Privacy Attacks in Decentralized Learning Systems that Exchange Chunked Models

Robust Decentralized Learning

Halil Betmezoğlu
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: Halil Betmezoğlu
Final project course: CSE3000 Research Project
Thesis committee: Bart Cox, Jérémie Decouchant, Anna Lukina

Privacy Attacks in Decentralized Learning Systems that Exchange Chunked Models

Halil Betmezoğlu, Bart Cox, Jérémie Decouchant
Delft University of Technology

Abstract—**Decentralized Learning (DL) is a key tool for training machine learning models on sensitive, distributed data. However, peer-to-peer model exchange in DL systems exposes participants to privacy attacks. Existing defenses often degrade model utility, introduce communication overhead, or are not applicable to a DL system. Model chunking (splitting a model into parts before sharing them individually) has been proposed as an alternative, but its standalone privacy implications have not been investigated. This work implements and evaluates three distinct model chunking methods (static, cyclic, and random) against two privacy attacks: membership inference and linkability. Our work introduces a Hungarian matching-based enhancement to the linkability attack, and relaxes prior assumptions by evaluating attack success under limited access to neighbor datasets. Our results show that chunking increases vulnerability to membership inference. However, static and random chunking are effective against linkability attacks under specific conditions, particularly when full epochs are used during training.**

I. Introduction

In recent years, there has been a rapid shift in how machine learning models are trained. Traditionally, the data used for machine learning training is gathered in one location where the model can be trained. However, as the applications of machine learning expanded, these models had to be trained on sensitive data that could not be relocated to a central location. Examples of such cases include medical records of patients in healthcare, financial data of customers in banking, and personal information on smartphones.

Confronted with these challenges, approaches where data is processed in its original location were developed, with federated learning (FL) [1–3] and decentralized learning (DL) [4, 5]. The main difference between these two methods is that in FL, there is a central server which coordinates model training between participants, whereas in DL, there is no central authority, and participants communicate in a peer-to-peer manner.

These approaches achieve privacy of the data during model training through local training. However, model parameters in DL and the global model in FL are still shared between the participants, which creates vulnerabilities regarding privacy [6–8]. In this regard, FL and DL systems remain vulnerable to membership and attribute inference, gradient inversion and reconstruction, and linkability attacks [9, 10]. These attacks aim to infer information about the system, which they achieve by exploiting the shared model parameters and target model behavior. In DL, these risks are amplified because there is no central aggregator to obfuscate individual contributions: each participant directly shares their own model updates with peers.

In order to overcome the challenges of privacy in DL, frameworks where noise is introduced into the system [11–13] or only part of the model update is shared [14, 15] have been developed. However, these frameworks have been shown to weaken the performance of the trained model [16]. In this regard, a framework called *Shatter* was introduced, which does not rely on noise [17]. Instead, it introduces a concept called *model chunking*, where model parameters are divided into smaller groups, which are then shared between the neighbors of each peer in each round of communication. Additionally, *Shatter* applies *virtualization* to the shared parameters via virtual nodes to obfuscate model origins. *Shatter* claims to improve privacy; however, as Biswas et al. [17] point out, "the privacy benefits of model chunking diminish without virtualization". The authors study the combined privacy benefits of virtualization and model chunking; however, a systematic investigation of model chunking in isolation is lacking, leaving its privacy implications unexplored.

This work investigates model chunking as a standalone privacy-preserving mechanism in DL, focusing on its resilience to membership inference and linkability attacks. Our contributions are as follows:

- We improve the linkability attack with Hungarian matching and relax its assumptions with partial neighbor dataset access.
- We quantify the privacy impact of model chunking in decentralized learning and evaluate its effectiveness against membership inference and linkability attacks.
- We compare the privacy implications of static, cyclic, and random chunking methods under varying training conditions.

The remainder of this paper is organized as follows. Section II and Section III provide background and related work. Section IV defines the threat model. Sections V, VI and VII cover our methodology, experimental setup, and results. Section VIII discusses our findings, followed by a conclusion in Section IX and ethical considerations in Section X.

II. Background

In this section, we describe the primary privacy risks in decentralized learning which we investigate in this paper. Then, we describe *Shatter* [17], a framework that aims to address these privacy challenges by implementing a decentralized learning system that exchanges chunked models.

A. *Privacy Risks*

Despite avoiding explicit data sharing between nodes, decentralized learning (DL) is vulnerable to privacy attacks [16]. As an honest-but-curious malicious attacker in the network, it is still possible to devise attacks to gain information about the data that belongs to other nodes. We investigate two attacks in our paper, membership inference and linkability, which we briefly describe here. Implementation details are provided in Section V.

Membership Inference. In a membership inference attack (MIA), the adversary determines whether a specific data point was used in training by exploiting the tendency of models to respond with higher confidence to previously seen data [18–20]. The attacker queries the target model with suspected training data points and compares responses to infer membership. Hu et al. demonstrate that black-box attacks remain performant threats despite the lack of internal model access [21]. Shokri et al. present a shadow-model-based attack, which we implement and discuss further in Section V-B [18].

Linkability. In a linkability attack (LA), the adversarial node attempts to associate the model parameters that it receives with the training sets of its neighbors [10, 17, 22]. In order to achieve this, it compares the loss of each parameter update it receives on each training set of its neighboring nodes. The training set with the lowest loss, and hence its node, is then associated with that parameter update. In case the received model parameters are incomplete (for instance, in model chunking) the parameters are completed with those from the local model.

B. *Shatter*

Confronted with the privacy risks discussed, *Shatter* was created as a privacy-preserving protocol for decentralized learning [17]. Rather than sharing full models, *Shatter* splits each model into disjoint chunks, which are shared via virtual nodes to neighbors.

In spite of its privacy-enhancing features, *Shatter* makes some assumptions in its design and the attacks performed against it. In terms of design, *Shatter* uses static chunking, which means that each real node chunks its model parameters from the same indices in all rounds. Other approaches are to rotate which chunks are shared in each round, called cyclic chunking, or to randomly distribute chunks, referred to as random chunking. *Shatter* also assumes non-collaborative malicious nodes, whereas collaboration could increase attack severity. Furthermore, the virtualization process creates communication overhead during parameter sharing, which is eliminated by using simple model chunking. Therefore, it is essential that the the privacy of model chunking is assessed without virtualization.

A comparison between *Shatter* and other related work is presented in Section III-C.

III. Related Work

In this section, we survey existing research on privacy in collaborative machine learning, evaluate the effectiveness of attacks, and discuss current defenses for privacy.

A. *Attacks in Collaborative Machine Learning*

We review privacy attacks, other than membership inference and linkability, in collaborative machine learning, specifically for federated learning (FL) and decentralized learning (DL).

Gradient Inversion. In gradient inversion, the attacker keeps track of parameter updates, and tries to directly reconstruct the data that would result in such an update to the parameters [24]. In DL, this type of attack is most dangerous in the first rounds of training, since any gradient update is directly influenced by the raw data owned by the victim node.

Gradient Reconstruction. DL widely uses Decentralized Gradient Descent (D-GD) or some variation of D-GD. El Mrini et al. describe a reconstruction attack to infer gradients from shared model parameters [25]. This information can further be used with gradient inversion to infer private data of other nodes, even those further than neighbors.

Since model chunking exchanges parameters rather than gradients, an inversion attack would have to be preceded by a reconstruction attack. In our work, we focus on exchanged parameters and exclude these attacks from our evaluation.

Attribute Inference. Attribute inference is another type of attack on machine learning models [26]. This attack enables adversaries to infer missing or sensitive attributes from partially known records through strategic queries to the trained model, performing particularly well against overfitted models.

This attack is particularly relevant for tabular data, where a sensitive attribute can be predicted. In this work we will be using image data; therefore, we choose not to investigate this attack as it is not relevant to image-based datasets.

B. *Effectiveness of Attacks*

Pasquini et al. examine the insecurities of DL [16]. They argue that it is less secure than FL because any node can act as an adversary. Membership inference is stronger in DL because attackers know more about nearby nodes than the whole FL system. Gradient inversion is equally effective as in FL and improves as the adversary connects to more neighbors.

Ji et al. theoretically and experimentally compare privacy in DL against FL [27]. They find that DL is safer than FL only when secure aggregation is used, and agree with Pasquini et al. [16] that denser networks leak more information.

Hallaji et al. survey the security of various FL and DL methods, against passive and active attacks [9]. Reaffirming Pasquini et al. [16], they conclude that decentralized learning offers a greater attack surface, primarily because peer-to-peer secure aggregation in DL is harder to achieve than in FL.

C. *Defenses*

Strengthening the privacy of decentralized learning models has been investigated and multiple methods have been proposed. These approaches fall into three main categories: noise-based approaches, sparsification, and secure aggregation. We summarize our analysis of these defenses in Table I.

Noise-based approaches. *Leader-Follower Elastic Averaging SGD* (LEASGD) aims to provide differential privacy by applying gradient clipping and Gaussian noise in gradients

Defense	Fully Decentralized	Full Utility	Efficient Communication	Full Sharing
LEASGD [11]	✗	✗	✓	✓
Zip-DL [12]	✓	✗	✓	✓
Muffliato [13]	✓	✗	✓	✓
Selective SGD [15]	✗	✗	✓	✗
TopK SGD [14]	✗	✗	✓	✗
Secure Aggregation [23]	✗	✓	✗	✓
Shatter [17]	✓	✓	✗	✓
Model Chunking [10, 17]	✓	✓	✓	✓

TABLE I: A summary of current defenses in DL and their capabilities. The well-rounded nature of model chunking highlights the need to evaluate its security performance against privacy attacks.

during training [11]. *Zip-DL* introduces noise to shared parameters of each node, such that they sum to zero within local neighborhoods, preserving convergence while improving privacy [12]. *Muffliato* applies noise directly to local data, making it increasingly indistinguishable through gossip averaging [13].

The main drawback of these approaches is that the introduction of noise into the system negatively impacts the utility of the trained model [16]. This is evaluated by comparing against *Decentralized Parallel Stochastic Gradient Descent* (D-PSGD) [4]. D-PSGD does not take into account security considerations; therefore, it is the baseline for the performance of the defenses.

Sparsification. Another approach to improve privacy in decentralized learning is to share a subset of parameter updates with neighboring nodes, keeping the rest always private, thereby limiting the information revealed about local data. This is called sparsification. Shokri and Shmatikov propose *Selective SGD*, where updates are chosen either randomly or based on gradient magnitude, with the latter preserving convergence better [15]. *TopK SGD* follows a similar idea by sharing only the top K largest gradients from local updates [14].

Sparsification is inherently similar to model chunking. In both, only certain sections of the local computation is shared with each neighbor. While sparsification is generally researched in the context of improving communication efficiency with privacy as an incidental benefit, model chunking has a direct focus on improving privacy.

Secure Aggregation. Secure aggregation is a cryptographic protocol that enables a central server to compute the sum of model updates from users, without learning any individual user's contribution [23]. It is highly applicable in an FL scenario in order to preserve privacy of users. However, while effective at hiding individual updates, secure aggregation is computation- and communication-heavy, and is difficult to deploy in fully decentralized settings without a trusted coordinator.

IV. Threat Model

In this section, we describe our threat model by explaining the adversary's capability and behavior.

Capability. Attacks against machine learning models are broadly categorized into white- and black-box attacks [21]. In white-box attacks, the adversary has complete information about the training data, target model architecture, learned parameters and hyper-parameters. In contrast, black-box attacks provide much less information: the adversary can only query the model and observe its outputs. A third category, called gray-box attacks, lies in between these two categories [28]. It allows the adversary to have access to only the target model's architecture, hyper-parameters, and training data distribution. We use a *gray-box* model for membership inference attacks (with only knowledge of the target model architecture and training data distribution), and the *white-box* model for linkability attacks (with only access to the training data of neighbors).

Behavior. There are two adversarial behaviors: active and passive. Active attacks involve behaving outside the learning protocol, including data poisoning and backdoor attacks. Passive attacks involve honest-but-curious behavior, where the adversary obeys the protocol but infers information about others through observations. Our attacks are executed *passively*, as an honest-but-curious adversary.

V. Methodology

In this section, we present how we use and extend `DecentralizePy` for our research. Then, we explain the implementations of the two different privacy attacks that we execute against `DecentralizePy` with model chunking.

A. DecentralizePy and Model Chunking

`DecentralizePy` is the foundation on which our work is built [29]. It is a framework that allows for the creation of a network of nodes that collaboratively train their own machine learning models. The network consists of n nodes, which are directly connected to each other in an r-regular graph, where each node has exactly r neighbors. Other static or dynamic topologies can also be created, which effectively simulate epidemic learning [30].

It has been proposed that model parameters are separated into disjoint subsets, named chunks, and that each chunk is shared with a neighbor [10, 17]. Since model chunking is not a part of the original `DecentralizePy` framework, we provide our own implementation. This splits model parameters into k chunks, with the number of chunks equal to the number of neighbors r of each node. We implement three different chunking methods: static, cyclic, and random. In static chunking, the same chunk is sent to a specific neighbor in all training rounds. In cyclic chunking, the chunk that is sent to each node is rotated over the training rounds. Finally, random chunking selects which chunk to send randomly. We also implement no chunking (which shares all model parameters with all neighbors) as a baseline.

In each training round, received parameters from a chunk are then aggregated with the node's own local parameters by averaging; otherwise the local parameters are left unchanged.

B. *Privacy Attacks*

We implement two privacy attacks, membership inference and linkability, which we explain here. Table II presents a summary of the properties of each attack.

	Membership Inference	**Linkability**
Dataset Access	Disjoint subset	Neighbor datasets
Model Access	Model architecture	None
Timing	After training[1]	During training
Target Node(s)	Any node	Neighbors

TABLE II: A summary of our attacks.

Membership Inference Attack. The goal of a membership inference attack (MIA) is to successfully predict whether a given data point was used as part of the training dataset of a model [19]. In this attack, the adversary has data records that could belong to the training dataset of the model. In our threat model, as described in Section IV, the attacker has access to a distribution of the dataset (as a disjoint subset) and the model as a gray-box, where only its architecture is known. In order to accomplish this attack, we employ the use of shadow models as proposed by Shokri et al. [18]. The target models will train with the chunking methods described in Section V-A, and the attack will be executed on these nodes.

The structure of this attack can be viewed in four steps: shadow data collection, shadow model training, attack data generation, and attack model training, as explained below. An overview of the attack process is illustrated in Figure 1.

1) ***Shadow data collection.*** Our shadow models train on separate shadow datasets, which are created by sampling the original dataset. Both a training and a test set are created for each shadow model. To guarantee that the data is only accessible to the attacker and no other nodes, the sampling is done on a distinct subset of the whole dataset.

[1]Refers to model parameters being used to update the local model in that round. Can be executed after every training round.

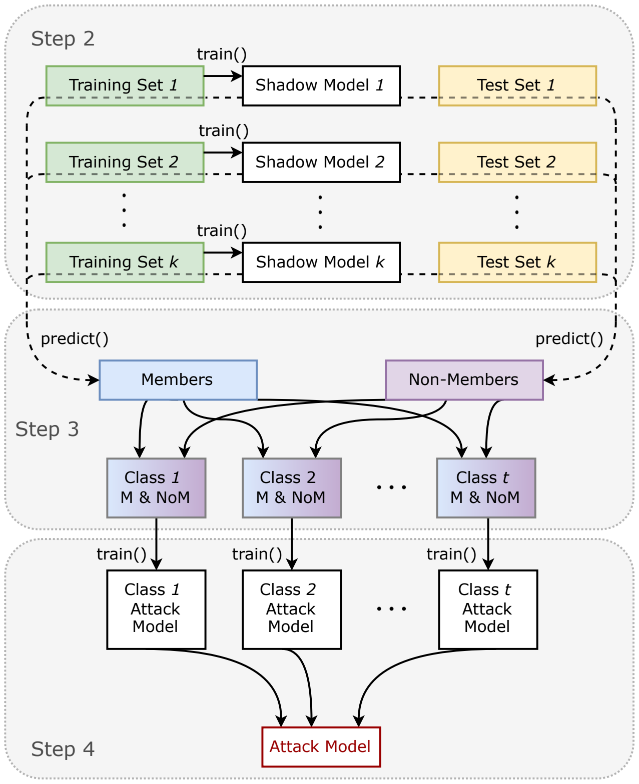


Fig. 1: Overview of our Membership Inference Attack. Steps after shadow data collection are shown. Datasets are represented by colored blocks, and models are represented by white blocks.

2) ***Shadow model training.*** For our membership inference, it is necessary to replicate the behavior of the target model. We achieve this by training a number of shadow models, each with shadow data that was gathered into training sets as described above.
3) ***Attack dataset generation.*** Each shadow model is run on its training and test sets to produce vectors of probabilities, grouped as members and non-members. These vectors are augmented with confidence, prediction correctness, loss, and entropy, inspired by other membership inference attacks [21] to improve performance. The resulting samples are then organized by true class label.
4) ***Attack model training.*** Each attack dataset is used to train an attack model for a single output class of the target model. Our final attack model consists of the collection of all class-specific attack models. When presented with a predicted data point and its true label, it can predict whether it was in the training set of the target model.

A more efficient attack strategy that relaxes the assumption of having access to the underlying distribution of the data has been proposed by Salem et al. [20]. Despite the efficiency and assumption improvements of this approach, its performance is worse in most cases, especially with image datasets. Therefore, we choose not to use this adversary model and its accompanying attack.

Linkability Attack. The linkability attack (LA) aims to discover the origin node of a received chunk [17]. Similar attacks were also devised for federated learning, where the server links the layers of the model with participants [10, 22]. The main assumption of this attack is that the adversary has access to the training sets of its neighbors, and can make use of them to execute its attack. In our experiments, we adopt this assumption but also consider a relaxed version of this assumption where the adversary is granted access to only 10% of the training set fo each neighbor, chosen randomly.

The structure of the linkability attack can be viewed in three steps: parameter patching, loss evaluation, and neighbor-chunk matching, explained below. Figure 2 gives a summary of this process.

1) ***Parameter patching.*** Each neighbor sends the attacker a single chunk from its calculated model parameters. The attacker then patches its own model parameters with the received chunk from that neighbor.
2) ***Loss evaluation.*** As the attacker has access to the training set of its neighbors, it evaluates its patched model parameters in the training sets of all its neighbors. This evaluation results in loss values, which the attacker records with that chunk for use at the end of that round.
3) ***Neighbor-chunk matching.*** The attacker repeats the above steps for all received chunks, then assigns each to the lowest loss value. This is done in two ways: (i) matching each chunk to the training set with the lowest loss [10, 17], or (ii) using the Hungarian algorithm [31], a combinatorial optimization algorithm that is used to compute an optimal one-to-one assignment between the elements of two sets by minimizing the total matching cost. We use these to match each loss value to a chunk, and thus each neighbor to a chunk.

VI. Experimental Setup

In this section, we present the goals and the setup of our experiments. Section VI-A describes the objectives the experiments we conduct, and Section VI-B details the conditions under which our experiments were conducted.

A. Objectives

We design our experiments to evaluate the effectiveness of model chunking as a standalone defense against membership inference and linkability attacks. Our objectives are as follows:

1) For both attacks, we evaluate the privacy impact of chunking, using a comparison between four different chunking methods (static, cyclic, random, and no chunking).
2) For both attacks, we study the effect of full-epoch training on different chunking methods in terms of preserving privacy.
3) For the linkability attack, we evaluate the effectiveness of the Hungarian algorithm for chunk-neighbor matching compared to minimum-loss matching.
4) For the linkability attack, we assess the impact of the relaxed assumption of 10% access to neighbor datasets compared to 100% access.

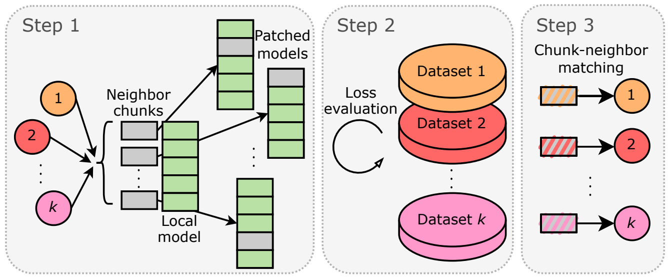


Fig. 2: Overview of our Linkability Attack. The attacker node has k local chunks, but we omit visualizing them for simplicity.

B. Conditions

In our experiments, we use the MNIST and CIFAR-10 datasets. MNIST is an image dataset of 70,000 handwritten digits of size 28×28, and CIFAR-10 is a dataset of 60,000 color images of size 32×32. Both datasets consist of 10 classes, with 7,000 images per class in MNIST and 6,000 per class in CIFAR-10.

The experiments are run using configurations in `DecentralizePy`. For both attacks, we use a LeNet model in the system, a batch size of 8, IID data partitioning, and a random seed of 90.

Our linkability attack is executed on a 16-node, 8-regular graph, where each node is allocated 1% of the dataset and is trained for 10 rounds on every iteration. Each node in the network executes the linkability attack, and we measure the mean attack accuracy.

Our membership attacks use 0.5% of the dataset per node on a 16-node, 3-regular graph. Each node uses 10 shadow models with a shadow dataset of 2000 images allocated to them. Three nodes in the network that are not neighbors execute the attack, and we measure the mean attack success as before. In this attack, we use different rounds of training per iteration on each dataset. On MNIST data, every iteration does 10 rounds of training, whereas on CIFAR-10 data, 50 rounds per iteration is preferred. This is because training on CIFAR-10 typically requires more iterations to reach accuracies comparable to MNIST due to the more varied and complex images of the former [32]. For the same reason, we use 50 epochs to train the shadow models on CIFAR-10 data, but 25 epochs for the MNIST data.

We provide a discussion on the implications of using a LeNet model and IID data in Section VIII.

VII. Evaluation

In this section, we present and evaluate our results, in the aim of answering our objectives as described in VI-A. This is done separately for membership inference and linkability attacks.

A. Membership Inference Attack

In the membership inference experiments, our evaluation metrics are attack precision, recall, F1-Score, and area under the ROC curve (AUC). Precision indicates the ratio of true members

TABLE III: Membership Inference Attack results (%). Lower values indicate better attack resilience. FE denotes full epochs. Highest values per metric are underlined; lowest are boldfaced. Each value includes the standard error of the mean.

Dataset	Chunking	Precision	Recall	F1-Score	AUC	Test Acc.	Val. Acc.
MNIST (FE)	None	51.37 ± 0.24	94.00 ± 1.02	66.43 ± 0.32	53.1 ± 0.79	97.48 ± 0.58	96.87 ± 0.09
	Static	55.24 ± 0.06	98.44 ± 0.87	70.77 ± 0.21	60.31 ± 0.27	93.88 ± 0.17	93.87 ± 0.29
	Cyclic	**51.10 ± 0.20**	**92.67 ± 1.35**	**66.11 ± 0.30**	**52.53 ± 1.41**	97.34 ± 0.07	96.70 ± 0.10
	Random	55.08 ± 0.22	98.89 ± 0.40	70.75 ± 0.12	58.2 ± 0.98	93.87 ± 0.19	93.97 ± 0.35
MNIST (Non-FE)	None	47.66 ± 1.07	**41.78 ± 2.84**	**44.40 ± 1.54**	**47.29 ± 0.32**	96.99 ± 0.10	96.33 ± 0.15
	Static	50.27 ± 0.48	50.56 ± 1.72	50.39 ± 0.95	50.73 ± 0.55	93.83 ± 0.22	93.27 ± 0.55
	Cyclic	**47.17 ± 0.69**	46.89 ± 0.29	47.02 ± 0.41	47.74 ± 0.43	96.60 ± 0.28	95.93 ± 0.52
	Random	50.91 ± 0.89	51.22 ± 5.50	50.83 ± 0.03	51.19 ± 0.74	93.80 ± 0.20	93.23 ± 0.41
CIFAR-10 (FE)	None	82.86 ± 0.45	90.40 ± 3.82	86.40 ± 1.98	89.70 ± 1.40	52.59 ± 0.09	52.90 ± 0.30
	Static	93.22 ± 1.16	85.73 ± 6.25	89.01 ± 2.88	95.54 ± 0.72	37.91 ± 0.64	37.40 ± 0.85
	Cyclic	**82.34 ± 1.38**	86.67 ± 6.81	**84.28 ± 3.85**	**89.11 ± 2.84**	51.62 ± 0.31	52.57 ± 0.34
	Random	94.19 ± 0.52	**82.8 ± 7.17**	87.76 ± 3.83	94.81 ± 0.90	38.28 ± 0.76	37.33 ± 0.67
CIFAR-10 (Non-FE)	None	72.37 ± 3.19	17.47 ± 0.27	28.11 ± 0.11	**67.56 ± 1.77**	54.77 ± 0.07	55.50 ± 0.32
	Static	74.88 ± 9.13	**5.07 ± 0.96**	**9.48 ± 1.75**	70.24 ± 0.38	37.80 ± 1.41	37.20 ± 1.56
	Cyclic	75.00 ± 2.67	21.07 ± 3.27	32.74 ± 4.17	68.45 ± 4.01	52.23 ± 0.31	53.03 ± 0.50
	Random	**67.02 ± 9.18**	6.13 ± 1.62	11.21 ± 2.86	70.33 ± 0.02	37.93 ± 1.41	38.67 ± 1.61

among all data points predicted as members, while recall reflects the proportion of actual members that were correctly identified. The threshold to flag a data point as a member is a predicted membership probability of 0.5. F1-Score (the harmonic mean of precision and recall) is a balance between these two metrics. Finally, AUC captures attack success across all possible thresholds. We also report test and validation accuracies for effective comparison of attack success and model performance.

Table III shows the results of our experiments after 200 training rounds. Insights into how the attack success evolves over the training rounds can be found in Appendix A.

No chunking. The performance of the attack when chunking is not used depends on both the dataset that was used and whether full epochs (FE) were used. When FEs were not used, especially with the MNIST data, chunking proved to be relatively more secure. When FEs were used, no chunking proved again to be one of the more secure options.

Cyclic chunking. It can clearly be seen that cyclic chunking was the most resilient to attacks on almost all metrics when FEs were used. On the contrary, it was one of the worst performers when it came to defending against attacks for the non-FE CIFAR-10 data, and moderately defensive for the non-FE MINST data.

Random chunking. Random chunking was also usually poor for attack resilience. It usually scored close to the higher end for attack accuracies, especially apparent in the non-FE MNIST run. In cases where it shows satisfactory precision, such as non-FE CIFAR-10, this score is counterbalanced by the lack of adequate recall performance of the attack.

Static chunking. Static chunking did not show good performance with FE datasets. Its performance was also poor for the non-FE MNIST dataset, placing itself slightly closer to the scores of the poorer-performing chunking methods. Despite this, it performed relatively well on the non-FE CIFAR-10 dataset, where the attack scored the lowest recall.

It is worth noting that test and attack accuracies of no chunking and cyclic chunking have a strong contrast with the performance of static and random chunking. In other words, whenever the former ones prove to be more resilient to an attack, the latter ones do not, and vice versa.

B. Linkability Attack

Our linkability attacks are run on a 16-node, 8-regular graph. We define attack accuracy as the percentage of chunks that were correctly associated with training datasets of the neighbors. All nodes act as attackers in each experiment, and the attack accuracy is calculated as the mean of all accuracies, as can be seen in Figure 3.

Matching method. Firstly, we evaluate the overall performance of minimum-loss matching compared to Hungarian matching. In all experiments and across all variables, Figure 3 clearly demonstrates that Hungarian matching outperforms minimum-loss matching consistently.

Full epochs. Our experiments also demonstrate the performance of the attack against the usage of full epochs (FE) during training. We see in Figure 3 that the attack success is better when FE are used, especially when matching with the

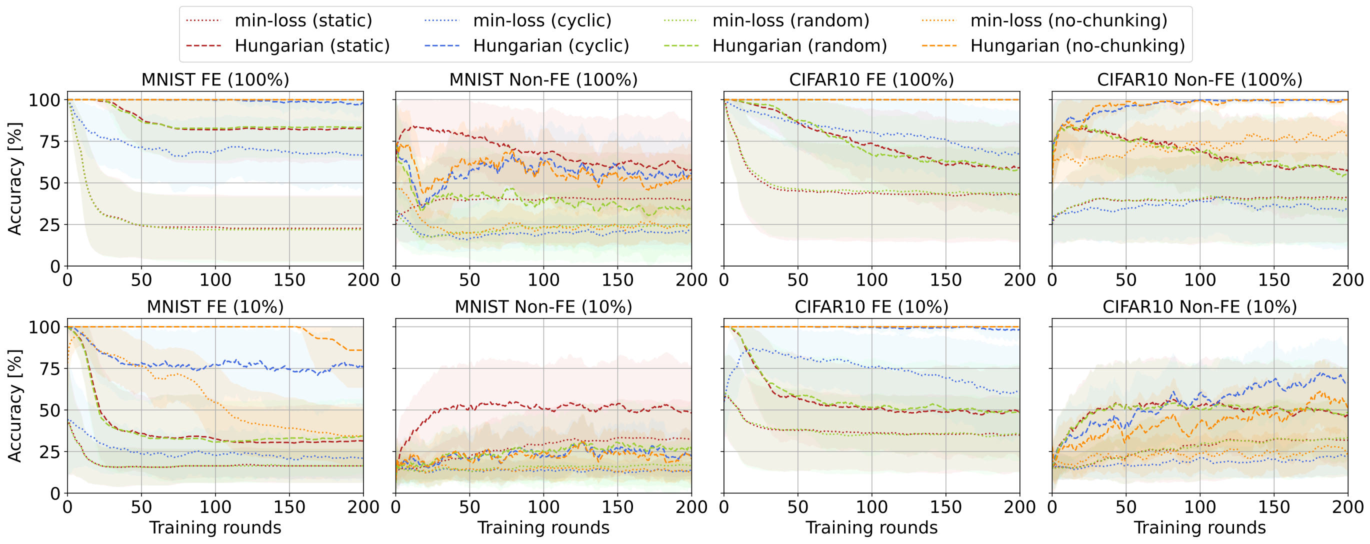


Fig. 3: Comparison of chunking methods during our Linkability Attack. The top row is with access to 100% of data in all neighbor datasets, and the bottom row with 10%. Results are a rolling average with a window size of 10. Results without a rolling average and a table of the rolling averages after 200 iterations can be found in Appendix B.

Hungarian algorithm: for example, the attack performs almost always at 100% accuracy against no chunking, and always above 50% when there is 100% access to neighbor datasets.

Chunking methods. We observe that attack accuracy is almost always higher when chunking is not used. The only exception to this is the MNIST non-FE case, where it is highly outperformed by static chunking. This case is also special because it is the only case where the performance of static and random chunking differ significantly: attack accuracy is around 50% with static chunking, and slightly above 25% with random chunking (using Hungarian matching). In all other cases, we observe that static and random chunking perform almost identically.

However, this identical performance of the attack against static and random chunking is always worse than the attack success against cyclic chunking. There is the exception of the MNIST non-FE case again, where cyclic chunking performs similarly to random and no chunking. We also note that in many cases, cyclic and no chunking perform similarly.

Dataset access. The top row in Figure 3 shows the results for 100% access to the dataset of each neighbor, and the bottom row for 10% access. It is clear that the attack success drops significantly, especially in the non-FE cases. In the FE experiments with CIFAR-10, the attack performs almost as good as it did with 100% of the data, whereas with MNIST there is a noticeable drop in attack accuracy for all chunking methods: even with no chunking whose attack accuracy drops after training round 150.

VIII. Discussion

In this section, we put our results into context, exploring the reasons behind them. We also discuss the limitations of our research when it comes to our attack implementations and datasets.

A. *Interpretation of Results*

There are different implications of our results when it comes to membership inference attacks (MIA) and linkability attacks (LA). We discuss these below.

Chunking as a defense in MIA. Our results indicate that any chunking method performs overall worse than non-chunked model exchange when it comes to membership inference. The reason for this is because when chunking is used, only part of the local model is updated: hence the model leaks more information about its internal state. Conversely, the local model is heavily influenced by neighbors when chunking is not used, diluting membership inference signals. Therefore, chunking alone is not a viable defense against MIA.

Chunking as a defense in LA. On the contrary, chunking improves resilience against linkability attacks, especially with static and random chunking. This is for a similar reason as mentioned above: the less information shared about the model, the harder it is to link that result to its origin. As such, chunking could be a way to improve privacy against linkability, especially when full epochs (FE) are used.

Training without FE as mitigation. Our results clearly demonstrate that using FE during training significantly increases the vulnerability of the system to attacks, in both membership inference and linkability, and in all methods of chunking. This is because when full epochs are used, each chunk contains much more information about the model. This directly improves attack performance in linkability. In membership inference, the higher information content of the chunks is overshadowed by the model remembering more of its own data with FE, resulting in greater attack success.

Therefore, avoiding FE can be a mitigation against both attacks. In Table III, we already see that both FE and non-FE cases reach similar accuracies in each respective dataset.

Nonetheless, in order not to compromise model performance, precautions such as training for more iterations or increasing the rounds of learning on each iteration can be taken.

Identical performance of some chunking methods. We observe that static and random chunking display almost identical results, in both attacks. The same is also true for cyclic chunking and no chunking. In order to explain this, we have to take into account the information that is embedded in a chunk, and which chunks get shared in each training round. For example, cyclic chunking shares all of its chunks with a given neighbor over a number of rounds. This is a similar behavior to no chunking, since this shares all of its chunks with a given neighbor over a single round.

When it comes to static chunking, each neighbor sends the same chunk to the same neighbor: however, different neighbors might send different chunks, which means that the attacker still has a good view of the whole system. This is similar to random chunking in the sense that any neighbor can send any chunk, instead of any neighbor sending one chunk. Overall, the information that the adversary can infer about the system emerges to be similar in both cases.

This behavior also affects the model accuracy, where we see in Table III that no chunking and cyclic chunking achieved higher model accuracies.

Impact of data homogeneity on LA under static chunking. Figure 3 shows that the attack performs best under static chunking with non-FE MNIST data, particularly with 10% neighbor dataset access. Since MNIST data is highly homogeneous, any given chunk produces very similar loss values across neighbors. Static chunking compares the same chunk from each neighbor, amplifying the subtle loss differences. Random and cyclic chunking shuffle the chunks, and full model exchange dilutes these signals because of the homogeneity of MNIST, reducing attack accuracy.

B. Limitations

There are certain limitations of the datasets that were used in the experiments. Specifically for the membership inference attack, the more classes of the data, the better the performance of the attacker [18]. This is because a higher number of classes produces more signals about the internal state of the model, hence the attacker has more grounds to make a decision on membership. The datasets used for our experiments, MNIST and CIFAR-10, only have 10 classes, but a more comprehensive evaluation would involve datasets with more classes.

Furthermore, our overall results indicate that membership inference attacks on the MNIST dataset performed particularly poorly. Similarly, the linkability attack also performed worse on this dataset. The reason for this is the lack of randomness of each class in MNIST. The images are black and white, and have little texture. Therefore, it becomes much harder to distinguish whether a given data point is a member of the model, regardless of the chunking method used [18]. A more realistic classification task would likely involve more randomness in its data, hence would be more vulnerable to membership inference.

Our model accuracies when training with the CIFAR-10 dataset were limited to slightly above 50% at best. In order to overcome this, the experiments could be run with more data per node and for more training rounds.

When it comes to the design of our attacks, our membership inference attack uses a disjoint subset of the whole dataset for training its shadow models. A more realistic scenario would be for the attacker to derive the data through model- or statistic-based synthesis [18]. Furthermore, we use IID data in experiments. Non-IID data will contribute positively to attack performance [22, 33], and would be a more realistic scenario in decentralized learning.

We conducted all our experiments based on the LeNet model, which lacks depth when it comes to classification tasks when compared with other CNNs. Since it was designed specifically for handwritten digit recognition, its use is acceptable for MNIST data. However, a more complex model could have been a more realistic and appropriate choice for CIFAR-10. Nonetheless, using the same model for both datasets enabled us to do cross-dataset comparisons.

IX. Conclusion

This research investigated the effectiveness of model chunking as a standalone privacy-preserving mechanism in decentralized learning systems. While prior frameworks such as *Shatter* combine chunking with virtualization, we isolate chunking to evaluate its privacy guarantees independently and under various conditions.

Our results demonstrate that model chunking offers limited protection against some privacy attacks, and only under certain circumstances. Static and random chunking, in particular, were shown to significantly reduce the success of linkability attacks when full epochs were used. However, full epochs diminish the privacy of the system, and we advise against its use from a privacy perspective. When it comes to membership inference, chunking is not a reliable solution because its partial model sharing nature leads to local models exposing more information about themselves. In conclusion, we find that chunking alone does not eliminate privacy risks.

For linkability attacks, we demonstrate that the Hungarian matching algorithm is more effective than minimum-loss matching for a linkability attack. We also show that the relaxed assumption of access to only a portion of the neighbor datasets (10%) still results in successful linkability in certain contexts.

Future work may explore this domain in two main directions. Firstly, our work can be expanded to include experiments with other datasets (particularly those with a larger number of classes), non-IID data, different target models, and larger networks. These conditions would assist in reflecting real-world scenarios. Furthermore, the evaluation of our attacks and others would benefit from an analysis of the information density and similarities between chunks. Secondly, privacy attacks discussed in Section III can be explored, since attribute inference, gradient inversion and reconstruction remain unexplored in decentralized learning systems that exchange chunked models.

X. Responsible Research

We conducted this research in accordance with the principles outlined in the Netherlands Code of Conduct for Research Integrity [34]. Below, we reflect on the ethical aspects of our work, the measures taken to ensure integrity, reproducibility, and the use of LLMs and AI tools.

Ethical considerations. Our experiments focus on evaluating privacy vulnerabilities in decentralized learning systems. These experiments rely completely on MNIST and CIFAR-10 datasets, which are publicly available and standard for this domain. Therefore, they do not raise ethical concerns of consent for collection or usage. Due to the necessity for high computation power, external resources were used to run experiments. This does not present ethical concerns with publicly available datasets, but is a concern if experiments are run on sensitive datasets. Furthermore, we acknowledge that the domain of this work is privacy attacks, which raises concerns of exploitation and misuse. However, an analysis of vulnerabilities in DL systems is necessary to understand the limitations of current defenses, and to develop more robust privacy-preserving approaches.

Reproducibility. Our implementation builds on the open-source `DecentralizePy` framework, which we extend with model chunking and privacy attacks. Our implementations, including the additions on top of this framework, evaluation tools, and configuration files for the experiments are made available in a public repository in order to make our results reproducible. Furthermore, the procedures for aggregating our results and generating figures are documented in this repository, to support the reproduction of and further work on our results.

Integrity. Throughout this research, we were cautious about presenting results transparently and avoiding overstating the results of our experiments. We clearly state the limitations of our work in Section VIII. When interpreting results, we made efforts to explain unexpected patterns rather than omitting them.

Use of LLMs and AI tools. We made limited and transparent use of LLMs and AI tools during the research process. Specifically, we used these tools to understand the structure and operation of the `DecentralizePy` and *Shatter* frameworks, assist in the integration of privacy attacks to `DecentralizePy`, and assist in writing code to plot the graphs of our experimental results. All consultations to these tools were reviewed by the authors, and corrected where necessary.

References

[1] 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.

[2] B. Cox, L. Y. Chen, and J. Decouchant. "Aergia: leveraging heterogeneity in federated learning systems". In: *Proceedings of the 23rd ACM/IFIP International Middleware Conference*. 2022, pp. 107–120.

[3] Y. Zuo, B. Cox, L. Y. Chen, and J. Decouchant. "Spyker: Asynchronous Multi-Server Federated Learning for Geo-Distributed Clients". In: *Proceedings of the 25th International Middleware Conference*. 2024, pp. 367–378.

[4] 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]`.

[5] I. Hegedundefineds, G. Danner, and M. Jelasity. "Gossip Learning as a Decentralized Alternative to Federated Learning". In: *Distributed Applications and Interoperable Systems: 19th IFIP WG 6.1 International Conference, DAIS 2019, Held as Part of the 14th International Federated Conference on Distributed Computing Techniques, DisCoTec 2019, Kongens Lyngby, Denmark, June 17–21, 2019, Proceedings*. Kongens Lyngby, Denmark: Springer-Verlag, 2019, pp. 74–90.

[6] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava. "A survey on security and privacy of federated learning". In: *Future Generation Computer Systems* 115 (2021), pp. 619–640.

[7] 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.

[8] R. Wang, X. Wang, H. Chen, J. Decouchant, S. Picek, N. Laoutaris, and K. Liang. "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.

[9] 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.

[10] 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.

[11] H.-P. Cheng, P. Yu, H. Hu, S. Zawad, F. Yan, S. Li, H. Li, and Y. Chen. "Towards Decentralized Deep Learning with Differential Privacy". In: *Cloud Computing – CLOUD 2019*. Cham: Springer International Publishing, 2019, pp. 130–145.

[12] S. Biswas, D. Frey, R. Gaudel, A.-M. Kermarrec, D. Lerévérend, R. Pires, R. Sharma, and F. Taïani. *Low-Cost Privacy-Preserving Decentralized Learning*. 2025. arXiv: 2403.11795 `[cs.LG]`.

[13] E. Cyffers, M. Even, A. Bellet, and L. Massoulié. *Muffliato: Peer-to-Peer Privacy Amplification for Decentralized Optimization and Averaging*. 2024. arXiv: 2206.05091 `[cs.CR]`.

[14] D. Alistarh, T. Hoefler, M. Johansson, S. Khirirat, N. Konstantinov, and C. Renggli. *The Convergence of Sparsified Gradient Methods*. 2018. arXiv: 1809.10505 `[cs.LG]`.

[15] R. Shokri and V. Shmatikov. "Privacy-Preserving Deep Learning". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. CCS '15. Denver, Colorado, USA: Association for Computing Machinery, 2015, pp. 1310–1321.

[16] D. Pasquini, M. Raynal, and C. Troncoso. *On the (In)security of Peer-to-Peer Decentralized Machine Learning*. 2023. arXiv: 2205.08443 `[cs.CR]`.

[17] S. Biswas, M. Even, A.-M. Kermarrec, L. Massoulié, R. Pires, R. Sharma, and M. de Vos. "Noiseless Privacy-Preserving Decentralized Learning". In: *Proceedings on Privacy Enhancing Technologies* 2025.1 (Jan. 2025), pp. 824–844.

[18] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. *Membership Inference Attacks against Machine Learning Models*. 2017. arXiv: 1610.05820 [cs.CR].

[19] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer. *Membership Inference Attacks From First Principles*. 2022. arXiv: 2112.03570 [cs.CR].

[20] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes. *ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models*. 2018. arXiv: 1806.01246 [cs.CR].

[21] H. Hu, Z. Salcic, L. Sun, G. Dobbie, P. S. Yu, and X. Zhang. "Membership inference attacks on machine learning: A survey". In: *ACM Computing Surveys (CSUR)* 54.11s (2022), pp. 1–37.

[22] H. Hu, X. Zhang, Z. Salcic, L. Sun, K.-K. R. Choo, and G. Dobbie. "Source Inference Attacks: Beyond Membership Inference Attacks in Federated Learning". In: *IEEE Transactions on Dependable and Secure Computing* 21.4 (2024), pp. 3012–3029.

[23] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. "Practical Secure Aggregation for Privacy-Preserving Machine Learning". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17. Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 1175–1191.

[24] L. Zhu, Z. Liu, and S. Han. *Deep Leakage from Gradients*. 2019. arXiv: 1906.08935 [cs.LG].

[25] A. E. Mrini, E. Cyffers, and A. Bellet. *Privacy Attacks in Decentralized Learning*. 2024. arXiv: 2402.10001 [cs.LG].

[26] B. Z. H. Zhao, A. Agrawal, C. Coburn, H. J. Asghar, R. Bhaskar, M. A. Kaafar, D. Webb, and P. Dickinson. *On the (In)Feasibility of Attribute Inference Attacks on Machine Learning Models*. 2021. arXiv: 2103.07101 [cs.LG].

[27] C. Ji, S. Maag, R. Heusdens, and Q. Li. *Re-Evaluating Privacy in Centralized and Decentralized Learning: An Information-Theoretical and Empirical Study*. 2024. arXiv: 2409.14261 [cs.CR].

[28] B. Hui, Y. Yang, H. Yuan, P. Burlina, N. Z. Gong, and Y. Cao. "Practical Blind Membership Inference Attack via Differential Comparisons". In: *Proceedings 2021 Network and Distributed System Security Symposium*. NDSS 2021. Internet Society, 2021.

[29] A. Dhasade, A.-M. Kermarrec, R. Pires, R. Sharma, and M. Vujasinovic. "Decentralized Learning Made Easy with DecentralizePy". In: *Proceedings of the 3rd Workshop on Machine Learning and Systems*. EuroMLSys '23. Rome, Italy: Association for Computing Machinery, 2023, pp. 34–41.

[30] M. de Vos, S. Farhadkhani, R. Guerraoui, A.-M. Kermarrec, R. Pires, and R. Sharma. *Epidemic Learning: Boosting Decentralized Learning with Randomized Communication*. 2023. arXiv: 2310.01972 [cs.LG].

[31] H. W. Kuhn. "The Hungarian method for the assignment problem". In: *Naval Research Logistics Quarterly* 2.1-2 (1955). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.3800020109, pp. 83–97.

[32] A. Gautam, Y. Lohumi, and D. Gangodkar. "Achieving Near-Perfect Accuracy in CIFAR-10 Classification". In: *2024 Second International Conference on Advances in Information Technology (ICAIT)*. Vol. 1. 2024, pp. 1–6.

[33] L. Bai, H. Hu, Q. Ye, H. Li, L. Wang, and J. Xu. "Membership Inference Attacks and Defenses in Federated Learning: A Survey". In: *ACM Comput. Surv.* 57.4 (Dec. 2024).

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

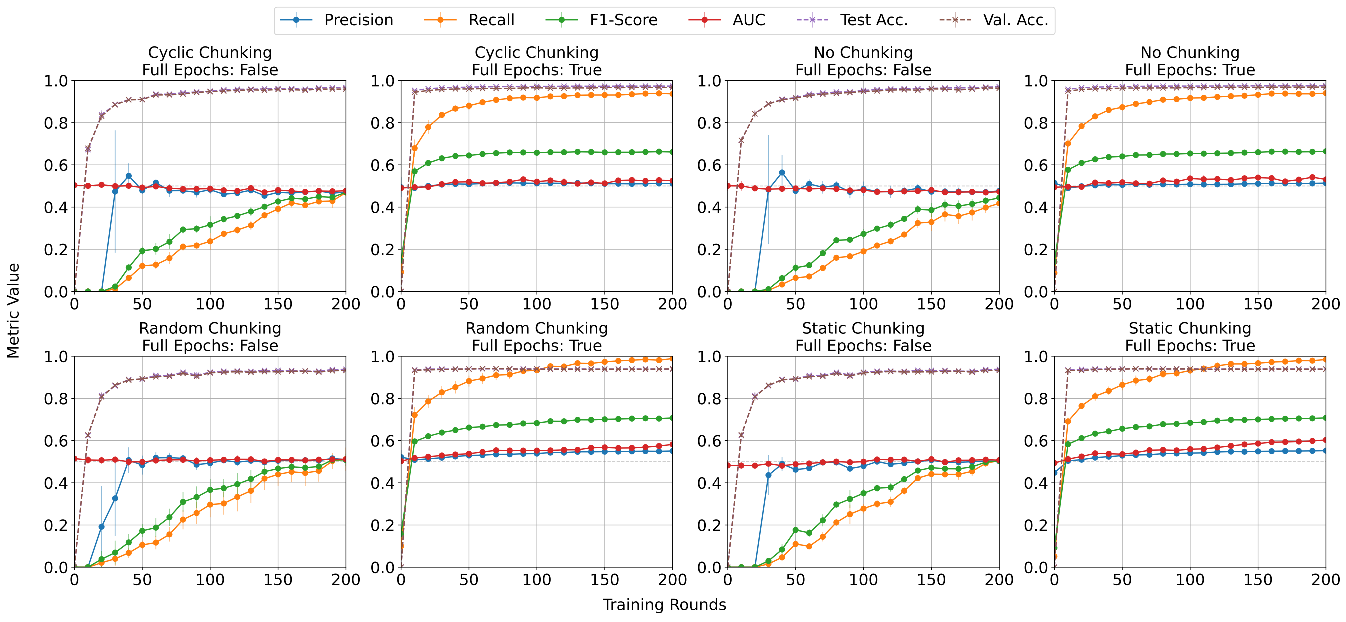
Appendix A
Membership Inference over Training Rounds

This section presents the attack and model metrics for the membership inference attack over all training rounds. The attack is executed every 10 iterations. Results can be seen in Figure 4.

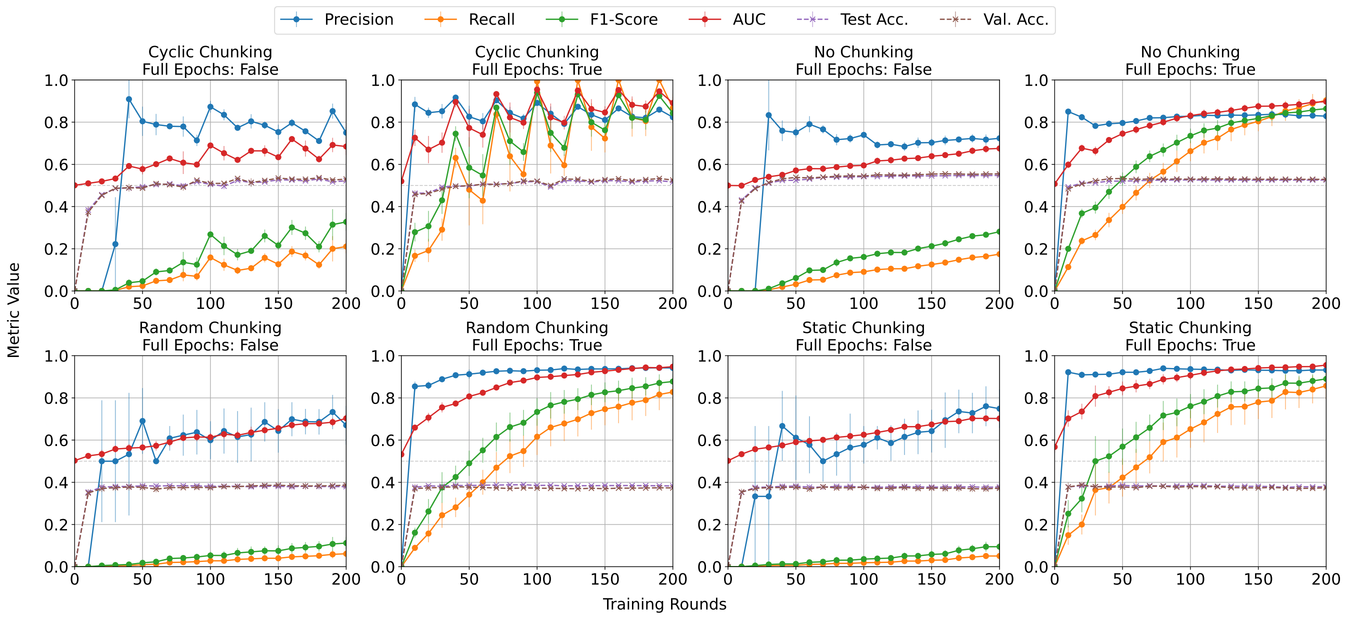
Overall, membership inference improves over training rounds. In the case of cyclic chunking on CIFAR-10 data, inference accuracies display fluctuating behavior. This is due to the nature of cyclic chunking: some chunks do not affect the local model as much as others, resulting in higher inference accuracies, and vice versa.

Appendix B
Linkability Attack Results

A table of the results at the end of 200 iterations can be found in Tables IV and V. Furthermore, Figure 5 shows the results without a rolling average.



(a) MNIST



(b) CIFAR-10

Fig. 4: Membership inference over training rounds.

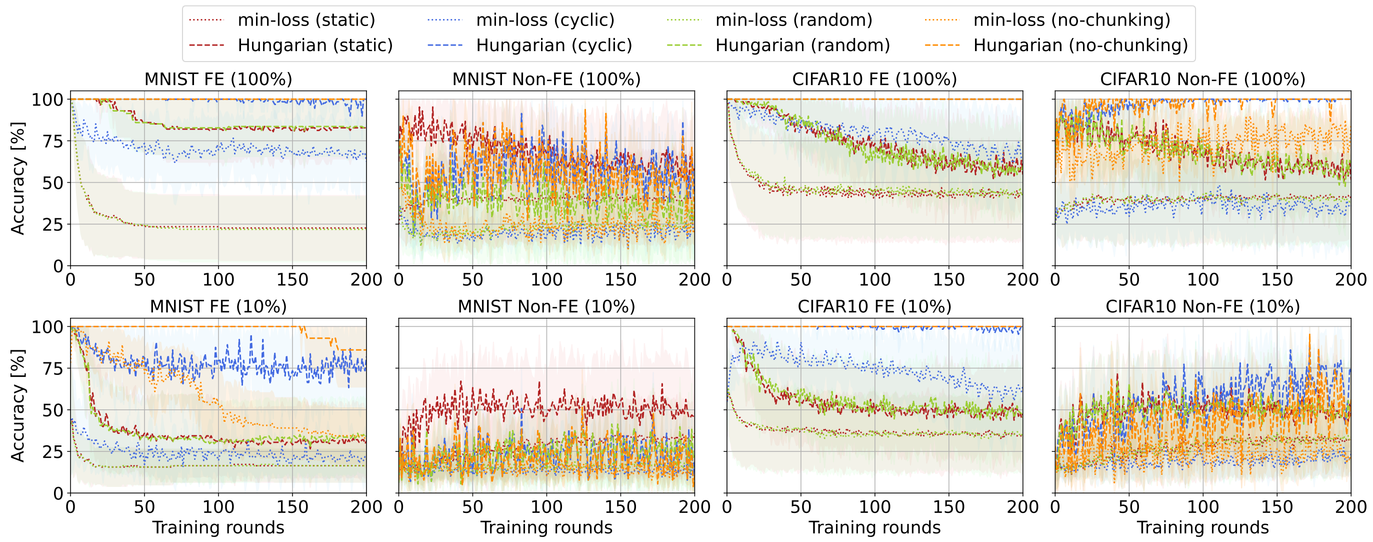


Fig. 5: Comparison of chunking methods during our Linkability Attack. The top row is with access to 100% of data in all neighbor datasets, and the bottom row is access to 10% of the data in each neighbor dataset, chosen randomly. Results for each chunking method are averages of 16 nodes.

TABLE IV: Linkability Attack accuracy (%) on MNIST across access levels, epoch settings, and chunking methods.

Access	Epochs	Matching	Static	Cyclic	Random	No Chunking
100%	FE	Min-Loss	22.66	66.09	21.88	**100.00**
		Hungarian	82.81	97.97	83.44	**100.00**
100%	Non-FE	Min-Loss	**40.00**	21.88	25.00	25.16
		Hungarian	**57.81**	55.70	33.28	51.48
10%	FE	Min-Loss	16.41	20.94	16.41	**34.06**
		Hungarian	31.41	76.33	34.38	**85.94**
10%	Non-FE	Min-Loss	**32.97**	13.75	16.25	12.03
		Hungarian	**48.20**	22.27	26.25	20.39

TABLE V: Linkability Attack accuracy (%) on CIFAR-10 across access levels, epoch settings, and chunking methods.

Access	Epochs	Matching	Static	Cyclic	Random	No Chunking
100%	FE	Min-Loss	42.97	66.80	43.59	**100.00**
		Hungarian	59.14	**100.00**	57.81	**100.00**
100%	Non-FE	Min-Loss	41.41	34.38	39.61	**78.36**
		Hungarian	57.11	**100.00**	55.94	**100.00**
10%	FE	Min-Loss	34.92	61.56	35.47	**100.00**
		Hungarian	48.91	98.36	48.44	**100.00**
10%	Non-FE	Min-Loss	31.80	22.11	**33.28**	22.89
		Hungarian	46.56	**66.41**	48.52	51.09