Byzantine Attacks and Defenses in Decentralized Learning Systems that Exchange Chunked Models

Robust Decentralized Learning

Atanas Donev
Supervisors: Bart Cox, Jérémie Decouchant
EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to the EEMCS Faculty of Delft University of Technology,
In Partial Fulfilment of the Requirements
For the **Bachelor of Computer Science and Engineering**
June 22, 2025

Name of the student: Atanas Donev
Final project course: CSE3000 Research Project
Thesis committee: Bart Cox, Jérémie Decouchant, Anna Lukina

Byzantine Attacks and Defenses in Decentralized Learning Systems that Exchange Chunked Models

Atanas Donev
Delft, Netherlands

***Abstract*—Decentralized learning (DL) enables collaborative model training in a distributed fashion without a central server, increasing resilience but still remaining vulnerable to the same adversarial model attacks, that Federated Learning is vulnerable to. In this paper, we test two defense mechanisms, specifically designed to protect fully decentralized learning, in the scenario of *chunked models* and introduce one new defense. We clearly define the threat model, where malicious peers either try to perform a backdoor attack or an untargeted label flipping. The defenses are Norm Clipping, Sentinel and we propose Adaptive Norm Clipping. We evaluate the effectiveness of these defenses on the CIFAR-10 dataset. Our results indicate that the model attacks (backdoor attack and untargeted label flipping) significantly harm model accuracy on vanilla DL with chunking. While static defenses like Norm Clipping and Adaptive Norm Clipping reduce the impact of the attack, they lower the final test average accuracy in the chunked scenario. Also, robust aggregators like Sentinel fail at mitigating the attack, but do not lower the average test accuracy.**

***Index Terms*—Decentralized Learning; Byzantine Attacks on Chunked Models; Chunked Models; Distributed Defense;**

I. Introduction

In an era of information and privacy, distributed machine learning enables collaborative model training without the need to have all the data collected in one place. In Federated Learning (FL), a set of nodes collaborate with a trusted central entity to train a model [1]. The nodes train a model locally on their private data and send only the updates to the central entity, which aggregates them into a global model [2]. The central server thus coordinates learning while never seeing individual data points. The same goes for nodes, who do not see other data points except their own. In decentralized learning, there is no central server. Instead, each node exchanges model parameters with its neighbors over a network topology and aggregates incoming updates to improve a shared model [2–4].

Several privacy attacks have been identified on Federated Learning systems [5]. To prevent these attacks in DL systems, the concept of model chunking has been introduced in several works [3, 6]. Under the model chunking paradigm a node sends only a subset (chunk) of his parameters (model) to each neighbor, ensuring no neighbor ever sees the whole model. While this approach may enhance privacy, it also remains weak to Byzantine attacks such as backdoor and label flipping attacks.

In centralized FL, many backdoor attacks have been studied [7–10], however, backdoors in DL remain largely unexplored [4, 11]. Existing defenses (like robust aggregators and anomaly detectors) generally assume a central aggregator or access to complete model updates [12, 13], and thus are not directly applicable to DL. Furthermore, in chunked setting these defenses never see the whole model update, which in most cases renders them not directly applicable. For example, robust aggregation rules such as Krum [14], Bulyan [15] and Catalyst [13] require collecting all client updates on a server [4], and monitoring full models for anomalies is difficult if nodes only see fragments of each model.

The specific research question we address in this work is the following: **How to defend against Byzantine attacks while exchanging chunked models in decentralized setting (e.g., via the Shatter[1] virtual-node framework or DecentralizePy[2])?**

As a summary, we make the following contributions:

First, we adapt two representative attacks to DL with chunked models exchanges, namely:

- Backdoor attack - injects a backdoor behaviour in the model
- Untargeted label flipping - flips labels, trying to sabotage performance

Then, we adapt two defenses that have been designed for the decentralized learning with full model echanges, and test whether they harm the performance of a non-attacked network and then test if they are also effective. We consider the following two defenses:

- Norm Clipping [4] - works by clipping the ℓ_2 norm of neighbors' updates to some predefined constant τ;
- Sentinel [16] - Sentinel is a three-phase aggregation protocol consisting of similarity filtering, bootstrap validation, and layer normalization.

Then, we modify Norm Clipping to store history of model updates and adapt its clipping factor. We call it Adaptive Norm Clipping.

Finally, we evaluate these attacks and defenses on the CIFAR10 dataset. Our results indicate that while Norm Clipping and Adaptive Norm Clipping are effective at defeding they lower the final average testing accuracy noticeably. Sentinel does not do that, but fails at defending completely.

II. Background

This section introduces the key technical pillars needed to understand our setting: *federated learning*, its *decentralized* (peer-to-peer) variant, the idea of *model chunking* with virtual nodes, and the *backdoor-poisoning threat model*.

[1] https://github.com/sacs-epfl/shatter
[2] https://github.com/sacs-epfl/decentralizepy

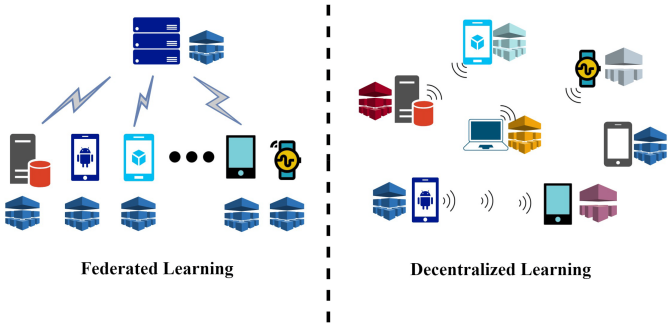

Fig. 1: Federated and Decentralized Learning

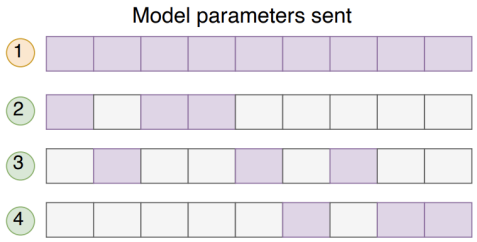

Fig. 2: Model Chunking: Neighbours (2, 3, 4) send parts of their model (in purple) to 1, who reconstitute a model that it then uses to update its local model

A. From Centralized to Decentralized Learning

Federated Learning: Federated Learning (FL) is a distributed machine learning paradigm that enables multiple clients (e.g., user devices or silos) to collaboratively train a global model without directly sharing their raw training data [17]. In FL, each client keeps its private dataset local and instead computes model updates (such as gradient updates or model weight deltas) on its own data. These local updates are then sent to a central server which aggregates them, for example, by averaging them as in the FedAvg algorithm, to produce an improved global model [18–20]. FL is illustrated in Figure 1. The key idea is that only model parameters or updates are exchanged, never the raw data, so that sensitive personal information remains on the source devices [17]. This process typically iterates over multiple rounds: the server distributes the latest global model to all clients, clients perform further local training on their data, and the clients' new updates are collected and averaged again [18]. Through this approach, FL addresses data privacy concerns and reduces the need to transmit large datasets, since the data never leaves the clients and only model updates are communicated. However, it also creates a single point of failure and a privacy bottleneck. Many attacks (e.g., model poisoning) and defenses (e.g., secure aggregation) are therefore studied under the assumption that a trusted coordinator exists [10].

Decentralized or peer-to-peer federated learning (DL/DFL/P2PFL). When a central aggregator is unavailable, untrusted, or simply too costly, learning can proceed in a fully distributed fashion: each participant maintains its own copy of the model, communicates only with graph neighbours, and acts both as a *learner* and a lightweight *parameter server* (cf. Figure 1). Canonical algorithms include D-PSGD [21], push-sum gossip learning [22], and numerous variants that improve communication or convergence speed [23–25]. Nodes iteratively mix models they receive with their own before taking a local SGD step, approximating the effect of a global average. This architecture improves fault tolerance and removes the server bottleneck, but each node now has only a *partial view* of the global state, complicating robustness and privacy [18, 26].

B. Model Chunking

One recent privacy-enhancing refinement of DL is **model chunking**, which we illustrate in Figure 2. Instead of transmitting the entire parameter vector, a node splits it into k *chunks*, and transmits each of them to one of their neighbors. Model chunking can have two variations: sequential and random. In sequential the parameter vector is split as if cut by a scissor. In random chunking, each time a parameter is selected from the parameter vector and added into a chunk, until the chunk has been completed. Throughout this paper we assume **full-sharing**. Full-sharing means that all parameters from the parameter vector are sent to at least one neighbor, ensuring all the learned information is shared and speeding up convergence. A high-level description of the algorithms is available in subsection IV-A.

C. Byzantine attacks

In this paper we primarily focus on two types of model attacks: targeted and untargeted model attacks. In the former the attacker's goal is to implant a *triggered misbehavior* without noticeably harming clean-task accuracy. More formally, the learned model f_θ should predict correctly on benign inputs x, but output an attacker-chosen label $y^\star$ whenever a secret pattern τ is present, i.e. $f_\theta(x \oplus \tau) = y^\star$ [27]. In untargeted attacks, malicious nodes flip labels to try and harm the global accuracy or send perturbed model updates, that reverse the training progress or generally disturb the performance. In decentralized settings the problem is exacerbated: there is no global "view" to compare updates against, and a small coalition can exploit network topology to disseminate poisoned parameters [4].

D. Why Defending in DL is Hard

Classical robust aggregators (e.g., Krum [14], trimmed mean[28]) require collecting *all* client updates in one place before filtering out the malicious ones [14, 28]. Without a server, no node sees such a complete batch, and the asynchronous, time-varying graph means different nodes may aggregate incompatible model sets. Moreover, techniques that rely on shared validation data or a public "clean" dataset are difficult to realize when each participant holds only private data. Consequently, DL requires lightweight, *local* defense

logic that operates with partial information, but designing such defenses without incurring high computational or communication cost remains open [11, 16]. Furthermore, with chunking this problem grows even harder, because no model ever sees the whole model update from their neighbor, making it harder for defenses to detect an anomaly within the updates or suppress it.

III. Related Work

Decentralized Federated Learning (DFL) [10, 23–25, 29, 30], otherwise known in literature as Peer-to-Peer or gossip Learning, is a decentralized version of federated learning, where there is no central coordinator that aggregates results. Each node sends its current model to selected peers and aggregates the models it receives, e.g., via gossip or consensus protocols [3, 31]. There are some variations of DFL that guarantee privacy, introducing extra layers of computation (Virtual Nodes) [3], but for now we will focus on pure decentralized learning with the small change of chunking [3].

Model Chunking in Decentralized Learning: Model chunking has emerged as a crucial technique in P2P learning, helping to reduce the burden of full-model exchanges and to provide privacy or fault tolerance. Shatter [3] is an example of such framework. Biswas et. al. [3] mentions two primary types of model chunking: sequential and random. In sequential chunking the model parameters are split into k equal chunks as if the parameters represent a line and a scissor is cutting it in pieces. In random chunking each time a random model parameter is chosen from the set of available parameters to place in a given chunk. In both cases Biswas et. al. [3] assumes full-sharing. Full-sharing means sending every parameter of the model to at least one neighbor. That way all of the progress is shared and convergence is sped up. In Federated Learning, Lebrun et al. propose MixNN [6], a framework that shows privacy can also be achieved by inter-client chunking of entire layers rather than fine-grained parameters. MixNN inserts a trusted proxy that randomly permutes whole layers among participants before the aggregation step. Since FedAvg is linear, this permutation leaves the aggregated update unchanged, so utility is unaffected [6]. While MixNN's layer mixing can be viewed as a form of random chunking across clients, Shatter complements it with intra-client chunking plus virtual nodes. Each real node spawns multiple virtual nodes (VNs), distributes different chunks through them, and thus prevents any single adversary from reconstructing the full model and from linking a chunk to its origin [3]. Biswas et. al. [3] claim that with 16 virtual nodes per participant Shatter simultaneously defeats linkability, membership-inference and gradient-inversion attacks while slightly improving convergence speed.

Byzantine model attacks: Byzantine model attacks have been extensively studied in the context of machine learning [8, 14, 27, 32–35]. These types of attacks have proven highly effective across many federated learning (FL) scenarios while remaining largely "invisible" in terms of main-task accuracy [7, 8, 36]. For example, Wang et al. [7] introduced "edge-case" backdoors, which force misclassifications on rare but semantically coherent inputs (living in the tail of the data distribution), and showed that such stealthy triggers can evade state-of-the-art defenses. Likewise, Mei et al. [36] note that a backdoor attack is "notoriously stealthy and threatening", since it changes model behavior only on specific inputs while leaving overall accuracy unchanged. Other detrimental attacks are Neurotoxins [37], Distributed Backdoor Attacks (DBA) [38] and stealthy model poisoning [39]. Recent studies extend these findings to non-traditional FL settings. Notably, Lyu et. al. [40] and Ye et. al. [41] demonstrate that personalized FL (PFL) is also vulnerable. Lyu et. al. [40] propose a PFedBA attack aligns the backdoor trigger with each client's local objective, achieving outstanding attack success across 10 state-of-the-art PFL algorithms and defeating six existing defenses.
In the context of decentralized federated learning backdoors have been investigated as well. Syros et al. [4] provide one of the first comprehensive studies of backdoors in DL. They show that even a small fraction of strategically-placed malicious nodes can succeed. In their experiments, compromising only 5% of nodes (selected by graph centrality) yields a high backdoor success rate while the model's clean accuracy remains undisturbed. Syros et al. [4] also confirm that classic FL defenses fail in DL.

Defenses: Several robust aggregation methods have been proposed to defend against backdoors in FL, but each has its own limitations, especially outside the centralized server model [14, 42]. BRIDGE adapts FL defenses like Coordinate Wise Trimmed Mean [28] and Krum [14] to decentralized learning, but they limit themselves to only IID data. Another example is FLAME [9]. It is effective in the standard server-coordinated FL setting (yielding backdoor-free global models with minimal accuracy loss [9]), but it crucially relies on a central aggregator that sees all client updates. Catalyst [13] recently adapted FLAME to the asynchronous learning setting but it still assumes a central server. Recent experiments in a gossip-based P2P FL architecture show that a small coalition of attackers can still achieve 100% backdoor success under trimmed-mean-style aggregation, and in fact the learning process converges more slowly with this "defense" than with no defense at all [4]. In other words, without a trusted server to perform the trimming and global aggregation, trimmed mean offers essentially no protection against backdoors [4].

FoolsGold [43] has been shown to work only under restrictive conditions: it assumes benign data are non-IID across clients and the attackers' data are IID [9]. Bulyan [15] is a hybrid robust aggregator that combines Krum [14] (selecting updates closest to each other) with coordinate trimming. It was designed to reduce high-dimensional attack strategies. However, empirical studies have found that Bulyan can even underperform Krum, because its aggressive filtering sometimes discards honest updates [16]. As with other Byzantine-robust rules, Bulyan requires collecting a pool of client updates at each round and applying a multi-step filter – an operation that intrinsically assumes a central server. In a decentralized FL (gossip) scenario, nodes do not have a synchronized view of all updates, so neither Krum nor Bulyan can be directly

executed.

Recently, Sentinel [16] was proposed specifically for DL. Other hybrid approaches have been suggested: for example, Cao et al. [44] propose that nodes compute trust scores based on neighbors' behavioral consistency and share these scores to form a global trust metric. Similarly to Sentinel, Fang et. al. propose to disregard neighbors' models if they deviate too much from the local one [45]. Another defense developed for fully decentralized learning is ByRDiE [46]. ByRDiE assumes a fully-connected and static topology. This does not realistically reflect typical decentralized learning networks, which tend to be sparsely connected graphs. Also, Belal et. al. [47] proposed a different view of the problem. They noted that in most cases byzantine participants can not only manipulate local updates, but also cheat the peer sampler. Therefore, Belal et. al. [47] proposed GRANITE, which combines History-aware Peer Sampling (HaPS) and Adaptive Probabilistic Threshold (APT). It bridges the gap between secure peer sampling and robust aggregation, offering formal analysis and strong empirical proof.

IV. System and Threat Models

A. System Model

We consider a fully decentralized learning system with n nodes connected in a fixed, sparse communication graph. The network topology is a 3-regular graph $G = (V, E)$, meaning each node has exactly 3 neighbors (peers) in the overlay network. We denote by $N(i)$ the set of neighbors of node i in G (so $|N(i)| = 3$ for all i). There is no central server or aggregator, and instead, nodes communicate directly with their neighbors.

Local Models and Data Distribution: Each node i maintains a local copy of the model parameters $w_i \in \mathbb{R}^d$ (for a model of dimension d) and has a private local dataset D_i. We assume that the local datasets can be either independent and identically distributed (IID) or non independent and identically distributed (non-IID).

Model Chunking and Message Passing: To enhance privacy and reduce communication overhead, nodes do not exchange their entire parameter vector in each communication round. Instead, each node partitions its model parameters into k disjoint chunks and transmits only one chunk to each neighbor. Formally, let $k = |N(i)|$ (the degree of node i, which is 3 in our 3-regular graph). We partition the index set of model parameters $1, 2, \ldots, d$ into k subsets $I_{i\to j}^{(t)}$ (for each neighbor $j \in N(i)$) at each round t. These index sets represent the chunk of parameters that node i will send to neighbor j in round t. In general, the chunks can be constructed in different ways (e.g. sequential chunking, where the parameter vector is split into contiguous segments, or random chunking, where indices are randomly assigned to chunks). In our scenario we assume full-sharing of the model (as per Biswas et al. [3]): every parameter index belongs to at least one chunk, so no parameter is completely withheld. A simple implementation is to assign each parameter index to exactly one neighbor's chunk (a disjoint partition), ensuring that the union of chunks covers the entire model: $\bigcup_{j\in N(i)} I_{i\to j} = 1, \ldots, d$ and $I_{i\to j} \cap I_{i\to \ell} = \emptyset$ for $j \neq \ell$. This means that each node shares one-third of its parameters with each of its 3 neighbors in each round. Other implementations could allow overlapping chunks or dynamically changing random chunks each round, but for clarity we describe the fixed disjoint chunk approach.

At every communication round $t = 1, 2, \ldots, T$, each node i performs the following steps in parallel:

1) Local Update: Node i performs local training on its current model $w_i^{(t-1)}$ using its private data D_i. This can be one or several steps of stochastic gradient descent (SGD) or Adaptive Moment Estimation (Adam). Let $\tilde{w}_i^{(t)} = \text{LocalUpdate}(w_i^{(t-1)}; D_i)$ denote the updated local model after this computation (we use $\tilde{w}_i^{(t)}$ to distinguish the post-local-training model from the final aggregated model $w_i^{(t)}$ after communication).
2) Chunk Partitioning: Node i splits $\tilde{w}_i^{(t)}$ into $k = 3$ parameter chunks. Let $I_{i\to j}^{(t)}$ be the index subset allocated to neighbor j. Node i extracts the corresponding parameters $\tilde{w}_i^{(t)}[I_{i\to j}^{(t)}]$ (the entries of $\tilde{w}_i^{(t)}$ with indices in $I_{i\to j}^{(t)}$).
3) Communication (Send): Node i sends the chunk $\tilde{w}_i^{(t)}[I_{i\to j}^{(t)}]$ to each neighbor $j \in N(i)$. (If a fixed partition is used, $I_{i\to j}^{(t)}$ may not depend on t; for random chunking, $I_{i\to j}^{(t)}$ could be freshly sampled each round. In either case, the size of each chunk is approximately $d/3$ parameters.)
4) Communication (Receive): Simultaneously, node i receives a chunk of parameters from each neighbor $j \in N(i)$. Let $I_{j\to i}^{(t)}$ be the index subset that neighbor j sends to i in round t, and let $\tilde{w}_j^{(t)}[I_{j\to i}^{(t)}]$ denote the values of j's updated model on those indices.
5) Aggregation: Node i integrates the received neighbor updates with its own update. For each parameter index $k \in 1, \ldots, d$, node i updates that coordinate of its model by averaging its own value with any received values for the same coordinate . Formally, for each k in $1, \ldots, d$:
$$w_i^{(t)}[k] = \frac{\tilde{w}_i^{(t)}[k] + \sum_{\{j\in N(i):\ k\in I_{j\to i}^{(t)}\}} \tilde{w}_j^{(t)}[k]}{1 + \left|\{j \in N(i) : k \in I_{j\to i}^{(t)}\}\right|}.$$

In other words, node i takes the average of its own parameter value and all neighbor values for that same parameter (if any). If the chunking scheme guarantees that each index k is provided by at most one neighbor (e.g. disjoint index chunks from neighbors), then the above formula simplifies to averaging between two values whenever a neighbor covers index k, or keeping the local value if no neighbor sent that index.

After this aggregation, node i sets $w_i^{(t)}$ as its model for the next round. All nodes proceed to round $t+1$ unless $t = T$, in which case the training completes.

B. Threat Model

Adversary Population: We assume a fixed fraction of the nodes is Byzantine. For example, in our experiments approximately 18.75% of the nodes are adversarial. These adversarial

nodes remain malicious throughout the whole training process: no additional nodes become compromised after initialization, and no compromised node reverts to honest behavior once it is corrupted.

Capabilities: Each malicious (Byzantine) node has complete control over its own local training pipeline. The adversary can perform the following actions:

1) Data Poisoning: The adversary can manipulate its private training dataset arbitrarily by inserting, deleting, or relabeling examples.
2) Update Manipulation: In each round and for each neighbor, the adversary can craft arbitrary parameter chunks. These chunks maintain the expected shape but may deviate arbitrarily from the honest gradient. The adversary may even send different poisoned chunks to different neighbors in the same round.
3) Computation Changes: A malicious node may execute additional local training iterations. By doing so, it can more effectively learn the backdoor trigger pattern and increase the attack's success.

Limitations: Attackers cannot modify the overlay graph, intercept communications between honest nodes, forge messages on behalf of other nodes, or break cryptographic primitives (e.g., message integrity checks). These restrictions confine the adversary to local manipulations within its own node.

Knowledge and Coordination: Each adversarial node has only local visibility. It knows its own data, the update history received from its direct neighbors, and its incident edges in the overlay graph. It does not know the full network topology or the internal states of non-neighboring nodes. Nevertheless, we assume that in the case of backdoor attack all compromised nodes share a common trigger and a common target label l_t. In particular, they agree on a patch trigger τ_{mask} and a target label y_t for the backdoor.

Goal: There are two types of goals for the adversaries. One is to implant a backdoor in the learned model. Specifically, it aims to ensure that any input x containing the trigger τ_{mask} is classified as the target label y by the final converged model of every honest node. At the same time, the overall test accuracy on clean (benign) inputs should remain essentially unchanged. We measure the success of the attack by an attack success rate (ASR), which is detailed in subsection V-C. The goal of the second attack is to sabotage overall performance by flipping labels randomly.

Why This Model Is Challenging: Exchanging model parameters in chunks conceals most of each model from individual participants. As a result, each honest node receives at most $\frac{1}{k}$ of any neighbor's parameters per round. This fragmentation reduces statistical redundancy and prevents straightforward global consistency checks. Traditional defenses— which rely on inspecting full gradient vectors or comparing updates across clients—become ineffective under these conditions. The attacker exploits this limited observability (and the absence of a trusted aggregator) by distributing small, carefully crafted perturbations across many chunks and rounds. Over time, these perturbations accumulate so that the backdoor dominates the global decision boundary, all without producing obvious anomalies that could be easily detected.

V. Methodology

A. Network and Framework

All experiments are conducted on the above 16-node, 3-regular peer-to-peer network topology. We implemented the decentralized learning algorithm with chunked model exchange using the DecentralizePy framework (an open-source library for decentralized learning) augmented with custom code to support model chunking [3]. In DecentralizePy's architecture, each node runs as an independent process executing the communication protocol and local training. We configured DecentralizePy such that each node splits its model into a number of chunks equal to its number of neighbors (in our case, 3 chunks per node). The chunking can be configured as sequential or random; in our implementation, we ensured disjoint chunks per neighbor and full parameter sharing, as described in the System Model. All machine-learning code (model definition, training, etc.) was implemented in PyTorch[4] (we integrated PyTorch models into DecentralizePy's Python environment).

Each node's behavior follows the aforementioned algorithm (subsection IV-A) during training rounds. The network topology remains static throughout all experiments (no peers are added or removed, and the neighbor relationships do not change). We emphasize that there is no central coordinator: model update averaging occurs locally at each node using only neighbor communications.

B. Datasets and Models

We evaluate our system on the CIFAR-10 image classification dataset, which has 10 classes of natural images. The global dataset is split among the 16 nodes either in an IID manner or a non-IID manner, to test both homogeneous and heterogeneous data distributions:

- IID Partitioning: The 50,000 training images of CIFAR-10 are shuffled and evenly divided into 16 shards of equal size (3125 images per node). Each node thus receives a random sample of the entire dataset, approximating an independent and identically distributed data split. This ensures that all nodes have a similar class distribution.
- Non-IID Partitioning: We simulate a more realistic heterogeneous scenario using a Dirichlet distribution-based partitioning. Following common FL practice, we use a Dirichlet(α) allocation to create skewed data shards [48]. In our experiments we choose $\alpha = 1.0$, which produces moderately non-IID splits: each node's dataset D_i is biased towards a subset of CIFAR-10 classes, though not completely single-class.

The model architecture used for all experiments is LeNet-5, a classic convolutional neural network (CNN). LeNet-5 consists of two convolutional layers (with pooling) followed

[3] Code based on DecentralizePy
[4] https://pytorch.org/

by two fully-connected layers, and was originally designed for digit classification. We adapt LeNet-5 to CIFAR-10 by using 3-channel input and 10 output classes. Despite its simplicity, LeNet-5 can achieve reasonable accuracy on CIFAR-10 with proper training, and its small size (around 60k parameters) makes it suitable for extensive experimental runs. All nodes initialize their local model $w_i^{(0)}$ to the same random initial weights before training begins (ensuring a common start point for learning).

We fix a training schedule of 300 communication rounds (iterations of the decentralized protocol). In each round, nodes perform local SGD on their data. We set the local computation such that each node completes 1 local epoch (pass through its local dataset) per round, which in our setup corresponds to 30 training steps per round (since with batch size B, number of batches ≈ 30 covers the node's data; effectively we train 30 minibatches locally each round). We use a fixed learning rate and no global learning rate decay for simplicity, since our focus is on comparative robustness with and without defenses (all models, baseline and defended, see the same training schedule).

During training, 3 out of the 16 nodes (18.75%) are designated as Byzantine (malicious). These adversarial nodes are chosen arbitrarily at the start and remain malicious throughout the training (no dynamic compromise or recovery). The honest 13 nodes are not aware of which peers are malicious and treat all incoming messages normally.

C. Backdoor Attack Suite

Backdoor attack: In this attack, adversarial nodes aim to implant a specific backdoor trigger into the global model without significantly affecting normal accuracy. Each malicious node alters a fraction of its training data by inserting a small trigger pattern into the images and changing their labels to a predetermined target label y_t. Specifically, a certain percentage $p\%$ (poisoning rate) of the node's local examples are modified: for each such example, a fixed trigger pattern τ_{mask} is stamped onto the image, and the image's label is flipped to y_t (the attacker-chosen target class). The malicious node then proceeds with local training on this poisoned dataset, which causes its model update $\tilde{w}_i^{(t)}$ to learn the backdoor. The attack success rate is:

$$ASR = \frac{\sum_{j=0}^{|L|} c_{j,t} - c_{t,t}}{|B| - c_{t,t}}$$

Untargeted label flipping: In this attack, the adversaries poison their data in a non-targeted way: by randomly flipping labels of training examples to incorrect classes. Each malicious node takes some portion of its local dataset and assigns random wrong labels to those examples (without using a specific trigger pattern in the input). The goal of untargeted poisoning is to degrade the overall model performance (lower the global accuracy) rather than to induce a particular misclassification. Malicious nodes still train on their poisoned data and send chunks of the resulting model updates.

D. Defense Suite

Norm Clipping: This defense, proposed by Syros et. al. [4], has each honest node limit the magnitude (Euclidean norm) of incoming neighbor updates. The idea is to prevent any single chunk from a neighbor from having an outsized influence due to extremely large values (which might indicate a poisoning attempt). In our implementation, before averaging a neighbor's chunk into the model, node i checks the ℓ_2 norm of that chunk $\tilde{w}_j^{(t)}[I_{j\to i}^{(t)}]$ (the vector of parameter differences sent by neighbor j). If $\|\tilde{w}_j^{(t)}[I_{j\to i}^{(t)}]\|_2$ exceeds a certain threshold τ, the chunk is scaled down to norm τ. Formally, node i replaces $\tilde{w}_j^{(t)}[I_{j\to i}^{(t)}]$ by

$$\tilde{w}_j^{(t)}[I_{j\to i}^{(t)}] \;=\; \tilde{w}_j^{(t)}[I_{j\to i}^{(t)}] \cdot \frac{\tau}{\max(\|\tilde{w}_j^{(t)}[I_{j\to i}^{(t)}]\|_2,\; \tau)}$$

, so that $\|\tilde{w}_j^{(t)}[I_{j\to i}^{(t)}]\|_2 \le \tau$. In our experiments we set $\tau_{\text{own}} = 1.0$ for the norm of an honest node's own model update (we never scale down its own update, assuming honest training remains within this norm), and $\tau_{\text{nbr}} = 0.1$ for the norm of each incoming neighbor update. Norm clipping is computationally cheap and distributed (each node does it independently). By capping update magnitudes, it can limit the damage from arbitrary poisoned chunks.

Adaptive Norm Clipping: This is a modified version of norm clipping that dynamically adjusts the clipping threshold based on recent history of neighbor behavior. The motivation is that a static τ_{nbr} might be too conservative if a neighbor usually provides updates similar in norm to the local node's, or too lenient if a clever attacker slowly increases norms.

- Max Norm Factor: 0.2,
- Min Norm Factor: 0.01,
- History Window Size: 20

Sentinel: Sentinel (proposed by Feng et al. [16]) is a more complex three-stage filtering mechanism originally designed for fully decentralized learning. We adapted Sentinel to the chunked-update scenario. Each honest node, at each round, will: (i) compute the cosine similarity between each neighbor's incoming chunk and the node's own update, and prune any neighbor chunks that are too dissimilar (below a cosine similarity threshold) – the idea is that honest updates should roughly agree in direction, whereas an outlier could be malicious; (ii) perform a bootstrap weighting of the remaining updates based on their loss contributions (giving less weight to updates that increase a validation loss, for example); and (iii) apply a scale normalization so that the weighted neighbor updates have bounded influence. Sentinel combines these steps to reject anomalous updates before aggregation. In our implementation, we set the Sentinel parameters according to the suggestions in the original paper (for instance, a certain cosine similarity cutoff, etc.). However, it is important to note that applying Sentinel to chunked models is challenging – since each node only sees a fragment of each neighbor's update, the cosine similarity is computed on a partial vector rather than the full model, which may reduce its effectiveness. We include Sentinel mainly as a reference to state-of-the-art:

each node runs the Sentinel filter on the three incoming chunks each round and then aggregates whatever remains.

VI. Performance Evaluation

All of the experiments were conducted in both IID and non-IID setting. Testing on non-IID data ensures representability of attacks and defenses, since in most real-world scenarios a participant does not have the same distribution of data as their neighbors. The partitioning function used was *dirichlet* with an alpha parameter ($\alpha = 1$) allowing for uniform selection over all possible classes.

A. Baselines

We consider the following baselines (Table I and Table II):

- Vanilla Chunked DL (CDL) with no attack and no defense employed - This experiment showcases the basis on which we judge, whether we have succeeded or not.
- Baseline Model under Backdoor Attack (BA) - we demonstrate stealthiness and efficiency of the attack on chunked models.
- Baseline Model under Untargeted Label Flipping attack (UF) - we the efficiency of the attack on chunked models.
- Baseline Model with Sentinel (SL) - we show the effect of running Sentinel on chunked models.
- Baseline Model with Norm Clipping (NC) - we show the effect of running Norm Clipping on chunked models.
- Baseline Model with Adaptive Norm Clipping (ANC) - we show the effect of running Adaptive Norm Clipping on chunked models.

B. Defenses vs Attacks

Now we demonstrate the defending capabilities of our defenses against every attacks. Each defense is tested against each attack to showcase their defending power(Table I and Table II):

- Sentinel under Backdoor Attack (SL-BA)
- Norm Clipping under Backdoor Attack (NC-BA)
- Adaptive Norm Clipping under Backdoor Attack (ANC-BA)
- Sentinel under Backdoor Attack (SL-UF)
- Norm Clipping under Backdoor Attack (NC-UF)
- Adaptive Norm Clipping under Backdoor Attack (ANC-UF)

VII. Discussion

A. Baselines

In the IID scenario, the baseline model (with no attacks or defenses) converged to a high accuracy on the CIFAR-10 task (see CDL in Table I). In the non-IID scenario, the model likewise reached a strong accuracy, albeit slightly lower due to the heterogeneous data distribution (again see CDL in Table II). These baseline results confirm that, under ideal conditions, the fully decentralized chunked-model approach can learn effectively. However, when attacks are introduced, significant vulnerabilities emerge. Without any defenses, the backdoor attack proved highly effective and stealthy in both

Algorithm	Accuracy	ASR
ANC	57.93±0.02%	7.01±0.11%
ANC-BA	57.92±0.02%	6.70±0.51%
ANC-UF	55.68±0.21%	N/A
CDL	73.04±0.10%	7.98±0.04%
BA	72.71±0.23%	85.07±0.13%
NC	57.33±0.07%	7.01±0.48%
NC-BA	57.14±0.25%	20.83±2.06%
NC-UF	54.30±0.21%	N/A
SL	73.17±0.08%	N/A
SL-BA	72.14±0.28%	85.48±0.06%
SL-UF	9.94±0.00%	N/A
UF	9.91±0.06%	N/A

TABLE I: IID Aggregated Results from 3 experimental run(s) (Mean ± Standard Deviation)

IID and non-IID settings (see BA in Table I and Table II): the attack success rate (ASR) at convergence was extremely high (exceeding 80%), while the model's accuracy on clean (non-triggered) inputs remained near the baseline. In contrast, the untargeted label-flip attack caused a sharp drop in overall test accuracy (see UF in Table I and Table II), illustrating the model's fragility when faced with random label noise. Unlike the backdoor, the label-flip attack visibly degraded the model's normal accuracy, underscoring the vulnerability of the chunked decentralized training to even blatant poisoning.

It is also important to note the effect of defenses on the baseline performance. In no-attack scenario, Norm Clipping and Adaptive Norm Clipping introduced a modest reduction in accuracy despite the absence of attackers (see NC and ANC in Table I and Table II). This indicates that while these defenses strengthen robustness, they do so at the expense of some model performance in attack-free settings. Opposed to that Sentinel (see SL in Table I and Table II) did not induce such cost in the no-attack scenario.

B. Defenses vs. Attacks

Both *Norm Clipping* and *Adaptive Norm Clipping* significantly mitigated the impact of attacks in our decentralized learning experiments. Although these defenses reduced the final clean accuracy, they were highly effective in curbing the backdoor attack's success. With either clipping defense in place, the backdoor's ASR dropped from near 85% (in the undefended case) to a much lower value (see NC-BA and ANC-BA in Table I and Table II). For example, Norm Clipping limited the ASR to 20% in the IID setting, and kept it similarly low in the non-IID setting (with the non-IID case showing only a marginally higher ASR than IID). Adaptive Norm Clipping performed better than the static Norm Clipping—by dynamically adjusting its per-neighbor threshold. It managed to lower the ASR to only 6.7% in the IID scenario and 17.8% in the non-IID scenario.

In addition to countering backdoors, the clipping defenses also helped withstand label-flipping attacks (see ANC-UF and

Algorithm	Accuracy	ASR
ANC	44.12±0.08%	8.22±0.10%
ANC-BA	40.94±0.38%	17.80±0.09%
ANC-UF	41.42±0.40%	N/A
CDL	63.13±1.04%	7.94±0.18%
BA	62.66±0.03%	85.06±0.60%
NC	43.75±0.06%	7.04±0.65%
NC-BA	41.96±0.09%	23.12±1.35%
NC-UF	41.92±0.10%	N/A
SL	63.74±0.27%	N/A
SL-BA	61.94±0.84%	85.94±1.44%
SL-UF	9.91±0.06%	N/A
UF	9.87±0.06%	N/A

TABLE II: NIID Aggregated Results from 3 experimental run(s) (Mean ± Standard Deviation)

NC-UF in Table I and Table II). Under the untargeted label-flip attack, an honest model without defenses suffered a severe accuracy collapse, whereas with Norm Clipping or its adaptive variant, the global model maintained much higher accuracy. The defenses capped the influence of malicious updates, so the random label noise from adversaries did not entirely derail training. Overall, Norm Clipping and Adaptive Norm Clipping proved robust across all tested attack types, in both IID and non-IID settings, demonstrating their effectiveness at protecting chunked model exchange against both subtle backdoors and noisy label poisoning.

By contrast, the *Sentinel* defense was largely ineffective in our fully decentralized chunked-model setting. Deploying Sentinel did not meaningfully reduce the backdoor ASR (see SL-BA in Table I and Table II): the backdoor remained as successful as in the no-defense case, indicating that Sentinel failed to detect or mitigate the attack. A similar lack of improvement was observed against the label-flipping attack (see SL-UF in Table I and Table II) — Sentinel provided little to no benefit in preventing accuracy degradation. The likely reason for this failure lies in Sentinel's design, which relies on comparing full model updates or computing cosine similarities across complete models. In a chunked update scenario, each node only receives a small fraction of each neighbor's model parameters, severely limiting the information available for anomaly detection. Sentinel's mechanism cannot aggregate a holistic view of neighbors' updates under these conditions, and thus it cannot effectively flag malicious deviations. This result highlights a crucial insight: defense techniques that assume access to entire model updates (as many federated learning defenses do) may not transfer to fully decentralized, partially-observed (chunked) environments. Some static defenses like Norm Clipping, which clip the updates of neighbors to a given percentage of the current ℓ_2 norm succeed in mitigating the attacks, but degrade the overall test performance noticeably. New or adapted defenses are required to handle the challenges introduced by model chunking.

C. Future work

Although experiments were comprehensive in this study, our computational power was severely limited. As a potential improvement we suggest changing the following things:

- Increasing the local steps per epoch - Currently, we only perform 1 local steps per epoch for 30 epochs on each training round. Increasing the training steps may increase convergence and improve baseline models. However, increasing training steps will likely increase the adversaries powers to inject backdoor behavior. The effects on benign nodes and malicious nodes need to be further studied.
- Adding more nodes - Currently, we are using 16 nodes. A more realistic scenario might include a bigger number up to 100 nodes (sparsely connected to reflect realistic situations).
- Varying the number of adversarial nodes - our assumptions is that the percentage of adversarial nodes is quite high, in order to stress test the system. In reality it may be lower, since the nodes may be better protected.
- Changing the model - right now because of computational constraints, we have used the LeNet5 model, which is known to achieve acceptable accuracy on CIFAR10 for quite short time frame. However, there are more powerful models for the classification task of CIFAR10 like ResNet models, ViT-H/14, ViT-L/16, MobileNet models, etc.

VIII. Conclusion

This thesis investigated the resilience of fully decentralized learning systems exchanging chunked models against Byzantine attacks, specifically backdoor and label-flipping attacks. Experiments conducted on a 16-node decentralized network demonstrated significant vulnerabilities, highlighting that adversarial nodes could implant effective backdoors or substantially degrade model performance through label poisoning. Norm Clipping and its adaptive variant emerged as highly effective defenses, significantly reducing the attack success rates, however they incurred a modest final test accuracy degradation. Conversely, Sentinel proved ineffective in this context, emphasizing the challenge of applying centralized federated learning defenses to partially observed decentralized updates. There is a need for further research into potential defense classes for chunked DFL, that are not based on robust aggregators (like Sentinel)and do incur utility cost (like Norm Clipping and Adaptive Norm Clipping). Limitations include the restricted model complexity, small network size, fixed adversarial proportion, and moderately non-IID data distribution. Future research should explore larger-scale and more dynamic networks, more powerful and complex models, varying adversarial scenarios, adaptive attacker-defender interactions, and defenses explicitly tailored to chunked decentralized settings. Overall, this work establishes foundational insights into the robustness of decentralized learning with chunked model exchange and outlines critical directions for advancing security in peer-to-peer machine learning.

IX. Responsible Research

This research was carried out with a careful eye on ethical standards and proper procedures. It follows the guidelines mentioned in the Netherlands Code of Conduct for Research Integrity[49]. Public, open-access dataset (CIFAR-10) were used to promote accessibility and reproducibility of results. These datasets are established standards for assessing machine learning performance and are free of personal and sensitive data and hence meet privacy and data handling responsibilities norm.

This study explores adversarial attacks comprehensively, both targeted label-flipping and backdoor attacks. Throughout the process, our aim was to advance decentralized learning systems' security and robustness. We recognize that harmful exploitation is a possibility. The methods and conclusions drawn from this research may be applied with ill intent to sabotage or backdoor a decentralized system. For this reason, it is imperative that future researchers and practitioners are themselves responsible custodians of these methods and the code.

This study's source code and detailed documentation are securely retained in an access-restricted private GitLab repository. Access for interested scholars and researchers needs to be requested through contacting the authors and/or TU Delft. The in-depth documentation of the code encourages openness, reproducibility and facilitates future researchers' ability to extend our research responsibly. In terms of environmental effects, the current research indicates an absence of obvious ecological harm. Systematic computational analyses were conducted on controlled and optimized computer architectures according to established standards for efficient resource utilization and handling.

Large Language Models (ChatGPT, Claude 4 Sonnet and Gemini) were used solely for grammar checking, generating documentation and minimally assisting in implementations. All optimizations of code and text was written without the help of LLM.

References

[1] M. de Goede, B. Cox, and J. Decouchant. "Training diffusion models with federated learning". In: *arXiv preprint arXiv:2406.12575* (2024).

[2] I. Hegedűs, G. Danner, and M. Jelasity. "Decentralized learning works: An empirical comparison of gossip learning and federated learning". In: *Journal of Parallel and Distributed Computing* 148 (2021), pp. 109–124.

[3] S. Biswas et al. "Noiseless Privacy-Preserving decentralized learning". In: *Proceedings on Privacy Enhancing Technologies* 2025.1 (Nov. 2024), pp. 824–844.

[4] G. Syros, G. Yar, S. Boboila, C. Nita-Rotaru, and A. Oprea. "Backdoor Attacks in Peer-to-Peer Federated Learning". In: *ACM Transactions on Privacy and Security* 26.4 (2023), Article 57.

[5] J. Xu, C. Hong, J. Huang, L. Y. Chen, and J. Decouchant. "Agic: Approximate gradient inversion attack on federated learning". In: *2022 41st International Symposium on Reliable Distributed Systems (SRDS)*. IEEE. 2022, pp. 12–22.

[6] T. Lebrun, A. Boutet, J. Aalmoes, and A. Baud. "MixNN: protection of federated learning against inference attacks by mixing neural network layers". In: *Proceedings of the 23rd ACM/IFIP International Middleware Conference*. Middleware '22. ACM, Nov. 2022, pp. 135–147.

[7] H. Wang et al. "Attack of the Tails: Yes, You Really Can Backdoor Federated Learning". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 16070–16084.

[8] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov. "How To Backdoor Federated Learning". In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2020.

[9] T. D. Nguyen et al. "FLAME: Taming Backdoors in Federated Learning". In: *Proceedings of the USENIX Security Symposium*. 2022, pp. 1415–1432.

[10] P. Liu, X. Xu, and W. Wang. "Threats, attacks and defenses to federated learning: issues, taxonomy and perspectives". In: *Cybersecurity* 5.1 (Feb. 2022).

[11] T. Huang, S. Hu, F. Ilhan, S. F. Tekin, W. Wei, and L. Liu. *Silencer: Pruning-aware Backdoor Defense for Decentralized Federated Learning*. 2024.

[12] R. Wang et al. "MUDGUARD: Taming Malicious Majorities in Federated Learning using Privacy-Preserving Byzantine-Robust Clustering". In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 8.3 (2024), pp. 1–41.

[13] B. Cox, A. Mălan, L. Y. Chen, and J. Decouchant. "Asynchronous byzantine federated learning". In: *arXiv preprint arXiv:2406.01438* (2024).

[14] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer. "Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[15] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault. *The Hidden Vulnerability of Distributed Learning in Byzantium*. 2018. arXiv: 1802.07927 [stat.ML].

[16] C. Feng et al. *Sentinel: An Aggregation Function to Secure Decentralized Federated Learning*. 2024. arXiv: 2310.08097 [cs.DC].

[17] Q. Li, W. Yu, Y. Xia, and J. Pang. *From Centralized to Decentralized Federated Learning: Theoretical Insights, Privacy Preservation, and Robustness Challenges*. 2025. arXiv: 2503.07505 [cs.LG].

[18] E. Hallaji, R. Razavi-Far, M. Saif, B. Wang, and Q. Yang. "Decentralized Federated Learning: A Survey on Security and Privacy". In: *IEEE Transactions on Big Data* 10.2 (Apr. 2024), pp. 194–213.

[19] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Vol. 54. Proceedings of Machine Learning Research. PMLR, 20–22 Apr 2017, pp. 1273–1282.

[20] P. Kairouz et al. *Advances and Open Problems in Federated Learning*. 2021. arXiv: 1912.04977 [cs.LG].

[21] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu. *Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent*. 2017. arXiv: 1705.09056 [math.OC].

[22] R. Ormándi, I. Hegedűs, and M. Jelasity. "Gossip learning with linear models on fully distributed data". In: *Concurrency and Computation: Practice and Experience* 25.4 (May 2012), pp. 556–571.

[23] T. Sun, D. Li, and B. Wang. "Decentralized Federated Averaging". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.4 (2023), pp. 4289–4301.

[24] A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar. *Fully Decentralized Federated Learning*. arXiv preprint or unpublished manuscript. 2020.

[25] Y. Shi et al. *Improving the Model Consistency of Decentralized Federated Learning*. 2023. arXiv: 2302.04083 [cs.LG].

[26] D. Pasquini, M. Raynal, and C. Troncoso. "On the (In)security of Peer-to-Peer Decentralized Machine Learning". In: *2023 IEEE Symposium on Security and Privacy (SP)*. 2023, pp. 418–436.

[27] T. Gu, B. Dolan-Gavitt, and S. Garg. *BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain*. 2019. arXiv: 1708.06733 [cs.CR].

[28] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett. "Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates". In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2021.

[29] R. Dai, L. Shen, F. He, X. Tian, and D. Tao. *DisPFL: Towards Communication-Efficient Personalized Federated Learning via Decentralized Sparse Training*. 2022. arXiv: 2206 . 00187 [cs.LG].

[30] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger. *BrainTorrent: A Peer-to-Peer Environment for Decentralized Federated Learning*. 2019. arXiv: 1905.06731 [cs.LG].

[31] M. Fang, X. Cao, J. Jia, and N. Gong. "Local Model Poisoning Attacks to Byzantine-Robust Federated Learning". In: *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 1605–1622.

[32] R. Guerraoui, N. Gupta, and R. Pinot. "Byzantine Machine Learning: A Primer". In: *ACM Comput. Surv.* 56.7 (Apr. 2024).

[33] M. Baruch, G. Baruch, and Y. Goldberg. *A Little Is Enough: Circumventing Defenses For Distributed Learning*. 2019. arXiv: 1902.06156 [cs.LG].

[34] Z. Zhang, M. Fang, J. Huang, and Y. Liu. *Poisoning Attacks on Federated Learning-based Wireless Traffic Prediction*. 2025. arXiv: 2404.14389 [cs.NI].

[35] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu. *Data Poisoning Attacks Against Federated Learning Systems*. 2020. arXiv: 2007.08432 [cs.LG].

[36] H. Mei, G. Li, J. Wu, and L. Zheng. *Privacy Inference-Empowered Stealthy Backdoor Attack on Federated Learning under Non-IID Scenarios*. 2023. arXiv: 2306.08011 [cs.LG].

[37] Z. Zhang et al. "Neurotoxin: Durable Backdoors in Federated Learning". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 17–23 Jul 2022, pp. 26429–26446.

[38] C. Xie, K. Huang, P.-Y. Chen, and B. Li. "DBA: Distributed Backdoor Attacks against Federated Learning". In: *International Conference on Learning Representations*. 2020.

[39] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo. *Analyzing Federated Learning through an Adversarial Lens*. 2019. arXiv: 1811.12470 [cs.LG].

[40] X. Lyu et al. *Lurking in the shadows: Unveiling Stealthy Backdoor Attacks against Personalized Federated Learning*. 2024. arXiv: 2406.06207 [cs.LG].

[41] T. Ye, C. Chen, Y. Wang, X. Li, and M. Gao. "BapFL: You can Backdoor Personalized Federated Learning". In: *ACM Trans. Knowl. Discov. Data* 18.7 (June 2024).

[42] C. Fang, Z. Yang, and W. U. Bajwa. "BRIDGE: Byzantine-Resilient Decentralized Gradient Descent". In: *IEEE Transactions on Signal and Information Processing over Networks* 8 (2022), pp. 610–626.

[43] C. Fung, C. J. M. Yoon, and I. Beschastnikh. *Mitigating Sybils in Federated Learning Poisoning*. 2020. arXiv: 1808.04866 [cs.LG].

[44] X. Cao, M. Fang, J. Liu, and N. Z. Gong. *FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping*. 2022. arXiv: 2012.13995 [cs.CR].

[45] M. Fang et al. *Byzantine-Robust Decentralized Federated Learning*. 2024. arXiv: 2406.10416 [cs.CR].

[46] Z. Yang and W. U. Bajwa. "ByRDiE: Byzantine-Resilient Distributed Coordinate Descent for Decentralized Learning". In: *IEEE Transactions on Signal and Information Processing over Networks* 5.4 (Dec. 2019), pp. 611–627.

[47] Y. Belal, M. Maouche, S. B. Mokhtar, and A. Simonet-Boulogne. *GRANITE : a Byzantine-Resilient Dynamic Gossip Learning Framework*. 2025. arXiv: 2504.17471 [cs.LG].

[48] B. Cox, J. Galjaard, A. Shankar, J. Decouchant, and L. Y. Chen. "Parameterizing Federated Continual Learning for Reproducible Research". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2023, pp. 478–486.

[49] Netherlands Code Committee. *Netherlands Code of Conduct for Research Integrity*. Published by the Association of Universities in the Netherlands (VSNU). 2018.

APPENDIX

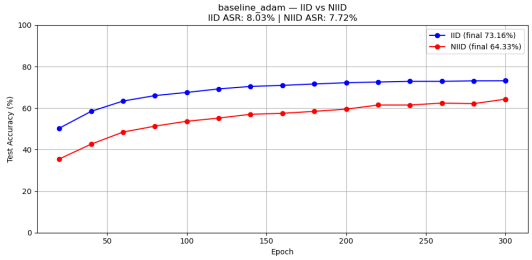

Fig. 3: Baseline (IID vs. non-IID)

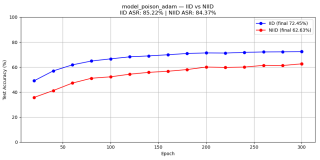

Fig. 4: Backdoor attack

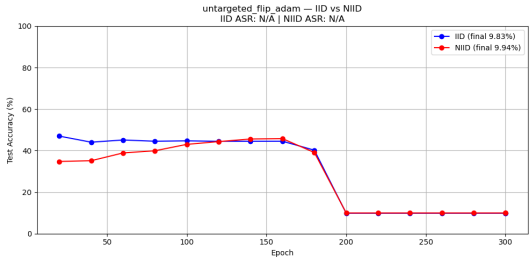

Fig. 5: Untargeted label flipping

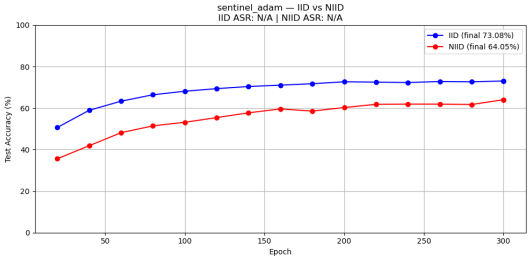

Fig. 6: Sentinel defense

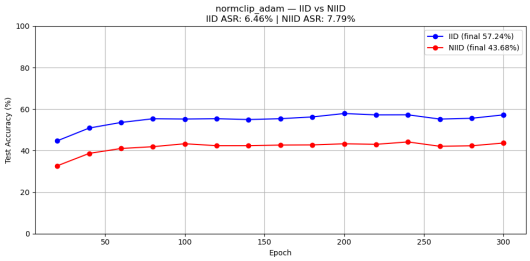

Fig. 7: Norm clipping defense

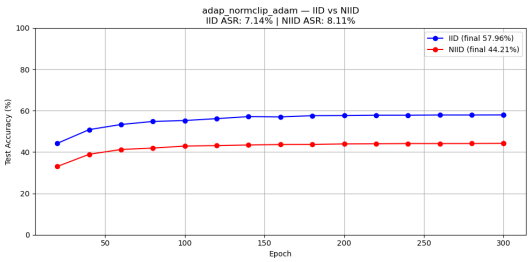

Fig. 8: Adaptive norm clipping defense

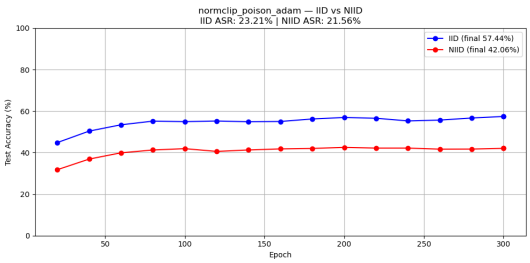

Fig. 9: Norm clipping under backdoor attack

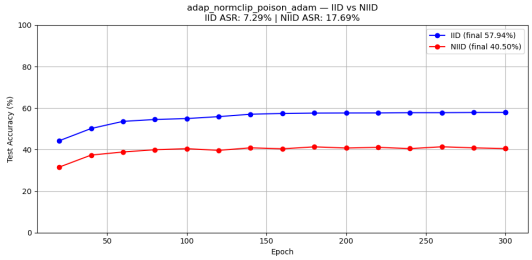

Fig. 10: Adaptive norm clipping under backdoor attack

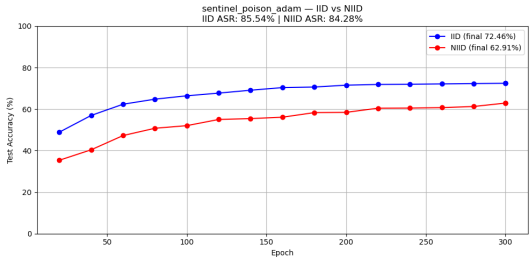

Fig. 11: Sentinel under backdoor attack

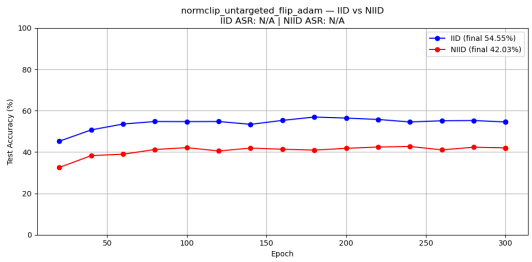

Fig. 12: Norm clipping under untargeted flip

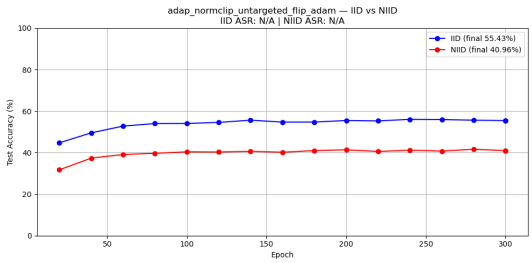

Fig. 13: Adaptive norm clipping under untargeted flip

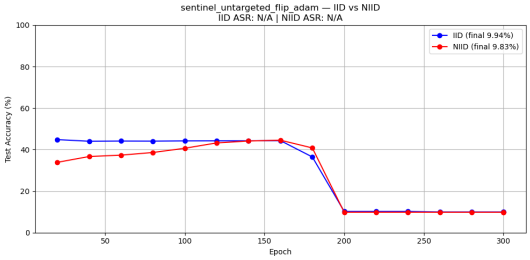

Fig. 14: Sentinel under untargeted flip