Sapienza
Università di Roma

Evaluating the Resilience of Decentralized Federated Learning to Model Poisoning Attacks

Ingegneria dell'informazione, informatica e statistica
Laurea Magistrale in Informatica

Giovanni Pica
ID number 1816394

Advisor
Prof. Tolomei

Academic Year 2022/2023

Evaluating the Resilience of Decentralized Federated Learning to Model Poisoning Attacks
Tesi di Laurea Magistrale. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: pica.1816394@studenti.uniroma1.it

Abstract

In recent years, *federated learning* (FL) has become a viral paradigm for training distributed, large-scale, and privacy-preserving machine learning (ML) systems. In contrast to standard ML, where data must be collected at the exact location where training is performed, FL takes advantage of the computational capabilities of millions of edge devices to collaboratively train a shared, global model without disclosing their local private data. Specifically, in a typical FL system, the central server acts only as an orchestrator; it iteratively gathers and aggregates all the local models trained by each client on its private data until convergence. Although FL undoubtedly has several benefits over traditional ML (e.g., it protects private data ownership by design), it suffers from several weaknesses. One of the most critical challenges is to overcome the centralized orchestration of the classical FL client-server architecture, which is known to be vulnerable to single-point-of-failure risks and man-in-the-middle attacks, among others. To mitigate such exposure, *decentralized* FL solutions have emerged where all FL clients cooperate and communicate without a central server. Nevertheless, these two configurations are susceptible to model poisoning attacks, as participants are granted the ability to alter the model parameters. In particular, the decentralized one is the least explored yet. This work summarizes the characteristics of decentralized federated learning and provides experiments to this scenario with model poisoning attacks and defences.

Contents

1 Introduction **1**

2 Background and Preliminaries **5**
2.1 Federated Learning 5
2.1.1 Data Partitioning 6
2.1.2 Machine Learning Models 7
2.1.3 Scale of Federation 8
2.1.4 Communication Architecture 8
2.2 Peer-to-Peer Systems (P2P) 8

3 Related Work **11**
3.1 Decentralized FL 11
3.2 Adversarial Attacks on FL 12
3.3 Defences on FL 15

4 Problem Formulation **17**
4.1 RQ1: Attacks on a decentralized FL scenario 17
4.2 RQ2: Decentralized Aggregation Schemes 19

5 Proposed Method **23**
5.1 PENS: Performance-based neighbor selection 23
5.2 Threat Model 24

6 Experimental Results **25**
6.1 Experimental Setup 25
6.2 Evaluation 26
6.3 Impact of Gossip Learning Algorithm 27

7 Conclusion and Future Works **33**

Bibliography **35**

Chapter 1

Introduction

The last decade has witnessed incredible advances in machine learning (ML) and artificial intelligence (AI), which in turn have led to the pervasive application of such techniques across several domains. One of the most remarkable successes of ML/AI is undoubtedly represented by the most recent generative large language models (LLMs), such as ChatGPT and GPT-4 [60], as well as text-to-image generators like DALL-E 2 [71], and Midjourney [61]. These tools have raised the bar of human-to-machine interaction to a *new* level, unforeseeable only a few years ago. The growing complexity of deep neural networks, which have millions or billions of adjustable parameters, demands extensive training data to prevent overfitting. Large datasets are typically gathered in a central location, either physically or in the cloud. However, this approach has downsides: transmitting large data volumes strains network bandwidth and energy efficiency, especially on devices like smartphones. Additionally, this data transfer gives control of the data to third parties, raising concerns about privacy and security, particularly in domains with strict regulations like GDPR [84] and HIPAA [58]. Companies using AI/ML tools must comply with data privacy and security laws, a stricter obligation for businesses heavily reliant on data-driven strategies for revenue generation. The focus of the research community has shifted to planning novel ML training procedures that protect data privacy and security *by design.* As a result of such effort, *federated learning* (FL) is a new ML paradigm first introduced by Google in 2017 [50] that addresses the challenges above. The authors advocate the need for a new *distributed* training procedure with a well-known app for smartphones, i.e., the smart Google keyboard (Gboard). Indeed, at the core of Gboard, there is a neural language model that can predict, hence suggest, the next word to enter while the user is typing. Of course, for such a model to be successful, it must be trained on vast text corpora collected from many user devices (e.g., smartphones). In contrast to the standard ML approach, which would require (*i*) transferring these user-generated data from remote devices into Google's infrastructure, (*ii*) training a model on those data at a central location, and (*iii*) distributing the learned model back to user devices, Google proposed FL. Generally speaking, FL trains a global model using distributed data, *without* the need for the data to be shared nor transferred to any central facility. In other words, FL takes advantage of a multitude of AI-enabled edge devices that cooperate with each other to *jointly* learn a predictive model using their own local private data. Specifically, an FL system consists of a central server and many edge clients; a typical FL round involves the following steps: *(i)* the server randomly picks some clients and sends them the current, global model; *(ii)* each selected client locally trains its model with its own private data; then, it sends the resulting local model to the server;[1] *(iii)* the server updates the global model by computing an *aggregation function* on the local models received from clients (by default, the average, FedAvg [49]). The process above continues until the global model converges. Since this paradigm smoothly integrates with ubiquitous, distributed infrastructures, FL has been successfully applied to several domains, such as IoT [91], Fog computing [102], autonomous vehicles [66], and wearable devices [10]. Although FL has unquestionable advantages over standard ML, it also presents its own problems. Broadly speaking, FL still faces the following critical challenges.

- ***Security and Privacy:*** Keeping local data at each client's end naturally preserves its privacy. Yet, FL can be vulnerable to data confidentiality, integrity, and availability threats [12]. We invite the reader to refer to [72] for an exhaustive survey on these issues, and also [46].
- ***Communication Efficiency:*** Avoiding large data transfers from their origins to a shared training facility improves communication efficiency. Nevertheless, FL requires several rounds of interactions, where updated model parameters are iteratively exchanged between edge clients and the central server until convergence. This may slow down the distributed training of very complex models with millions or even billions of parameters and therefore needs adequate strategies to reduce communication overhead (e.g., model compression techniques [88, 76, 97]). Interested readers should consider [67] for a survey on these aspects.
- ***Data and System Heterogeneity:*** One of the key characteristics of FL is that data generated by federated clients is usually non-independent and identically distributed (non-IID), which contrasts with the typical IID assumption of standard ML. In addition, the capabilities of FL clients may be very different from each other, e.g., in terms of network connectivity (WiFi, 5G, etc.) and hardware characteristics (memory, CPU, battery, etc.) Such differences also fluctuate due to the highly dynamic nature of FL systems, where new clients frequently join while others drop out. A useful reference to learn more about this matter is [47].
- ***Incentive Mechanisms:*** The success of FL depends on the ability to attract edge devices with high-quality data and strong computational capabilities to join the federation. Currently, there are not enough incentives for those clients to participate and contribute. Interesting attempts to engage "valuable" FL clients with the platform are discussed in [65, 31].
- ***Centralized Orchestration:*** In "vanilla" FL, a single, central server is responsible for handling the entire federated training process. This scheme suffers from the same weaknesses as any other client-server architecture, e.g., single-point failure and man-in-the-middle attacks.

[1]Whenever we refer to global/local model, we mean global/local model *parameters*.

To overcome these limitations *decentralized* FL solutions were studied. These solutions in the literature are based on two kind of approaches, one that obtain the decentralization through "standard" distributed computing techniques (e.g., P2P networks, graphs, gossip communication), and the other one achieves FL decentralization by exploiting Blockchain functionalities. We invite the reader to refer to our survey on the topic [18]. However, these two approaches suffer *model poisoning* attacks [1]. Model poisoning takes advantage of the inherent capability of federated learning, wherein malicious participants are able to exert direct control over the collective model. This allows for attacks that are notably more potent compared to traditional training-data poisoning methods. In particular, these attacks are studied mainly in centralized FL approaches. The centralized FL literature offers various strategies to address model poisoning attacks, with the goal of filtering out potentially malicious local updates during the server-side aggregation process. These approaches encompass a spectrum of techniques, including advanced statistical methods such as the Trimmed Mean and FedMedian [95], weighted federated averaging [54], heuristic-based outlier detection methods like Krum/MultiKrum [7] and Bulyan [13], data-driven approaches like K-means clustering [78], methods that rely on establishing a "source of trust," as seen in FLTrust [9], and matrix autoregression on client updates [85].

The present work will have the following contributions:

(*i*) We select a *decentralized* FL system called *PENS* [59], which is a performance-based neighbor selection mechanism in which clients, whose data distributions are similar, identify one another and collaborate by assessing the training losses incurred when using each other's data. This collaborative approach allows them to develop a model that aligns better with the specific characteristics of their local data distribution.

(*ii*) We integrate aggregation schemes and attacks on a library called *Gossipy*[2].

(*iii*) We compare the robustness on different aggregation methods under multiple settings such as different attack scenarios.

(*iv*) We publish our implementation with experiments and code[3].

Furthermore, this work will focus on two main research questions:

- ***RQ1:*** Does decentralized FL exhibit resilience against significant model poisoning attacks?
- ***RQ2:*** Do the adapted aggregation methods perform effectively in decentralized FL environments?

The remainder of this paper is organized as follows. In Chapter 2, we recall some background and preliminary concepts. Chapter 3 describes the related works. Chapter 4 and Chapter 5 describes the problem and the proposed method. Chapter 6 gathers experimental results. Finally, Chapter 7 concludes our work and illustrate some future directions.

[2] `https://github.com/makgyver/gossipy`

[3] `https://github.com/bonjon/gossipy`

Chapter 2

Background and Preliminaries

2.1 Federated Learning

The prototypical FL setting consists of a central server S and a set of distributed clients $\mathcal{C}$, such that $|\mathcal{C}| = K$, that jointly cooperate to solve a standard supervised learning task.[1] Each client $c \in \mathcal{C}$ has access to its own private training set $\mathcal{D}_c$, namely the set of its n_c local labeled examples, i.e., $\mathcal{D}_c = \{x_{c,i}, y_{c,i}\}_{i=1}^{n_c}$.

The goal of FL is to train a global predictive model whose architecture and parameters $\theta^* \in \mathbb{R}^d$ are shared amongst all the clients and found to solve the following objective:

$$\theta^* = \text{argmin}_\theta \mathcal{L}(\theta) = \text{argmin}_\theta \sum_{c=1}^{K} p_c \mathcal{L}_c(\theta; \mathcal{D}_c), \tag{2.1}$$

where $\mathcal{L}_c$ is the local objective function for client c. Usually, this is defined as the empirical risk calculated over the training set $\mathcal{D}_c$ sampled from the client's local data distribution:

$$\mathcal{L}_c(\theta; \mathcal{D}_c) = \frac{1}{n_c} \sum_{i=1}^{n_c} \ell(\theta; (x_{c,i}, y_{c,i})), \tag{2.2}$$

where ℓ is an instance-level loss (e.g., cross-entropy loss or squared error in the case of classification or regression tasks, respectively). Furthermore, each $p_c \geq 0$ specifies the relative contribution of each client. Since it must hold that $\sum_{c=1}^{K} p_c = 1$, two possible settings for it are: $p_c = 1/K$ or $p_c = n_c/n$, where $n = \sum_{c=1}^{K} n_c$.

The generic federated round at each time t is decomposed into the following steps and iteratively repeated until convergence, i.e., for each $t = 1, 2, \ldots, T$:

(*i*) S randomly selects a subset of clients $\mathcal{C}^{(t)} \subseteq \mathcal{C}$, so that $1 \leq |\mathcal{C}^{(t)}| \leq K$, and sends them the current, global model $\theta^{(t)}$. To ease of presentation, without loss of generality, in the following, we assume that the number of clients picked at each round is constant and fixed, i.e., $|\mathcal{C}^{(t)}| = m, \ \forall t \in \{1, 2, \ldots, T\}$.

(*ii*) Each selected client $c \in \mathcal{C}^{(t)}$ trains its local model $\theta_c^{(t)}$ on its own private data $\mathcal{D}_c$ by optimizing the following objective, starting from $\theta^{(t)}$:

$$\theta_c^{(t)} = \text{argmin}_{\theta^{(t)}} \mathcal{L}_c(\theta^{(t)}; \mathcal{D}_c). \tag{2.3}$$

[1] Notice that the FL paradigm can also be used to solve unsupervised learning tasks like K-means clustering [34].

The value $\theta_c^{(t)}$ is computed via gradient-based methods like stochastic gradient descent (SGD) and sent back to S.

(*iii*) S computes $\theta^{(t+1)} = \phi(\{\theta_c^{(t)} \mid c \in \mathcal{C}^{(t)}\})$ as the updated global model, where $\phi : \mathbb{R}^{d^m} \mapsto \mathbb{R}^d$ is an *aggregation function*; for example, $\phi = \frac{1}{m}\sum_{c\in\mathcal{C}^{(t)}} \theta_c^{(t)}$, i.e., FedAvg or one of its variants [45].

A few alternatives to the scheme above are possible. For example, in step (*ii*), instead of sending the vector of parameters $\theta_c^{(t)}$, each selected client could transmit to the server its displacement vector compared to the global model received at the beginning of the round, i.e., $u_c^{(t)} = \theta_c^{(t)} - \theta^{(t)}$. This way, in step (*iii*), the server will compute the new global model as $\theta^{(t+1)} = \theta^{(t)} + \phi(\{u_c^{(t)} \mid c \in \mathcal{C}^{(t)}\})$, namely the aggregation function is calculated on the local update vectors rather than the actual local models. Similarly, sending local models (or their updates) $\theta_c^{(t)}$ ($u_c^{(t)}$) is equivalent to sending "raw" gradients $\nabla\mathcal{L}_c^{(t)}$ to the central server; in the latter case, S simply aggregates the gradients and uses them to update the global model, i.e., $\theta^{(t+1)} = \theta^{(t)} - \eta\phi(\{\nabla\mathcal{L}_c^{(t)} \mid c \in \mathcal{C}^{(t)}\})$, where η is the learning rate.

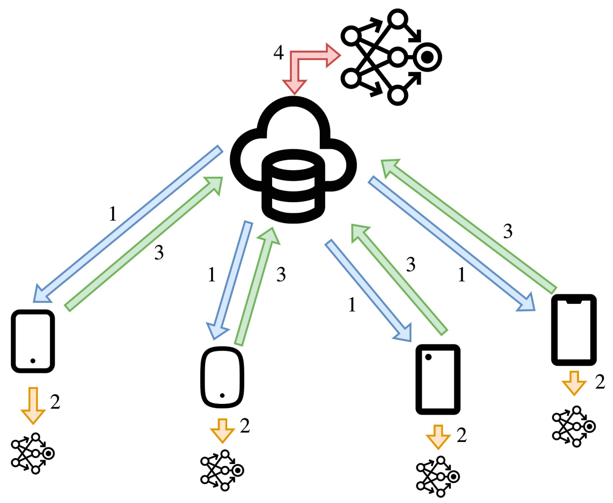


Figure 2.1. A general FL round. (1) The server samples a subset of available clients, initializes the model, and sends it to the selected workers. (2) Receiving clients train the model on their local datasets (3) and send back the updated models to the server. (4) The server aggregates the local models with some aggregation rule.

Broadly speaking, FL systems can be categorized according to four different axes: *data partitioning*, *machine learning model*, *scale of federation*, and *communication architecture*. Below, we review each of these aspects separately.

2.1.1 Data Partitioning

Data can be distributed over the sample and feature spaces, which creates a categorization in *horizontal*, *vertical*, and *hybrid* FL [94].

In *horizontal* FL, the datasets of different clients have the same feature space but little intersection on the sample space. The majority of studies on FL assume horizontal partitioning. Since the local data are in the same feature space, the parties can train the local models using their local data with the same model architecture, and the global model can simply be updated by averaging all the local models using standard FedAvg. A typical example where horizontal FL comes into play is when two (or more) regional branches of a bank want to collaboratively train a loan prediction model for their customers. Each branch may have very different user groups due to its geographical location, and the common set of users shared between any two branches is very small. However, their business is very similar, so the feature spaces are likely the same.

In *vertical* FL, instead, the datasets of different clients have the same or similar sample space but differ in the feature space. It usually adopts entity alignment techniques [11] to collect the overlapped samples of the parties, and then these data are used to train the machine learning model using encryption methods. As an example, consider two different companies in the same city, e.g., a bank and an e-commerce business. Their customer bases are likely to contain most of the residents of the area, so the intersection of their user space is large. However, the bank may record the user's revenue and expenditure behavior and credit rating, whereas the e-commerce collects the user's browsing and purchasing history; thus, their feature spaces are different.

Finally, *hybrid* FL (also referred to as *federated transfer learning* or FTL) is a combination of horizontal and vertical data partitioning. FTL applies when any pair of datasets from federated clients differ both in samples and in feature space. For instance, consider two different companies that are also geographically distant from each other, e.g., a bank located in China and an e-commerce company located in the United States. Due to geopolitical restrictions, the user groups of the two companies have a small intersection. Moreover, due to the different businesses, only a small portion of the feature space from both parties overlaps. In this case, *transfer learning* [63] techniques can be applied to provide solutions for the entire sample and feature space under a federation.

2.1.2 Machine Learning Models

FL is primarily used to jointly solve a machine learning task, which usually consists of collaboratively training a common model on several distributed private datasets. The choice of the specific ML model to train depends on the problem at hand and the dataset. The most popular family of models used within FL systems is Neural Networks (NNs) in all their flavors. Simpler models, such as Linear Regression (LinReg) or Logistic Regression (LogReg), and Decision Trees (DTs) or ensembles of those like the Gradient Boosting Decision Trees (GBDTs) are also successfully proposed in FL environments [23, 38], mostly due to their high efficiency and interpretability.

Generally speaking, an FL system can consist of homogenous or heterogenous ML models. In the former case, all clients have the same model, and aggregation of gradients comes into play at the server. In the latter scenario, there is no need for aggregation since each client has a different model. Therefore, at the server's end, aggregation methods are replaced by ensemble methods like majority voting.

2.1.3 Scale of Federation

FL systems can be categorized into *cross-silo* and *cross-device* [29]. The difference between the two concerns the number of parties involved and the overall amount of data available in the federation. The simplest way to understand them is to associate cross-silo with large organizations or data centers and cross-devices with mobile devices. In the case of cross-silo, the number of federated clients is usually small, but they have extensive computational abilities (e.g., a group of large medical institutions). The main challenge of cross-silo FL is to create an efficient distributed computation under the constraint of privacy [101]. When it comes to cross-device, instead, FL systems are made of a massive number of clients, each one with limited computational power (e.g., Google Gboard). In cross-device FL, the primary challenge relates to devices' energy consumption, which limits the complexity of training tasks that they can perform.

2.1.4 Communication Architecture

There are two main types of architectures for FL systems: centralized and decentralized. The centralized architecture follows a client-server model, where a central entity (referred to as the server) acts as the *orchestrator*. Its role involves coordinating the entire distributed training process, including the aggregation of individual local models sent by the clients into a single global model. On the other hand, in a decentralized architecture, there is no clear distinction between client and server roles. Each client has the potential to act as a server, taking turns in aggregating the current global model and transmitting it to other clients in a random manner during each round. Implementing *decentralized* FL systems poses significant challenges and can be categorized into two main categories: traditional distributed computing methods, such as peer-to-peer (P2P) approaches, and those utilizing Blockchain technology.

2.2 Peer-to-Peer Systems (P2P)

Generally speaking, peer-to-peer (P2P) computing offers an alternative to the traditional client-server architecture [2]. In a client-server model, clients connect to a central server to make requests while the server processes and responds to those requests. In contrast, P2P distributed systems allow nodes in the network to act as both servers and clients, promoting decentralization.

Although it is customary to associate P2P systems with (illegal) content sharing, they have broader applications beyond that. Increasingly, P2P solutions are being deployed to address various problems that were traditionally reliant on centralized server-based approaches. Some examples of P2P systems are BitTorrent [6], Gnutella [22], Napster [55], and Skype [80]. The fundamental principle of P2P networks is to enable resource sharing among end systems, including files, storage space, CPU cycles, and more. These systems create an overlay network over the Internet, facilitating communication between peers.

According to [73], there are three primary characteristics of a P2P system:

- ***Self-organizing***: Nodes must organize themselves to form an overlay network. There should be no assistance from a central node. Also, there should not be any global index that lists all the peers and/or the available resources.
- ***Symmetric communication***: All nodes must be equal (i.e., no node should be more important than any other node.) Also, peers should both request and offer services (i.e., they should act as both clients and servers).
- ***Decentralized control***: There should not be a central controlling authority that dictates behavior to individual nodes. Peers should be autonomous and must determine their level of participation in the network on their own.

P2P networks offer some advantages over classical client-server architectures, such as eliminating the single-point-of-failure and single-source bottleneck. The primary characteristics of P2P networks are *reliability* [40] against nodes that disconnect or have a low latency or bandwidth; *scalability* [32] as the workload is no more concentrated in a server; *privacy* [26] and *anonimity* [48] via cryptographic protocols.

Still following [73], some of the key challenges of P2P systems are as follows:

- ***Budget:*** The budget allocated to a solution influences the consideration of P2P architectures. If the budget is ample, the inefficiencies and complexities associated with P2P may not be deemed worthwhile. However, if the budget is limited, the low cost of entry for individual peers becomes an attractive factor despite the increased total system cost. Utilizing local components and surplus resources may be a justifiable approach within constrained budgets.
- ***Resource relevance to participants:*** The relevance of data to peers plays a significant role in P2P cooperation. If the probability of peers being interested in each other's data is high, cooperation naturally evolves. Conversely, if relevance is low, artificial or extrinsic incentives may be necessary to foster cooperation.
- ***Trust:*** Trust among peers varies depending on the specific problem requirements. Mutual distrust can be either essential or negligible. However, the cost of mutual distrust in P2P systems is high, and its necessity must be justified based on the problem's characteristics.
- ***Rate of system change:*** P2P systems may experience stable or rapidly changing participants, resources, and parameters. Rapid changes pose challenges in ensuring consistency guarantees, defending against flooding, and mitigating other attacks.
- ***Criticality:*** If the problem being solved is critical to users, centralized control may be demanded regardless of technical criteria. Even when P2P is not ruled out, the need for expensive security measures or extensive over-provisioning may render it economically unfeasible.
- ***Security:*** P2P networks are known to be vulnerable to various types of security attacks. For instance, malicious nodes can disrupt the network by flooding it with redundant data, manipulating trust values in trust-based

systems, coordinating with other malicious nodes for distributed denial-of-service (DDoS) attacks, or performing *Sybil* attacks by creating multiple fake identities within the network [68].

In recent years, *decentralized* machine learning has gained popularity within P2P networks. The focus has been on solving the distributed consensus problem, aiming to find a global model that minimizes the sum of local loss functions [70, 57]. Additionally, research has explored privacy-preserving approaches and scenarios where agents have distinct objectives [4]. These advancements enable collaborative learning and optimization in a decentralized manner.

Chapter 3

Related Work

3.1 Decentralized FL

The concept of Decentralized FL, has gained significant attention in recent times due to its ability to address the shortcomings of centralized approaches, such as vulnerability to *single-point failures* or man-in-the-middle attacks. One potential approach for mitigating the constraints of conventional centralized FL involves embracing decentralized frameworks influenced by established distributed computing technologies, like Peer-to-Peer (P2P) networks. According to this, several approaches came up. Each of them solve a particular problem like *fault-tolerance and scalability* [64, 69], *privacy and security* [30, 100, 19], *bandwidth utilization* [83, 25] and *data heterogeneity* [36, 90, 41]. Specifically, two works served as the initial sources of inspiration for the literature.

The first was proposed by Lalitha et. *al.* [35], which eliminates the requirement for a centralized controller, shifting the emphasis towards collaborative learning via local information exchange. To be more precise, clients employ a Bayesian-like methodology by incorporating a belief system concerning the model parameter space and employing a distributed learning algorithm. By aggregating information from neighboring users, individuals update their beliefs in order to acquire a model that optimally captures observations throughout the entire network. Furthermore, this framework permits users to exclusively sample points from smaller subspaces within the input space.

The other one proposed by Roy et. *al.* [74], is a starting point when we have to consider a decentralized framework. By establishing a fully connected network of peers, each participant maintains a vector that includes its own model version and the most recent models it received during merging. The training process follows these iterative steps:

(*i*) Each client independently trains its local model for several iterations using its own dataset.

(*ii*) A random client, denoted as $\widetilde{c}$, is chosen from the network and sends a "ping request" to collect the latest model versions from all other clients.

(*iii*) Clients that have updates transmit their model weights and training sample sizes to $\widetilde{c}$.

(*iv*) The models in this subset are merged with the current model of $\widetilde{c}$ through weighted averaging, resulting in a unified model.

The majority of these methods rely on a consensus mechanism to construct a single global model. Nonetheless, our aim is to focus on approaches that extend beyond the use of a single global model. Instead, we work with aggregated models for individual nodes. This approach enables the formation of a P2P network of nodes, each serving its distinct purposes. One example is *PANM* [41] which is a P2P clustered FL system. A clustered FL system operates under the Non-IID (Non-Independently and Identically Distributed) assumption, wherein distinct client groups possess their individual optimization objectives. This approach is typically employed to achieve enhanced accuracy performance or more efficient compression of model updates. To divide the clients into clusters they use a measurement called *client similarity*, based mainly on losses [20], gradients [75], and model weights [44]. In the context of loss-based measurement, clients have multiple models, and they evaluate these models on their respective local datasets. The model that yields the lowest loss is considered to have the highest similarity. Regarding the measurement of model weights and gradients, previous studies have employed metrics such as cosine distance or Euclidean distance. Clients with similar data distributions tend to exhibit smaller Euclidean distances and larger cosine similarities in terms of gradients or model weights. Since the dependency on a central server can introduce reliability and communication bandwidth problems [43], the researchers propose this P2P clustered FL system. Additionally, they propose two metrics that rely on loss and gradient measurements. It's divided in two stages: the first involves neighbor selection through Monte Carlo methods, and the second incorporates neighbor augmentation based on the Gaussian Mixture Model.

3.2 Adversarial Attacks on FL

Centralized FL (and for extension decentralized FL) encounters various adversarial threats that manifest in diverse forms, capable of negatively impacting output accuracy, disrupting model convergence, and even causing severe denial-of-service (DoS) issues for both servers and users. A major challenge in this context arises from the potential presence of malicious nodes. Their actions may include inundating other nodes with random or redundant data and collaborating with fellow malicious actors to execute DoS attacks. Such malicious nodes pose substantial security and reliability concerns within the FL process. These adversarial attacks can have different perspectives. For instance, *feature inference attacks* [103, 24, 96], *membership inference attacks* [37, 56], and *property inference attacks* [89, 51] aim to uncover information concerning the participants engaged in a machine learning task. In particular, the confidentiality measures of FL render it vulnerable to attacks seeking to alter a client's information to corrupt the global model. From another perspective, these attacks depending on the attacker's behaviour can be seen as *targeted* [81, 1] and *untargeted* [14, 7]. The primary objective of the first category is to introduce a secondary task into the model. In simpler terms, a successful attack achieves its goal by maintaining its performance in the original task while introducing an additional task. The other category aims to degrade the model's performance. The most extreme form of such attacks is where adversarial clients contribute randomly generated model updates or train on data with random modifications, resulting in unpredictable model updates [79]. A particular category of training-time adversarial attacks is called *poisoning attacks*. FL's confidentiality measures make it susceptible to these attacks, either by contaminating training data called *data poisoning attacks*, or manipulating model weights before aggregation called *model poisoning attacks*. In the following, we provide a detailed review of them.

Data poisoning attacks [27, 86]. It is assumed that the attacker has access to the training data of one or more clients and possesses the capability to make alterations to it. These attacks contaminate the training dataset by introducing new malicious examples or by tampering with existing ones. The primary objective in most data poisoning attacks is to undermine the global model, consequently affecting the local models of all clients. Nevertheless, there are instances where attackers aim not to impair all local models but only a specific subset of them. In the majority of attacks described in the literature, attackers are assumed to have access to the target nodes. By following [72] there are many types of data poisoning attacks, (*i*) *label flipping attack* [39], (*ii*) *poisoning samples attack* [98, 99], (*iii*) *out-of-distribution attack* [16].

The objective of the label flipping attack is to gain control over a specific target label or class within the FL system. Typically, the attacker focuses on this target label by altering the outputs of target data samples, effectively switching them from their original source label to the target label. It can take a targeted approach, where specific labels are exchanged [86], or an untargeted approach [79], where labels are shuffled randomly.

In contrast to the previous attack, poisoning samples attacks involve altering a portion of the training data samples. The poisoning can take various forms, including introducing patterns into the samples and associating them with a specific target class, or normalizing the samples and introducing uniform noise with the intention of degrading the model's performance.

The out-of-distribution attack is similar to poisoning sample attacks, but the key distinction lies in the fact that the poisoned training samples are not modifications of the original ones. Instead, they are samples sourced from outside the input distribution. These samples can come from either a different domain with similar characteristics or they can be entirely random noise-based samples.

Model poisoning attacks [5, 81]. According to [14] data poisoning attacks have limited effectiveness when attempting to undermine robust federated learning systems. Concerning this, model poisoning attacks were studied. Their objective is to tamper with local model updates before they are transmitted to the server or insert concealed vulnerabilities into the global model [1]. These model weights can be tampered with in different ways, for example with random weights generation [14], by minimizing an objective function between the corrupted and the global model [33], and by creating a secure communication protocol for attackers [93, 12].

Furthermore, federated models are constructed through the aggregation of model updates contributed by participants. In order to safeguard the confidentiality of the training data, the aggregator is intentionally designed to remain unaware of how these updates are generated. This design aspect renders FL susceptible to a model-poisoning attack that possesses considerably more power compared to attacks targeting solely the training data.

A malicious participant can exploit model replacement techniques to introduce backdoor functionality into the collective model. For instance, they can modify an image classifier in a way that it assigns an attacker-chosen label to images with specific characteristics, or manipulate a word predictor to complete particular sentences with an attacker-specified word. These attacks can be executed by a single participant or by coordinated efforts of multiple participants working in collusion.

In model poisoning, the attacker's goal is to deliberately make the FL model misclassify a specific set of inputs with a high level of confidence. Importantly, these inputs are not altered to induce misclassification during testing, as is the case with adversarial example attacks [82]. Instead, the misclassification results from manipulations during the training process. Recent research has delved into poisoning attacks on model updates, where a subset of updates sent to the server in each iteration is tainted, often by introducing hidden vulnerabilities. Surprisingly, even a single-shot attack can be sufficient to implant a vulnerability in a model.

Bhagoji et al. [5] demonstrated that model poisoning attacks are considerably more potent than data poisoning in FL scenarios, particularly in the context of targeted model poisoning attacks, where a single malicious participant, acting independently, attempts to make the model confidently misclassify specific inputs. To enhance the stealthiness of these attacks and avoid detection, they employ an alternating minimization strategy that optimizes alternatively for the training loss and the adversarial objective. They also use parameter estimation for benign participants' updates. As a result, these adversarial model poisoning attacks can successfully target and compromise FL models without detection.

In fact, it's worth noting that model poisoning encompasses data poisoning in FL settings since data poisoning attacks eventually lead to alterations in a subset of updates transmitted to the model during each iteration [17]. This essentially mirrors the behavior of centralized poisoning attacks, where a portion of the entire training dataset is tainted.

There is a notable absence of experimental studies examining poisoning attacks on decentralized FL systems in the existing literature. In our survey [18] we observed a limited number of studies attempting to address this issue. Most of these investigations primarily rely on Blockchain technology. For example, Biscotti [77] can withstand label flipping attacks [53] by employing Multi-Krum [7], which efficiently filters out model updates that significantly deviate from the majority direction of updates. BAFL [15] addresses Gaussian reverse attacks by ensuring security through blockchain technology and implementing a consensus mechanism similar to Algorand [21]. BytoChain [42] demonstrates its security resilience by withstanding random poisoning and reverse poisoning attacks. For instance, one of the traditional distributed approaches is Trusted DFL [19] that was examined in the context of a model poisoning attack. It effectively countered the attack by transmitting weights selected randomly within a specified range. The proposed algorithm showed enhanced performance when dealing with compromised agents compared to a system lacking trust incorporation.

3.3 Defences on FL

As the range and complexity of adversarial attacks on FL continue to expand, new defensive strategies are also emerging to counteract their harmful impact. While adversarial attacks can be categorized separately, the same cannot be said for their defences, as some of them prove effective against multiple attack types. Consequently, the majority of defence mechanisms are deployed on the federated server. The federated server has the assignment to perform the aggregation. Certain aggregation operators, like the classical FedAvg [49], are vulnerable to outliers. Hence, numerous aggregation operators relying on more resilient estimators have been suggested as a remedy. In the following, we describe some of them.

Krum and Multi-Krum [7]. This is a custom-designed aggregation operator that actively prevents attacks on the federated model by filtering out model updates from clients displaying extreme behavior. It accomplishes this by evaluating the geometric distances between their model update distributions and selecting the client closest to the majority as the aggregated model. In Multi-Krum, a parameter called d is introduced, which determines the number of clients included in the aggregation process (the first d clients after sorting) to generate the final aggregated model.

Median [95]. This is a robust aggregation operator that substitutes the arithmetic mean with the median of the model updates. The median represents the central value of the distribution.

Trimmed mean [95]. It is an adaptation of the arithmetic mean, which involves excluding a predetermined percentage of outlier values both below and above the data distribution.

Bulyan [52]. As the Euclidean distance between two high-dimensional vectors can be significantly influenced by just one component, Krum might be impacted by a small number of unusual local model weights, rendering it ineffective for intricate, high-dimensional parameter spaces. The authors have created a federated aggregation operator to counteract this problem by combining the Multi-Krum federated aggregation operator with the trimmed mean. Consequently, it organizes clients based on their geometric distances and employs an f parameter to eliminate the outermost $2f$ clients from the sorted client distribution, aggregating the remaining clients.

Adapted Federated Averaging (AFA) [54]. A proposal for defending against Byzantine attacks includes a mechanism where clients are assigned weights using a Hidden Markov model. This weighting process employs cosine similarity to evaluate the quality of model updates during training. The authors have found that this approach filters out both underperforming and potentially malicious clients, leading to improvements in computational and communication efficiency.

Federated Learning leveraging ANomaly DEtection for Robust and Secure (FLANDERS) [85]. In their research, they propose an innovative aggregation method designed to withstand model poisoning attacks. This approach is rooted in a data-driven strategy that takes into account time dependency. Specifically, the authors approach the challenge of identifying malicious client updates as a task involving multidimensional time series anomaly detection, represented as a matrix-valued problem. Inspired by the matrix autoregressive (MAR) framework, the authors have developed a practical implementation of this novel Byzantine-resilient aggregation method. One key advantage of FLANDERS, when compared to existing approaches, lies in its elevated flexibility in swiftly detecting deviations in local models that may result from the actions of malicious clients.

Chapter 4

Problem Formulation

In centralized machine learning (ML) systems, implementing model poisoning is generally considered challenging because it necessitates the adversary's access to the target model. This implies that the adversary must possess either grey-box or white-box knowledge about the model. In contrast, model poisoning becomes notably more feasible in the context of Federated Learning (FL) [12]. In fact, the adversary's ultimate objective is to induce incorrect predictions from the FL model. However, their specific aim is to force classification errors during inference, all while refraining from modifying the test examples themselves. Moreover, these attacks could exhibit significant strength in a P2P context, primarily due to the emphasis on anonymity within such systems and the lack of a server equipped with an anomaly detection mechanism [87]. As mentioned in the previous chapters the literature lacks of analysis of the robustness of the decentralized FL approach. In this chapter, we provide the integration of some model poisoning attacks and decentralized aggregation schemes in order to attempt to answer the research questions.

4.1 RQ1: Attacks on a decentralized FL scenario

To address *RQ1*, we must conduct experiments involving model poisoning attacks within a decentralized FL framework. Specifically, we have opted for a "standard" attack that involves predetermined modifications to model parameters (e.g., introducing random weight generalizations), as well as two more advanced attack. Below, we provide a detailed explanation of how these attacks are implemented.

Gaussian Attack [14]. The primary objective of the Gaussian Attack is to compromise the training process of machine learning models by introducing controlled variations (perturbations) into either the input data or model parameters. This constitutes a white-box attack in which the attacker selects model weights by sampling from the Gaussian distribution generated by the model updates of the other clients. When considering a malicious node, denoted as m, among the set of nodes N, this node's model update, labeled as θ_m, is of interest. On a generic i-th round it will send a manipulated model update denoted as θ'_m to the other nodes. Specifically, in this attack, the new model update is:

$$\theta'_m = \theta_m + \epsilon, \text{ with } \epsilon \sim \mathcal{N}(0, \sigma^2) \tag{4.1}$$

A Little Is Enough Attack [3]. Most defence mechanisms share a similar assumption and aim to employ statistically robust techniques for identifying and discarding values with reported gradients that significantly deviate from the population mean. Baruch et. *al.* [3] in their work observed that when the empirical variance among worker gradients is sufficiently high, an attacker could exploit this situation and execute a non-omniscient attack that operates within the population variance. The research demonstrates that this variance is indeed substantial, even in the case of relatively simple datasets like MNIST. This permits an attack that not only remains undetected by existing defences but also turns their capabilities against them. Consequently, these defence mechanisms tend to consistently select malicious workers for retention while discarding legitimate ones. The study showcases the effectiveness of this attack method, not only in terms of impeding convergence but also in manipulating the model's behavior, a technique known as "backdooring". In this study, their primary focus is directed towards the approach that addresses convergence. Assume that with b_{nodes} we refer to the number of malicious nodes, and with n_{nodes} the number of normal nodes. First of all, we must compute the number of supporters s given by:

$$s = \lfloor \frac{n_{nodes}}{2} + 1 \rfloor - b_{nodes} \tag{4.2}$$

After this, we have to compute the standard deviation and the mean vector of the parameters denoted as $\mu[layer]$, $\sigma[layer] \in \mathcal{R}^{|n|}$ with $layer$ referred to as the model parameter and $|n|$ the number of model parameters (dimensions of the model). The attacker will focus on the Cumulative Standard Normal Function $\phi(z)$ implemented with the *percent point function* and aim to identify the maximum value of z, denoted as z^{max}. This maximum value indicates a scenario where a group of non-corrupted workers is situated farther from the mean. Consequently, they tend to favor the selection of the byzantine parameters over those that are more distant but correct. This value is given by:

$$z^{max} = max_z\Big(\phi(z) < \frac{n-m-s}{n-m}\Big) \tag{4.3}$$

Finally, we set all corrupted workers model parameters with a perturbation range given by $(\mu - z^{max}\sigma, \mu + z^{max}\sigma)$. In this way, the defences will not be able to differentiate the corrupted workers from the benign. Specifically, we set the model parameters as:

$$model[layer] = \mu[layer] - z^{max}\sigma[layer] \tag{4.4}$$

Fall of Empires Attack [92]. In their paper, the authors investigate the vulnerabilities of two commonly employed aggregation methods. They specifically demonstrate that robust aggregation techniques for synchronous Stochastic Gradient Descent (SGD), namely Median and Krum, can be compromised through the implementation of novel attack strategies centered around inner product manipulation. These findings are substantiated through theoretical proofs and empirical validation and in particular, they describe a definition. Assume that U is the set of malign nodes, and V is the set of benign nodes. If we have $m - q$ gradients, with q defined as malicious gradients, and $\mathbb{E}[v_i] = g$ with $v_i \in V$, $\forall i \in [m - q]$, an aggregation rule is tolerant to malicious nodes if $\langle g, \mathbb{E}[Aggr(V \cap U)]\rangle \geq 0$.

The methodology employed to execute this attack is very simple. In fact, if ϵ is a number called magnitude greater than 0 and μ is the mean of the model parameters, we have that the new model parameters are:

$$model[layer] = -\epsilon\mu[layer]$$

4.2 RQ2: Decentralized Aggregation Schemes

Model aggregation in FL traditionally relies on a central server, which employs a selected algorithm to construct a global model. However, in a decentralized scenario, the absence of a central server and a global model poses a challenge. To address *RQ2*, we must adapt some of the aggregation schemes discussed in Section 3.3. Our approach involves each client in the network performing its own aggregation for its specific purposes, acting as a local server for itself and its local model or the aggregated model in subsequent steps. Below, we provide a detailed explanation of the implementation of three aggregation schemes: *(i) FedAvg* (*(ii) Multi-Krum*, and *(iii) Median.*

FedAvg [49]. This aggregation method serves as the conventional and initial approach used in the development of FL. In this process, each client independently performs one gradient descent step on the current model using its local data, and subsequently, the server computes a weighted average of the resultant models. This averaging process allows the global model to benefit from the insights acquired across diverse clients while preserving data privacy.

In a decentralized scenario where there is neither a global model nor a central server, the aggregation process operates in a similar fashion. In fact, in PENS [59], the node that receives the models performs an averaging operation with its own model. Consequently, each client obtains its newly aggregated model for the next round. This is summarized in algorithm 1.

Algorithm 1 Decentralized FedAvg, w_i^t model parameters of client i at round t and W^t set of the received model parameters at round t.

```
function decFedAvg($w_i^t$, $W^t$)
    $n \leftarrow |W^t| + 1$
    $s \leftarrow 0$
    for $w_j^t \in W^t$ do
        $s \leftarrow s + w_j^t$
    end for
    $w_i^{t+1} \leftarrow \frac{w_i^t + s}{n}$
    return $w_i^{t+1}$
end function
```

Krum [7]. Selecting the appropriate method for aggregating the models presents a significant challenge. One might be tempted to consider a non-linear aggregation rule based on squared distances, which involves choosing the vector that is "closest to the barycenter". For instance, this could be achieved by selecting the vector that minimizes the sum of squared distances to all other vectors. However, it's important to note that such a squared-distance-based aggregation rule can only tolerate a single malicious node. If two malicious nodes collaborate, with one assisting the other to get selected, they can collectively move the barycenter of all vectors further away from the "correct area". The researchers propose Krum to overcome this problem. To achieve such resilience they consider a majority-based method, where all possible subsets of $n - f$ vectors are examined, and the subset with the smallest diameter is chosen. Interestingly, by combining the insights from both the majority-based and squared-distance-based methods, we can select the vector that is, in some sense, closest to its $n - f$ neighbors. In other words, we choose the vector that minimizes the sum of squared distances to its $n - f$ nearest vectors. Krum picks one among the m local models received from clients, specifically the one that exhibits the highest similarity with all other models, to serve as the global model. This choice is based on the rationale that even if the selected local model originates from a malicious client, its influence would be constrained because of its similarity to other local models, which may be from benign clients. In a decentralized context, it becomes necessary to implement Multi-Krum, which represents a generalization involving the selection of more than one model and then executing FedAvg on these selected models. Moreover, in the absence of a central server, the similarity distance matrix simplifies into a vector, eliminating the need for scores. This is because each client independently executes this aggregation method for their own purposes. This idea is summarized in algorithm 2.

Algorithm 2 Decentralized Multi-Krum, w_i^t model parameters of client i at round t and W^t set of the received model parameters at round t, tk is how many models you want to perform aggregation, and b the number of malicious nodes.

```
function DECMULTIKRUM($w_i^t$, $W^t$, $tk$, $b$)
    $d_i^t \leftarrow$ COMPUTE_DISTANCES($w_i^t, W^t$)
    $num\_closest \leftarrow max(1, |W^t| - b - 2)$
    $closest\_idxs \leftarrow argsort(d_i^t[(num\_closest + 1)...|W^t|])$
    $best\_idxs \leftarrow closest\_idxs[tk...|W^t|]$
    for $i \in best\_idxs$ do
        $best\_results.insert(W^t[i])$
    end for
    $w_i^{t+1} =$ DECFEDAVG($w_i^t, best\_results$)
    return $w_i^{t+1}$
end function
function COMPUTE_DISTANCES($w_i^t$, $W^t$)
    set $vec$ as an empty list of dimension $|W^t|$
    $n \leftarrow |W^t|$
    for $j \in [n]$ do
        $vec[j] \leftarrow ||w_i^t - W^t[j]||^2$
    end for
    return $vec$
end function
```

Median [95]. FL relies on the aggregation of updates provided by participating devices, ensuring that this aggregation process maintains privacy. However, a vulnerability inherent in the typical approach lies in its susceptibility to corrupted updates. These corruptions can occur due to malicious adversaries aiming to disrupt the system or as a result of failures in low-cost hardware. The standard method of arithmetic mean aggregation in federated learning lacks robustness when it comes to handling these corruptions. Even a single corrupted update in a round can lead to the deterioration of the global model for all devices. In one dimension, the median emerges as an appealing alternative for aggregation due to its resilience against outliers. In fact, in the case of this aggregation method, the central server arranges the parameters from all the m local models for the j-th parameter and selects the median value to assign as the j-th parameter for the global model. Given the absence of a central server, each node calculates the median with its own model and the models it receives. In algorithm 3 this idea is illustrated.

Algorithm 3 Decentralized Median, w_i^t model parameters of client i at round t and W^t set of the received model parameters at round t.

```
function DECMEDIAN($w_i^t$, $W^t$)
    set $list$ as an empty list of dimension $|W^t| + 1$
    $list.insert(W^t, w_i^t)$
    $w_i^{t+1} \leftarrow Median(list)$
    return $w_i^{t+1}$
end function
```

Chapter 5

Proposed Method

5.1 PENS: Performance-based neighbor selection

In centralized FL, a central server coordinates the learning process among clients and handles the aggregation of parameters once it receives updates from these clients. On the other hand, decentralized FL operates without a global model state. Instead, the participating clients adhere to a communication protocol designed to establish a consensus on the model during training. Commonly used methods for decentralized learning include gradient-based algorithms are made through gossip learning techniques [8, 28, 62]. In these methods, clients individually train their models using local data and use a communication protocol where they randomly share (gossip) their model parameters with neighboring clients. The ultimate objective for the participating clients is to converge toward a consensus on a high-quality model. Both centralized and decentralized FL approaches tackle a crucial question of how to develop a personalized model that performs effectively when faced with varying client data distributions, commonly referred to as the non-iid data setting.

In *PENS* [59] they investigate the performance of deep neural networks within decentralized peer-to-peer networks, particularly in scenarios where data exhibits non-iid characteristics. More specifically, their focus centers on addressing the challenge of covariate shift, where the distribution of $\mathcal{D}_i(x)$ varies, but the conditional distributions $\mathcal{D}_i(y|x) = \mathcal{D}_j(y|x)$ remain consistent across all clients $i \neq j$. Within the context of this work, clients who share similar data distributions are more likely to engage in collaboration, while those with dissimilar data distributions are less inclined to collaborate.

Specifically, the algorithm is divided into two distinct phases:

(i) In the initial phase, the objective is to identify peers with data distributions $\mathcal{D}_i^r(x)$ that closely match each other.

(ii) The second phase involves selecting peers who have been chosen more frequently than expected and labeling them as "neighbors with similar data distribution". During this phase, the algorithm employs gossip learning techniques.

To identify peers with the same data distributions they propose a loss-based selection, in which each client performs random communication within the network for a predetermined number of rounds denoted as T. At arbitrary intervals, a client i

sends its model to j, followed by the calculation of $\mathcal{L}_j(w_i, z)$, where z is randomly sampled from $\mathcal{D}_j$. This represents the loss of the client's model w_i on the training dataset of client j. Each client awaits the collection of a specified number of models and maintains a list of their corresponding losses. Subsequently, the top-performing clients, those with the lowest loss values, are identified as potential neighbors with similar data distributions. The model parameters of these selected clients are then aggregated to create a new model.

5.2 Threat Model

Adversarial Model. We investigate a scenario where a subset of participants can be identified as malicious, and this subset is quantified as a percentage of the total participants. The primary objective of these malicious participants is to send poisoned models with the intent of degrading the overall accuracy of the framework. These attacks are classified as untargeted, in contrast to targeted poisoning attacks where the aim is to deliberately cause prediction errors for particular test instances.

For instance, let's suppose that N is the number of total nodes in the network, and b is the number of malicious nodes, and $p \in [0, 1]$ is the percentage of malicious clients. We have that $b = \lfloor p \cdot N \rfloor$ and $0 \leq b \leq N$. The attacker possesses the capability to manipulate the local models and send them to the other clients in an arbitrary manner.

Attacker's Knowledge. We assume that the attacker possesses information about the code, local training datasets, and local models residing on the clients under their control. Specifically, they perform these attacks on the received models from the other clients. This contrasts with attacks conducted in a centralized scenario, where malicious nodes have knowledge of all the models. In centralized scenarios, the primary goal is typically to manipulate the global model. However, in this context, our objective is to alter the model sent from the malicious node to the benign one. Additionally, since our focus in this work pertains to the resilience of the decentralized idea, we consider the worst-case scenario, which, from the perspective of an honest FL system, involves the attacker having knowledge of the aggregation function employed by the central server.

Attacker's Behaviour. We assume that from the first round, a portion of clients are acting maliciously. Consequently, their actions can weaken the average accuracy of the clients' models. Additionally, given the absence of a global model, this situation can also have a harmful impact on the models of honest clients who cannot be selected during neighbor selection. To ensure generality, we make the assumption that the distribution of both malicious and benign clients is similar.

Chapter 6

Experimental Results

6.1 Experimental Setup

Datasets and Models. We work with three datasets and a classification task, consisting of two image-based datasets and one tabular dataset, namely MNIST, Fashion-MNIST, and Spambase. Our dataset preprocessing involves splitting them into two parts: a training subset and a test subset. To accomplish this, we randomly shuffle each dataset and allocate 80% of the data for training while reserving the remaining 20% for testing. In the end, for each combination of datasets, we select two simple FL models, Logistic Regression (LogReg), and Multilayer Perceptron (MLP). All these models experience training with the objective of minimizing cross-entropy loss, and in each round of FL, every client carries out one epoch of stochastic gradient descent (SGD). As an evaluation metric, we decided to use the average accuracy of all models of the nodes. In particular:

- ***MNIST.*** This dataset is based on handwritten digits that are commonly used for training various image processing systems. It contains 60,000 training images and 10,000 testing images. Each of these examples is a 28x28 grayscale image, associated with 10 classes. For this particular dataset, our choice is to employ a Multilayer Perceptron (MLP) implemented within the Gossipy framework. Specifically, we configure the learning rate at $\eta = 0.1$, choose the SGD optimizer, and adopt the cross-entropy loss criterion.

- ***Fashion-MNIST.*** Fashion-MNIST is a dataset featuring images of Zalando's fashion items, and it comprises a training set containing 60,000 examples and a test set comprising 10,000 examples. Each example is a grayscale image measuring 28x28 pixels and is associated with one of 10 different class labels, representing various fashion categories. Zalando has designed Fashion-MNIST as a suitable alternative to the original MNIST dataset for the purpose of benchmarking machine learning algorithms. It mimics the same image size and follows a similar structure for training and testing data division. For Fashion-MNIST we use the same MLP used for MNIST, the same optimizer, but we change only the learning rate $\eta = 10^{-4}$.

- ***Spambase.*** The Spambase dataset contains a total of 4601 email messages, with 1813 of them categorized as spam. This dataset encompasses 57 distinct features, which encompass attributes such as word frequency, character occurrences, and other characteristics extracted from the emails. For this dataset based on continuous features, we decided to use a LogisticRegression (LogReg) implemented within the Gossipy framework. We set the optimizer as SGD and the learning rate $\eta = 0.1$.

Decentralized FL simulation environment. To create a realistic decentralized FL environment, we use Gossipy[1], which is one of the most suited frameworks for our purpose. Specifically, this framework is a Python module for simulating gossip learning and decentralized federated learning.

With Gossipy, we possess the capability to determine the number of FL clients, and for our research, we've designated this number as 50. Furthermore, for PENS, we set $n_{sampled} = 5$ as the specified number of received models and $m_{top} = 2$ as the number of top-performing clients. Additionally, we've made the assumption that in every FL round, attacks are launched, signifying that b (the count of malicious clients) constitutes a fixed percentage of the total nodes N. Our experiment includes a total of 50 rounds. Notably, we have chosen to deactivate the second step of PENS, which involves gossip learning. This decision arises from the belief that, in the case of attacks, employing this algorithm would not yield significant benefits. We investigate deeper into this specific aspect in Section 6.3.

6.2 Evaluation

We compare the robustness of the decentralized framework employer with the three aggregation schemes illustrated in 4.2: FedAvg, Multi-Krum, and Median, and the attacks described in 4.1: Gaussian Attack, A Little is Enough Attack, and Fall of Empires Attack. Specifically, for each of them, we set some parameters. Concerning the attacks, we set for the Gaussian Attack the magnitude $\sigma = 1$, for Fall of Empires Attack we set $\epsilon = 10$. For the defences, in Multi-Krum we set $tk = 1$. Finally, we decided to set the number of malicious nodes as $b = \{3, 5, 15, 25\}$, respectively the probabilities $p = \{0.05, 0.1, 0.3, 0.5\}$. Notably, when the proportion of attackers does not exceed 50% of the total number of nodes, PENS proves to be resilient against model poisoning attacks.

RQ1. To demonstrate the robustness of decentralized FL when it comes to defending against model poisoning attacks, our objective is to evaluate the effectiveness of the approach using the FedAvg algorithm, because this is the "standard" aggregation scheme. The robustness of this method is shown below in Fig 6.1, Fig 6.2, Fig 6.3, specifically, the subfigures with a, d, g as letters. In particular, these plots depict the average accuracy of the models changes for each communication round and for each of the attacks described before. As evident from the results, the decentralized FL approach exhibits resilience against all the proposed attacks. This is evident from the fact that not only does the accuracy remain consistently high, but it also demonstrates improvement as the communication rounds progress.

[1]`https://github.com/makgyver/gossipy`

RQ2. Above, we demonstrate that the decentralized FL approach is resilient even with a classical aggregation scheme as FedAvg. In this work, we also want to demonstrate that the adapted defence aggregation methods perform effectively in decentralized FL environments. Concerning this from the plots in Fig 6.1, Fig 6.2, Fig 6.3, more precisely the subfigures with letters different that a, d, g, we can say that also with other kinds of aggregation schemes, the approach performs effectively. Moreover, the decentralized approach exhibits significant resilience against these attacks, whether employing simpler aggregation schemes (like Median) or more robust alternatives (like Multi-Krum). Furthermore, the performances are similar between each aggregation scheme, so a more simpler and efficient one is the greatest choice.

6.3 Impact of Gossip Learning Algorithm

In PENS [59] they propose two steps of their algorithm and the second step is based on *gossip learning*. This algorithm randomly selects a peer that receives the local model update initiated by the calling node. When the receiver accumulates a number of models exceeding the count of peers, it proceeds to aggregate them and train the resulting aggregated model.

However, from our studies, we have seen that this algorithm is not suited when there are malicious nodes inside the network. For instance, let's assume that n is a benign node, and m a malicious node. When PENS executes the second step, the gossip learning protocol is called. Following the initial step, each node within the network possesses a set of "highly chosen" neighbors, which serves as the basis for the random selection process during gossip learning. In this context, even the malicious node m maintains a group of best-suited neighbors, and it engages in gossip learning just like a benign node. When a benign node n receives the model from m during the aggregation step, it is constrained to include this model in the aggregation process, leading to a decline in accuracy.

According to the theoretical assumptions outlined earlier, the following experiments serve to validate them. These experiments employ the MNIST dataset and incorporate gossip learning after a specified number of rounds, denoted as $T = 30$. The experiments also involve the aggregation schemes and model poisoning attacks detailed in Sections 4.1 and 4.2. As evident from the graphs, there is a noticeable decrease in accuracy following the completion of the rounds denoted as T. This drop is particularly evident in cases where DecMedian was selected as the aggregation scheme (Fig 6.4c, 6.4f, 6.4i). However, when we opt for DecFedAvg and DecMultiKrum, the accuracy exhibits somewhat noisy fluctuations but doesn't exhibit a distinct decline (Fig 6.4a, 6.4b, 6.4d, 6.4e), especially in scenarios involving Gaussian or LIE attacks. Furthermore, when FOE attack is performed, we can see a significant accuracy drop in all three aggregation schemes (Fig 6.4g, 6.4h, 6.4i).

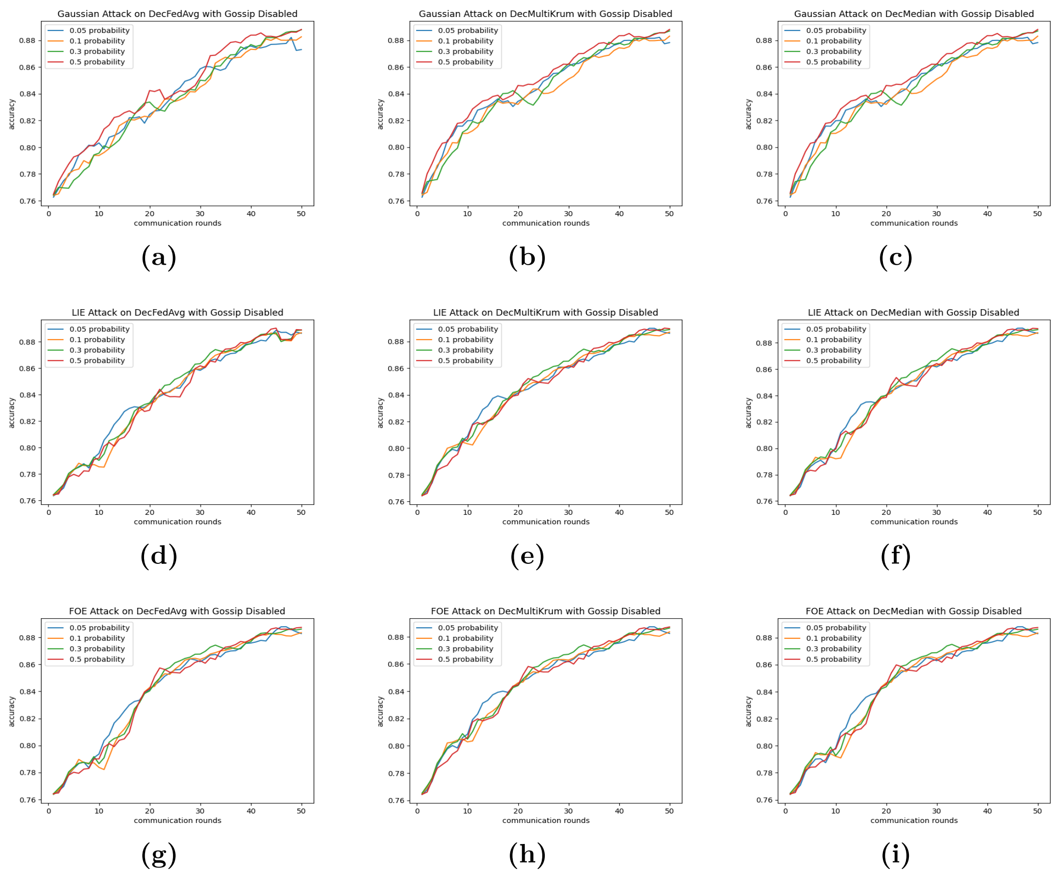
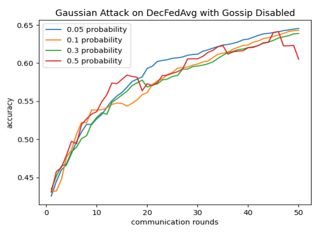
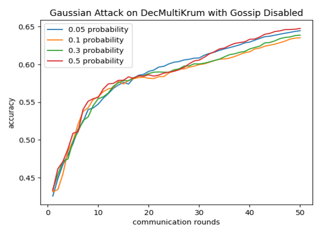


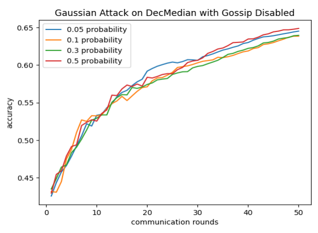
Figure 6.1. Comparing the robustness of PENS, with MNIST dataset. For each row we have the aggregation schemes, and for each column the attacks.



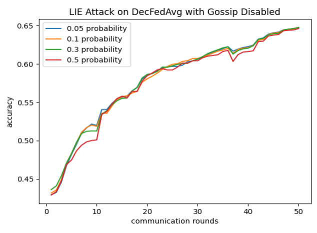
(a)



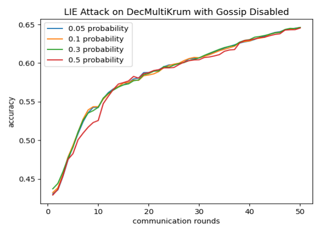
(b)



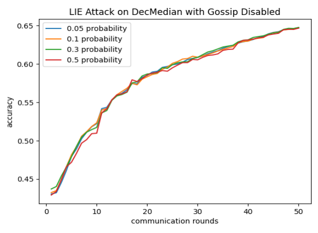
(c)



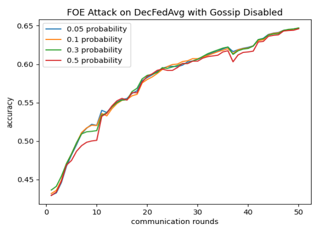
(d)



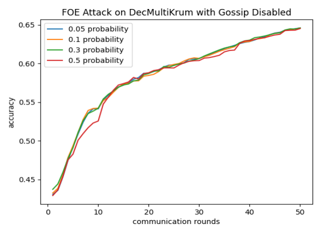
(e)



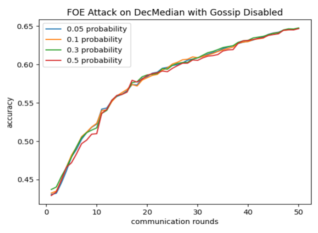
(f)



(g)

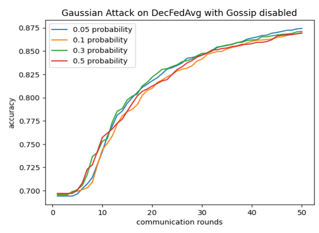


(h)

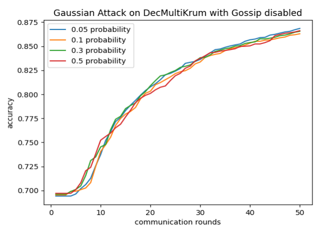


(i)

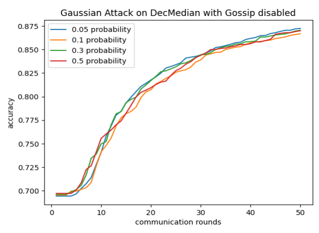
Figure 6.2. Comparing the robustness of PENS, with Fashion-MNIST dataset. For each row we have the aggregation schemes, and for each column the attacks.



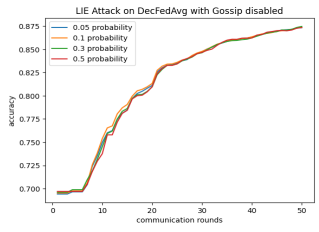
(a)



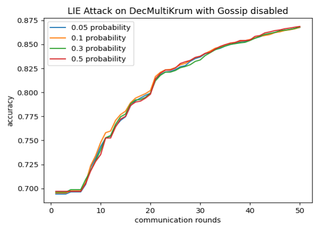
(b)



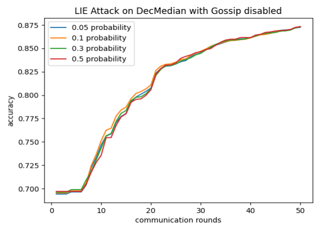
(c)



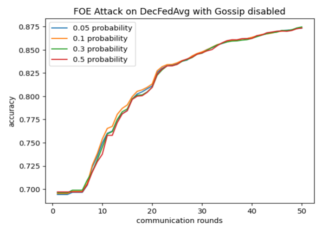
(d)



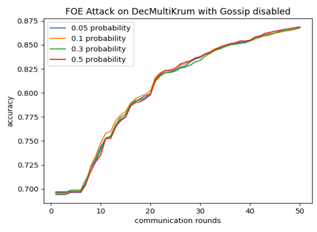
(e)



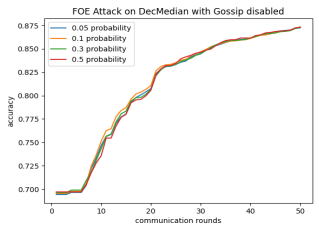
(f)



(g)

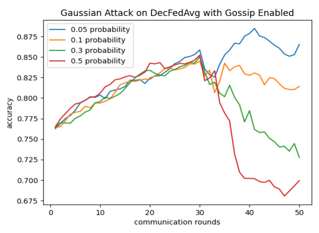


(h)

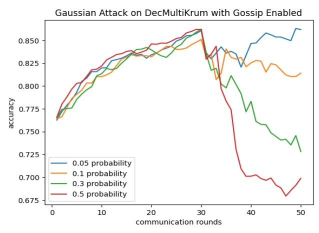


(i)

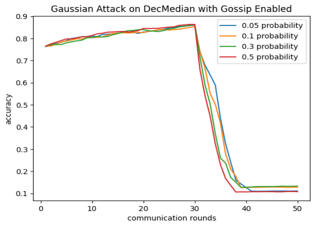
Figure 6.3. Comparing the robustness of PENS, with Spambase dataset. For each row we have the aggregation schemes, and for each column the attacks.



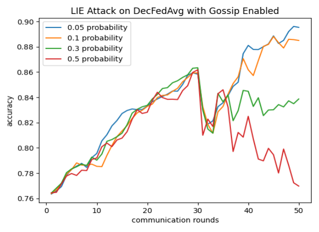
(a)



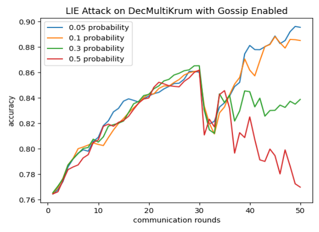
(b)



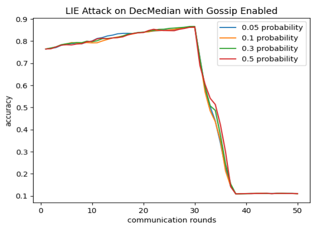
(c)



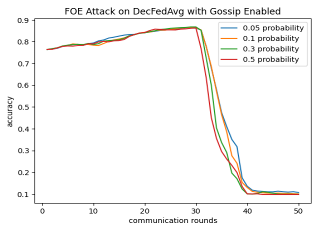
(d)



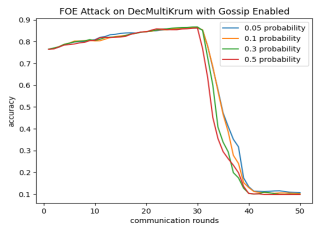
(e)



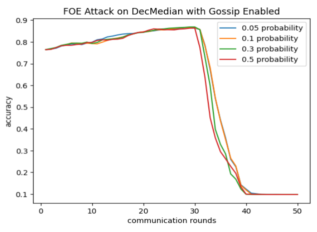
(f)



(g)



(h)



(i)

Figure 6.4. Comparing the robustness of PENS, with MNIST dataset and gossip learning enabled. For each row we have the aggregation schemes, and for each column the attacks.

Chapter 7

Conclusion and Future Works

In recent times, federated learning (FL) has obtained popularity as a paradigm for training extensive, distributed, and privacy-centric ML systems. However, FL encounters various challenges. Among these, a paramount concern revolves around mitigating the vulnerabilities associated with the centralized orchestration found in conventional FL client-server architectures, which are susceptible to risks related to a single point of failure. To tackle these concerns, decentralized FL solutions have arisen, offering FL clients the ability to collaborate and communicate independently, without relying on a central server. Nevertheless, both of these methodologies are susceptible to a form of attack known as model poisoning. Model poisoning exploits the intrinsic feature of federated learning, wherein malicious participants possess the ability to exert direct influence over the shared model. This opens the door to attacks that are considerably more powerful compared to conventional data poisoning methods typically employed in traditional training scenarios. It's noteworthy that these attacks have predominantly been studied within the framework of standard federated learning approaches.

In this research, we have presented a novel study investigating the resilience of decentralized approaches. This study involves adapted implementations derived from centralized FL and includes empirical findings. Notably, we have demonstrated that the decentralized approach, which relies on neighbor selection, exhibits robustness against three distinct types of attacks. Furthermore, this approach maintains its robustness even when utilizing a standard adapted aggregation scheme like FedAvg. Looking ahead, our future objectives include verifying the resilience of these approaches against newly developed attacks specifically tailored for this approach. Additionally, we aim to extend our experimental findings to envelop other datasets and various neural network architectures. Finally, we plan to expand our experiments to incorporate other decentralized FL approaches that do not have the requirement for gossip learning or incorporate a more generalized variant of it, and may not necessarily rely on neighbor selection.

Bibliography

[1] Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. How to backdoor federated learning. In *Proc. of AISTATS '20* (edited by S. Chiappa and R. Calandra), vol. 108, pp. 2938–2948. PMLR (2020). Available from: `https://proceedings.mlr.press/v108/bagdasaryan20a.html`.

[2] Barkai, D. An introduction to peer-to-peer computing. *Intel Developer update magazine*, (2000), 1.

[3] Baruch, G., Baruch, M., and Goldberg, Y. A Little Is Enough: Circumventing Defenses for Distributed Learning. In *Proc. of NeurIPS '19*, pp. 8632–8642 (2019). Available from: `https://proceedings.neurips.cc/paper/2019/hash/ec1c59141046cd1866bbbcdfb6ae31d4-Abstract.html`.

[4] Bellet, A., Guerraoui, R., Taziki, M., and Tommasi, M. Personalized and private peer-to-peer machine learning. In *Proc. of AISTATS '18* (edited by A. Storkey and F. Perez-Cruz), vol. 84, pp. 473–481. PMLR (2018). Available from: `https://proceedings.mlr.press/v84/bellet18a.html`.

[5] Bhagoji, A. N., Chakraborty, S., Mittal, P., and Calo, S. Analyzing federated learning through an adversarial lens. In *Proc. of ICML* (edited by K. Chaudhuri and R. Salakhutdinov), vol. 97, pp. 634–643. PMLR (2019). Available from: `https://proceedings.mlr.press/v97/bhagoji19a.html`.

[6] BitTorrent. `https://www.bittorrent.com` (2023).

[7] Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Adv. in NeurIPS* (edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett), vol. 30. Curran Associates, Inc. (2017). Available from: `https://proceedings.neurips.cc/paper_files/paper/2017/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf`.

[8] Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. Randomized gossip algorithms. *IEEE transactions on information theory*, **52** (2006), 2508.

[9] Cao, X., Fang, M., Liu, J., and Gong, N. Z. Fltrust: Byzantine-robust federated learning via trust bootstrapping (2022). `arXiv:2012.13995`.

[10] Chen, Y., Qin, X., Wang, J., Yu, C., and Gao, W. Fedhealth: A federated transfer learning framework for wearable healthcare. *IEEE Intell. Syst.*, **35** (2020), 83. `doi:10.1109/MIS.2020.2988604`.

[11] Christen, P. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection.* Springer Publishing Company, Incorporated (2012). ISBN 3642311636.

[12] Costa, G., Pinelli, F., Soderi, S., and Tolomei, G. Turning federated learning systems into covert channels. *IEEE Access*, **10** (2022), 130642.

[13] El Mhamdi, E. M., Guerraoui, R., and Rouault, S. The hidden vulnerability of distributed learning in Byzantium. In *Proceedings of the 35th International Conference on Machine Learning* (edited by J. Dy and A. Krause), vol. 80 of *Proceedings of Machine Learning Research*, pp. 3521–3530. PMLR (2018). Available from: `https://proceedings.mlr.press/v80/mhamdi18a.html`.

[14] Fang, M., Cao, X., Jia, J., and Gong, N. Z. Local model poisoning attacks to byzantine-robust federated learning (2021). `arXiv:1911.11815`.

[15] Feng, L., Zhao, Y., Guo, S., Qiu, X., Li, W., and Yu, P. Bafl: A blockchain-based asynchronous federated learning framework. *IEEE Trans. Comput.*, **71** (2022), 1092. `doi:10.1109/TC.2021.3072033`.

[16] Fort, S., Ren, J., and Lakshminarayanan, B. Exploring the limits of out-of-distribution detection (2021). `arXiv:2106.03004`.

[17] Fung, C., Yoon, C. J. M., and Beschastnikh, I. Mitigating sybils in federated learning poisoning (2020). `arXiv:1808.04866`.

[18] Gabrielli, E., Pica, G., and Tolomei, G. A survey on decentralized federated learning (2023). `arXiv:2308.04604`.

[19] Gholami, A., Torkzaban, N., and Baras, J. S. Trusted decentralized federated learning. In *IEEE CCNC*, pp. 1–6 (2022). `doi:10.1109/CCNC49033.2022.9700624`.

[20] Ghosh, A., Chung, J., Yin, D., and Ramchandran, K. An efficient framework for clustered federated learning (2021). `arXiv:2006.04088`.

[21] Gilad, Y., Hemo, R., Micali, S., Vlachos, G., and Zeldovich, N. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proc. of SOSP '17*, p. 51–68. Association for Computing Machinery, New York, NY, USA (2017). ISBN 9781450350853. Available from: `https://doi.org/10.1145/3132747.3132757`, `doi:10.1145/3132747.3132757`.

[22] Gnutella. `https://rfc-gnutella.sourceforge.net/` (2023).

[23] Hardy, S., Henecka, W., Ivey-Law, H., Nock, R., Patrini, G., Smith, G., and Thorne, B. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *CoRR*, **abs/1711.10677** (2017). Available from: `http://arxiv.org/abs/1711.10677`, `arXiv:1711.10677`.

[24] Hitaj, B., Ateniese, G., and Perez-Cruz, F. Deep models under the gan: Information leakage from collaborative deep learning (2017). arXiv:1702.07464.

[25] Hu, C., Jiang, J., and Wang, Z. Decentralized federated learning: A segmented gossip approach (2019). arXiv:1908.07782.

[26] Isdal, T., Piatek, M., Krishnamurthy, A., and Anderson, T. Privacy-preserving p2p data sharing with oneswarm. In *Proc. of SIGCOMM '10*, p. 111–122. Association for Computing Machinery, New York, NY, USA (2010). ISBN 9781450302012. Available from: https://doi.org/10.1145/1851182.1851198, doi:10.1145/1851182.1851198.

[27] Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., and Li, B. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 19–35 (2018). doi:10.1109/SP.2018.00057.

[28] Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.-M., and Van Steen, M. Gossip-based peer sampling. *ACM Transactions on Computer Systems (TOCS)*, **25** (2007), 8.

[29] Kairouz, P., et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, **14** (2021), 1.

[30] Kalra, S., Wen, J., Cresswell, J. C., Volkovs, M., and Tizhoosh, H. Decentralized federated learning through proxy model sharing. *Nature Communications*, **14** (2023), 2899.

[31] Kang, J., Xiong, Z., Niyato, D., Yu, H., Liang, Y.-C., and Kim, D. I. Incentive design for efficient federated learning in mobile networks: A contract theory approach. In *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*, pp. 1–5 (2019). doi:10.1109/VTS-APWCS.2019.8851649.

[32] Kermarrec, A.-M. and Taiani, F. Want to scale in centralized systems? think p2p. *Journal of Internet Services and Applications*, **6** (2015). doi:10.1186/s13174-015-0029-1.

[33] Koh, P. W., Steinhardt, J., and Liang, P. Stronger data poisoning attacks break data sanitization defenses (2021). arXiv:1811.00741.

[34] Kumar, H. H., V R, K., and Nair, M. K. Federated k-means clustering: A novel edge ai based approach for privacy preservation. In *IEEE CCEM*, pp. 52–56 (2020). doi:10.1109/CCEM50674.2020.00021.

[35] Lalitha, A. Fully decentralized federated learning (2018).

[36] Li, C., Li, G., and Varshney, P. K. Decentralized federated learning via mutual knowledge transfer. *IEEE Internet of Things Journal*, **9** (2022), 1136. doi:10.1109/JIOT.2021.3078543.

[37] Li, O., Sun, J., Yang, X., Gao, W., Zhang, H., Xie, J., Smith, V., and Wang, C. Label leakage and protection in two-party split learning (2022). arXiv:2102.08504.

[38] Li, Q., Wen, Z., and He, B. Practical federated gradient boosting decision trees. *Proceedings of the AAAI Conference on Artificial Intelligence*, **34** (2020), 4642. Available from: https://ojs.aaai.org/index.php/AAAI/article/view/5895, doi:10.1609/aaai.v34i04.5895.

[39] Li, X., Qu, Z., Zhao, S., Tang, B., Lu, Z., and Liu, Y. Lomar: A local defense against poisoning attack on federated learning (2022). arXiv:2201.02873.

[40] Li, X., Zhao, P., and Li, L. Resilience and reliability analysis of p2p network systems. *Operations Research Letters*, **38** (2010), 20. Available from: https://www.sciencedirect.com/science/article/pii/S0167637709001084, doi:https://doi.org/10.1016/j.orl.2009.09.006.

[41] Li, Z., Lu, J., Luo, S., Zhu, D., Shao, Y., Li, Y., Zhang, Z., Wang, Y., and Wu, C. Towards effective clustered federated learning: A peer-to-peer framework with adaptive neighbor matching. *IEEE Transactions on Big Data*, (2022), 1. doi:10.1109/TBDATA.2022.3222971.

[42] Li, Z., Yu, H., Zhou, T., Luo, L., Fan, M., Xu, Z., and Sun, G. Byzantine resistant secure blockchained federated learning at the edge. *IEEE Network*, **35** (2021), 295. doi:10.1109/MNET.011.2000604.

[43] Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Adv. in NeurIPS* (edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett), vol. 30. Curran Associates, Inc. (2017). Available from: https://proceedings.neurips.cc/paper_files/paper/2017/file/f75526659f31040afeb61cb7133e4e6d-Paper.pdf.

[44] Long, G., Xie, M., Shen, T., Zhou, T., Wang, X., and Jiang, J. Multi-center federated learning: clients clustering for better personalization. *World Wide Web*, **26** (2022), 481. Available from: https://doi.org/10.1007%2Fs11280-022-01046-x, doi:10.1007/s11280-022-01046-x.

[45] Lu, Y. and Fan, L. An efficient and robust aggregation algorithm for learning federated cnn. In *Proceedings of the 2020 3rd International Conference on Signal Processing and Machine Learning*, pp. 1–7 (2020).

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

[47] Ma, X., Zhu, J., Lin, Z., Chen, S., and Qin, Y. A State-of-the-Art Survey on Solving Non-IID Data in Federated Learning. *Future Generation Computer Systems*, **135** (2022), 244. Available from: https://

www.sciencedirect.com/science/article/pii/S0167739X22001686, doi: https://doi.org/10.1016/j.future.2022.05.003.

[48] Marti, S. and Garcia-Molina, H. Identity crisis: anonymity vs reputation in p2p systems. In *Proceedings Third International Conference on Peer-to-Peer Computing (P2P2003)*, pp. 134–141 (2003). doi:10.1109/PTP.2003.1231513.

[49] McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proc. of AISTATS '17'* (edited by A. Singh and J. Zhu), vol. 54, pp. 1273–1282. PMLR (2017). Available from: https://proceedings.mlr.press/v54/mcmahan17a.html.

[50] McMahan, B. and Ramage, D. Federated learning: Collaborative machine learning without centralized training data. *Google Research Blog*, **3** (2017).

[51] Melis, L., Song, C., Cristofaro, E. D., and Shmatikov, V. Exploiting unintended feature leakage in collaborative learning (2018). arXiv:1805.04049.

[52] Mhamdi, E. M. E., Guerraoui, R., and Rouault, S. The hidden vulnerability of distributed learning in byzantium (2018). arXiv:1802.07927.

[53] Miller, B., Kantchelian, A., Afroz, S., Bachwani, R., Dauber, E., Huang, L., Tschantz, M. C., Joseph, A. D., and Tygar, J. Adversarial active learning. In *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, AISec '14, p. 3–14. Association for Computing Machinery, New York, NY, USA (2014). ISBN 9781450331531. Available from: https://doi.org/10.1145/2666652.2666656, doi:10.1145/2666652.2666656.

[54] Muñoz-González, L., Co, K. T., and Lupu, E. C. Byzantine-robust federated machine learning through adaptive model averaging (2019). arXiv:1909.05125.

[55] Napster. https://www.napster.com/it (2023).

[56] Nasr, M., Shokri, R., and Houmansadr, A. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE (2019). Available from: https://doi.org/10.1109%2Fsp.2019.00065, doi:10.1109/sp.2019.00065.

[57] Nedic, A. and Ozdaglar, A. Distributed subgradient methods for multi-agent optimization. *Automatic Control, IEEE Transactions on*, **54** (2009), 48 . doi:10.1109/TAC.2008.2009515.

[58] Nosowsky, R. and Giordano, T. J. The health insurance portability and accountability act of 1996 (hipaa) privacy rule: implications for clinical research. *Annu. Rev. Med.*, **57** (2006), 575.

[59] Onoszko, N., Karlsson, G., Mogren, O., and Zec, E. L. Decentralized federated learning of deep neural networks on non-iid data (2021). arXiv:2107.08517.

[60] OpenAI. Gpt-4 technical report (2023). arXiv:2303.08774.

[61] Oppenlaender, J. The creativity of text-to-image generation. In *Proceedings of the 25th International Academic Mindtrek Conference*. ACM (2022). Available from: https://doi.org/10.1145%2F3569219.3569352, doi:10.1145/3569219.3569352.

[62] Ormándi, R., Hegedűs, I., and Jelasity, M. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, **25** (2013), 556.

[63] Pan, S. J. and Yang, Q. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, **22** (2010), 1345. doi:10.1109/TKDE.2009.191.

[64] Pappas, C., Chatzopoulos, D., Lalis, S., and Vavalis, M. Ipls: A framework for decentralized federated learning. In *2021 IFIP Networking Conference (IFIP Networking)*, pp. 1–6 (2021). doi:10.23919/IFIPNetworking52078.2021.9472790.

[65] Peng, Z., Xu, J., Chu, X., Gao, S., Yao, Y., Gu, R., and Tang, Y. Vfchain: Enabling verifiable and auditable federated learning via blockchain systems. *IEEE Transactions on Network Science and Engineering*, **9** (2022), 173. doi:10.1109/TNSE.2021.3050781.

[66] Pokhrel, S. R. and Choi, J. A decentralized federated learning approach for connected autonomous vehicles. In *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 1–6 (2020). doi:10.1109/WCNCW48565.2020.9124733.

[67] Pouriyeh, S., Shahid, O., Parizi, R. M., Sheng, Q. Z., Srivastava, G., Zhao, L., and Nasajpour, M. Secure Smart Communication Efficiency in Federated Learning: Achievements and Challenges. *Applied Sciences*, **12** (2022). Available from: https://www.mdpi.com/2076-3417/12/18/8980, doi:10.3390/app12188980.

[68] Pretre, B. Attacks on peer-to-peer networks. *Dept. of Computer Science Swiss Federal Institute of Technology (ETH) Zurich Autumn*, (2005).

[69] Qu, Y., Dai, H., Zhuang, Y., Chen, J., Dong, C., Wu, F., and Guo, S. Decentralized federated learning for uav networks: Architecture, challenges, and opportunities. *IEEE Network*, **35** (2021), 156. doi:10.1109/MNET.001.2100253.

[70] Ram, S., Nedic, A., and Veeravalli, V. Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of Optimization Theory and Applications*, **147** (2010), 516. doi:10.1007/s10957-010-9737-7.

[71] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents (2022). `arXiv:2204.06125`.

[72] Rodríguez-Barroso, N., Jiménez-López, D., Luzón, M. V., Herrera, F., and Martínez-Cámara, E. Survey on federated learning threats: Concepts, taxonomy on attacks and defences, experimental study and challenges. *Information Fusion*, **90** (2023), 148.

[73] Roussopoulos, M., Baker, M., Rosenthal, D. S. H., Giuli, T. J., Maniatis, P., and Mogul, J. C. 2 p2p or not 2 p2p? In *International Workshop on Peer-to-Peer Systems* (2003).

[74] Roy, A. G., Siddiqui, S., Pölsterl, S., Navab, N., and Wachinger, C. Braintorrent: A peer-to-peer environment for decentralized federated learning. *CoRR*, **abs/1905.06731** (2019). Available from: `http://arxiv.org/abs/1905.06731`, `arXiv:1905.06731`.

[75] Sattler, F., Müller, K.-R., and Samek, W. Clustered federated learning: Model-agnostic distributed multi-task optimization under privacy constraints (2019). `arXiv:1910.01991`.

[76] Seide, F., Fu, H., Droppo, J., Li, G., and Yu, D. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Interspeech* (2014).

[77] Shayan, M., Fung, C., Yoon, C. J. M., and Beschastnikh, I. Biscotti: A blockchain system for private and secure federated learning. *IEEE Transactions on Parallel and Distributed Systems*, **32** (2021), 1513. `doi:10.1109/TPDS.2020.3044223`.

[78] Shen, S., Tople, S., and Saxena, P. Auror: Defending against poisoning attacks in collaborative deep learning systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pp. 508–519 (2016).

[79] Shi, J., Wan, W., Hu, S., Lu, J., and Zhang, L. Y. Challenges and approaches for mitigating byzantine attacks in federated learning (2022). `arXiv:2112.14468`.

[80] Skype. `https://www.skype.com` (2023).

[81] Sun, Z., Kairouz, P., Suresh, A. T., and McMahan, H. B. Can you really backdoor federated learning? (2019). `arXiv:1911.07963`.

[82] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks (2014). `arXiv:1312.6199`.

[83] Tang, Z., Shi, S., Li, B., and Chu, X. Gossipfl: A decentralized federated learning framework with sparsified and adaptive communication. *IEEE Transactions on Parallel and Distributed Systems*, **34** (2023), 909. `doi:10.1109/TPDS.2022.3230938`.

[84] Tikkinen-Piri, C., Rohunen, A., and Markkula, J. Eu general data protection regulation: Changes and implications for personal data collecting companies. *Computer Law & Security Review*, **34** (2018), 134.

[85] Tolomei, G., Gabrielli, E., Belli, D., and Miori, V. A byzantine-resilient aggregation scheme for federated learning via matrix autoregression on client updates (2023). arXiv:2303.16668.

[86] Tolpegin, V., Truex, S., Gursoy, M. E., and Liu, L. Data poisoning attacks against federated learning systems (2020). arXiv:2007.08432.

[87] Vucovich, M., et al. Anomaly detection via federated learning (2022). arXiv:2210.06614.

[88] Wang, H., Sievert, S., Charles, Z., Liu, S., Wright, S., and Papailiopoulos, D. Atomo: Communication-efficient learning via atomic sparsification. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, p. 9872–9883. Curran Associates Inc., Red Hook, NY, USA (2018).

[89] Wang, L., Xu, S., Wang, X., and Zhu, Q. Eavesdrop the composition proportion of training labels in federated learning (2019). arXiv:1910.06044.

[90] Wang, X., Lalitha, A., Javidi, T., and Koushanfar, F. Peer-to-peer variational federated learning over arbitrary graphs. *IEEE Journal on Selected Areas in Information Theory*, **3** (2022), 172. doi:10.1109/JSAIT.2022.3189051.

[91] Wu, Q., He, K., and Chen, X. Personalized federated learning for intelligent iot applications: A cloud-edge based framework. *IEEE Open Journal of the Computer Society*, **1** (2020), 35. doi:10.1109/OJCS.2020.2993259.

[92] Xie, C., Koyejo, S., and Gupta, I. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation (2019). arXiv:1903.03936.

[93] Xu, X., Wu, J., Yang, M., Luo, T., Duan, X., Li, W., Wu, Y., and Wu, B. Information leakage by model weights on federated learning. In *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*, PPMLP'20, p. 31–36. Association for Computing Machinery, New York, NY, USA (2020). ISBN 9781450380881. Available from: https://doi.org/10.1145/3411501.3419423, doi:10.1145/3411501.3419423.

[94] Yang, Q., Liu, Y., Chen, T., and Tong, Y. Federated machine learning: Concept and applications. **10** (2019). Available from: https://doi.org/10.1145/3298981, doi:10.1145/3298981.

[95] Yin, D., Chen, Y., Kannan, R., and Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. In *Proceedings of the 35th International Conference on Machine Learning* (edited by J. Dy and A. Krause), vol. 80 of *Proceedings of Machine Learning Research*, pp. 5650–5659. PMLR (2018). Available from: https://proceedings.mlr.press/v80/yin18a.html.

[96] Yuan, X., Ma, X., Zhang, L., Fang, Y., and Wu, D. Beyond class-level privacy leakage: Breaking record-level privacy in federated learning. *IEEE Internet of Things Journal*, **9** (2022), 2555. Available from: `https://api.semanticscholar.org/CorpusID:236745405`.

[97] Zhang, H., Li, J., Kara, K., Alistarh, D., Liu, J., and Zhang, C. Zipml: Training linear models with end-to-end low precision, and a little bit of deep learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, p. 4035–4043. JMLR.org (2017).

[98] Zhang, J., Chen, B., Cheng, X., Binh, H. T. T., and Yu, S. Poisongan: Generative poisoning attacks against federated learning in edge computing systems. *IEEE Internet of Things Journal*, **8** (2021), 3310. Available from: `https://api.semanticscholar.org/CorpusID:226519574`.

[99] Zhang, J., Chen, J., Wu, D., Chen, B., and Yu, S. Poisoning attack in federated learning using generative adversarial nets. *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, (2019), 374. Available from: `https://api.semanticscholar.org/CorpusID:207830094`.

[100] Zhao, J., Zhu, H., Wang, F., Lu, R., Liu, Z., and Li, H. Pvd-fl: A privacy-preserving and verifiable decentralized federated learning framework. *IEEE Transactions on Information Forensics and Security*, **17** (2022), 2059. `doi:10.1109/TIFS.2022.3176191`.

[101] Zhou, A. C., Xiao, Y., Gong, Y., He, B., Zhai, J., and Mao, R. Privacy regulation aware process mapping in geo-distributed cloud data centers. *IEEE Trans. Parallel Distributed Syst.*, **30** (2019), 1872. Available from: `https://doi.org/10.1109/TPDS.2019.2896894`, `doi:10.1109/TPDS.2019.2896894`.

[102] Zhou, C., Fu, A., Yu, S., Yang, W., Wang, H., and Zhang, Y. Privacy-preserving federated learning in fog computing. *IEEE Internet of Things Journal*, **7** (2020), 10782. `doi:10.1109/JIOT.2020.2987958`.

[103] Zhu, L., Liu, Z., and Han, S. Deep leakage from gradients (2019). `arXiv:1906.08935`.