Towards Sybil Resilience in Decentralized Learning

Thomas Werthenbach

Student number:	4772466
Thesis committee:	Dr. ir. J.A. Pouwelse (supervisor)
	Dr. D.M.J. Tax

4 July 2023

To obtain the degree of Master of Science in Computer Science
Software Technology Track
To be defended publicly on July 4, 2023

Delft University of Technology
Faculty of Electrical Engineering, Mathematics & Computer Science
Distributed Systems Group

Towards Sybil Resilience in Decentralized Learning

— MSc. Thesis —

Thomas Werthenbach
Delft University of Technology
Delft, The Netherlands
T.A.K.Werthenbach@student.tudelft.nl

Johan Pouwelse
Delft University of Technology
Delft, The Netherlands
J.A.Pouwelse@tudelft.nl

***Abstract*—Decentralized learning has recently been emerging as a promising alternative to federated learning, a privacy-enforcing machine learning technology. The scalability of federated learning is limited by the internet connection and memory capacity of the central parameter server and the complexity of the aggregation function. Decentralized learning eliminates the need for a central parameter server by decentralizing the aggregation process across all participating nodes. Numerous studies have been conducted on improving the resilience of federated learning against poisoning and Sybil attacks, whereas the resilience of decentralized learning remains largely unstudied. This research gap serves as the main motivator for this study, in which our objective is to improve the resilience of decentralized learning.**

We present SybilWall, an innovative algorithm focused on increasing the resilience of decentralized learning against targeted Sybil poisoning attacks. By combining a Sybil-resistant aggregation function based on similarity between Sybils, with a novel probabilistic gossiping mechanism, we establish a new benchmark for scalable, Sybil-resilient decentralized learning.

Through comprehensive empirical evaluation, we found that SybilWall outperforms existing state-of-the-art solutions designed for federated learning scenarios and is the only algorithm to obtain consistent accuracy over a range of adversarial strategies. We also found SybilWall to discourage adversaries from creating many Sybils, as our evaluations demonstrate a higher success rate among adversaries employing fewer Sybils. Finally, we suggest a number of possible improvements to SybilWall and highlight promising future research directions.

***Index Terms*—Decentralized applications, Adversarial machine learning, Federated learning, Decentralized learning, Sybil attack, Poisoning attack**

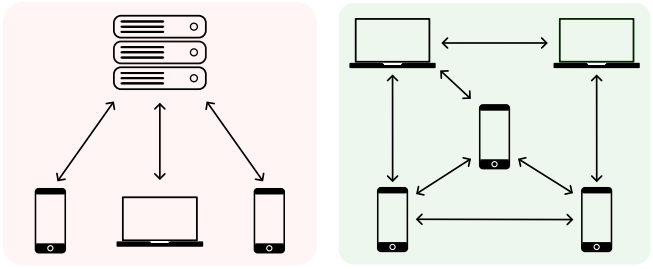
I. Introduction

The rise of machine learning has resulted in an increasing number of everyday-life intelligent applications. As such, machine learning has been used in personal assistants [1], recommendations on social media [2] and music [3], and cybersecurity [4]. However, accurate machine learning models require large training datasets [5], [6], which can often be difficult to obtain and store due to recent privacy legislation [7]. Federated learning [8] has become a promising option for distributed machine learning, having been proposed for the training of numerous industrial machine learning models [9]–[13]. Federated learning ensures the protection of privacy, as the user's data will not leave their device during training.

With federated learning, in contrast to centralized machine learning, training takes place on end-users' personal devices, which are often referred to as *edge devices* or *nodes*. The resulting trained models are communicated to a central server, commonly referred to as the *parameter server*, which aggregates these models using some predefined methodology. By only sharing the end-user-trained models with the parameter server, the user's privacy is preserved, while obtaining comparable performance compared to centralized machine learning [14]. Although there exist attacks in which training data can be reconstructed based on the gradient of trained models [15], [16], defense mechanisms against this attack have been proposed [17], [18].

However, federated learning suffers from some disadvantages. For example, the parameter server aggregates the models of all participating nodes, inducing high communication costs and a potential bottleneck in the learning process, affecting the overall convergence time [19]. Second, the scalability of the chosen aggregation function in terms of the number of nodes may vary greatly. In robust and secure federated learning aggregation methods, the incorporation of additional nodes during aggregation can result in a significantly increased computational effort for the parameter server [20]. Third, the parameter server performing the aggregation poses a single point of failure [21], [22]. Disruptions to the parameter server can cause downtime and hinder the overall model training process, particularly in architectures where nodes require the globally aggregated model before proceeding the training. An upcoming alternative that aims to resolve these issues is *decentralized learning* [23]–[26], also commonly referred to as *decentralized federated learning*. In decentralized learning, there exists no dedicated parameter server performing the aggregation, and the nodes form a distributed network, e.g., a peer-to-peer network, in which each node individually performs aggregation using their neighbors' models (Figure 1). Although the information available during aggregation is more limited relative to federated learning, decentralized learning has been shown to have the potential to obtain similar results compared to federated learning [27]. Models are exchanged between individual nodes and aggregated on a smaller scale using some predefined aggregation method, alleviating the communicative bottleneck and single point of failure issues imposed on federated learning, and paving the path for boundless scalability.

Although decentralized learning solves the scalability challenges faced in federated learning, it is still vulnerable to



(a) Federated learning (b) Decentralized learning

Figure 1: Examplary network topologies in federated learning and decentralized learning.

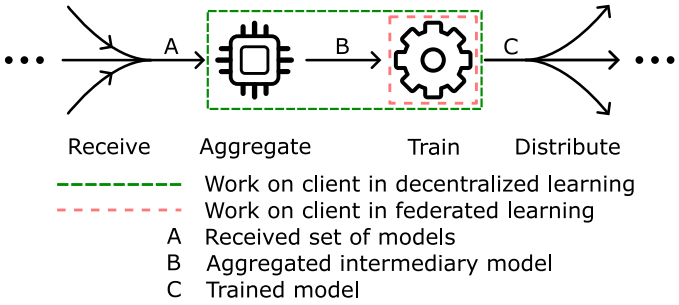


Figure 2: The general train-aggregate loop executed by all nodes participating in decentralized learning, highlighting the difference in work performed by nodes in federated learning and decentralized learning.

Byzantine environments [22]. Since the predefined aggregation method in decentralized learning does not have access to all models in the network, aggregation is performed with less information compared to federated learning, resulting in relatively lower resistance against possible poisoning attacks [28]. Poisoning attacks can generally be classified in two categories, namely those of *targeted poisoning attacks* and *untargeted poisoning attacks*. Targeted poisoning attacks focus on a specific goal that an adversary aims to achieve, such as the label-flipping attack [29], [30] and the backdoor attack [31]–[33]. On the other hand, untargeted poisoning attacks aim to hinder the result of the training process in some way without any particular goal in mind. The effect of these attacks can often be amplified by combining them with the Sybil attack [34], in which an adversary creates a substantial number of virtual nodes to increase its influence. As such, an adversary may deploy the Sybil attack to rapidly spread their poisoned model through the network. In this work, we focus exclusively on targeted poisoning attacks amplified by Sybil attacks in decentralized learning.

Prior work on resilience against Sybil poisoning attacks in distributed machine learning has mainly been done in federated learning settings. One popular example of such work is *FoolsGold* [35], which aims to increase Sybil resilience under the assumption that all Sybils will broadcast similar gradients during each round of training. By dynamically adapting the aggregation weights of peers' models based on their similarity with others, experimental results suggest that FoolsGold has the potential to provide effective protection against Sybil attacks in small-scale federated learning settings.

In this work, we experimentally demonstrate *FoolsGold*'s inability to scale to an unbounded number of nodes in federated learning and inept defensive capabilities against targeted poisoning attacks in decentralized learning.

We suggest an improved version of FoolsGold, named SybilWall, which shows significant resilience towards defending against targeted poisoning attacks while enjoying the boundless scalability offered by decentralized learning. More specifically, we achieve this by introducing a probabilistic gossiping mechanism for data dissemination. We empirically evaluated SybilWall on numerous types of Sybil attacks and showed its ability to obtain increased Sybil resilience.

To the best of our knowledge, this work is the first to propose a defensive algorithm against poisoning attacks amplified by the Sybil attack in decentralized learning. In short, our contributions are the following:

- We reproduce FoolsGold's results in federated learning and demonstrate its unpredictable performance in decentralized learning in Section III.
- We define the Spread Sybil Poisoning attack for effective Sybil poisoning attacks in decentralized learning and decompose it into three distinct adversarial scenarios.
- We present SybilWall, a pioneering algorithm for Sybil poisoning resilience with boundless scalability in decentralized learning, in Section V.
- We performed an empirical evaluation of the performance of SybilWall in VI on various datasets and against competitive alternatives.

II. Background

Federated learning was initially proposed by Google [8] as a means of training machine learning models on real user data without compromising user privacy. However, federated learning is associated with limitations in scalability. Decentralized learning is a promising alternative as it resolves scalability limitations through decentralization. Both distributed machine learning technologies are prone to poisoning attacks, of which the effects can be amplified by employing the Sybil attack. In this work, we consider two of these attacks: the label-flipping attack and the backdoor attack.

A. Federated learning

Federated learning achieves privacy-enforcing distributed machine learning by training all machine learning models on the edge devices (nodes) of the participants, which contain real user data. Training proceeds in synchronous rounds, each consisting of a predefined number of epochs, during which the trained models are sent to a central parameter server at the end of each round. The role of the parameter server is to aggregate all trained models into a global model without the need of training data. After aggregation, the parameter server

communicates the global model to all nodes, immediately followed by the start of the next training round. Figure 1a illustrates a simplified federated learning network topology. The original federated learning paper [8] suggests the usage of *FedAvg*, which adopts a weighted average function as the aggregation function, such that the next global model w^{t+1} is calculated as follows:

$$w^{t+1} = \sum_{i \in N} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} w_i^t \tag{1}$$

where w_i^t is the model of node i in round t, N is the set of nodes, $\mathcal{D}_i$ corresponds to node i's local dataset and $\mathcal{D}$ is the global distributed collection of data, such that $\mathcal{D} = \bigcup_{j \in N} \mathcal{D}_j$.

The goal of the training process is to minimize the global loss function such that the global model x approaches the optimal model x^*. More formally, the search for a global optimal model can approximately be defined as:

$$w^* = \arg\min_w \sum_{i \in N} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} \mathcal{L}_i(w) \tag{2}$$

where $\mathcal{L}_i$ is a node's loss function, e.g. cross-entropy loss or negative log likelihood loss, using the node i's local dataset.

In federated learning, all participating nodes are only connected to the parameter server, such that the network graph $\mathcal{G}$ is defined as a tuple of nodes and undirected edges $\langle N, E \rangle$, for which there exists a one-to-one mapping $N \rightarrow E$, such that for parameter server s, $\forall n \in N, \langle n, s \rangle \in E$.

B. Decentralized learning

Decentralized learning is an upcoming alternative to federated learning [23]–[26]. In contrast to federated learning, which relies on a parameter server to aggregate locally trained models, aggregation in decentralized learning takes place on a smaller scale and is performed by every participating node on their own model and those of its neighbors (Figures 1b and 2). By doing so, decentralized learning does not suffer from the scalability limitations encountered in federated learning. These improvements in scalability can be decomposed into three distinct aspects:

1) *Communication costs*: In federated learning, all models are downloaded and uploaded by the parameter server every training round, forming a communication bottleneck bounded by the parameter server's internet connection. Such bottlenecks are reduced in decentralized learning depending on a node's number of neighbors.
2) *Memory*: Storing all models in memory during aggregation may result in substantial memory usage. In decentralized learning, aggregation likely coincides with a significantly reduced number of models compared to federated learning, thus diminishing memory-related limitations.
3) *Aggregation time*: The time complexity of the more sophisticated aggregation functions may not scale linearly with respect to the number of participating nodes. Due to the decentralization of the aggregation, the number of models in each aggregation is greatly reduced.

In contrast to federated learning, the aggregation function of decentralized learning is generally applied on a node's own model and that of its neighbors. For example, a basic average-based aggregation function in decentralized learning can be defined as the following function:

$$w_i^{t+1} = \frac{1}{|\mathcal{N}_i| + 1} \sum_{j \in \{\mathcal{N}_i \cup i\}} w_j^t \tag{3}$$

where $\mathcal{N}_i$ is the set of neighbours of node i. Note the lack of the local datasets' size ratio as a weight factor in the averaging function, caused by the challenging task of obtaining a trustworthy measure of the size of any node's dataset without sharing its local dataset. This complexity of this task is further increased by to the anonymity of participating nodes and the lack of a central authority capable of verifying one's identity.

Furthermore, nodes may not have access to all information in the network, causing a decrease in informativeness. More specifically, the convergence rate may be impaired compared to federated learning [26] and nodes may experience increased vulnerability to Byzantine attacks due to the lack of global information [22]. However, decentralized learning has been shown to obtain comparable performance results compared to federated learning [27] and may sporadically outperform federated learning altogether [36], [37].

The network graph $\mathcal{G}$ in decentralized learning, in contrast to federated learning, does not contain a parameter server. Nodes are also generally not limited to a subset of the set of nodes N to form a connection, allowing for more diverse network topologies.

C. Targeted poisoning attacks

A *poisoning attack* is a type of Byzantine attack, which encapsulates all methods by which an adversary may attempt to compromise the integrity of the global model in decentralized learning and federated learning. Poisoning attacks typically include *data poisoning* or *model poisoning*. Data poisoning involves malicious alteration of the training data on which a model is trained, while model poisoning entails adapting the process in which the model is trained to produce a malicious model [28]. In this work, we only consider data poisoning. Furthermore, poisoning attacks can be classified into two categories: *targeted poisoning attacks* and *untargeted poisoning attacks*. With untargeted poisoning attacks, the adversary aims to decrease the performance metric of the global model without any particular goal in mind. On the other hand, targeted poisoning attacks are employed to achieve a specific goal by manipulating the global model to behave in a deterministic manner that deviates from objectively correct behavior. In this paper, we exclusively consider two types of targeted poisoning attacks: the label-flipping attack [29], [30] and the backdoor attack [31]–[33].

The label-flipping attack can be deployed as an attempt to increase the probability of two targeted classes being misclassified. More specifically, given two target classes t_1 and t_2, the aim of the label-flipping attack is to manipulate

the model such that an arbitrary sample $x \in X_{t_1}$ belonging to class t_1 is more likely to be classified as class t_2 by the global model and vice versa. One logical way of achieving this is to explicitly transform the adversary's local dataset $\mathcal{D}$ into an adversarial dataset $\mathcal{D}'$ and train the adversarial model on this dataset. Given two target classes t_1 and t_2, this transformation can be defined as:

$$\begin{aligned}\mathcal{D}' = &\{(x,y) \in \mathcal{D} \mid y \neq t_1 \wedge y \neq t_2\} \\ &\cup \{(x,t_1) \mid (x,y) \in \mathcal{D},\ y = t_2\} \\ &\cup \{(x,t_2) \mid (x,y) \in \mathcal{D},\ y = t_1\}\end{aligned} \tag{4}$$

The backdoor attack requires a more sophisticated manipulation of the training data. The objective of a backdoor attack is to alter the global model such that any sample containing a specific predefined pattern is misclassified to a chosen target class. In the domain of image classification, this adversarial pattern could, for example, correspond to a small square or triangle in the top left corner of the input image [38]. Given a target class t and a function f that introduces a hidden pattern to input samples, the transformation applied on the adversary's local dataset $\mathcal{D}$ can be defined as:

$$\mathcal{D}' = \{(f(x), t) \mid (x,y) \in \mathcal{D}\} \tag{5}$$

D. The Sybil attack

The Sybil attack, first introduced by Douceur [34], is an adversarial strategy in decentralized environments in which the attacker exploits the anonymity of nodes, caused the inability to verify the authenticity of any node's identity. Through the effortless creation of fake nodes, named *Sybils*, and strategical edges to honest nodes in the targeted decentralized network, the attacker may gain significantly more influence compared to honest nodes. We denote the edges between Sybils and honest nodes as *attack edges*. A typical example of a scenario in which the Sybil attack may be deployed is *majority voting* [39], [40]. In such a case, an attacker can trivially generate sufficient nodes to outnumber all honest voters.

Methods for mitigating the Sybil attack through an admission control system to the decentralized network have been proposed [41]–[43], but are often not frictionless or are based on an invite-only system. Adoption of such systems may take place at a slower rate due to its decreased accessibility and usability [44], especially considering that decentralized learning can be deployed as a background task [12], highlighting the importance of frictionless admission.

A network graph on which a Sybil attack is deployed can be defined as $\mathcal{G} = \langle N', E' \rangle$, such that $N' = N \cup \mathcal{S}$, where $\mathcal{S}$ is the unbounded set of Sybils created by the adversary. Note that Sybils and honest nodes are indistinguishable from the typical point of view. The modified set of edges E' is defined as $E' = E \cup E_{\mathcal{S}}$, where $E_{\mathcal{S}}$ is the set of *attack edges*, which is highly dependent on the strategy of the adversary. Note that attack edges always consist of at least one Sybil, such that $\forall \langle i,j \rangle \in E_{\mathcal{S}}, i \in \mathcal{S} \vee j \in \mathcal{S}$.

In this work, we consider the targeted Sybil poisoning attack, in which an adversary aims to amplify the effects of a targeted poisoning attack by creating Sybils. These Sybils help spread the adversary's malicious model more rapidly and effectively throughout the network.

III. Related work

Numerous studies have been conducted in order to improve poisoning resilience in a form of distributed machine learning. This section provides an overview of three defense mechanisms suggested in prior work.

A. FoolsGold

FoolsGold [35] is an algorithm designed to mitigate Sybil poisoning attacks in federated learning settings. It builds on the assumption that Sybil model gradients show a substantially higher degree of similarity relative to that of honest model gradients. Through the computation of similarity between a node's gradient history and that of others, and subsequently transforming this to the gradient's weight in an average-based aggregation, FoolsGold successfully manages to mitigate targeted Sybil poisoning attacks.

During aggregation, FoolsGold first computes the pairwise cosine similarity score for all gradient histories. The gradient history of node i in round T is defined as $h_i^T = \sum_{t=0}^{T} g_i^t$, where g_i^t are the gradients of a model obtained by training the model on node i in round t. Given that Sybil gradient histories show a high degree of similarity, the number of false positives can be reduced by multiplying each similarity score s_{ij} by the ratio of the maximum score of node i' and the maximum score of node j' in the cases where the latter is greater, such that s_{ij} is multiplied by $\frac{\max_v s_{iv}}{\max_v s_{jv}}$ if $\max_v s_{iv} < \max_v s_{jv}$.

Subsequently, the scores are aggregated for each node by taking the inverse of the maximum, such that node i's aggregated score s'_i can be defined as $s'_i = 1 - \max_v s_{iv}$. After which the aggregated scores are rescaled such that the maximum aggregated score equals 1, as FoolsGold assumes the existence of at least one honest node. Each node now has a weight which is close to zero if its gradient history shows high similarity to another node's gradient history and vice versa, the weight is close to 1 if a node's gradient history shows little similarity to any other node's gradient history.

The aggregated scores are then amplified and transformed through the use of a bounded logit function. This function can be considered a gradual decision boundary for determining a node's benevolence based on its similarity with others. Finally, the weights are normalized and the aggregated model is computed by a weighted average.

A reproduction of FoolsGold's results can be found in Figure 3, where the attack score represents the extent to which the attack was successful, e.g., the percentage of labels that are successfully flipped in the label-flipping attack. It becomes clear that FoolsGold shows significantly higher Sybil resilience compared to FedAvg. However, as discussed in Section II-B, federated learning can be considered unscalable as the number of participating nodes increases, particularly in

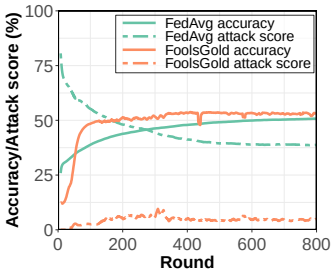


Figure 3: FoolsGold and FedAvg in federated learning setting using the CIFAR-10 [46] dataset on a LeNet-5 [45] model.

view of the $\mathcal{O}(n^2)$ pairwise cosine similarity computation and the memory capacity required to store these models. Figure 4 demonstrates the $\mathcal{O}(n^2)$ time complexity of the pairwise cosine similarity computation on the LeNet-5 model [45]. We further note that the generation of this figure was restricted to a maximum memory usage of 16 GB, thereby highlighting another limitation of a pairwise comparison-based aggregation function. Furthermore, Figure 5 shows the performance of FoolsGold in a decentralized setting against the performance of our improved solution, SybilWall, based on FoolsGold's intuitions. When comparing both Figures 5a and 5b it becomes clear that FoolsGold's performance heavily depends on the network topology, while SybilWall demonstrates relatively higher and more consistent Sybil resilience.

B. Krum

Krum [47] attempts to improve the general Byzantine resilience in distributed machine learning. This approach operates on the assumption that Byzantine model gradients are prone to deviate from the gradients produced by honest nodes. More specifically, the aggregation involves computing a score $s(w)$ for every received model w, which corresponds to the sum of the squared distances between i and its $n - f - 2$ nearest neighbours, where f corresponds to the maximum number of Byzantine nodes Krum is designed to protect against. Finally, the model m with the lowest score, such that $m = \arg\min_w s(w)$, is chosen as the next global model.

C. Resilient Averaging Gradient Descent

Resilient Averaging Gradient Descent (RAGD) [48] uses distance-based intuitions similar to Krum, but was specifically

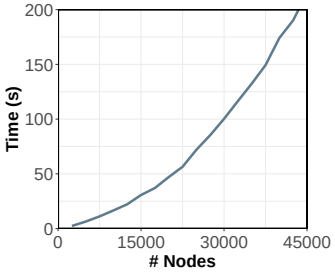
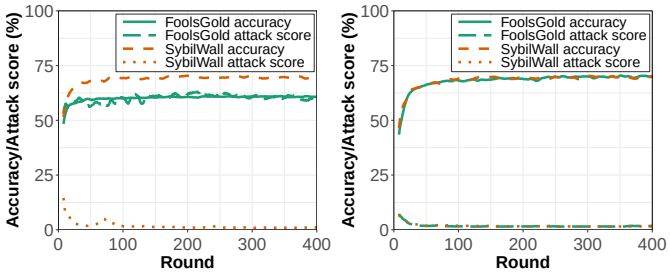


Figure 4: Pairwise cosine similarity computation time against the number of nodes (LeNet-5 [45]).



(a) Network topology 1 (b) Network topology 2

Figure 5: FoolsGold and SybilWall in decentralized learning using the FashionMNIST [49] dataset on a single-layer softmax neural network.

designed for decentralized learning. By introducing additional assumptions, it guarantees convergence of an approximately optimal model in the presence of poisoning attacks. Firstly, it assumes that all nodes are honest and that only their local datasets might be compromised, thereby still participating in aggregation benevolently, but training malicious models. Second, RAGD assumes the existence of a weighted global adjacency matrix, corresponding to the network graph. The weights in this adjacency matrix are considered *trust values* and correspond to the influence that nodes have during aggregation. Third, RAGD assumes that the edge weights from some node i to some attacked neighboring node j is limited by a predefined global constant ϵ, such that $0 < \epsilon < \frac{1}{2}$, $a_{ij} < \epsilon$, where a_{ij} corresponds to the edge weight assigned by node i to its edge with attacked node j.

A typical round of training in RAGD can be decomposed into a number of steps. 1) Nodes attempt to reach a global consensus on the aggregated model through repeatedly broadcasting and averaging models aggregated by neighboring nodes, weighted by the corresponding edge weight. 2) Every node trains the aggregated model and broadcasts the gradients. Note that malicious models may be produced by the attacked nodes during this step. 3) While some value g, which is initialized to 1, remains larger than $1 - \epsilon$, RAGD selects two of the received gradients, such that the Euclidean distance between the two selected gradients g_i and g_j is maximized, and eliminates the gradient that has the largest sum of distances to all other gradients. The edge weight corresponding to the node that produced the eliminated gradient is subtracted from the value g. When $g \leq 1 - \epsilon$, every node computes the weighted average of the remaining gradients. 4) Finally, the weighted average of the remaining gradients is applied to the (pre-training) aggregated model and the next round begins.

IV. Threat Model and Preliminaries

This section provides an overview of the assumptions and threat model used throughout this work.

A. Adversarial assumptions

Assumption 1. *The adversary can only communicate with other nodes through the default decentralized learning API.*

As the adversary can only communicate with other nodes through the default decentralized learning API, it does not possess the ability to manipulate other nodes' local models or data. We also assume that the decentralized learning API enforces homogeneous model broadcasting. That is, when some node i broadcasts its model to its neighbors at the end of every training round, all of i's neighbors receive the same model. In practice, this can be enforced by adopting existing algorithms [50]. Lastly, we assume that the default decentralized learning API adopts the use of signatures to prevent spoofing.

Assumption 2. *All used cryptographic primitives are secure.*

The signatures used by the decentralized learning API, as well as any other cryptographic primitives employed throughout this work, are assumed to be secure.

Assumption 3. *The adversary is unrestricted in both the quantity of Sybil nodes it can create and the selection of honest nodes it can form attack edges to.*

Assumption 4. *Sybil models show high similarity compared to honest models.*

Given the context of targeted poisoning attacks, Sybils are created by an adversary to achieve a specific goal during decentralized learning. As these Sybils share their training dataset, their trained models will likely show a high similarity.

In contrast to prior work [35], we assume a high similarity between the trained models of Sybils, rather than the model gradients, i.e. the difference between the aggregated intermediary model and the trained model. Due to the lack of knowledge of the aggregated intermediary model between the aggregation and training phases (Figure 2), no node can ascertain the model gradients of another node in decentralized learning.

Assumption 5. *The creation of Sybils by the adversary does not increase its adversarial computing capabilities.*

Following Assumption 4 and the lack of knowledge of the aggregated intermediary model, we must assume that each Sybil utilizes the same aggregated intermediary model. This assumption is enforced through Assumption 5, that is, the adversary does not have sufficient adversarial computing capabilities to execute the train-aggregate loop for each Sybil each round.

B. Network restrictions

Assumption 6. $\exists\ e \in \mathbb{N}$ *such that* $d_i \leq e$, $\forall i \in N$, *where* d_i *represents the degree of node* i.

We restrict the impact that any individual node may exercise on the network, by assuming existence of an upper bound on the degree of any node. Such bounds may arise naturally due to internet connection speeds, but may also be detected through existing algorithms. For example, a network latency-based avoidance mechanism [51] can be used to discover multiple edges of a node. Another alternative is to perform a random walk or a breadth-first search, which are known to be biased toward high-degree nodes [52].

Assumption 7. *Every node has at least one honest neighbour.*

This final assumption is inherited from prior work [35], as the cosine similarity function requires a baseline for *honest* work for measuring relative similarity. This might be achieved through an invite-only network with accountability [42]. We note that Eclipse attacks [53] are out of the scope of this work.

C. Adversarial strategy

We define an intuitive and effective type of worst-case attack in similarity-based aggregation techniques in decentralized learning as *Spread Sybil Poisoning Attacks* (SSP attacks). That is, the adversary aims to avoid detection by maximizing the distance between its attack edges while increasing the influence of the attack by minimizing the distance between any honest node and the nearest attack edge. The latter part of this problem resembles the *Maximal Covering Location Problem* [54], which is known to be an NP-Hard problem [55]. To determine the attack edge positions for SSP attacks, we propose a heuristic approach using the unsupervised clustering algorithm K-medoids [56], assigning attack edges to the medoids.

Furthermore, we define a parameter for SSP attacks, ϕ, which represents the average density of attack edges per node. Note that the attack edges are as spread out as possible, such that $\forall a_i, a_j \in \mathcal{A}, |a_i - a_j| \leq 1$, where $\mathcal{A}$ represents the set of the number of attack edges per node. For any value of ϕ, each honest node receives $\lfloor\phi\rfloor$ or $\lceil\phi\rceil$ attack edges. Therefore, the total number of attack edges is denoted as $\lceil|N| \cdot \phi\rceil$. The remainder, defined by $\phi \bmod 1$, is distributed according to the K-medoids clustering algorithm. The resulting attack edge positions are then grouped and distributed over the Sybils while maintaining Assumption 6. We define three attack scenarios for specific ranges of ϕ. These attack scenarios are the following:

i. Dense Sybil poisoning attack. $\phi \geq 2$. Every honest node has at least two attack edges, whereas any distinct Sybil cannot form more than one attack edge to any given node. As a result, each honest node is a direct neighbor of at least two distinct Sybils.
ii. Distributed Sybil poisoning attack. $\epsilon < \phi < 2$. There exists at least one node which is connected to fewer than 2 attack edges and will therefore only be connected to at most one Sybil.
iii. Sparse Sybil poisoning attack. $\phi \leq \epsilon$. A low ϕ will result in sparse and distant attack edges. Any node has a probability of ϕ of being directly connected to a Sybil.

V. Design of SybilWall

Our proposed algorithm, SybilWall, takes inspiration from federated learning but was meticulously designed to enable boundless scalability through decentralization. We used the state-of-the-art FoolsGold algorithm (federated learning) as a starting point for SybilWall. Moreover, we integrate a probabilistic gossiping mechanism for data dissemination to

aid the aggregation function. In short, SybilWall was designed to mitigate the three attack scenarios of the SSP attack strategy, as listed in Section IV-C. These scenarios are as follows:

i. Dense Sybil poisoning attack: The FoolsGold-based aggregation function detects Sybils directly through its similarity mechanism and is capable of mitigating the attack.
ii. Distributed Sybil poisoning attack: The probabilistic gossiping mechanism serves as a channel for data propagation of probabilistically selected nodes, thereby providing the aggregation function with context on indirect neighbors.
iii. Sparse Sybil poisoning attack: In the case that the probabilistic gossiping mechanism does not provide sufficient context for detecting distant attack edges, we argue that the distance between these edges limits the attack's impact. This is due to a natural dampening effect originating from the train-aggregate loop on each node.

A. Aggregation function

We improve upon the intuitive direction of FoolsGold (Section III-A), designed for inherently unscalable federated learning. By exploiting the high degree of similarity between Sybil models, FoolsGold detects and diminishes the impact of Sybils on the training process. Based on this promising heuristic and Assumption 4, we adopt a modified version of FoolsGold as SybilWall's aggregation function. Our aggregation function improves on FoolsGold in two dimensions.

Firstly, we modify FoolsGold to always trust the aggregator. As a node's training dataset and their training function cannot be compromised by Assumption 1, nodes can trust themselves and may therefore exclude their own work from the similarity and logit scoring function. Its own model is reintroduced into the aggregation with the maximum weight, after all other models have been assigned a score. An additional rationale for this modification is that neighbors with similar datasets should not be penalized during aggregation, as they will possibly produce a similar model as the aggregator.

Secondly, we support the addition of an arbitrary number of model histories in the similarity function. The purpose of the gossiping mechanism (Section V-B) is to spread information about indirect neighbors. By including this information in the cosine similarity function, FoolsGold's similarity heuristic potentially gains the ability to detect new Sybils among its direct neighbors. Note that only the models of direct neighbors are considered for aggregation, and the additional information obtained through gossiping is exclusively used for judging direct neighbors in the similarity function.

B. Probabilistic gossiping mechanism

SybilWall makes use of a probabilistic gossiping mechanism, which allows the dissemination of data among indirect neighbors. By doing so, the sensitivity of the aggregation function improves, as it increases the amount of reference material for the similarity function. This gossiped information consists of the model history h_i^T of some node i in round T, which is defined by $h_i^T = \sum_{t=0}^{T} w_i^t$, where w_i^t is a trained model produced by node i in round t.

1) Probabilistic model selection: First, let us define the method in which model histories are probabilistically selected to be propagated to a neighboring node, for which SybilWall employs a weighted random selection algorithm.

More specifically, let $\mathcal{H}_i$ denote the local database of model histories of node i. $\mathcal{H}_i$ consists of a list of tuples, with each tuple of the form $\langle p, h, r, d, f \rangle \in \mathcal{H}_i$, where h corresponds to the model history of node p, r is the identifier of the synchronous training round from which the model history originates, d is the distance from node i to node p in the number of hops the model history has traveled, and f is the neighbor of node i from which this model history was received. Given the current node i and its neighboring node j, let the filtered database of model histories $\mathcal{H}_i^j$ be defined as $H_i^j = \{(p, h, r, d, f) \mid (p, h, r, d, f) \in H_i, p \notin \{i, j\} \wedge f \neq j\}$. This filtered database is used in a weighted random selection to determine which model history will be gossiped to node j.

To perform the weighted random selection, the entries of the filtered database of model histories are first assigned weights. These weights directly correspond to the distance d and are assigned according to the exponential distribution:

$$P(d) = \lambda e^{-\lambda d} \tag{6}$$

where λ can be considered a hyperparameter representing the relevance of propagating the model history of distant nodes. The selection of the exponential distribution is not arbitrary, as it prioritizes the propagation of the model history of nearby nodes over that of distant nodes. This approach assumes that the sparse Sybil poisoning attack is mitigated through a natural dampening effect, thus reducing the utility of propagating model histories originating from distant nodes. After the weights have been assigned to the filtered database of model histories, a weighted random selection is performed to select which model history is propagated.

A node's local database of model histories can be updated in two distinct methods. First, if a node i receives a model history through gossiping from some other node j, which it has not seen before, it is added to i's local database. Second, if node i receives a model history from some node k that is more recent than the prior model history of k known to node i, it is updated accordingly. Note that the model histories of direct neighbors are updated every round, as each training round will result in a more recent model history. It is possible that a node's local database of model histories may grow to a significant size over time, resulting in a decrease in performance during aggregation. In such a scenario, SybilWall supports dropping outdated model histories to mitigate performance loss.

2) Secure and efficient communication: SybilWall replaces the traditional model communication discussed in Section II-B with a more sophisticated communication protocol. The previously described probabilistic gossiping mechanism requires model histories to be propagated to neighbors, which allows for the forgery of false information by malicious nodes if implemented naively. Such adversarial strategies could be employed to increase the similarity of some target node with another node, thereby potentially increasing the utility of

the adversary. To mitigate this vulnerability, we propose a modification to the default decentralized learning API, which entails the use of signed histories (secure by Assumption 2).

To enable the use of signed histories, the model history and the corresponding signature need to be constructed on the originating node and communicated to its neighbors. These neighbors are now capable of propagating a signed model history of an indirect neighbor to their neighbors through the use of the probabilistic gossiping mechanism. However, this induces additional communication costs, as the trained model, the signed model history, and a gossiped model history all need to be communicated to neighbors every training round. We decrease these communication costs by omitting the trained model, as it can be inferred from the comparison of the two most recently received models histories.

More specifically, we alter the message composition such that a message $m_{i \to j}$ from node i to j can be decomposed into $\langle h_i, S_i(h_i), g_k, S_k(g_k), r_k \rangle$, where h_i represents the updated model history of node i' signed by its signature function S_i and g_k corresponds to the gossiped model history of node k signed by node k originating from round r_k.

3) Downtime tolerance: Due to the adoption of the aforementioned altered decentralized learning communication protocol, SybilWall tolerates downtime of nodes in the network by setting an upper bound on the waiting time for each training round. In contrast to a pull-based communication scheme, where nodes stochastically request a (distant) node's signed model history for context improvement of the cosine similarity function, SybilWall's communication protocol supports arbitrary downtime or the presence of private networks, both resulting in unreachable nodes. Nodes are not responsible for the propagation of their own model history and therefore do not need to be reachable for the probabilistic gossiping mechanism to function properly.

In the event that a node experiences downtime, its aggregation function will start operating properly again once the node is online again and skips an additional training round. This allows for inference of the trained model from two distinct model histories produced by its neighbor.

VI. Evaluation

We evaluate SybilWall by answering the following questions: *(1) How does the complexity of the dataset and the model affect the performance of SybilWall? (2) How does SybilWall perform compared to other existing algorithms? (3) How does the attack density ϕ influence the performance of SybilWall? (4) What is the effect of the distribution of data among nodes on the performance of SybilWall? (5) Can SybilWall be further enhanced by combining it with different techniques?*

A. Experimental setup

We SybilWall implemented in Python3 in the context of a fully operational decentralized learning system for experimental evaluation and is available online [60]. We have used the PyTorch [61] library for the training of machine learning models. Regarding communication between individual nodes, we leveraged IPv8 [62], which provides an API for constructing network overlays in order to simulate P2P networks. Furthermore, we adopted the Gumby library [63] as the experimental execution framework, which was specifically designed for sophisticated experiments with IPv8 involving many nodes. All experiments were performed on the Distributed ASCI Supercomputer 6 (DAS-6) [64]. Each node in the compute cluster has access to a dual 16-core CPU, 128 GB RAM, and either an A4000 or A5000 GPU. Furthermore, all default hyperparameters for the experiments can be found in Table II. Except where mentioned otherwise, these default hyperparameters define the configuration of all experiments.

Table I: The datasets used in the evaluation of SybilWall.

Dataset	Model	Learning rate
MNIST [57]	Single soft-max layer	$\eta = 0.01$ [35]
FashionMNIST [49]	Single soft-max layer	$\eta = 0.01$ [35]
CIFAR-10 [46]	LeNet-5 [45]	$\eta = 0.004$ [58]
SVHN [59]	LeNet-5 [45]	$\eta = 0.004$ [58]

Table II: The default hyperparameters used during the evaluation of SybilWall.

Hyperparameter	Value
# honest nodes	99
Attack edge density ϕ	1
Gossip mechanism parameter λ	0.8
Dirichlet concentration parameter α	0.1
Max node degree d	8
Local epochs	10
Batch size	8

In all experiments, we measure the accuracy of the trained models by averaging the accuracy of the models of all honest nodes. Simultaneously, we measure the success rate of the attacker by averaging the attack score achieved on the models of all honest nodes. The attack score is defined as the accuracy that a model obtains on the altered segment of the data obtained by transforming the test dataset by the data transformation functions defined in equation 4 or 5. Note that both metrics are measured each round directly after aggregation.

1) Datasets: The datasets used during evaluation can be found in Table I. These datasets were chosen for a number of reasons. First of all, MNIST [57] is a widely used dataset for the evaluation of machine learning algorithms [35], [65]–[67], serving as an adequate baseline algorithm for SybilWall. FashionMNIST was developed as a more challenging variant of MNIST, thus serving as an ideal candidate to demonstrate the direct correlation between the complexity of classification tasks and the performance of SybilWall. The choice for SVHN and CIFAR-10 is motivated by the increased complexity of the models required to obtain satisfactory accuracy, which may affect the performance of SybilWall. The use of complex multilayer models in evaluation is frequently overlooked in related work or is performed only on a single dataset [35], [65]–[69]. Moreover, when multilayer models are used, they are regularly pre-trained and trained solely through transfer learning [35], [67], [70]. While we recognize that all the

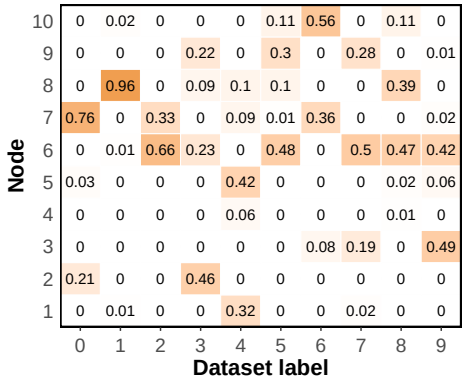
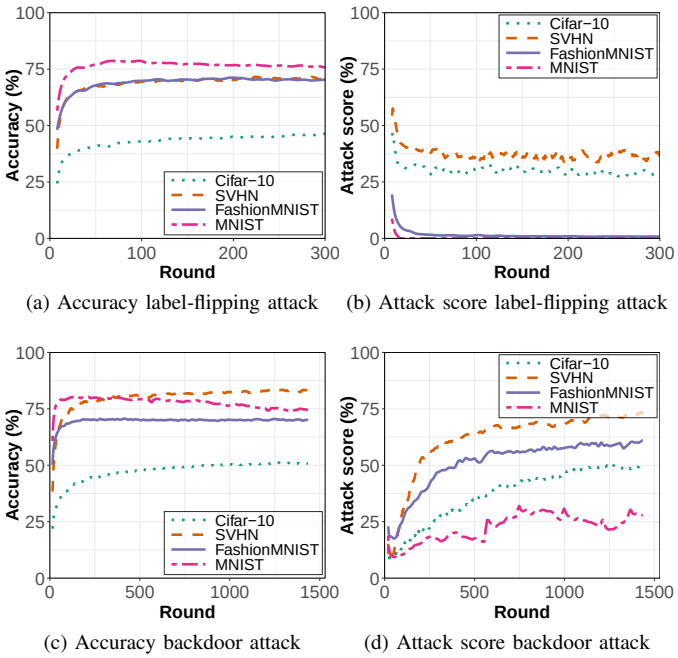


Figure 6: Example distribution for non-i.i.d. data generated with the Dirichlet distribution with concentration parameter $\alpha = 0.1$ for 10 nodes and a dataset containing 10 labels.



(a) Accuracy label-flipping attack (b) Attack score label-flipping attack

(c) Accuracy backdoor attack (d) Attack score backdoor attack

Figure 7: Accuracy and attack score for the label-flipping attack (300 rounds) and the backdoor attack (1450 rounds) on different datasets.

datasets employed in this experimental evaluation focus on image classification, we argue that focusing on image classification is justifiable as it is known as a well-established task in machine learning. Furthermore, image classification frequently serves as a benchmark for evaluating novel distributed machine learning algorithms [26], [35], [65]–[69], and there exists a variety of widely available datasets constructed specifically for this task.

The models that are trained using the aforementioned datasets can also be found in Table I, as well as the corresponding learning rate η. Note that all these models are trained using stochastic gradient descent (SGD).

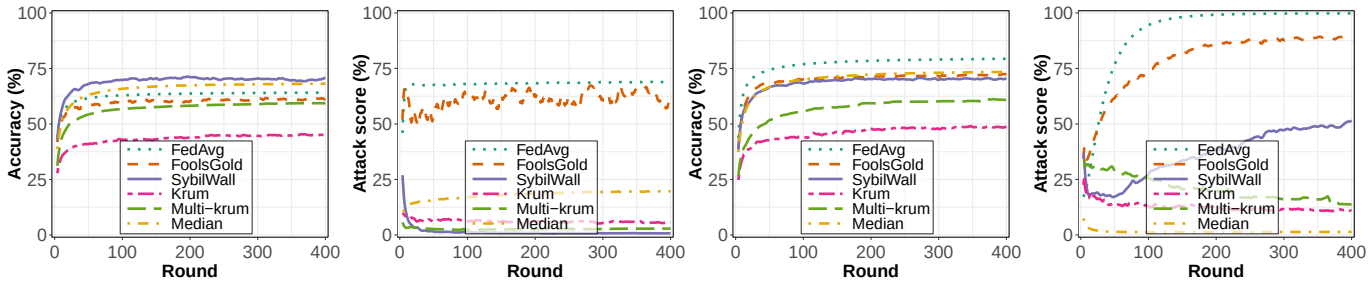
2) Data distribution: The aforementioned datasets are designed for centralized machine learning and require to be distributed among the participating nodes. During our evaluations, we assume that the data is *not* identically and independently distributed (non-i.i.d.), which more closely resembles real-world data than uniformly distributed data (i.i.d) [71], [72]. Although some works employ the use of a K-shard data distribution [8], [67], [73], [74] or simply assign each node a predefined number of classes of the training data [35], [73], [75], we utilize the Dirichlet distribution [76], which has recently gained more popularity for generating non-i.i.d. distributions [26], [77], [78]. More specifically, given the *concentration parameter* α, we compute for each class the fraction of data every node possesses, creating seemingly naturally unfair and irregular data distributions. Lower values of α result in more non-i.i.d. data. Figure 6 illustrates an example distribution for a dataset of 10 labels distributed over 10 nodes with a concentration parameter of $\alpha = 0.1$.

3) Network topology: To generate the necessary network topologies, defining the relations between nodes, we employed *random geometric graphs*. Random geometric graphs are constructed by randomly placing points, which correspond to nodes, on a grid. Two nodes are connected by an edge when the Euclidean distance between the corresponding points of these nodes is smaller than some predefined constant. To enforce the upper bound on a node's degree (Assumption 6), random edges are removed from the random geometric graph, such that all nodes remain connected through a single connected component. The locations of the attack edges are found using the methodology based on K-medoids described in Section IV-C. The code used to generate these network topologies can be found in our published code repository [60]. Furthermore, during our experiments, we assume a static network topology. That is, no nodes will leave or join the network during training, including Sybils. Lastly, we employ the SSP attack (Section IV-C) as the adversarial strategy in the simulated Sybil attacks, as we hypothesize that more distant attack edges will result in a lower detection rate, thereby approximating the optimal attack scenario.

B. Effect of dataset

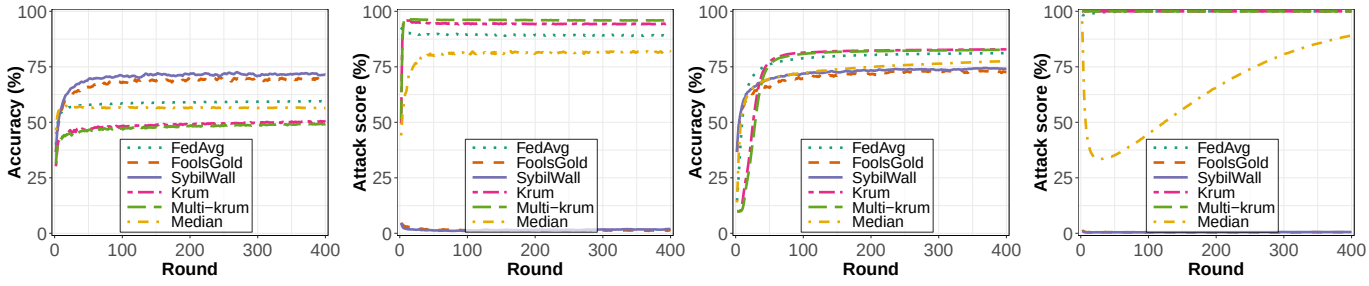
1) Setup: We evaluated the performance of SybilWall on different datasets, allowing us to observe how SybilWall is affected by varying the complexity in both the dataset and the model. This experiment was carried out using the default parameters listed in Table II and using the datasets, models and learning rates listed in Table I.

2) Results: Figure 7 demonstrates the effect of varying the dataset on the trend of accuracy and attack score. We clearly observe that CIFAR-10, arguably the most challenging dataset used in this work, obtains a significantly lower accuracy compared to simpler datasets (Figure 7a), such as MNIST. This can be explained by the reduced overlap of training samples over the different output classes of easier datasets, making



(a) Accuracy label-flipping attack (b) Attack score label-flipping attack (c) Accuracy backdoor attack (d) Attack score backdoor attack

Figure 8: Comparison of SybilWall against different techniques on $\phi = 1$. Results generated using the FashionMNIST [49] dataset.



(a) Accuracy label-flipping attack (b) Attack score label-flipping attack (c) Accuracy backdoor attack (d) Attack score backdoor attack

Figure 9: Comparison of SybilWall against different techniques on $\phi = 4$. Results generated using the FashionMNIST [49] dataset.

individual nodes less likely to counteract their neighbors in a non-i.i.d. setting.

A noteworthy observation with regard to the attack score of the label-flipping attack in Figure 7b is that datasets that require more sophisticated models, such as convolutional neural networks, are generally more susceptible to the label-flipping attack compared to simpler models, such as single-layer neural networks. Due to the smaller number of trainable weights in simpler models, it likely becomes easier to distinguish similarities between Sybil model histories of simpler models compared to more sophisticated models, which may show more diversity due to the increased number of weights. Although some fraction of the higher attack score in complex models can be attributed to the more challenging classification task, subsequent experiments demonstrate that the attack score can be significantly reduced under certain conditions (Sections VI-E and VI-F).

Taking into account the results of the backdoor attack depicted in Figures 7c and 7d, it is apparent that all attack scores demonstrate an increasing trend over a prolonged period of time. Note the difference in the number of rounds between the evaluation on the label-flipping attack and the backdoor attack. This finding suggests that the aggregation technique performed to obtain a node's model history does not always provide a reliable reflection of the node's intentions. However, the time period required for the attack score to achieve convergence is significantly longer than the time required for convergence of the accuracy for most datasets.

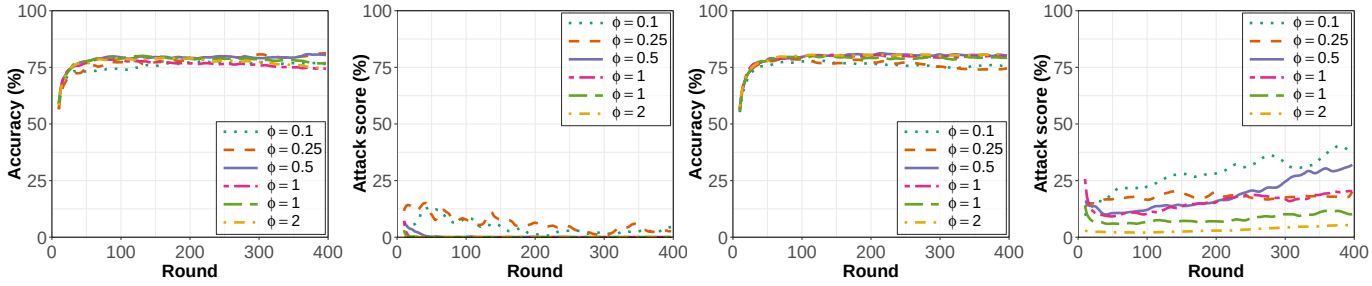
C. Comparison with different techniques

1) Setup: We evaluate the performance of SybilWall relative to a number of different techniques focused on mitigating Sybil poisoning attacks or Byzantine attacks in general. These techniques are the following:

i. FedAvg [8]: naively averages all models. This algorithm was the first proposed federated learning aggregation algorithm and will serve as a baseline during our evaluation.
ii. FoolsGold [35]: detects Sybils among its neighbors by assuming that Sybils produce highly similar models. This algorithm is the main inspiration for SybilWall.
iii. Krum [47]: Excludes Byzantine models by filtering for the model which has the smallest sum of Euclidean distances to its $n - f - 2$ closest neighbors.
iv. Multi-krum [47]: Similar to krum. Averages the m models with the lowest sum of euclidian distances to its $n-f-2$ closest neighbors.
v. Median [79]: Computes the element-wise median of all models and thereby excludes outliers.

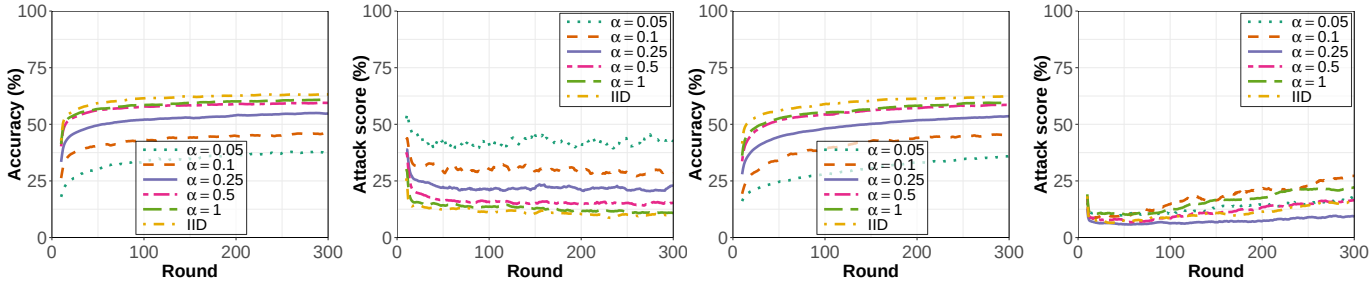
During this experiment, we alternated the attack edge density $\phi \in \{1, 4\}$ and fixed the dataset on FashionMNIST.

2) Results: Figure 8 shows the results of SybilWall compared to different techniques using attack edge density $\phi = 1$. We observe that SybilWall always scores among the best performing algorithms in terms of accuracy. Especially con-



(a) Accuracy label-flipping attack (b) Attack score label-flipping attack (c) Accuracy backdoor attack (d) Attack score backdoor attack

Figure 10: Accuracy and attack score for the label-flipping attack and backdoor attack on different attack edge densities. Results generated using the MNIST [57] dataset.



(a) Accuracy label-flipping attack (b) Attack score label-flipping attack (c) Accuracy backdoor attack (d) Attack score backdoor attack

Figure 11: Accuracy and attack score for the label-flipping attack and backdoor attack of different data distributions, indicated by the concentration parameter α of the Dirichlet distribution. Results generated using the CIFAR-10 dataset [46].

sidering the label-flipping attack, SybilWall achieves the highest accuracy among all evaluated techniques. We also find that SybilWall successfully mitigates the label-flipping attack, similarly to some of the other techniques evaluated. In the backdoor attack, we observe that SybilWall exhibits the same increasing pattern as in the prior experiment on the effect of the datasets in Section VI-B; the attack score starts at a low point and gradually increases as the training progresses.

Figure 9 shows the results of SybilWall compared to different techniques using a higher attack edge density $\phi = 4$. These results clearly demonstrate how most aggregation algorithms succumb under the use of a large-scale Sybil attack. Taking into account the accuracy of both label-flipping attack and backdoor attack, we observe that the accuracy of most algorithms increases significantly when employing the backdoor attack. This phenomenon can be explained by the fact that the adversary is not actively attempting to decrease the accuracy of the model, but only tries to insert an activation pattern, which was highly successful for the algorithms with an increased accuracy. On the other hand, both FoolsGold and SybilWall seem to be unaffected by both attacks. Regarding SybilWall, this is likely caused by the integration of a modified version of FoolsGold, which was specifically designed to mitigate dense Sybil poisoning attack.

Considering both the results in Figure 8 and 9, we find that SybilWall does not outperform all the alternative evaluated techniques in all scenarios, but it is the only technique to consistently score among the best. Furthermore, most other algorithms surprisingly score significantly better under a backdoor attack with a high attack edge density compared to a lower attack edge density, while the accuracy of SybilWall remains constant in both scenarios.

D. Effect of attack edge density

1) Setup: We evaluate SybilWall in a number of different attack edge density configurations. This experiment aims to demonstrate the effect that an attacker can exercise on the network by employing a variety of Sybil attack strategies. MNIST is fixed as the dataset during this experiment and the attack edge density ϕ is varied within the range $\phi \in [0.1, 2]$.

2) Results: Figure 10 illustrates the effect of various attack edge density values on the label-flipping attack and backdoor attack. It is apparent that the attack edge density has little effect on the convergent accuracy (Figures 10a and 10c). On the other hand, the trend of the attack score shows that network topologies with lower attack edge densities are more prone to the label-flipping attack despite the smaller number of generated Sybils for lower values of ϕ (Figure 10b). This clearly demonstrates the effect of the gossiping mechanism on reducing the impact of Sybil poisoning attacks. Strengthening this observation, the results of the backdoor attack in Figure 10d demonstrate how the attack score decreases as the attack edge density increases.

E. Effect of data distribution

1) Setup: The method in which the data is distributed over the nodes might influence the attack score and the accuracy of the trained models. To explore this effect, we evaluate SybilWall's performance under a variety of data distributions. More specifically, we vary the data distribution between i.i.d. and non-i.i.d. (Dirchlet-based). For the non-i.i.d. scenario, we vary the concentration parameter α within the range $\alpha \in [0.05, 1]$. Furthermore, we fixate the dataset on CIFAR-10.

2) Results: Figure 11 shows the effects of different data distributions on the convergence of the training process. We observe in both the label-flipping attack and backdoor attack that the accuracy increases as the data is more uniformly distributed (Figures 11a and 11c). Furthermore, the attack score of the label-flipping attack demonstrates how the attacker becomes less successful with more i.i.d. data (Figure 11b). Lastly, the data distribution does not appear to have a significant effect on the attack score of the backdoor attack, as no clear trend emerges when varying the data distribution (Figure 11d).

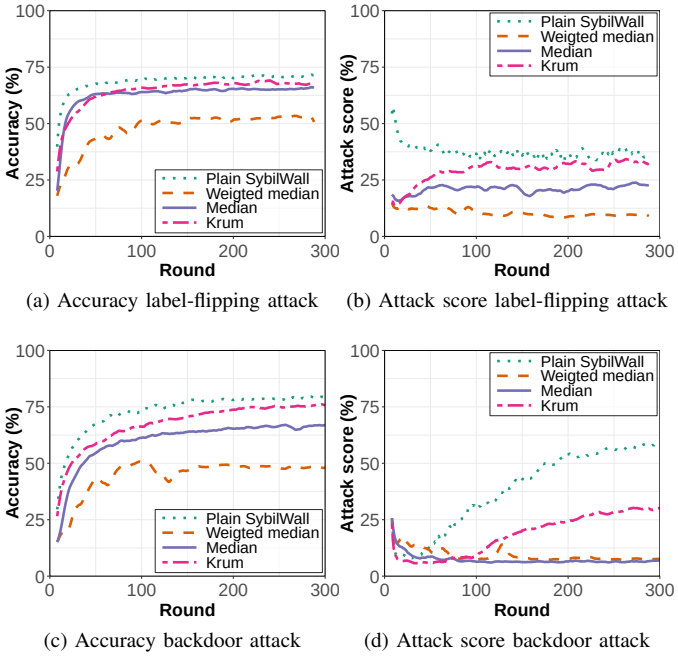
F. Further enhancing SybilWall

1) Setup: Given the increasing, although impeded, attack score demonstrated for the backdoor attack in Section VI-B, we consider several techniques for additional enhancement of the defensive capabilities of SybilWall. These augmentations include the following:

i. Median: given the resilience of the Median [79] algorithm in Section VI-C against attack edge density $\phi = 1$, we implement a combined version of the Median approach and SybilWall. This is achieved by initially employing SybilWall to compute a non-normalized aggregation weight in the range $[0, 1]$, followed by the execution of the Median algorithm on the 50% highest scoring models.
ii. Weighted median: a variant of the Median-based approach, in which scores computed by SybilWall are adopted as weights for a weighted median aggregation.
iii. Krum-filter: based on the suggestion of [35], we combine SybilWall with Krum, such that the model with the lowest Krum score receives an aggregation weight of 0.

We integrate these augmentations through chaining the aggregation functions, such that the last step of SybilWall's aggregation method, a weighted average, is substituted with an augmentation. We also provide the trends for plain SybilWall to serve as a baseline. The dataset is fixed to SVHN.

2) Results: Figure 12 illustrates the effect of enhancing SybilWall with various methodologies. First, we find that plain SybilWall achieves the highest accuracy overall, but the worst Sybil resilience. While each of the evaluated methodologies improves SybilWall's defensive capabilities, a trade-off occurs in which accuracy is sacrificed to obtain improved Sybil resilience. In particular, the Sybil resilience of the weighted median is unmatched, but achieves considerably lower accuracy compared to the alternative methodologies. The Krum-filter-based approach appears to obtain an accuracy comparable



(a) Accuracy label-flipping attack (b) Attack score label-flipping attack

(c) Accuracy backdoor attack (d) Attack score backdoor attack

Figure 12: Accuracy and attack score of the label-flipping attack and backdoor attack for different possible enhancements of SybilWall. Results generated using the SVHN [59] dataset.

to plain SybilWall, but it obtains the worst Sybil resilience of the evaluated enhancements. Arguably, the median-based methodology shows the most promising results, as it achieves to consistently limit the attack score to levels comparable to those of the weighted median methodology, while showing significant improvement on the obtained accuracy.

VII. Discussion

During the experimental evaluation of SybilWall, we found that SybilWall obtains a satisfactory accuracy and convergence rate on 4 widely adopted datasets. Furthermore, the converged accuracy obtained by SybilWall is similar to that achieved by the FedAvg algorithm in a federated learning setting (Figures 3 and 7). In addition to obtaining satisfactory accuracy on all datasets, we also found that SybilWall outperforms alternative algorithms both in obtained accuracy and attack score, as it was the only evaluated algorithm that consistently scored among the best algorithms in all scenarios. SybilWall thereby arguably exhibits the overall strongest resilience to Sybil poisoning attacks and possesses the attributes to be considered state of the art. Although the attack score of the backdoor attack shows a rising trend when employing SybilWall, we note that the rate at which this occurs is significantly reduced, allowing honest nodes to stop the training process once the accuracy has converged, thus limiting the success of potential adversaries.

We argue that the aforementioned rising trend demonstrated by the attack score of the backdoor attack mainly originates from the lack of knowledge of the aggregated intermediary

model induced by the train-aggregate loop depicted in Figure 2. This lack of knowledge prohibits nodes to ascertain the aggregated intermediary model of another node, thereby complicating the extraction of model gradients. Similarly to prior work [35], summing a model's gradients, rather than the model itself, would arguably improve the representation of a node's history. Such an approach would provide a more accurate representation of a node's intentions, as the model history would directly correspond with a node's training data, thus more accurately representing the *direction* in which a node aims to contribute to the aggregated model. Eventually, this improvement might lead to the omission of Assumption 5. Adversaries with extensive computational capabilities violating this assumption, such as click farms [80], may well be able to train numerous malicious models within one training round, thus possibly violating Assumption 4. This highlights another motivator for adopting the use of the sum of model gradients as a node's history. However, obtaining a node's post-training gradients is a non-trivial task in the setting of decentralized learning, as there exists no method of validating the aggregated intermediary model, which was trained to generate the gradients, without sharing the corresponding training data. As an example, a Sybil could claim to start training on an arbitrary model m_r, resulting in seemingly diverse training gradients g, such that the sum of these is equal to the malicious model $m_s = m_r + g$, where m_s is highly similar to the trained model of other Sybils. An adversary could trivially manipulate the sums of gradients of its Sybils to make their work seem more diverse. This drawback is absent in federated learning, as the aggregated intermediary model is equal for all nodes every round and was created by a central authority.

To obtain the model gradients, nodes require the ability to ascertain the aggregated intermediary model of their neighbors. RAGD [48] (Section III-C) achieves this by reaching a global consensus on the aggregated intermediary model. By repeatedly averaging the model with that of neighbors, nodes converge to a globally coherent model under a number of assumptions. However, we argue that these assumptions do not realistically reflect a deployed decentralized setting and are therefore not applicable to this work. We leave the required analysis for a robust method for ascertaining the aggregated intermediary model for future work.

During the evaluation of the effect of the number of Sybils on the attack score in Section VI-D, we found that decreasing the number of Sybils increases the attack score. This implies that we eliminated the need to amplify a poisoning attack with the Sybil attack, as employing Sybils would result in a lower attack score. However, reducing the Sybil poisoning attack to a simple poisoning attack, which cannot be deflected by SybilWall, requires the integration of alternative poisoning attack mitigation algorithms. During evaluation, we considered further enhancing SybilWall with a number of such alternative algorithms through chained aggregation in Section VI-F. Although all enhancements demonstrated an increased resilience to Sybil poisoning, they sacrifice in terms of accuracy. Considering that accuracy is often the primary goal in machine learning [81], the use of such enhancements is likely not justifiable in most applications. We leave further enhancing SybilWall with a poisoning attack mitigation algorithm for increased resilience against single attackers, without compromising accuracy, as a possible research direction for future work.

Furthermore, adversaries may employ a strategy to generate more diverse Sybil model histories. By introducing random noise to the irrelevant weights of the model [35], adversaries may be able to significantly increase the diversity among Sybil model histories, resulting in a violation of Assumption 4. Additionally, it may be possible to generate sufficient diversity between Sybils to form multiple attack edges to the same honest node. More research is required to accurately filter exclusively relevant weights, which could be achieved through a number of approaches, such as layer-wise relevance propagation [82], weight magnitude filtering [83], or empirical weight importance [84].

VIII. Conclusion

We have presented SybilWall, a pioneering algorithm in the mitigation of Sybil poisoning attacks in decentralized learning. Building on the popular federated learning Sybil poisoning mitigation algorithm, FoolsGold [35], we exploit the increased similarity between the models produced by Sybils over that of honest nodes. We proposed a probabilistic gossiping mechanism to facilitate data dissemination. The disseminated data aids in the mitigation of a poisoning attack amplified by distributing Sybils over the decentralized network. We found that SybilWall achieves satisfactory performance on four widely adopted datasets and obtains similar accuracy to federated learning. Furthermore, SybilWall was compared with a number of alternative algorithms and was found to be the only algorithm to consistently score among the best in all the evaluated scenarios, thus arguably outperforming all the alternative evaluated algorithms. Although SybilWall does not fully mitigate targeted poisoning attacks in the form of a backdoor attack, it manages to greatly decrease the convergence rate of the attacker's success. This enables honest nodes to complete the training process prior to the attack having substantial impact.

We proposed a number of promising future research directions, such as further improving SybilWall to successfully mitigate single attackers, or exploring potential improvements to mitigate the backdoor attack by adopting the usage of summed model gradients in the similarity metric.

References

[1] E. V. Polyakov, M. S. Mazhanov, A. Y. Rolich, L. S. Voskov, M. V. Kachalova, and S. V. Polyakov, "Investigation and development of the intelligent voice assistant for the internet of things using machine learning," in *2018 Moscow Workshop on Electronic and Networking Technologies (MWENT)*, 2018, pp. 1–5.

[2] B. T.K., C. S. R. Annavarapu, and A. Bablani, "Machine learning algorithms for social media analysis: A survey," *Computer Science Review*, vol. 40, p. 100395, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013721000356

[3] X. Wang and Y. Wang, "Improving content-based and hybrid music recommendation using deep learning," in *Proceedings of the 22nd ACM International Conference on Multimedia*, ser. MM '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 627–636. [Online]. Available: https://doi.org/10.1145/2647868.2654940

[4] S. A. Salloum, M. Alshurideh, A. Elnagar, and K. Shaalan, "Machine learning and deep learning techniques for cybersecurity: A review," in *Proceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV2020)*, A.-E. Hassanien, A. T. Azar, T. Gaber, D. Oliva, and F. M. Tolba, Eds. Cham: Springer International Publishing, 2020, pp. 50–57.

[5] J. Prusa, T. M. Khoshgoftaar, and N. Seliya, "The effect of dataset size on training tweet sentiment classifiers," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 2015, pp. 96–102.

[6] J. Hestness, S. Narang, N. Ardalani, G. F. Diamos, H. Jun, H. Kianinejad, M. M. A. Patwary, Y. Yang, and Y. Zhou, "Deep learning scaling is predictable, empirically," *CoRR*, vol. abs/1712.00409, 2017. [Online]. Available: http://arxiv.org/abs/1712.00409

[7] A. Goldsteen, G. Ezov, R. Shmelkin, M. Moffie, and A. Farkash, "Data minimization for gdpr compliance in machine learning models," *AI and Ethics*, pp. 1–15, 2021.

[8] 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*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[9] J. Janai, F. Güney, A. Behl, A. Geiger *et al.*, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," *Foundations and Trends® in Computer Graphics and Vision*, vol. 12, no. 1–3, pp. 1–308, 2020.

[10] P. Navarro, C. Fernández, R. Borraz, and D. Alonso, "A machine learning approach to pedestrian detection for autonomous vehicles using high-definition 3d range data," *Sensors*, vol. 17, no. 12, p. 18, Dec 2016. [Online]. Available: http://dx.doi.org/10.3390/s17010018

[11] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *CoRR*, vol. abs/1811.03604, 2018. [Online]. Available: http://arxiv.org/abs/1811.03604

[12] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," *CoRR*, vol. abs/1812.02903, 2018. [Online]. Available: http://arxiv.org/abs/1812.02903

[13] M. Chen, R. Mathews, T. Ouyang, and F. Beaufays, "Federated learning of out-of-vocabulary words," *CoRR*, vol. abs/1903.10635, 2019. [Online]. Available: http://arxiv.org/abs/1903.10635

[14] Y. Cheng, Y. Liu, T. Chen, and Q. Yang, "Federated learning for privacy-preserving ai," *Communications of the ACM*, vol. 63, no. 12, pp. 33–36, 2020.

[15] L. Lyu and C. Chen, "A novel attribute reconstruction attack in federated learning," *CoRR*, vol. abs/2108.06910, 2021. [Online]. Available: https://arxiv.org/abs/2108.06910

[16] H. Yang, M. Ge, K. Xiang, and J. Li, "Using highly compressed gradients in federated learning for data reconstruction attacks," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 818–830, 2023.

[17] H. S. Sikandar, H. Waheed, S. Tahir, S. U. R. Malik, and W. Rafique, "A detailed survey on federated learning attacks and defenses," *Electronics*, vol. 12, no. 2, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/2/260

[18] P. Qiu, X. Zhang, S. Ji, Y. Pu, and T. Wang, "All you need is hashing: Defending against data reconstruction attack in vertical federated learning," 2022. [Online]. Available: https://arxiv.org/abs/2212.00325

[19] J. Hamer, M. Mohri, and A. T. Suresh, "FedBoost: A communication-efficient algorithm for federated learning," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 3973–3983. [Online]. Available: https://proceedings.mlr.press/v119/hamer20a.html

[20] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, and K. Ramchandran, "Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning," *CoRR*, vol. abs/2009.11248, 2020. [Online]. Available: https://arxiv.org/abs/2009.11248

[21] Y. Qi, M. S. Hossain, J. Nie, and X. Li, "Privacy-preserving blockchain-based federated learning for traffic flow prediction," *Future Generation Computer Systems*, vol. 117, pp. 328–337, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X2033065X

[22] J. Hou, F. Wang, C. Wei, H. Huang, Y. Hu, and N. Gui, "Credibility assessment based byzantine-resilient decentralized learning," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–12, 2022.

[23] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," *CoRR*, vol. abs/1908.07782, 2019. [Online]. Available: http://arxiv.org/abs/1908.07782

[24] I. Hegedűs, G. Danner, and M. Jelasity, "Decentralized learning works: An empirical comparison of gossip learning and federated learning," *Journal of Parallel and Distributed Computing*, vol. 148, pp. 109–124, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731520303890

[25] Z. Tang, S. Shi, B. Li, and X. Chu, "Gossipfl: A decentralized federated learning framework with sparsified and adaptive communication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 3, pp. 909–922, 2023.

[26] M. de Vos, A. Dhasade, A.-M. Kermarrec, E. Lavoie, and J. Pouwelse, "Modest: Bridging the gap between federated and decentralized learning with decentralized sampling," 2023.

[27] I. Hegedűs, G. Danner, and M. Jelasity, "Decentralized learning works: An empirical comparison of gossip learning and federated learning," *Journal of Parallel and Distributed Computing*, vol. 148, pp. 109–124, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731520303890

[28] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Computer Security – ESORICS 2020*, L. Chen, N. Li, K. Liang, and S. Schneider, Eds. Cham: Springer International Publishing, 2020, pp. 480–501.

[29] N. M. Jebreel, J. Domingo-Ferrer, D. Sánchez, and A. Blanco-Justicia, "Defending against the label-flipping attack in federated learning," 2022. [Online]. Available: https://arxiv.org/abs/2207.01982

[30] D. Li, W. E. Wong, W. Wang, Y. Yao, and M. Chau, "Detection and mitigation of label-flipping attacks in federated learning systems with kpca and k-means," in *2021 8th International Conference on Dependable Systems and Their Applications (DSA)*, 2021, pp. 551–559.

[31] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 2938–2948. [Online]. Available: https://proceedings.mlr.press/v108/bagdasaryan20a.html

[32] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *CoRR*, vol. abs/1911.07963, 2019. [Online]. Available: http://arxiv.org/abs/1911.07963

[33] C. Wu, X. Yang, S. Zhu, and P. Mitra, "Mitigating backdoor attacks in federated learning," *CoRR*, vol. abs/2011.01767, 2020. [Online]. Available: https://arxiv.org/abs/2011.01767

[34] J. R. Douceur, "The sybil attack," in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260.

[35] C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," *CoRR*, vol. abs/1808.04866, 2018. [Online]. Available: http://arxiv.org/abs/1808.04866

[36] I. Hegedűs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *Distributed Applications and Interoperable Systems*, J. Pereira and L. Ricci, Eds. Cham: Springer International Publishing, 2019, pp. 74–90.

[37] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning," *CoRR*, vol. abs/1905.06731, 2019. [Online]. Available: http://arxiv.org/abs/1905.06731

[38] C. Wu, X. Yang, S. Zhu, and P. Mitra, "Mitigating backdoor attacks in federated learning," *CoRR*, vol. abs/2011.01767, 2020. [Online]. Available: https://arxiv.org/abs/2011.01767

[39] B. N. Levine, C. Shields, and N. B. Margolin, "A survey of solutions to the sybil attack," *University of Massachusetts Amherst, Amherst, MA*, vol. 7, p. 224, 2006.

[40] D. N. Tran, B. Min, J. Li, and L. Subramanian, "Sybil-resilient online content voting." in *NSDI*, vol. 9, no. 1, 2009, pp. 15–28.

[41] H. Rowaihy, W. Enck, P. McDaniel, and T. La Porta, "Limiting sybil attacks in structured p2p networks," in *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, 2007, pp. 2596–2600.

[42] Y. Xie, F. Yu, Q. Ke, M. Abadi, E. Gillum, K. Vitaldevaria, J. Walter, J. Huang, and Z. M. Mao, "Innocent by association: Early recognition of legitimate users," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 353–364. [Online]. Available: https://doi.org/10.1145/2382196.2382235

[43] F. Lesueur, L. Mé, and V. V. T. Tong, "A sybil-resistant admission control coupling sybilguard with distributed certification," in *2008 IEEE 17th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2008, pp. 105–110.

[44] M. Moradi and M. Keyvanpour, "Captcha and its alternatives: A review," *Security and Communication Networks*, vol. 8, no. 12, pp. 2135–2156, 2015. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.1157

[45] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[46] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)." [Online]. Available: http://www.cs.toronto.edu/ kriz/cifar.html

[47] 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*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[48] Y. Mao, D. Data, S. Diggavi, and P. Tabuada, "Decentralized learning robust to data poisoning attacks," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 6788–6793.

[49] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. Https://github.com/zalandoresearch/fashion-mnist. [Online]. Available: https://github.com/zalandoresearch/fashion-mnist

[50] M. de Vos and J. Pouwelse, "Contrib: Maintaining fairness in decentralized big tech alternatives by accounting work," *Computer Networks*, vol. 192, p. 108081, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621001705

[51] Q. Stokkink, C. U. Ileri, D. Epema, and J. Pouwelse, "Web3 sybil avoidance using network latency," *Computer Networks*, vol. 227, p. 109701, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128623001469

[52] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou, "Walking in facebook: A case study of unbiased sampling of osns," in *2010 Proceedings IEEE INFOCOM*, 2010, pp. 1–9.

[53] A. Singh, T.-W. Ngan, P. Druschel, and D. S. Wallach, "Eclipse attacks on overlay networks: Threats and defenses," in *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, 2006, pp. 1–12.

[54] R. Church and C. ReVelle, "The maximal covering location problem," in *Papers of the regional science association*, vol. 32, no. 1. Springer-Verlag Berlin/Heidelberg, 1974, pp. 101–118.

[55] N. Megiddo, E. Zemel, and S. L. Hakimi, "The maximum coverage location problem," *SIAM Journal on Algebraic Discrete Methods*, vol. 4, no. 2, pp. 253–261, 1983. [Online]. Available: https://doi.org/10.1137/0604028

[56] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.

[57] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[58] C. Thapa, M. A. P. Chamikara, and S. Camtepe, "Splitfed: When federated learning meets split learning," *CoRR*, vol. abs/2004.12088, 2020. [Online]. Available: https://arxiv.org/abs/2004.12088

[59] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. [Online]. Available: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf

[60] T. Werthenbach, "Sybil-resilient-decentralized-learning," https://github.com/ThomasWerthenbach/Sybil-Resilient-Decentralized-Learning, 2023.

[61] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[62] Tribler, "Python implementation of tribler's ipv8 p2p-networking layer," https://github.com/Tribler/py-ipv8, 2023.

[63] ——, "Experiment runner framework for ipv8 and tribler," https://github.com/Tribler/gumby, 2022.

[64] H. Bal, D. Epema, C. de Laat, R. van Nieuwpoort, J. Romein, F. Seinstra, C. Snoek, and H. Wijshoff, "A medium-scale distributed system for computer science research: Infrastructure for the long term," *Computer*, vol. 49, no. 05, pp. 54–63, may 2016.

[65] C. Pappas, D. Chatzopoulos, S. Lalis, and M. Vavalis, "Ipls: A framework for decentralized federated learning," in *2021 IFIP Networking Conference (IFIP Networking)*, 2021, pp. 1–6.

[66] S. Alqahtani and M. Demirbas, "Performance analysis and comparison of distributed machine learning systems," *CoRR*, vol. abs/1909.02061, 2019. [Online]. Available: http://arxiv.org/abs/1909.02061

[67] J. Verbraeken, M. de Vos, and J. Pouwelse, "Bristle: Decentralized federated learning in byzantine, non-i.i.d. environments," *CoRR*, vol. abs/2110.11006, 2021. [Online]. Available: https://arxiv.org/abs/2110.11006

[68] H. Ye, L. Liang, and G. Y. Li, "Decentralized federated learning with unreliable communications," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 3, pp. 487–500, 2022.

[69] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," *CoRR*, vol. abs/1908.07782, 2019. [Online]. Available: http://arxiv.org/abs/1908.07782

[70] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[71] T.-C. Chiu, Y.-Y. Shih, A.-C. Pang, C.-S. Wang, W. Weng, and C.-T. Chou, "Semisupervised distributed learning with non-iid data for aiot service platform," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9266–9277, 2020.

[72] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-IID data quagmire of decentralized machine learning," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 4387–4398. [Online]. Available: https://proceedings.mlr.press/v119/hsieh20a.html

[73] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *CoRR*, vol. abs/1806.00582, 2018. [Online]. Available: http://arxiv.org/abs/1806.00582

[74] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-iid data," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 15–24.

[75] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–9.

[76] G. L. Dirichlet, "Über die reduction der positiven quadratischen formen mit drei unbestimmten ganzen zahlen." *Journal für die reine und angewandte Mathematik (Crelles Journal)*, vol. 1850, no. 40, pp. 209–227, 1850. [Online]. Available: https://doi.org/10.1515/crll.1850.40.209

[77] L. Gao, H. Fu, L. Li, Y. Chen, M. Xu, and C.-Z. Xu, "Feddc: Federated learning with non-iid data via local drift decoupling and correction," 2022.

[78] X. Mu, Y. Shen, K. Cheng, X. Geng, J. Fu, T. Zhang, and Z. Zhang, "Fedproc: Prototypical contrastive federated learning on non-iid data," *Future Generation Computer Systems*, vol. 143, pp. 93–104, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X23000262

[79] D. Yin, Y. Chen, K. Ramchandran, and P. L. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," *CoRR*, vol. abs/1803.01498, 2018. [Online]. Available: http://arxiv.org/abs/1803.01498

[80] E. Drott, "Fake streams, listening bots, and click farms: Counterfeiting attention in the streaming music economy," *American Music*, vol. 38, no. 2, pp. 153–175, 2020.

[81] S. Kaur and S. Jindal, "A survey on machine learning algorithms," *Int J Innovative Res Adv Eng (IJIRAE)*, vol. 3, no. 11, pp. 2349–2763, 2016.
[82] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLOS ONE*, vol. 10, no. 7, pp. 1–46, 07 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0130140
[83] M. C. Mozer and P. Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," in *Advances in Neural Information Processing Systems*, D. Touretzky, Ed., vol. 1. Morgan-Kaufmann, 1988.
[84] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.