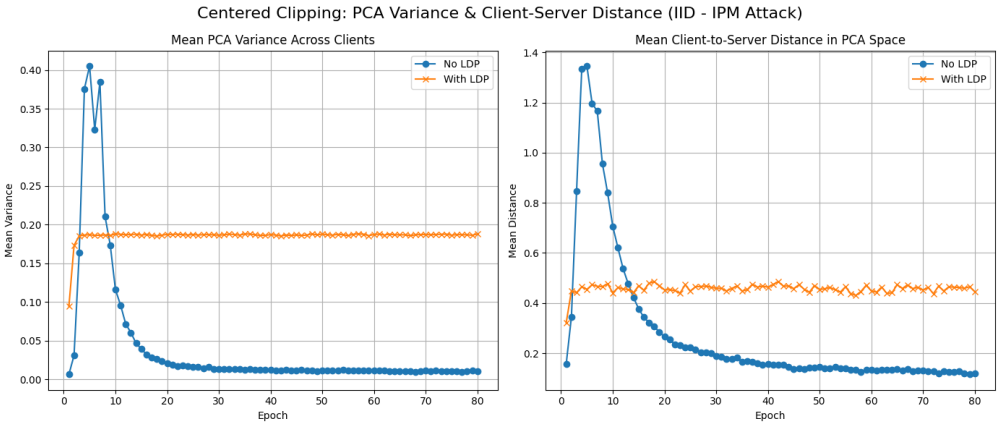

Figure 4.3: Centered Clipping: PCA Variance & Client-Server Distance – With vs Without LDP (IID - IPM Attack).

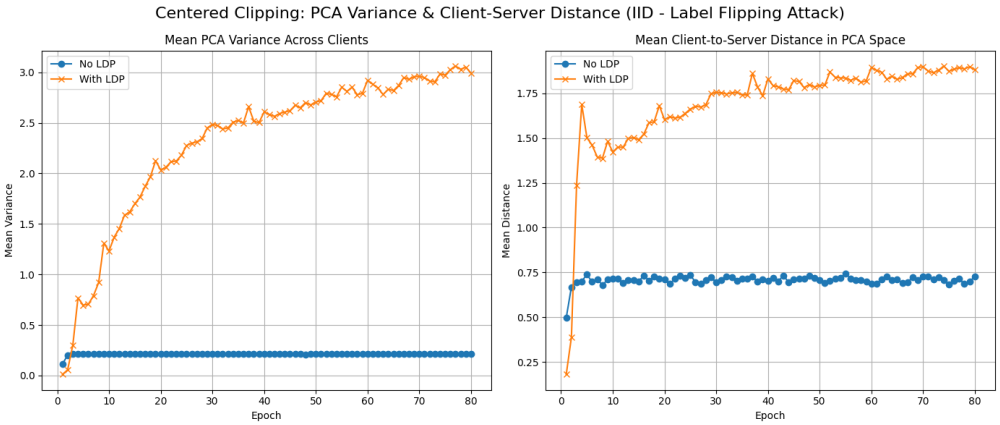

Figure 4.4: Centered Clipping: PCA Variance & Client-Server Distance – With vs Without LDP (IID - LF Attack).

Moreover, a similar accuracy drop happens in Centered Clipping when a Random Gaussian attack (from 98.84% to 66.38%) is performed under the LDP. The reason is the same, but it cannot be shown through the mean PCA variance across clients' plots because of the scaling issues. Figure 4.5 shows the density of PCA variances, comparing the effect

of LDP when Random Gaussian attack is performed. When LDP is applied (red bin), the density of PCA variance is higher than in the case where LDP is not applied (blue bin). Since the work in [SGS+22] states that large variance across clients can slow down the convergence of the global model, Centered Clipping suffers a performance drop when LDP is applied under Random Gaussian attack.

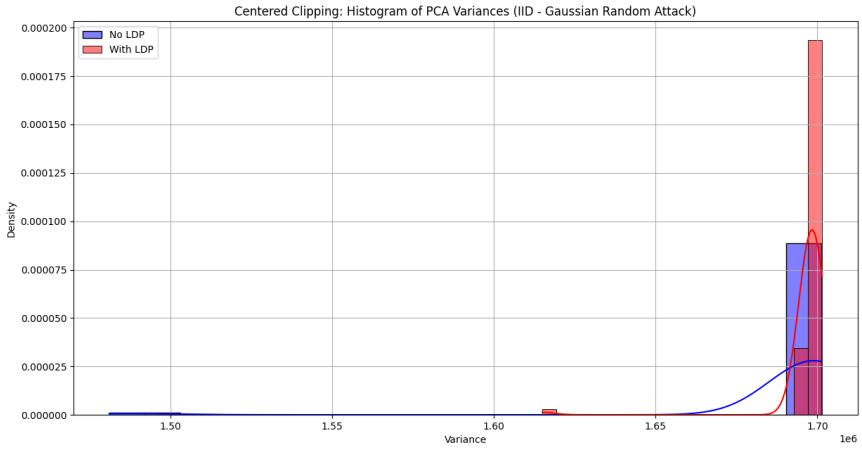

Figure 4.5: Centered Clipping: Density of PCA Variances – With vs Without LDP (IID - Gaussian Random Attack).

4.1.3 Clustering Aggregation under IPM Attack

Clustering achieves 93.94% test accuracy under IPM attack when LDP is not applied but sharply declines to 0.84% in the LDP case. Figure 4.6 shows, for Epochs 1 and 2, the projections of the clients' local updates (benign and malicious) and the server's aggregated update onto the first two principal components, comparing the cases without and with LDP under the IPM attack.

As can be seen from Figure 4.6, without LDP (left subplots), there is a distinct separation between benign and malicious clients. Since Clustering works well when the benign clients are tightly clustered and malicious clients are distinct from benign clients, it can achieve high test accuracy under IPM attack when LDP is not applied. On the other hand, when LDP is applied (right subplots), the benign clients are scattered around, and the malicious clients blend in with the benign clients. As a result, the Clustering method cannot distinguish malicious and benign clients from each other and uses all malicious clients to create the global update. Therefore, malicious clients mislead the Clustering method, and the method achieves low test accuracy when applying LDP.

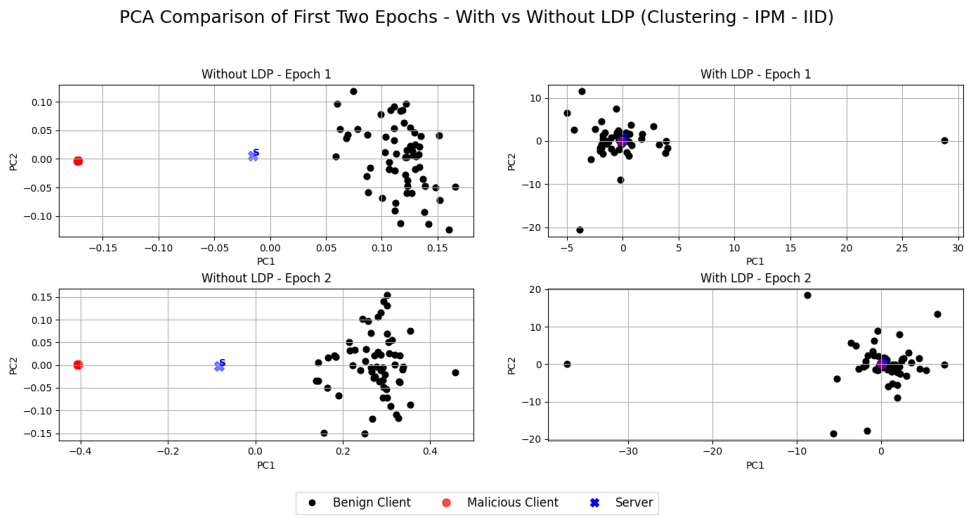

Figure 4.6: Clustering: PCA Comparison of First Two Epochs – With vs Without LDP (IID - IPM).

4.1.4 Clipped Clustering Aggregation under Random Gaussian Attack

Clipped Clustering achieves 95.13% test accuracy under Random Gaussian attack in the No-LDP case but sharply declines to 65.36% in the LDP case. Figure 4.7 shows the mean cosine similarity within the benign and malicious clusters and the mean cosine similarity between benign and malicious clients across 80 epochs, comparing the effect of LDP.

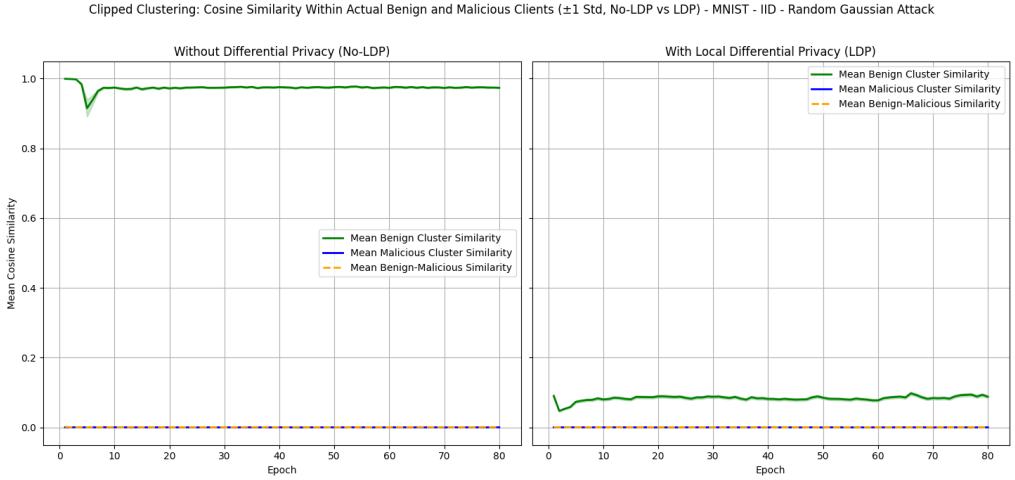

Figure 4.7: Clipped Clustering: The Mean of Cosine Similarity Within Benign and Malicious Clusters – With vs Without LDP (IID - Random Gaussian Attack).

On the left subplot, which shows the No-LDP case, the mean cosine similarity of the

benign cluster (green line) is high, meaning benign clients are highly similar throughout the 80 epochs. In contrast, the mean cosine similarity of the malicious cluster (blue line) is low, meaning they are highly dissimilar throughout the 80 epochs. The mean cosine similarity between malicious and benign clients (orange dashed line) is also low, meaning that benign and malicious clients are highly dissimilar. As a result, the Clipped Clustering method can correctly identify benign clients to use when creating the global update in the No-LDP case. On the other hand, on the right subplot, which shows the LDP case, both the mean cosine similarity of benign cluster (green line) and malicious cluster (blue line) are low, which means that they are dissimilar from each other. Since LDP clips the clients' gradients and adds calibrated Gaussian noise to the clients' clipped gradients, benign clients are not similar to each other anymore, and the Clipped Clustering is based on clustering the benign and malicious clients into two clusters using the cosine similarity scores. However, the Clipped Clustering method cannot cluster benign and malicious clients correctly because none of them are similar anymore. As a result, the method uses malicious clients to create the global update, which disrupts the global model's performance.

Additionally, the Agglomerative Clustering method used in the Clipped Clustering algorithm uses average linkage, which computes the similarity between two clusters as the mean distance of all pairs of points found in the clusters to merge clusters. With the effect of LDP, the mean distance calculation is affected significantly, leading the Agglomerative Clustering algorithm to misclassify the malicious clients as benign clients. On the other hand, Agglomerative Clustering with complete linkage, which clusters nodes by calculating the similarity as the maximum distance between any two points from the two clusters, was experimented with the same settings to be sure that the problem occurs because of the usage of average linkage. From the experiment result, the Clipped Clustering method achieved higher test accuracy when switching to the complete linkage because complete linkage forms a tighter and more compact cluster and excludes outliers such as malicious clients or some benign clients that became outliers because of LDP.

4.1.5 CosDefense Aggregation under Random Gaussian Attack

CosDefense achieves 90.44% test accuracy under Random Gaussian attack in the No-LDP case but sharply declines to 56.03% in the LDP case. Figure 4.8 shows the mean of normalized cosine similarity scores between benign and malicious clients and the server across 80 epochs, comparing the effect of LDP.

The dashed line is the threshold, which means that if the cosine similarity score of the client exceeds that point, it is considered a benign client by the aggregation method.

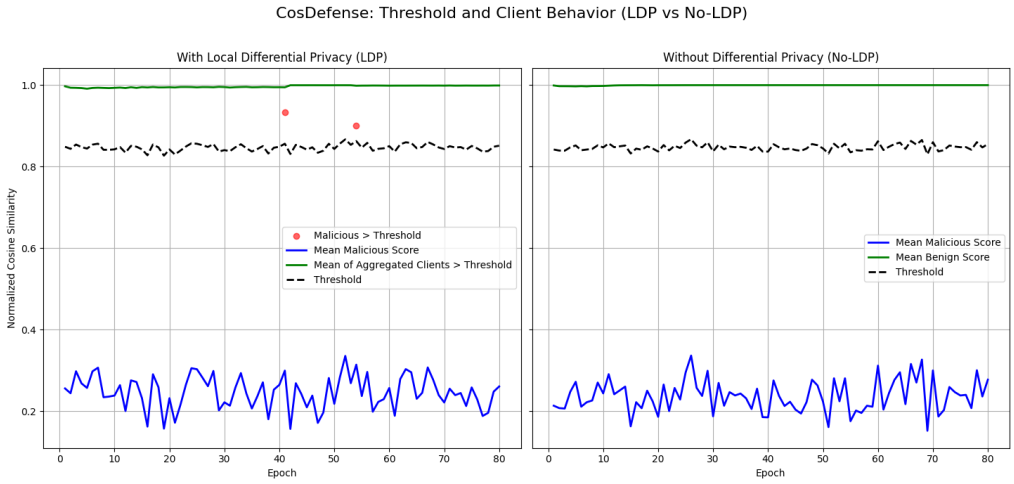

Figure 4.8: CosDefense: The Mean of Cosine Similarity Scores between Benign and Malicious Clients and the Server – With vs Without LDP (IID - Random Gaussian Attack).

On the left subplot, which shows the case where LDP is applied, the mean of the cosine similarity score between malicious clients and the server (blue line) is consistently lower than the threshold, and the mean of the cosine similarity score between benign clients and the server (green line) is consistently higher than the threshold. However, in two epochs, the aggregation method misses some malicious clients because of LDP. Since LDP clips the gradients and adds calibrated Gaussian noise to those gradients, and the attack is a Random Gaussian attack, some of the malicious clients could be similar to the server, and they could be used in the aggregation, which results in the disruption of the global model. On the right subplot, which shows the case where LDP is not applied, again, the mean of the cosine similarity score between malicious clients and the server (blue line) is consistently lower than the threshold. The mean of the cosine similarity score between benign clients and the server (green line) is consistently higher than the threshold. However, when LDP is not applied, the aggregation method does not consider any malicious client as a benign client and only uses benign clients in the aggregation.

4.2 Robust Aggregation Methods - MNIST - non-IID

This section refers to Table 3.3 and analyzes performance drops of robust aggregation methods when data partition is non-IID under LDP.

4.2.1 Krum Aggregation under All Attack Scenarios

In the No-LDP case, Krum achieves 95.19%, 94.22%, and 93.64% test accuracies under No-attack, Random Gaussian attack, and LF attack, respectively, but in the LDP case, these test accuracies decline sharply to 30.14%, 29.94%, and 30.14%, respectively. Throughout the training, the Krum method chooses the same client to update the global model under LDP. This client has one of the smallest sample sizes among the other clients. Most of its samples belong to a single class, digit 7, and some other classes do not have any samples. Therefore, the local update that the client creates is simple and concentrated on one class. However, to be sure that the selection of that client is not specific to the data partition among clients, different seed numbers are tried. In other experiments with different seed numbers, the Krum method again chooses a specific client throughout the training to update the global model under LDP. The client chosen by the Krum method in different seeds has the smallest sample size compared to the other clients. Moreover, to understand which part of LDP, gradient clipping or adding calibrated Gaussian noise to those gradients causes this, the same experiment was performed only with the gradient clipping part of LDP, and calibrated Gaussian noise was set to 0. The result showed that the effect of gradient clipping was too low, and the Krum method was able to choose different clients throughout the training and achieved higher test accuracies. Later, the same experiment was performed only by adding calibrated noise to the gradients, and the norm clipping value was set to high values to restrict the effect of gradient clipping. The result showed that the Krum method chose the same client throughout the training. Consequently, the local updates created by the client that has the smallest sample size are simple, and the effect of LDP on those local updates would be simpler. Therefore, the client is the closest one to the other clients and has been chosen to update the global model. Since the local data distribution of the selected client is highly skewed, and it is chosen to update the global model, the global model has poor generalization. Therefore, when the data partition is non-IID, Krum suffers a performance drop under LDP in No-attack, Random Gaussian attack, and LF attack scenarios.

4.2.2 Centered Clipping Aggregation under All Attack Scenarios

Centered Clipping achieves 98.79%, 98.72%, 98.11%, and 98.65% test accuracies in the No-LDP case but has a sharp decline to 67.67%, 58.30%, 21.52%, and 52.00% in the LDP case when there is No-attack, Random Gaussian attack, IPM attack, and LF attack, respectively. For each case, the reason is precisely the same as the IID case described in Section 4.1.2. Corresponding plots for No-attack, Random Gaussian attack, IPM attack, and LF attack are shown in Figures 4.9, 4.10, 4.11, and 4.12, respectively.

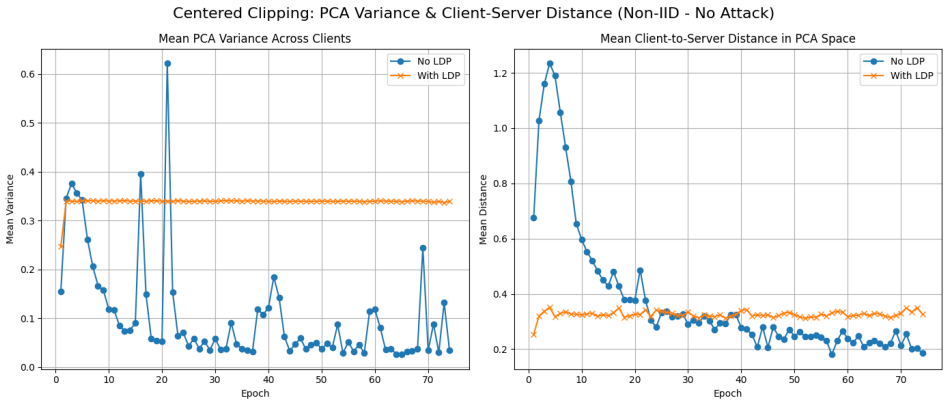

Figure 4.9: Centered Clipping: PCA Variance & Client-Server Distance – With vs Without LDP (non-IID - No-attack).

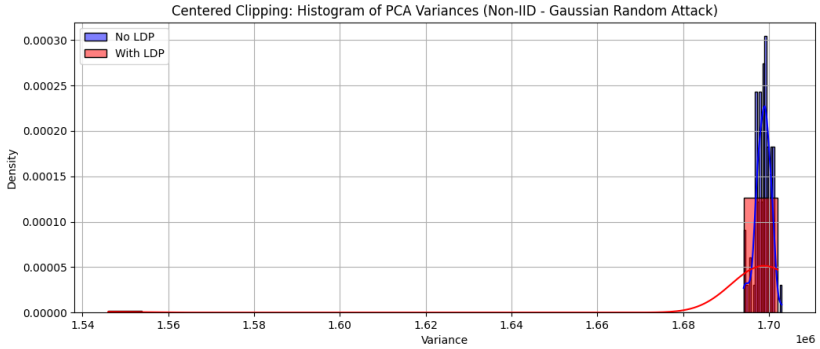

Figure 4.10: Centered Clipping: Density of PCA Variances – With vs Without LDP (non-IID - Random Gaussian Attack).

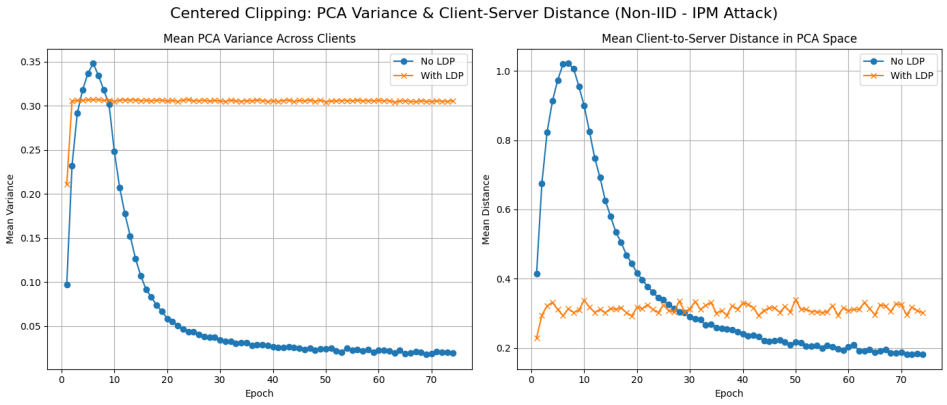

Figure 4.11: Centered Clipping: PCA Variance & Client-Server Distance – With vs Without LDP (non-IID - IPM Attack).

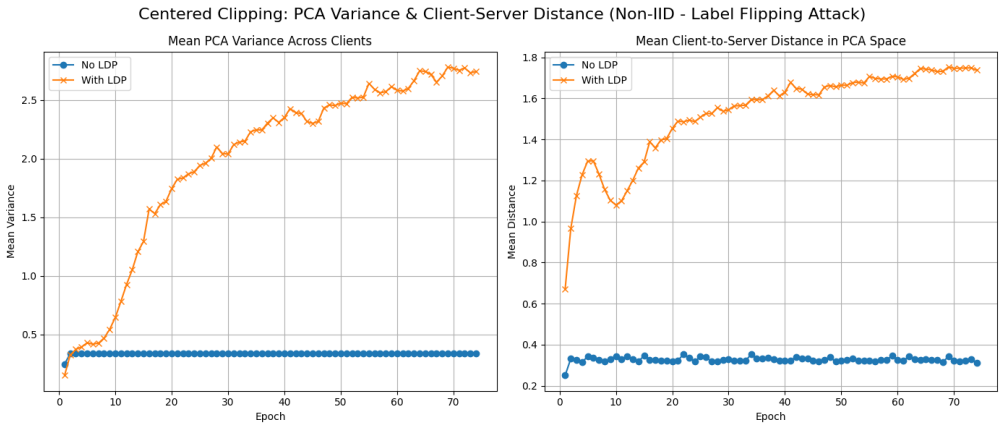

Figure 4.12: Centered Clipping: PCA Variance & Client-Server Distance – With vs Without LDP (non-IID - LF Attack).

4.2.3 Clipped Clustering Aggregation under Random Gaussian Attack

Clipped Clustering achieves 93.85% test accuracy in the No-LDP case but has a sharp decline to 61.05% in the LDP case when Random Gaussian attack is performed. The reason is exactly the same as the IID case described in Section 4.1.4. The corresponding plot can be seen in Figure 4.13.

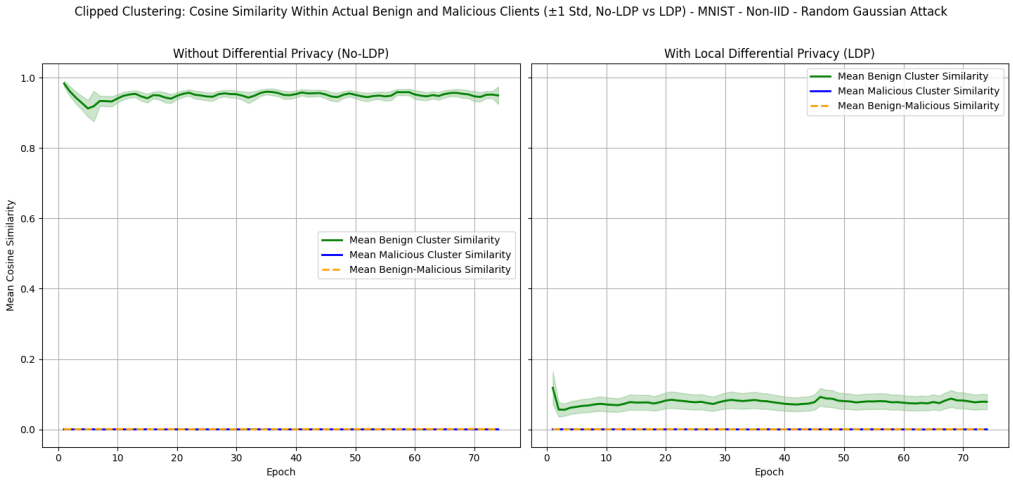

Figure 4.13: Clipped Clustering: The Mean of Cosine Similarity Within Benign and Malicious Clusters, Comparing the Effect of LDP (non-IID - Random Gaussian Attack).

4.2.4 Clipped Clustering Aggregation under IPM Attack

Clipped Clustering achieves 98.09% test accuracy in the No-LDP case but has a sharp decline to 10.10% in the LDP case when IPM attack is performed. Figure 4.14 shows the mean of cosine similarity within the benign and malicious clusters and the mean of cosine similarity between benign and malicious clients across 80 epochs, comparing the effect of LDP.

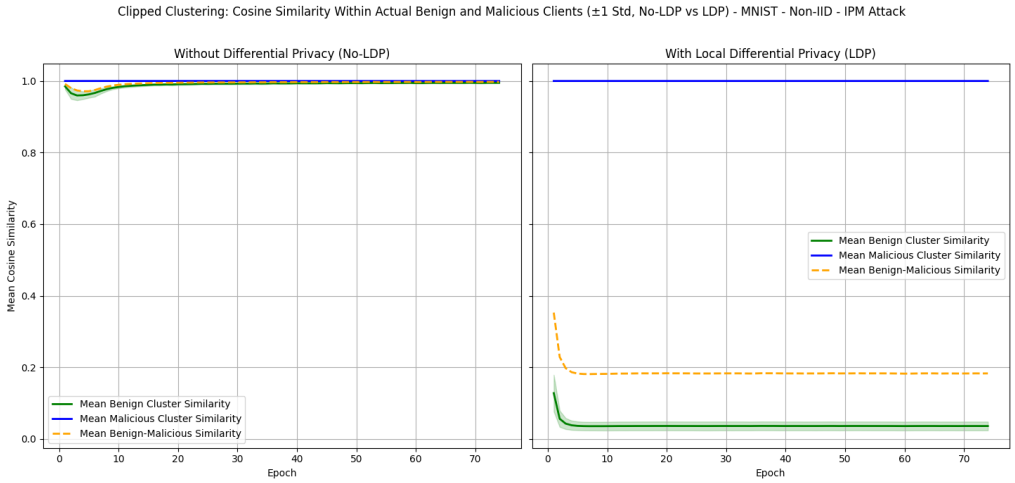

Figure 4.14: Clipped Clustering: The Mean of Cosine Similarity Within Benign and Malicious Clusters – With vs Without LDP (non-IID - IPM Attack).

On the left subplot, which shows the No-LDP case, the mean cosine similarity of the benign cluster (green line) is high, meaning they are similar throughout the 80 epochs. Also, the mean cosine similarity of the malicious cluster (blue line) is high because the attack type is IPM, meaning that all malicious clients send exactly the same gradients to the server for aggregation. The mean cosine similarity between malicious and benign clients (orange dashed line) is also high, meaning that benign and malicious clients are highly similar. Since malicious clients use the means of benign clients' gradients for the attack, their gradients are similar to those of benign ones. On the other hand, on the right subplot, which shows the LDP case, the mean cosine similarity of the malicious cluster (blue line) is always one since the attack is IPM. However, the mean cosine similarity of the benign cluster (green line) is low, which means that they are dissimilar from each other. Moreover, the mean cosine similarity between malicious and benign clients (orange dashed line) is low, close to 0.2, meaning that benign clients and malicious clients are no longer similar under LDP. Since LDP clips the clients' gradients and adds calibrated gradients, benign clients are no longer similar. Clipped Clustering method is based on clustering the benign and malicious clients into two clusters using the cosine similarity

scores. However, since malicious clients send the same gradients, they are highly similar, and benign clients are dissimilar due to LDP. The method uses malicious clients' gradients to create the global update, disrupting the global model's performance.
Moreover, the Agglomerative Clustering method used in the Clipped Clustering algorithm uses average linkage, which computes the similarity between two clusters as the mean distance of all pairs of points found in the clusters. With the effect of LDP, the mean distance calculation is affected significantly, leading the Agglomerative Clustering algorithm to misclassify the malicious clients as benign clients. On the other hand, Agglomerative Clustering with complete linkage, which clusters nodes by calculating the similarity as the maximum distance between any two points from the two clusters, was experimented with the same settings to be sure that the problem occurs because of the usage of average linkage. From the experiment result, the Clipped Clustering method achieved higher test accuracy when switching to the complete linkage because complete-linkage forms a tighter and more compact cluster and excludes outliers such as malicious clients or some benign clients that became outliers because of LDP.

4.3 Robust Aggregation Methods - CIFAR-10 - IID

This section refers to Table 3.4 and analyzes performance drops of robust aggregation methods when data partition is IID under LDP.

4.3.1 MKrum Aggregation under IPM Attack

MKrum achieves 74.81% test accuracy under IPM attack in the No-LDP case but sharply declines to 14.43% in the LDP case. Figure 4.15 shows the average MKrum scores of benign and malicious clients throughout 100 epochs, comparing the effect of LDP. Figure 4.16 shows for Epochs 1 and 2, the projections of the clients' local updates (benign and malicious) and the server's aggregated update onto the first two principal components, comparing the cases without and with LDP under an IPM attack.
MKrum effectively chooses benign clients to update the global model when LDP is not applied because, under IID data partition, benign clients are well clustered, as shown in Figure 4.16 (left subplots). As a result, benign clients have lower average MKrum scores than malicious ones, as shown in 4.15 (left subplot). However, under LDP, each client adds random Gaussian noise to their gradients, which means that benign clients are distant from each other. On the other hand, malicious clients perform an IPM attack. They send exactly the same gradients to the server, which means they are close to each other, as shown in 4.16 (right subplots), and their average MKrum score is lower compared to the average MKrum score of benign clients at each epoch, as shown in 4.15

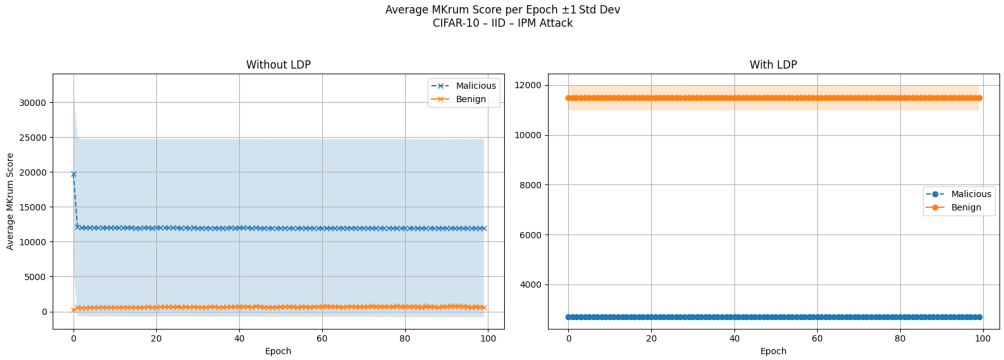

Figure 4.15: MKrum: The Average of MKrum Scores of Benign and Malicious Clients, Comparing the Effect of LDP (IID - IPM Attack).

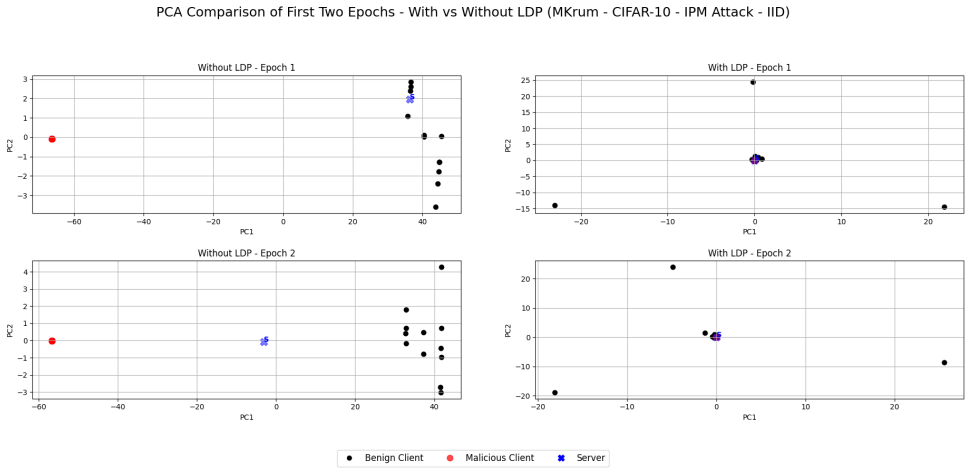

Figure 4.16: MKrum: PCA Comparison of First Two Epochs – With vs Without LDP (IID - IPM Attack).

(right subplot). Since benign clients are distant from each other but malicious clients are close to each other, MKrum chooses all of the malicious clients' gradients to update the global model. Additionally, for the CIFAR-10 benchmark dataset, the first rounds are important for the model to learn [FSM20]. If it cannot learn efficiently and uses malicious clients in the aggregation part, the model's learning stops in the upcoming rounds. As a result, MKrum suffers a performance drop when LDP is applied under the IPM attack.

4.3.2 MKrum Aggregation under LF Attack

MKrum achieves 76.32% test accuracy under LF attack in the No-LDP case but sharply declines to 31.81% in the LDP case. Figure 4.17 shows the average MKrum scores of

benign and malicious clients throughout 100 epochs, comparing the effect of LDP. Figure 4.18 shows for Epochs 1 and 2, the projections of the clients' local updates (benign and malicious) and the server's aggregated update onto the first two principal components, comparing the cases without and with LDP under a LF attack.

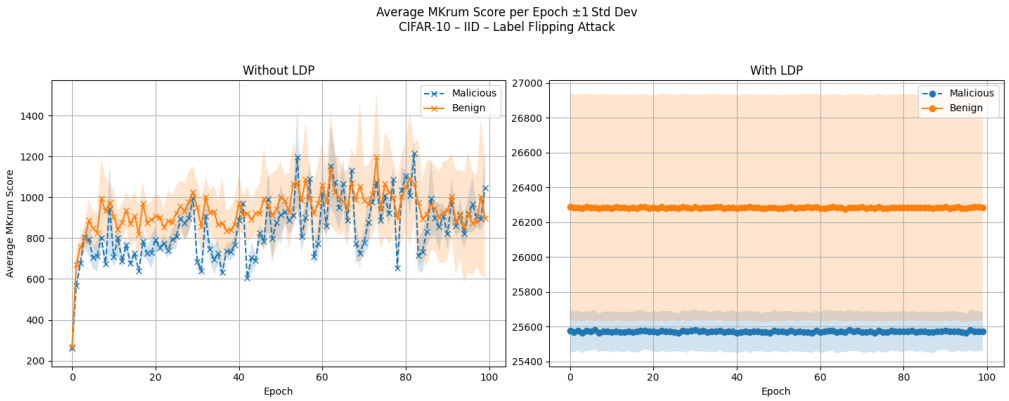

Figure 4.17: MKrum: The Average of MKrum Scores of Benign and Malicious Clients, Comparing the Effect of LDP (IID - LF Attack).

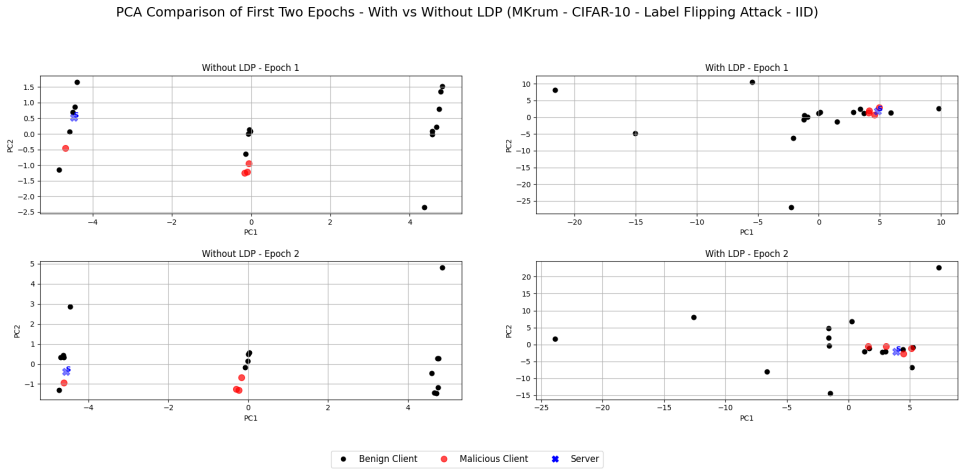

Figure 4.18: MKrum: PCA Comparison of First Two Epochs – With vs Without LDP (IID - LF Attack).

MKrum considerably chooses benign clients to update the global model when LDP is not applied because, under the IID partition, benign clients are clustered well, as shown in Figure 4.18 (left subplots). However, the method can also use malicious clients while updating the global model, as shown in Figure 4.17 (left subplot). On the other hand, under LDP, each client adds calibrated Gaussian noise to their gradients, meaning that

benign clients become distant. However, malicious clients perform a LF attack, and their flipping strategies are the same. As a result, they create similar updates, as shown in Figure 4.18 (right subplots), and their average MKrum score is lower compared to the average MKrum score of benign clients at each epoch, as shown in Figure 4.17 (right subplot). Since malicious clients are close to each other, but benign clients are far away from each other under LDP, MKrum chooses all of the malicious clients' gradients to update the global model. Additionally, for the CIFAR-10 benchmark dataset, the first rounds are important for the model to learn [FSM20], and if it cannot learn efficiently and uses malicious clients in the aggregation part, in the upcoming rounds, the model's learning stops. As a result, MKrum suffers a performance drop when LDP is applied under a LF attack.

4.3.3 Median Aggregation under All Attack Scenarios

Median achieves 78.18%, 72.32%, and 75.53% test accuracies in the No-LDP case but has a sharp decline to 10.19%, 11.27%, and 11.87% in the LDP case under there is No-attack, Random Gaussian attack, and LF attack, respectively. Figures 4.19, 4.20, and 4.21 show the evolution of the percentage of model parameters for which each of the two top clients contributes the element-wise median across training under No-attack, Random Gaussian attack, and LF attack, respectively. In each figure, Median aggregation without LDP (left subplot) is compared to LDP (right subplot).

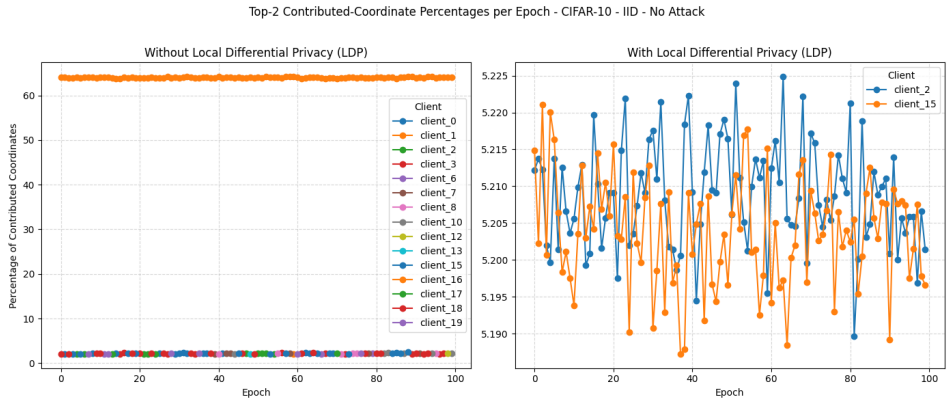

Figure 4.19: Median: Epoch-wise share of median-contributed coordinates (Top 2 Clients), comparing the effect of LDP (IID - No-attack).

As seen from the Figures, when LDP is not applied, the top-2 clients contribute more than half of the percentage of the coordinates in the global model at each epoch. In other words, more than 50% of the global model's parameters are directly selected from only those two clients at each epoch. Since their local updates are consistent and fall in the middle

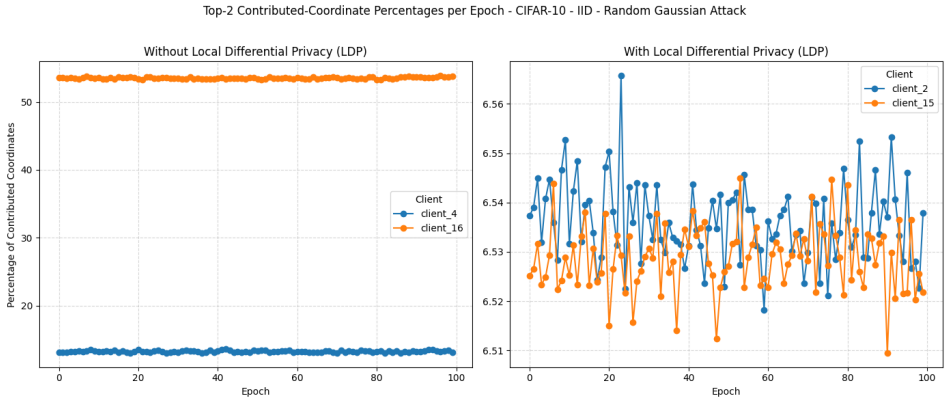

Figure 4.20: Median: Epoch-wise share of median-contributed coordinates (Top 2 Clients), comparing the effect of LDP (IID - Random Gaussian Attack).

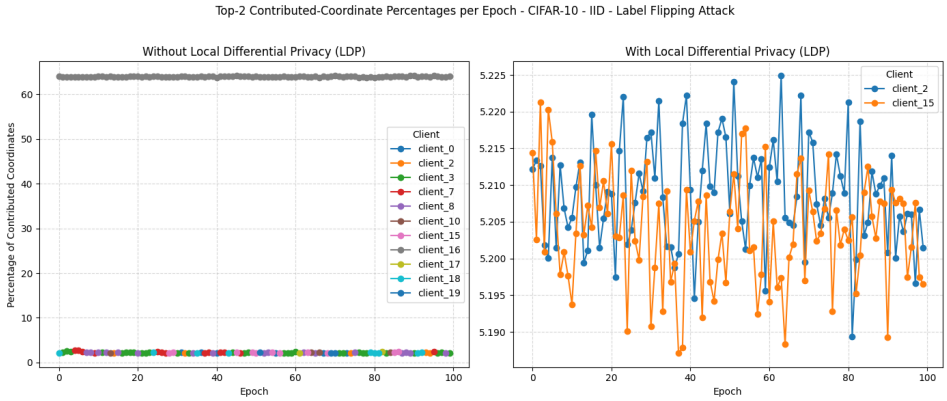

Figure 4.21: Median: Epoch-wise share of median-contributed coordinates (Top 2 Clients), comparing the effect of LDP (IID - LF Attack).

of the other clients, their effect is high on the global model. Due to this concentration on just two clients, the global model converges, and since there are no random fluctuations, the accuracy of the global model increases steadily. Conversely, when LDP is applied, the top-2 clients contribute around 10% of the coordinates in the global model, meaning that the Median selects coordinates from each of the client models uniformly. Due to this uniformity, coherent aggregation cannot be created, and the global model cannot converge. As a result, the method fails under LDP. Additionally, in FL experiments performed on the MNIST dataset, there was no significant decrease in the test accuracy of the global model when the aggregation method was Median under LDP. However, when the epoch-wise contribution of the best two clients in the coordinates selected by the Median was examined after applying LDP, it was found that the resulting plot was similar to Figure 4.19. The CNN model that is trained on the MNIST dataset is small

compared to the ResNet18 model that is trained on the CIFAR-10 dataset, considering the total number of parameters. Additionally, it should be noted that the MNIST dataset is a much simpler task for a model to learn compared to CIFAR-10 [GK24]. Consequently, it is expected that the Median aggregation to achieve higher test accuracy when the CNN model is trained on the MNIST dataset under LDP.

4.3.4 CosDefense Aggregation under Random Gaussian Attack

CosDefense achieves 80.10% test accuracy in the No-LDP case but has a sharp decline to 11.75% in the LDP case. Figure 4.22 shows the mean of normalized cosine similarity scores between benign and malicious clients and the server across 100 epochs, comparing the effect of LDP.

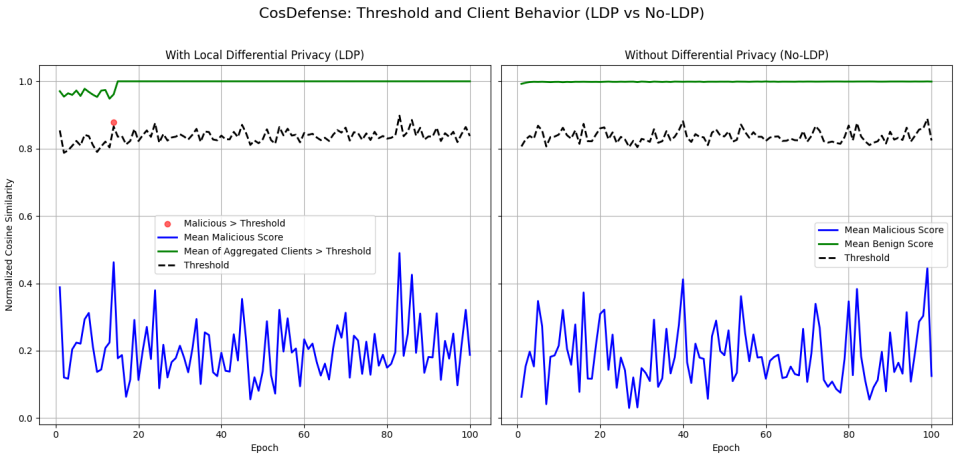

Figure 4.22: CosDefense: The Mean of Cosine Similarity Scores between Benign and Malicious Clients and the Server – With vs Without LDP (IID - Random Gaussian Attack).

The dashed line is the threshold, which means that if the cosine similarity score of the client exceeds that point, it is considered a benign client by the aggregation method. The left subplot shows the case where LDP is applied. On the left subplot, the mean of the cosine similarity score between malicious clients and the server (blue line) is consistently lower than the threshold, and the mean of the cosine similarity score between benign clients and the server (green line) is consistently higher than the threshold. However, in one epoch, the aggregation method misses some malicious clients because of LDP. Since LDP clips the gradients and adds calibrated Gaussian noise to those gradients, and our attack is a Random Gaussian attack, some of the malicious clients could be similar to the server, and they could be used in the aggregation, which results in the disruption of the global model. The right subplot shows the case where LDP is not

applied. On the right subplot, again, the mean of the cosine similarity score between malicious clients and the server (blue line) is consistently lower than the threshold, and the mean of the cosine similarity score between benign clients and the server (green line) is consistently higher than the threshold. Additionally, when LDP is not applied, the aggregation method correctly identifies malicious clients, discards them, and only uses benign clients, achieving higher test accuracy.

4.4 Robust Aggregation Methods - CIFAR-10 - non-IID

This section refers to Table 3.5 and analyzes performance drops of robust aggregation methods under LDP when data partition is non-IID.

4.4.1 MKrum Aggregation under All Attack Scenarious

MKrum achieves 70.81%, 68.78%, 61.48%, and 66.24% test accuracies in the No-LDP case but has a sharp decline to 39.77%, 39.67%, 13.49%, and 37.15% in the LDP case when there is No-attack, Random Gaussian attack, IPM attack, and LF attack, respectively. In order to understand that LDP noise or LDP norm clipping causes a severe accuracy drop, two different experiments were performed. In the first experiment, the LDP noise was set to 0, and MKrum achieved higher test accuracy (close to the No-LDP case) for each attack scenario. In the second experiment, the norm clipping of LDP is set to 100 to restrict the clipping effect, and MKrum failed under each attack scenario, meaning that the LDP noise caused the accuracy drop. As discussed in Section 4.2.1, the Krum aggregation method tends to choose a client that has the lowest sample size compared to the other clients to update the global model when the data distribution is non-IID under LDP. Since MKrum is an extension of Krum, it also chooses clients with the smallest number of sample sizes compared to the other clients that are not chosen to update the global model. Therefore, the global model has poor generalization. In order to test this idea for the MKrum method, clients' local updates chosen by the No-LDP case were used in the aggregation, and MKrum achieved higher test accuracies under each attack scenario when LDP was applied. Furthermore, there are many different approaches in the MKrum method to specify how many client updates will be used in the aggregation, which could be another reason why MKrum achieves lower test accuracy under LDP. In our experiments, MKrum chooses less than half of the total number of client updates for aggregation. In order to understand whether the number of client updates used in the aggregation is another cause for the low test accuracy under LDP or not, MKrum was forced to use more client updates than it normally uses in the aggregation, and MKrum's test accuracies increased under each of the attack scenarios when LDP is applied. Consequently, the number of

client updates used in the aggregation and the chosen clients' sample sizes cause the test accuracy drop under LDP.

4.4.2 Median Aggregation under All Attack Scenarios

Median achieves 72.86%, 62.30%, and 68.71% test accuracies in the No-LDP case but has a sharp decline to 10.00%, 10.00%, and 10.00% in the LDP case under there is No-attack, Random Gaussian attack, and LF attack, respectively. For each case, the reason is precisely the same as the IID case described in Section 4.3.3. Corresponding plots for No-attack, Random Gaussian attack, and LF attack are shown in Figures 4.23, 4.24, and 4.25, respectively.

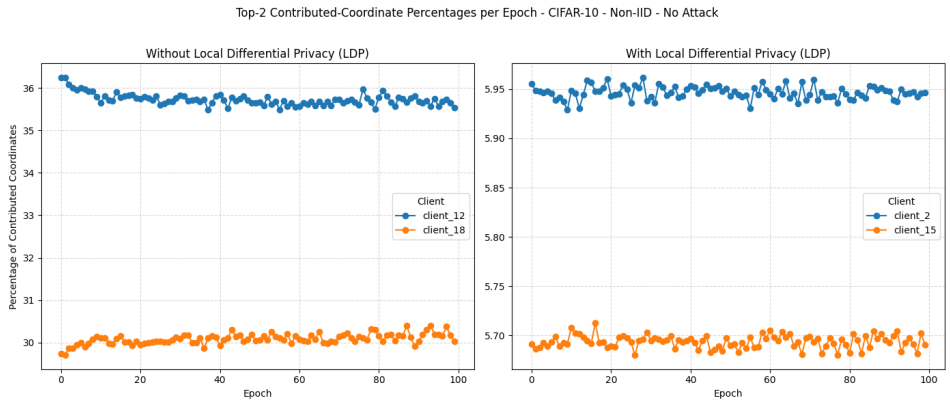

Figure 4.23: Median: Epoch-wise share of median-contributed coordinates (Top 2 Clients), comparing the effect of LDP (non-IID - No-attack).

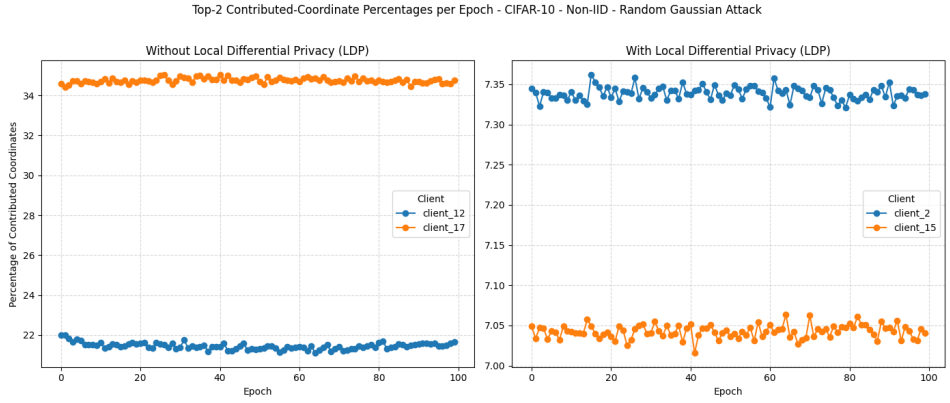

Figure 4.24: Median: Epoch-wise share of median-contributed coordinates (Top 2 Clients), comparing the effect of LDP (non-IID - Random Gaussian Attack).

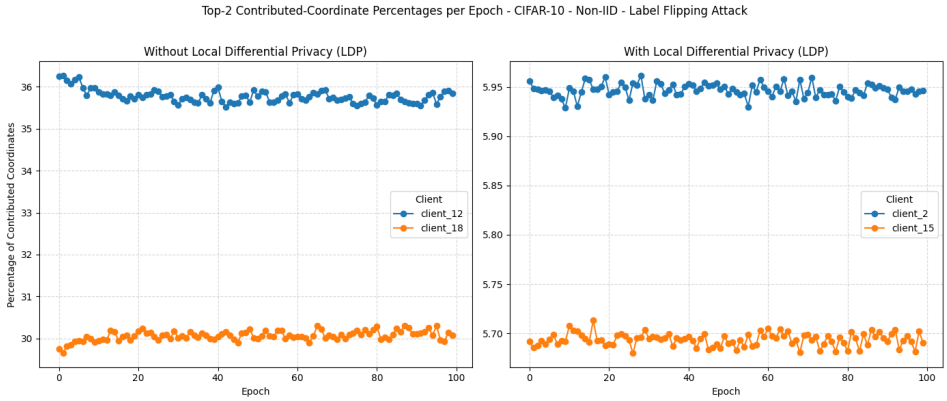

Figure 4.25: Median: Epoch-wise share of median-contributed coordinates (Top 2 Clients), comparing the effect of LDP (non-IID - LF Attack).

4.4.3 Clipped Clustering Aggregation under IPM Attack

Clipped Clustering achieves 59.20% test accuracy in the No-LDP case but has a sharp decline to 37.55% in the LDP case when IPM attack is performed. Figure 4.26 shows the mean cosine similarity within the benign and malicious clusters and the mean cosine similarity between benign and malicious clients across 100 epochs, comparing the effect of LDP.

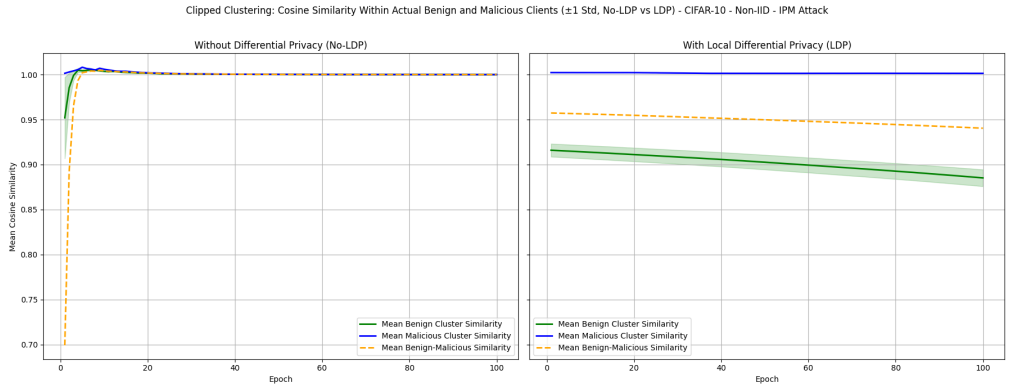

Figure 4.26: Clipped Clustering: The Mean of Cosine Similarity Within Benign and Malicious Clusters – With vs Without LDP (non-IID - IPM Attack).

On the left subplot, which shows the No-LDP case, the mean cosine similarity of the benign cluster (green line) is high, meaning they are similar throughout the 100 epochs. Also, the mean cosine similarity of the malicious cluster (blue line) is high because the attack type is IPM, meaning that all malicious clients send exactly the same gradients to the server for aggregation. The mean cosine similarity between malicious and benign

clients (orange dashed line) is also high because malicious clients are using mean of benign clients' gradients to create their gradients which results in highly similar client gradients. On the other hand, on the right subplot, which shows the LDP case, the mean cosine similarity of the malicious cluster (blue line) is always one since the attack is IPM. However, the mean cosine similarity of the benign cluster (green line) is low compared to the mean of cosine similarity of the malicious cluster, which means that they are dissimilar. Since LDP clips the clients' gradients and adds calibrated noise to those gradients, benign clients are no longer highly similar. Clipped Clustering is based on clustering the benign and malicious clients into two clusters using the cosine similarity scores. However, since malicious clients are exactly the same and benign clients are dissimilar, the method uses malicious clients to create the global update, which disrupts the global model's performance.

Moreover, the Agglomerative Clustering method used in the Clipped Clustering algorithm uses average linkage, which computes the similarity between two clusters as the mean distance of all pairs of points found in the clusters. With the effect of LDP, the mean distance calculation is affected significantly, leading the Agglomerative Clustering algorithm to misclassify the malicious clients as benign clients. On the other hand, Agglomerative Clustering with complete linkage, which clusters nodes by calculating the similarity as the maximum distance between any two points from the two clusters, was experimented with the same settings to be sure that the problem occurs because of the usage of average linkage. From the experiment result, the Clipped Clustering method achieved higher test accuracy when switching to the complete linkage because complete-linkage forms a tighter and more compact cluster and excludes outliers such as malicious clients or some benign clients that became outliers because of LDP.

4.4.4 Bulyan Aggregation under LF Attack

Bulyan achieves 68.87% test accuracy in the No-LDP case but has a sharp decline to 38.22% in the LDP case when LF attack is performed. Figure 4.27 shows, for Epochs 1 and 2, the projections of the clients' local updates (benign and malicious) and the server's aggregated update onto the first two principal components, comparing the cases without and with LDP under a LF attack.

As seen from Figure 4.27, without LDP (left subplots), benign clients are close to each other, and malicious clients are considerably far away from the center. Since Bulyan uses the Krum aggregation method, it chooses clients that are closest to their neighbors. As a result, it chooses benign clients' updates to update the global model. On the other hand, when LDP is applied (right subplots), the malicious clients' updates are close to

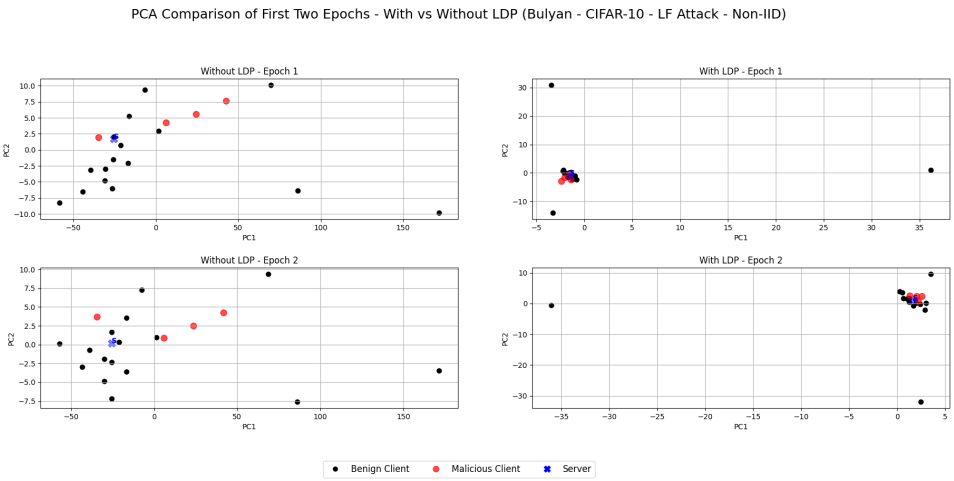

Figure 4.27: Bulyan: PCA Comparison of First Two Epochs – With vs Without LDP (non-IID - LF Attack).

the benign clients' updates due to LDP noise and norm clipping. Since the attack is LF, malicious clients create similar updates and tend to be centered. Consequently, Bulyan uses malicious clients' updates in the aggregation, meaning that the test accuracy of the Bulyan method suffers from a performance drop under LDP.

4.5 Common Patterns

Euclidean distance-based robust aggregation methods such as Krum and MKrum are highly affected by LDP noise when the data distribution is non-IID. Under LDP, the Krum method tends to choose a local update that is created by the client that has the smallest sample size, meaning that the local update is simple, and the effect of LDP noise on the local update is simple. As a result, the local update is the one that is closest to other local updates. Since the data distribution is non-IID and the chosen local update is created by the client that has the smallest sample size, the global model has poor generalization. This causes Krum to fail when it is experimented on the MNIST dataset when the data distribution is non-IID. However, MKrum is able to achieve high performances and is not affected by this because it can choose more clients' updates, and the MNIST dataset is a simpler task for a model to learn. However, MKrum fails under LDP when the model is trained on CIFAR-10, and the data distribution is non-IID. The reason is the same, and the MKrum method chooses local updates that are created by clients that have small sample sizes. Since the CIFAR-10 dataset is not a simple task for a model to learn compared to the MNIST dataset, MKrum is not able to achieve high

performance and is highly affected by LDP noise. Only Bulyan aggregation, which is also a Euclidean distance-based method, is not affected by LDP and achieves high test accuracies (except in the case when the model is trained on CIFAR-10 under LF attack and the data distribution is non-IID) because it aggregates more local updates compared to Krum and MKrum methods.

Cosine similarity-based robust aggregation methods, such as Clustering, CosDefense, and Clipped Clustering, are highly affected by LDP in different attack scenarios and data distributions on both datasets. Even if the usage of cosine similarity scores in each aggregation is different from each other, they are based on cosine similarity calculations. LDP makes benign clients dissimilar and malicious clients become similar depending on their attack type. If malicious clients use an IPM attack, they send exactly the same gradients to the server, meaning they are the same. If malicious clients use a LF attack and their flipping strategy is the same, they send similar updates to the server, meaning they become similar. Therefore, cosine similarity-based methods cannot distinguish benign clients from malicious clients, causing methods to choose malicious clients while updating the global model. As a result, they suffer significant performance drops when LDP is applied under different attack scenarios and different data distributions on both datasets.

5 Conclusion

This thesis investigated how LDP, particularly DP-SGD mechanism, affects the performance of various robust aggregation methods (Krum, MKrum, Trimmed Mean, Median, Geometric Median, Centered Clipping, Clustering, Clipped Clustering, CosDefense, and Bulyan) under different attack scenarios (No-attack, Random Gaussian attack, IPM attack, LF attack) in an FL framework, both through extensive experimental evaluations and in-depth empirical analysis. The main findings and contributions are summarized below.

For the extensive experimental evaluations, we evaluated these robust aggregation methods on an FL framework, trained on MNIST and CIFAR-10 datasets, in both IID and non-IID data distributions under different attack scenarios, with and without LDP. The experimental results showed that LDP-enhanced FL achieved lower test accuracies than the FL framework, where LDP was not applied, under different robust aggregation methods. Since LDP distorts clients' updates due to gradient clipping and calibrated noise addition to those gradients, minor performance degradation was expected. However, some robust aggregation methods had a severe performance drop when LDP was applied. These robust aggregation methods are analyzed further to understand how LDP causes these accuracy drops.

When the model was trained on the MNIST (IID), MKrum, Trimmed Mean, Geometric Median, and Bulyan performed well under LDP in different attack scenarios. On the other hand, Median, Centered Clipping, Clustering, Clipped Clustering, and CosDefense aggregation methods had significant performance drops under LDP when different attack scenarios were applied. For these five robust aggregation methods, the analysis showed the following:

- *Median under IPM Attack:* Due to LDP, the statistical center of benign clients' updates is shifted, and malicious clients' updates are centered, leading to the median no longer being able to provide a realistic representation.

- *Centered Clipping under all attack scenarios:* Since LDP performs clipping on the

clients' gradients and the method itself clips them further, the clients' final updates are no longer similar to each other, leading to degradation of the global model's performance.

- *Clustering under IPM Attack:* Due to LDP, the benign clients become dissimilar and malicious clients are highly similar because of IPM attack, resulting in the method choosing malicious clients' updates to update the global model.
- *Clipped Clustering under Random Gaussian Attack:* Due to LDP, the benign clients become dissimilar, and malicious clients become similar, leading the method to choose malicious clients' updates to update the global model.
- *CosDefense under Random Gaussian Attack:* Because of LDP, malicious clients' updates become similar to the global model, leading to some malicious clients being chosen to update the global model in some epochs.

When the model was trained on the MNIST (non-IID), Trimmed Mean, Geometric Median, and Bulyan performed well under LDP in different attack scenarios, which is similar to the MNIST (IID) case. On the other hand, Krum, Centered Clipping, and Clipped Clustering aggregation methods had significant performance drops under LDP when different attack scenarios were applied. For these three robust aggregation methods, the analysis showed the following:

- *Krum under all attack scenarios:* Due to LDP, Krum tends to choose a client's update that was trained on the smallest sample size compared to the other clients, leading to a poor generalization of the global model.
- *Centered Clipping under all attack scenarios:* Since LDP performs clipping on the clients' gradients and the method itself clips them further, the clients' final updates are no longer similar to each other, leading to degradation of the global model's performance.
- *Clipped Clustering under Random Gaussian and IPM Attacks:* Due to LDP, the benign clients become dissimilar, and malicious clients become similar, leading the method to choose malicious clients' updates to update the global model.

When the model was trained on the CIFAR-10 (IID), Trimmed Mean, Geometric Median, and Bulyan performed well under LDP in different attack scenarios. On the other hand, MKrum, Median, and CosDefense aggregation methods had significant performance drops under LDP when different attack scenarios were applied. For these three robust aggregation methods, the analysis showed the following:

- *MKrum under IPM and LF Attacks:* Due to LDP, benign clients' updates became distant, and malicious clients' updates were similar because the attacks that they were using were designed to create the same or similar local updates. As a result, MKrum used malicious clients' updates to create the global model, which resulted in a performance drop in the global model.

- *Median under all attack scenarios:* Due to LDP, the coordinate-wise contribution of top-2 clients' local updates could not exceed 10% of the total coordinate of the global model, meaning that the method selects coordinates from each of the client models uniformly. Consequently, the global model could not create a coherent aggregation or converge.

- *CosDefense under Random Gaussian Attack:* Because of LDP, malicious clients' updates become similar to the global model, leading to some malicious clients being chosen to update the global model in some epochs.

When the model was trained on the CIFAR-10 (non-IID), Trimmed Mean and Geometric Median performed well under LDP in different attack scenarios, which is similar to IID case. On the other hand, MKrum, Median, Clipped Clustering, and Bulyan aggregation methods had significant performance drops under LDP when different attack scenarios were applied. For these four robust aggregation methods, the analysis showed the following:

- *MKrum under all attack scenarios:* Due to LDP, Krum chooses the client's update created by the local model trained on a small sample. Since MKrum is an extension of Krum, it also chooses clients' updates that are created by training the local models on small samples, resulting in poor generalization of the global model.

- *Median under all attack scenarios:* Due to LDP, the coordinate-wise contribution of top-2 clients' local updates could not exceed 10% of the total coordinate of the global model, meaning that the method selects coordinates from each of the client models uniformly. Consequently, the global model could not create a coherent aggregation or converge.

- *Clipped Clustering under IPM Attack:* Due to LDP, the benign clients' local updates became dissimilar, and malicious clients' local updates were the same because of the IPM attack, leading the method to choose malicious clients' updates to update the global model.

- *Bulyan under LF Attack:* Because of LDP, malicious clients' updates became similar due to the flipping strategy that they used for the LF attack, whereas benign clients'

updates became dissimilar, leading to some malicious clients being chosen to update the global model in some epochs.

The analysis reveals common patterns between some of the robust aggregation methods. Euclidean distance-based robust aggregation methods such as Krum and MKrum (except Bulyan) are highly affected by LDP noise when the data distribution is non-IID. Both methods tend to choose clients' local updates that are created by clients who have the smallest sample sizes. Since the created local updates are simple because clients have the smallest sample sizes, the effect of LDP on those local updates is simple, meaning that the local updates are closest to the others. Since those local updates are created by clients that have highly skewed local data, the global model has poor generalization, and methods suffer under LDP when the data distribution is non-IID. Additionally, there are some commonalities in how LDP impacts robust aggregation methods based on cosine similarity, such as Clustering, CosDefense, and Clipped Clustering. LDP makes benign clients dissimilar. However, malicious clients depend on their attack types; if they are performing an IPM attack, they send exactly the same local updates to the server, or if they are performing a LF attack and their flipping strategy is the same, they send similar local updates to the server, becoming similar. As a result, cosine similarity-based robust aggregation methods cannot distinguish benign clients from malicious clients. They use malicious clients' local updates for the aggregation, resulting in high-performance drops when applying LDP.

In summary, the analysis provides explanations on why robust aggregation methods suffer a performance drop under LDP when applying different attack scenarios and also pinpoints which robust aggregation methods survive under LDP when applying different attack scenarios, which aligns with the central goal of the thesis. It must be noted that the findings of this thesis depend on empirical analysis rather than theoretical analysis, naturally limiting the generalization of the results to different datasets or models. Additionally, most of the analyses were performed through PCA. Since PCA is a dimensionality reduction technique, additional information that could be useful for further analysis could be lost by performing PCA.

Potential research directions that could take this work forward are using different ϵ values for LDP and using different benchmark datasets for test scalability and generalization. In the thesis, a constant privacy budget (ϵ) value was used for the datasets. Different ϵ value usage could enable us to reason in which range the robust aggregation methods remain balanced and achieve higher test accuracies. Additionally, while LDP was applied through Gaussian noise in the thesis, different LDP mechanisms, such as the correlated noise mechanism [PUCC+25], could be tested to see the effect on the robust aggregation methods. In addition to MNIST and CIFAR-10 benchmark datasets, more complex

datasets such as ImageNet and CIFAR-100 [KH09] could be empirically analyzed to test scalability and generalization. Depending on the complexity of the benchmark datasets, the robust aggregation methods that failed under LDP when the model is trained on CIFAR-10 would fail under the new benchmark datasets. Yet, the empirical analysis could show us additional information regarding how LDP breaks their statistical assumptions or algorithmic steps. Moreover, to bring the FL system close to a real-world heterogeneity, other datasets, which are non-IID by design, could be experimented with.

Bibliography

[ACG+16] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS'16. ACM, Oct. 2016. [Online]. Available: http://dx.doi.org/10.1145/2976749.2978318

[AZELA21] Z. Allen-Zhu, F. Ebrahimian, J. Li, and D. Alistarh, "Byzantine-resilient non-convex stochastic gradient descent," 2021. [Online]. Available: https://arxiv.org/abs/2012.14368

[BBG19] M. Baruch, G. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," 2019. [Online]. Available: https://arxiv.org/abs/1902.06156

[BEG+19] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," 2019. [Online]. Available: https://arxiv.org/abs/1902.01046

[BIK+16] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for federated learning on user-held data," 2016. [Online]. Available: https://arxiv.org/abs/1611.04482

[BMGS17] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Byzantine-tolerant machine learning," 2017. [Online]. Available: https://arxiv.org/abs/1703.02757

[CLE+19] N. Carlini, C. Liu, U. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," 2019. [Online]. Available: https://arxiv.org/abs/1802.08232

[CSX17] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," 2017. [Online]. Available: https://arxiv.org/abs/1705.05491

[DDS+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[Dwo06] C. Dwork, "Differential privacy," in *International Colloquium on Automata, Languages, and Programming (ICALP)*. Springer, 2006, pp. 1–12. [Online]. Available: https://link.springer.com/chapter/10.1007/11787006_1

[FCJG20a] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-Robust federated learning," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 1605–1622. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/fang

[FCJG20b] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *Proceedings of the 29th USENIX Conference on Security Symposium*, ser. SEC'20. USA: USENIX Association, 2020.

[FSM20] J. Frankle, D. J. Schwab, and A. S. Morcos, "The early phase of neural network training," 2020. [Online]. Available: https://arxiv.org/abs/2002.10365

[GGP24] R. Guerraoui, N. Gupta, and R. Pinot, *Robust Machine Learning, Distributed Methods for Safe AI*. Springer Singapore, 2024, reviewed. [Online]. Available: https://infoscience.epfl.ch/handle/20.500.14299/240850

[GK24] S. Greydanus and D. Kobak, "Scaling down deep learning with mnist-1d," 2024. [Online]. Available: https://arxiv.org/abs/2011.14439

[HRM+19] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," 2019. [Online]. Available: https://arxiv.org/abs/1811.03604

[HZRS15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[Jol02] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed., ser. Springer Series in Statistics. New York, NY: Springer, 2002, hardcover ISBN: 978-0-387-95442-4; Softcover ISBN: 978-1-4419-2999-0; eBook ISBN: 978-0-387-22440-4 Springer Book Archive. [Online]. Available: https://doi.org/10.1007/b98835

[KGS21] M. Kim, O. Günlü, and R. F. Schaefer, "Federated learning with local differential privacy: Trade-offs between privacy, utility, and communication," 2021. [Online]. Available: https://arxiv.org/abs/2102.04737

[KH09] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, Ontario, Tech. Rep. 0, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[KHJ21] S. P. Karimireddy, L. He, and M. Jaggi, "Learning from history for byzantine robust optimization," 2021. [Online]. Available: https://arxiv.org/abs/2012.10333

[KHJ23] ——, "Byzantine-robust learning on heterogeneous datasets via bucketing," 2023. [Online]. Available: https://arxiv.org/abs/2006.09365

[KMA+21] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," 2021. [Online]. Available: https://arxiv.org/abs/1912.04977

[LBH15] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[LCB10] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[LLZL21] Z. Li, L. Liu, J. Zhang, and J. Liu, "Byzantine-robust federated learning through spatial-temporal analysis of local model updates," 2021. [Online]. Available: https://arxiv.org/abs/2107.01477

[LNV24] S. Li, E. C.-H. Ngai, and T. Voigt, "An experimental study of byzantine-robust aggregation schemes in federated learning," *IEEE Transactions on Big Data*, vol. 10, no. 6, p. 975–988, Dec. 2024. [Online]. Available: http://dx.doi.org/10.1109/TBDATA.2023.3237397

[LSTS20] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, p. 50–60, May 2020. [Online]. Available: http://dx.doi.org/10.1109/MSP.2020.2975749

[MGR18] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The hidden vulnerability of distributed learning in byzantium," 2018. [Online]. Available: https://arxiv.org/abs/1802.07927

[MMR+23] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2023. [Online]. Available: https://arxiv.org/abs/1602.05629

[Mü11] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," 2011. [Online]. Available: https://arxiv.org/abs/1109.2378

[Nar91] A. Narayanan, "Algorithm as 266: Maximum likelihood estimation of the parameters of the dirichlet distribution," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 40, no. 2, pp. 365–374, 1991. [Online]. Available: http://www.jstor.org/stable/2347605

[NHC22] M. Naseri, J. Hayes, and E. D. Cristofaro, "Local and central differential privacy for robustness and privacy in federated learning," 2022. [Online]. Available: https://arxiv.org/abs/2009.03561

[PUCC+25] K. Pillutla, J. Upadhyay, C. A. Choquette-Choo, K. Dvijotham, A. Ganesh, M. Henzinger, J. Katz, R. McKenna, H. B. McMahan, K. Rush, T. Steinke, and A. Thakurta, "Correlated noise mechanisms for differentially private learning," 2025. [Online]. Available: https://arxiv.org/abs/2506.08201

[PyT] PyTorch Team, "Opacus: Differential privacy library for pytorch," https://github.com/pytorch/opacus, 2020–, accessed: 25 May 2025.

[QWH24] T. Qi, H. Wang, and Y. Huang, "Towards the robustness of differentially private federated learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 18, pp. 19 911–19 919, Mar. 2024. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/29967

[RARROR+24] I. Reyes-Amezcua, M. Rojas-Ruiz, G. Ochoa-Ruiz, A. Mendez-Vazquez, and C. Daul, "Leveraging pre-trained models for robust federated learning for kidney stone type recognition," 2024. [Online]. Available: https://arxiv.org/abs/2409.19934

[SFUK22] M. Shaheen, M. S. Farooq, T. Umer, and B.-S. Kim, "Applications of federated learning; taxonomy, challenges, and research trends," *Electronics*, vol. 11, no. 4, 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/4/670

[SGS+22] G. Shen, D. Gao, D. Song, L. Yang, X. Zhou, S. Pan, W. Lou, and F. Zhou, "Fast heterogeneous federated learning with hybrid client selection," 2022. [Online]. Available: https://arxiv.org/abs/2208.05135

[SSBD14] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[SSSS17] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," 2017. [Online]. Available: https://arxiv.org/abs/1610.05820

[Tiw24] A. Tiwari, "Differential privacy: 6 key equations explained," Jan. 2024. [Online]. Available: https://doi.org/10.59350/ntarj-tg210

[XHCL20] C. Xie, K. Huang, P. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," 2020, publisher Copyright: © 2020

8th International Conference on Learning Representations, ICLR 2020. All rights reserved.; 8th International Conference on Learning Representations, ICLR 2020 ; Conference date: 30-04-2020.

[XKG19] C. Xie, S. Koyejo, and I. Gupta, "Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation," 2019. [Online]. Available: https://arxiv.org/abs/1903.03936

[YCRB21] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," 2021. [Online]. Available: https://arxiv.org/abs/1803.01498

[YZA23] D. N. Yaldiz, T. Zhang, and S. Avestimehr, "Secure federated learning against model poisoning attacks via client filtering," 2023. [Online]. Available: https://arxiv.org/abs/2304.00160

[ZLH19] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," 2019. [Online]. Available: https://arxiv.org/abs/1906.08935

[ZLH+25] H. Zhang, Y. Liu, X. He, J. Wu, T. Cong, and X. Huang, "Sok: Benchmarking poisoning attacks and defenses in federated learning," 2025. [Online]. Available: https://arxiv.org/abs/2502.03801

[ZLL+18] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," 2018. [Online]. Available: https://arxiv.org/abs/1806.00582