Topology-Aware Privacy Amplification in Decentralized Learning:

A Hybrid Chunking and Differential Privacy Approach Against Membership Inference Attacks

Niels Tomassen

Delft University of Technology

TU Delft

Topology-Aware Privacy Amplification in Decentralized Learning:

A Hybrid Chunking and Differential Privacy Approach Against Membership Inference Attacks

by

Niels Tomassen

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Monday June 15, 2026 at 15:00 PM.

Student number:	5101190	
Project duration:	November 17, 2025 – June 15, 2026	
Thesis committee:	Georgios Smaragdakis,	Chair / Thesis Advisor
	Jérémie Decouchant,	Core Member 2
	Lilika Markatou,	Core Member 3 / Daily Supervisor
	Andreas Athanasiou,	Advisor / Daily Co-Supervisor

Cover:	Canadarm 2 Robotic Arm Grapples SpaceX Dragon by NASA under CC BY-NC 2.0 (Modified)
Style:	TU Delft Report Style, with modifications by Daan Zwaneveld

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

Decentralized learning allows data owners to collaboratively train machine learning models without relying on a central server, making it attractive for privacy sensitive and distributed environments. However, despite keeping data on-premises, model updates exchanged between peers can still leak private information about the underlying dataset. In particular, Membership Inference Attacks (MIAs) allow an adversary to determine whether a specific data sample was used in the training process, posing a significant privacy risk. Differential privacy (DP) is a common defense against this leakage, which injects carefully calibrated noise into model updates, but this inevitably hurts utility. Recent studies have shown that chunking, where model updates are split into chunks and only a subset of chunks is shared to neighboring nodes, can also mitigate leakage. Existing approaches include topology-aware chunking, where the number of chunks for a specific node is dependent on the communication topology, and topology-independent fixed-K chunking, where a fixed number of chunks K is used for all nodes. However, it remains unclear how the underlying topology influences the effectiveness of such defenses.

This thesis investigates whether topology-aware chunking can improve the privacy-utility tradeoff compared to topology-independent chunking strategies. We study decentralized image classification on CIFAR-100 across several communication topologies, including ring, star, grid, fully connected, d-regular, and Erdős-Rényi graphs. Privacy leakage is measured through the accuracy of the MIA (Area Under the Curve), while utility is measured by global test accuracy. The results show that the effectiveness of topology-aware chunking is strongly influenced by the underlying communication graph. Without defenses, MIA AUC remains high across all graph families (around 0.97-0.99). Topology-aware chunking reduces leakage significantly in dense graphs, for example, lowering AUC to 0.61 in the fully connected graph, but introduces uneven protection for sparse or heterogeneous topologies, where low-degree nodes remain vulnerable.

Compared to topology-aware chunking, topology-independent fixed-K chunking proves to be a stronger and more uniform graph-independent baseline. It often achieves equal or better privacy-utility tradeoffs, especially in utility-focused settings. To address the key limitation of topology-aware chunking, we propose ChunkDP, a defense that combines topology-aware chunking with degree-scaled DP noise. ChunkDP improves over DP-only by recovering a portion of the lost accuracy while keeping leakage close to random guessing performance (AUC 0.53). We show that ChunkDP can outperform fixed-K chunking in balanced privacy-utility settings.

Overall, the results show that topology-awareness alone does not guarantee a better privacy-utility tradeoff. Its effectiveness depends on graph density, node degree, and the desired privacy-utility balance. Fixed-K remains a robust defense, while the topology-aware ChunkDP can be useful in balanced privacy-utility scenarios.

Contents

List of Figures

Nomenclature

Abbreviations

Abbreviation	Definition
AUC	Area under the ROC curve
CNN	Convolutional Neural Network
D-PSGD	Decentralized Parallel Stochastic Gradient Descent
DL	Decentralized Learning
DP	Differential Privacy
DP-SGD	Differentially Private Stochastic Gradient Descent
ER	Erdős–Rényi
FL	Federated Learning
IID	Independent and Identically Distributed
MIA	Membership Inference Attack
SGD	Stochastic Gradient Descent

Symbols

Symbol	Definition
a	Mean maximum MIA AUC
C	Gradient clipping threshold
d_i	Degree of node i
K	Number of chunks used in fixed-K chunking
p	Edge probability in the Erdős–Rényi graph
r	Privacy risk score, defined as $r = \max\{0,\ 2a - 1\}$
S	Number of chunks sent to each node
S_λ	Privacy–utility score
u	Mean top-5 test accuracy
W	Communication weight matrix
w_{ij}	Communication weight between nodes i and j
$x_i^{(t)}$	Model parameters of node i at communication round t
$x_i^{r,\tau}$	Model parameters of node i at round r after τ local SGD steps
x_i^+	Aggregated model parameters of node i after communication
α	Dirichlet label-skew parameter
β	Mixing parameter controlling aggregation strength
λ	Trade-off parameter of the privacy–utility score
$\mathcal{N}_i$	Set of neighboring nodes of node i
σ	Global noise multiplier
σ_i	Noise multiplier for node i in the hybrid mechanism
σ_0	Noise multiplier for degree-1 nodes in the hybrid mechanism
$\tilde{w}_{ij}$	Normalized communication weight between nodes i and j

1

Introduction

Federated Learning (FL) [1] is a powerful paradigm that allows the training of machine learning models on distributed datasets, without centralizing them. Because it allows data owners to keep their data on-premises, it has been widely adopted in industry and has attracted substantial academic interest [2], [3]. For example, Google uses FL for mobile keyboard personalization, while healthcare institutions use collaborative learning to train models without sharing patient records. In these frameworks, multiple nodes collaboratively train a shared global model under the coordination of a central server that aggregates the model updates of each participant.

FL was initially thought to be privacy-preserving, as the data of each node never leaves the local device. However, it has been shown that the shared model updates (gradients or parameters), that are communicated with the central server, carry information about the data itself. This enables various inference and reconstruction attacks, including Membership Inference Attacks (MIAs) [4], Attribute Inference Attacks (AIA) [5] or even data reconstruction attacks [6], [7]. In a MIA, for example, an adversary attempts to infer if a specific sample was used in the training process of a model, potentially revealing sensitive information about individuals participating in the learning process. These attacks pose significant privacy risks, especially when updates are shared with untrusted entities.

To mitigate this, several privacy-preserving mechanisms have been proposed. One common approach is Differential Privacy (DP), in which data owners inject noise into their model updates before sharing, a technique known as model obfuscation. For example, a node may add Gaussian noise to its gradients before sharing them with neighboring nodes, making it more difficult for an adversary to infer information about the local training data. While this can reduce leakage, it inevitably leads to a decrease in utility, as the noise will affect the final accuracy. This tradeoff between privacy and utility is one of the main challenges in collaborative learning: adding more noise improves privacy, but reduces accuracy of the model.

Another class of defenses focuses on limiting the exposure of model information during communication. In these so-called chunk-based approaches, nodes only share a subset of model parameters during each communication round. For instance, instead of sharing the full model update, a node may split their model into 10 parts (chunks) and share only a single part with its neighbors. By exposing only a part of the model at a time, the amount of information that a neighbor can observe from a single interaction is reduced. DP and chunk-based defenses represent two contrasting approaches to privacy preservation. While DP obfuscates the full communicated update through noise injection, chunking restricts communication to only partial model updates.

The major limitation of FL is the strong trust assumption on the central server. If the server is compromised or malicious, private and potentially sensitive information can be inferred from the received updates. Furthermore, centralized architectures can create communication bottlenecks and limit scalability, as all traffic must pass through a single central server [8].

These are the problems that Decentralized Learning (DL) [8] tries to solve. In DL, the central server is eliminated, and the nodes both communicate and aggregate the model updates in a peer-to-peer manner, collaboratively training a global model. Although the trust assumption is relaxed in DL, privacy risks remain, as now, instead of the centralized server, malicious peers may attempt to infer or reconstruct data based on the received model updates. The difference between the centralized communication structure of FL and the peer-to-peer communication structure of DL is shown in Figure 1.1.

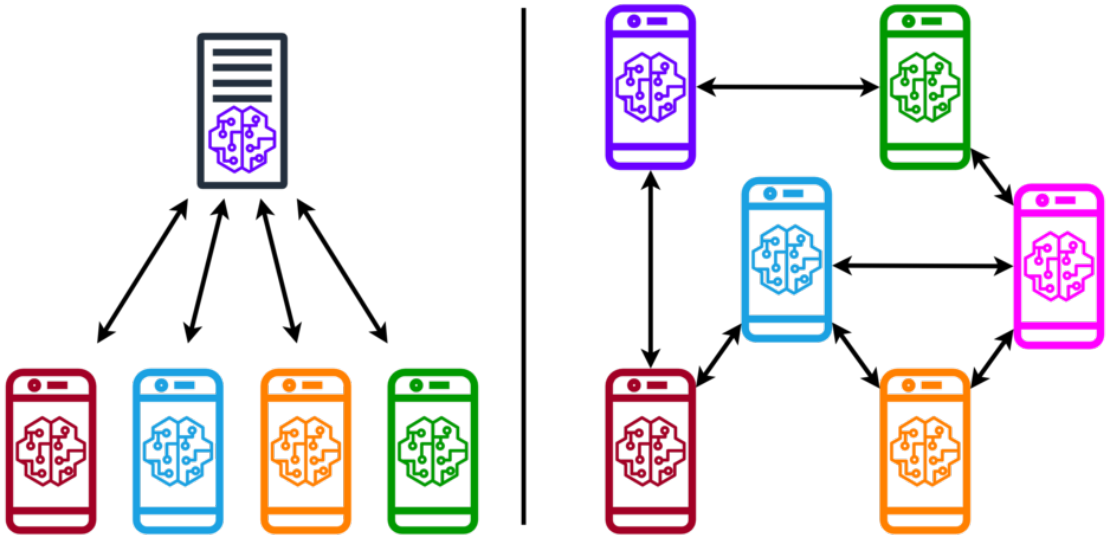


Figure 1.1: Communication structure in FL (left) and DL (right). Source: MarcT0K [9].

The underlying communication topology of DL plays an important role in both learning efficiency and privacy risks. Dense topologies (e.g. fully connected networks) converge to an optimal model faster but can increase data leakage due to frequent interactions. Whereas sparse topologies (e.g. rings or grids) reduce communication overhead but increase risk for low-degree nodes that may share their updates with only one or two other nodes, making it easier for a compromised neighbor to correlate updates to that client. Thus, the network topology directly impacts the model's convergence rate and its vulnerability to adversarial attacks.

Despite extensive research on privacy mechanisms such as Differential Privacy [10], [11], Secure Aggregation [12], [13], and Randomized Communication [14], [15], the effect of network topology on privacy leakage in decentralized learning remains relatively underexplored. Understanding how specific properties of a communication network, such as degree distribution, amplify or mitigate privacy risks is crucial to design topology-aware privacy amplification strategies. This thesis aims to systematically study the relationship between network topology and MIA attack vulnerability and evaluate mechanisms that exploit the network structure to improve the privacy-utility tradeoff. However, the extent to which topology-aware mechanisms can achieve this improvement remains unclear and requires empirical validation.

In particular, chunk-based defenses can be implemented in different ways. Many existing approaches do not account for structural differences between nodes, instead applying a fixed partitioning of the model regardless of node connectivity. As a result, poorly-connected nodes transmit the same limited portion of the model as well-connected nodes, but with fewer communication opportunities. This can reduce their influence on the global model, as their updates propagate more slowly through the network.

Topology-aware chunking variants address this issue by adapting the number of chunks to the node degree. By allowing low-degree nodes to share a larger fraction of their local model, these approaches aim to improve information flow for poorly-connected nodes. However, increasing the information shared by low-degree nodes also increases their vulnerability to privacy attacks,

resulting in uneven protection across nodes.

To address this limitation, this thesis proposes *ChunkDP*, a hybrid mechanism that combines topology-aware chunking with degree-scaled differential privacy noise, providing stronger protection for low-degree nodes to compensate for the uneven protection introduced by topology-aware chunking.

Research Questions

To evaluate the effectiveness of this approach, this thesis investigates the following research question:

> *In decentralized learning, how does communication topology influence vulnerability to membership inference attacks and model utility, and can topology-aware defense mechanisms improve the privacy-utility tradeoff compared to topology-agnostic approaches?*

This question aims to bridge the gap between existing privacy mechanisms and the structural properties of decentralized communication networks, by systematically analyzing how network topology interacts with privacy defenses under adversarial settings.

To answer this research question in a structured manner, it has been decomposed into three subquestions. The first subquestion examines the influence of communication topology on privacy leakage and model utility in decentralized learning. The second compares the influence and effectiveness of topology-aware chunking to topology-independent chunking strategies. Finally, the third evaluates whether the proposed ChunkDP mechanism improves the overall privacy-utility tradeoff compared to existing approaches.

The following subquestions are addressed in this thesis:

1. How do different communication topologies affect membership inference vulnerability and model utility in decentralized learning?
2. How does topology-aware chunking influence the privacy-utility tradeoff compared to topology-agnostic fixed-K chunking?
3. Can ChunkDP improve the privacy-utility tradeoff compared to differential privacy-only and topology-agnostic chunking approaches?

Main Contributions

To address these research questions, this thesis makes the following contributions:

- We systematically analyze how communication topology influences membership inference vulnerability and model utility in decentralized learning (Section 5.1). We show that communication topology has a strong influence on utility but only a limited effect on privacy leakage when no defenses are applied. However, topology becomes a dominant factor when topology-aware chunking is used, creating large disparities in privacy protection across nodes. In particular, we show that well-connected nodes receive substantially greater privacy benefits than poorly-connected nodes.
- Based on these findings (Section 5.1), we identify a key limitation of topology-aware chunking: privacy protection is distributed unevenly across nodes, with poorly-connected nodes remaining highly vulnerable.
- Having identified this limitation, we propose *ChunkDP* (Section 4.4), a hybrid defense mechanism that combines topology-aware chunking with degree-scaled differential privacy noise. ChunkDP adapts the amount of injected noise to node connectivity: poorly-connected nodes receive stronger noise for additional protection, while well-connected nodes receive less noise to preserve utility.
- We evaluate ChunkDP against DP-only and topology-agnostic fixed-K chunking across multiple communication topologies and privacy-utility preference scenarios (Section 5.2 and 5.3). We show that fixed-K chunking is a strong graph-independent baseline that

consistently outperforms topology-aware chunking in isolation, while ChunkDP can achieve more favorable privacy-utility tradeoffs in sparse communication graphs under balanced privacy-utility settings.

2

Background

This section introduces the foundational concepts needed to understand the rest of the thesis: decentralized learning and consensus, membership inference attacks, differential privacy and the idea of topology aware obfuscation.

2.1. Decentralized Learning and Consensus

Federated learning is a distributed learning paradigm in which multiple nodes collaboratively train a machine learning model under the coordination of a central server. This structure allows nodes to keep their data on-premise and collaborate with other nodes to train a shared global model. In this setting, the server receives model updates from participating nodes and aggregates these before distributing the updated global model. However, using a central server to aggregate the updates can introduce limitations such as communication bottlenecks, single points of failure, and potential privacy risks. Decentralized learning addresses these limitations by removing the central server and allowing nodes to collaboratively train a shared global model through peer-to-peer communication.

In decentralized learning, n nodes collaborate to train a shared global model without the help of a central server. The communication pattern is defined by an undirected graph $G = (V, E)$ with node set $V = \{0, \dots, n-1\}$ and edge set E. Each node i has a set of neighbors $\mathcal{N}_i = \{j \mid (i, j) \in E\}$ and degree $d_i = |\mathcal{N}_i|$. During communication, only model parameters (or updates) are exchanged between neighbors meaning that the underlying training data remains on-device.

A typical training round consists of the following steps:

1. **Local update**: each node performs stochastic gradient descent on its local dataset.
2. **Communication**: each node sends its updated model (or the update) to its neighbors.
3. **Aggregation**: each node combines its own model with the received ones using a consensus scheme.

Under suitable conditions (e.g., graph connectivity, data skew, aggregation weights) the nodes converge to a common model that is an approximation of the model that would result from a centralized approach.

A common choice for aggregation is a *doubly stochastic* weight matrix W, where $w_{ij} > 0$ if $(i, j) \in E$, $w_{ij} = 0$ otherwise, and $w_{ij} = w_{ji}$. This means that nodes assign nonzero weights to their neighbors and that the influence between neighbors is symmetric. In addition, the matrix is doubly stochastic, meaning that each row (and column) sums to one:

$$\sum_{j=0}^{n-1} w_{ij} = 1, \qquad \sum_{i=0}^{n-1} w_{ij} = 1.$$

Metropolis weights are a simple way to obtain such a matrix. For $i \neq j$, define

$$w_{ij} = \begin{cases} \frac{1}{1+\max(d_i,d_j)}, & \text{if } (i,j) \in E, \\ 0, & \text{otherwise,} \end{cases}$$

and choose the diagonal self-weights as

$$w_{ii} = 1 - \sum_{j \in \mathcal{N}_i} w_{ij}.$$

This weight construction depends only on local degree information and yields a valid consensus weight matrix, meaning that each node can compute the correct weights for itself and its neighbors without requiring global knowledge of the network. The structure of the communication graph G, e.g., dense vs. sparse or regular vs. heterogeneous degree, affects both the convergence speed and the amount of information each node exposes to others. Understanding this relationship is central to the contribution of this thesis.

Algorithm 1 formally defines the baseline decentralized learning algorithm, which serves as a reference point for all subsequent methods. At communication round r, node i first performs T local SGD updates using mini-batches of size B. With $x_i^{r,\tau}$ we denote the model of node i at communication round r after τ local SGD steps. After local training is completed, i.e., when $\tau = T$, the nodes communicate and update their parameters using mixing matrix W, resulting in $x_i^{r+1,0}$. This process is repeated for R communication rounds.

Algorithm 1 Decentralized parallel SGD at node i (baseline)

Require: Initial parameters $x_i^{0,0} = x_0$; learning rate γ; batch size B; T local steps per round; R rounds; mixing matrix W

1: **for** communication round $r = 0, 1, \ldots, R-1$ **do**
2: **for** local step $\tau = 1, \ldots, T$ **do**
3: Randomly sample minibatch $\mathcal{B}$ of B examples from node i's local data
4: $g \leftarrow \frac{1}{B} \sum_{\xi \in \mathcal{B}} \nabla F_i(x_i^{r,\tau}; \xi)$
5: Local update: $x_i^{r,\tau+1} \leftarrow x_i^{r,\tau} - \gamma\, g$
6: **end for**
7: Exchange full parameters with neighbors and set

$$x_i^{r+1,0} \leftarrow \sum_{j=1}^{n} W_{ij}\, x_j^{r,T}$$

8: **end for**
9: Output $x_i^{R,0}$, or $\frac{1}{n} \sum_{j=1}^{n} x_j^{R,0}$ as a consensus estimate

2.2. Membership Inference Attacks

Although collaborative learning was thought to be private, as data is not shared, the shared model updates can still leak information about the underlying training data. The model updates can be exploited by an attack known as a *Membership Inference Attack* [4]. This attack aims to determine whether a specific data sample was used to train a given model (or was part of the data that produced an update).

An attacker who has access to the model (or a proxy built from received updates) can query it on candidate samples and use the model output, such as the loss, the confidence in the label assigned, or the output probability vector, to decide membership. Typically, lower loss or higher confidence on a sample suggests that it was part of the training set. More advanced attacks train *shadow models*

to approximate the behavior of the target model and use their outputs to train an attack classifier that distinguishes between members and non-members.

The attack is formalized as a binary classification problem (member vs. non-member). Following the work of Carlini et al. [16], its performance is often reported as the *area under the ROC curve* (AUC). The receiver operating characteristic (ROC) curve plots the true positive rate (TPR) against the false positive rate (FPR), while varying the decision threshold used by the attack classifier to distinguish members from non-members. The TPR measures the fraction of correctly identified members, while the FPR measures the fraction of non-members incorrectly classified as members. The AUC summarizes this curve into a single value. An AUC of 0.5 corresponds to random guessing, while higher values indicate a more effective attack, and thus more privacy leakage. The AUC is widely used in machine learning as a threshold-independent measure for evaluating binary classifiers.

In decentralized settings, a malicious peer only has access to the messages it receives from its direct neighbors. So, depending on the protocol, the attacker must build a proxy of the victim's model from those messages (and possibly its own state) before running the MIA. The amount of information in the received messages, e.g., full model vs. a subset of parameters, directly affects attack success.

Membership inference attacks are widely used as a benchmark for measuring privacy leakage in collaborative learning systems. If an attacker is able to reliably distinguish training samples from non-training samples using only the messages received from its neighbors, it means information about the underlying training data has leaked. Because the MIAs only use information obtained from participating in the collaborative learning protocol, they represent realistic adversarial capability and provide a practical way to quantify privacy leakage in decentralized learning settings.

2.3. Differential Privacy

To quantify privacy leakage in learning algorithms, *Differential Privacy* [17] is widely used as a formal privacy framework. DP provides guarantees that the output of a randomized mechanism (e.g., a database query or local training step in a learning algorithm) does not depend strongly on any single individual data point. Intuitively, this means that whether an individual's data is included in the dataset or not should not significantly change the output, thus limiting the information that can be learned about any individual from the released result (e.g., query result or communicated model update).

In practice, many mechanisms achieve differential privacy by injecting carefully calibrated randomness into the computation, for example by adding noise to gradients or model updates during training.

Formally, a randomized mechanism $\mathcal{M}$ satisfies (ε, δ)-differential privacy if for any two datasets D and D' differing in a single record and any measurable set of outputs S,

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in S] + \delta. \tag{2.1}$$

Here, the privacy loss is controlled by ε, where smaller values correspond to stronger privacy guarantees. The parameter δ is often included to allow for a small probability that the privacy bound may be violated. Together, these parameters quantify the level of privacy provided by the mechanism.

Differentially Private Stochastic Gradient Descent (DP-SGD) [11] is an algorithm for training machine learning models with differential privacy guarantees. The key idea is that the influence of any single training sample on the gradient update should be limited by clipping per-sample gradients and then injecting carefully calibrated noise into the aggregated gradient.

Thus, DP-SGD builds upon the standard stochastic gradient descent algorithm in two main ways:

- **Gradient clipping.** The gradient computed for each individual training example is clipped so that its ℓ_2 norm (Euclidean length) does not exceed a threshold C. This clipping is necessary

Algorithm 2 Decentralized differentially private stochastic gradient descent at node i

Require: Initial parameters $x_i^{0,0} = x_0$; learning rate γ; batch size B; gradient norm bound C; noise multiplier σ; T local steps per round; R rounds; mixing matrix W

1: **for** communication round $r = 0, 1, \ldots, R-1$ **do**
2: **for** local step $\tau = 1, \ldots, T$ **do**
3: Randomly sample batch $\mathcal{B}$ of B examples from node i's local data
4: **for** each $\xi \in \mathcal{B}$ **do**
5: $g(\xi) \leftarrow \nabla F_i(x_i^{r,\tau}; \xi)$
6: Clip per-sample gradient:

$$\bar{g}(\xi) \leftarrow \frac{g(\xi)}{\max\left(1, \frac{\|g(\xi)\|_2}{C}\right)}$$

7: **end for**
8: Noisy minibatch gradient:

$$\tilde{g} \leftarrow \frac{1}{B}\left(\sum_{\xi \in \mathcal{B}} \bar{g}(\xi) + \eta\right), \quad \eta \sim \mathcal{N}(0, \sigma^2 C^2 I)$$

9: Local update: $x_i^{r,\tau+1} \leftarrow x_i^{r,\tau} - \gamma\, \tilde{g}$
10: **end for**
11: Exchange full parameters with neighbors and set

$$x_i^{r+1,0} \leftarrow \sum_{j=1}^{n} W_{ij}\, x_j^{r,T}$$

12: **end for**
13: Output $x_i^{R,0}$, or $\frac{1}{n}\sum_{j=1}^{n} x_j^{R,0}$ as a consensus estimate

to provide an upper bound on the sensitivity of the update, ensuring that a single training sample cannot have an arbitrarily large influence on the gradient.

- **Noise addition.** After aggregating the clipped gradients, Gaussian noise sampled from $\mathcal{N}(0, \sigma^2 C^2 I)$ is added to the gradient before performing the optimizer update. Here, C is the clipping threshold and σ is the noise multiplier controlling the noise scale. This noise masks the contribution of individual training samples and has zero mean, meaning that is does not introduce systematic bias, although it increases the variance of the gradient updates. An increased variance of the gradient updates makes the optimization process noisier, which can slow convergence and lead to lower model accuracy.

The noise multiplier σ, together with the sampling rate (determined by the batch size) and the total number of training steps, determines the resulting privacy guarantee (ε, δ). In general, stronger privacy (i.e., smaller ε) requires adding more noise, which degrades model performance.

DP-SGD is formally defined in Algorithm 2. At each communication round r, node i performs T local updates starting from the model $x_i^{r,0}$. As in Algorithm 1, at every local step τ, a mini-batch $\mathcal{B}$ of size B is sampled from the local dataset. For each sample $\xi \in \mathcal{B}$, the gradient $\nabla F_i(x_i^{r,\tau}; \xi)$ is computed and is clipped to limit the influence of any individual sample. When all samples in $\mathcal{B}$ have been clipped, they are averaged and Gaussian noise $\eta \sim \mathcal{N}(0, \sigma^2 C^2 I)$ is added. The node's local model is then updated using the noisy gradient estimate of the batch.

After completing the T local steps, i.e., when $\tau = T$, nodes communicate and update their parameters following the same procedure as in Algorithm 1.

2.4. Model chunking

Besides adding noise, there are other ways of amplifying privacy in decentralized learning. These so-called noiseless [15] approaches reduce information leakage without injecting noise, for example by limiting what each peer sees. In standard decentralized learning, a node sends its full model (or full update) to all of its neighbors. This means that a single malicious neighbor has complete information about that node's model state. To mitigate this risk, techniques such as model chunking can be applied to improve privacy. With chunking, instead of sending the full model state to each neighbor, the sender partitions the model into chunks and sends only a subset of chunks to each neighbor. This means that no single neighbor can see the full model which makes reconstruction or inference attacks harder. Chunking can therefore provide significant privacy protection with relatively limited impact on utility.

A *chunk* is defined as a contiguous subset of scalar parameter values from the model of a node. Formally, a chunk corresponds to a subset of parameter indices together with the associated parameter values. In the chunking strategies considered in this thesis, chunks originating from the same node within a communication round are always disjoint.

However, different chunking strategies exist. In commonly used graph-independent implementations, which we refer to as fixed-K chunking in this thesis, nodes partition their model into a fixed number of chunks and broadcast the same subset of chunks to all neighbors. While simple and effective, this does not exploit the underlying network structure and can lead to slower information propagation, as all neighbors receive the same subset of chunks each round. This observation motivates the exploration of topology-aware variants of chunking, where the chunk partitioning or distribution depends on the network topology.

Algorithm 3 formally defines a generic decentralized learning procedure with chunked communication. At each communication round r, node i performs T local SGD updates starting from $x_i^{r,0}$, following the same local training procedure as in Algorithm 1. After local training, the node communicates only parts of its model with its neighbors $\mathcal{N}_i$. Instead of sending the full model, the parameters are partitioned into chunks and distributed across neighbors using the ChunkAndAssign procedure. The received chunks are then incorporated into the local model using the Merge procedure. The specific implementations of these subroutines are defined in Section 4.3.2 and depends on the specific chunking strategy used.

Algorithm 3 Decentralized stochastic training at node i with chunked communication

Require: Initial parameters $x_i^{0,0} = x_0$; learning rate γ; batch size B; T local steps per round; R rounds; neighbors $\mathcal{N}_i = \{j_1, \ldots, j_{d_i}\}$ with $d_i = |\mathcal{N}_i|$; mixing matrix W, number of chunks sent S;

for communication round $r = 0, 1, \ldots, R-1$ **do**
  for local step $\tau = 1, \ldots, T$ **do**
    Randomly sample minibatch $\mathcal{B}$ of B examples from node i's local data
    $g \leftarrow \frac{1}{B} \sum_{\xi \in \mathcal{B}} \nabla F_i(x_i^{r,\tau}; \xi)$
    Local update: $x_i^{r,\tau+1} \leftarrow x_i^{r,\tau} - \gamma\, g$
  end for
  Let $x \leftarrow x_i^{r,T}$
  $msg \leftarrow \textsc{ChunkAndAssign}(x, \mathcal{N}_i, S)$
  Send msg; receive $\{\text{recv}_j\}_{j \in \mathcal{N}_i}$
  $x_i^{r+1,0} \leftarrow \textsc{Merge}(x, \{\text{recv}_j\}_{j \in \mathcal{N}_i}, W)$
end for
Output $x_i^{R,0}$, or a network consensus estimate of the parameters

3

Related Work

This section provides an overview of the literature relevant to this thesis. We first introduce the decentralized learning paradigm and its associated algorithms, followed by a discussion on the role of the network topology in decentralized learning. We then review privacy threats in collaborative learning and privacy preserving techniques to mitigate the risk of these attacks. Finally, we discuss related work on Graph Neural Networks and their vulnerability to privacy attacks, highlighting the parallels with decentralized learning.

3.1. Decentralized Learning

Decentralized Learning has emerged as an alternative to Federated Learning to eliminate the dependency on a central server and improve scalability [8]. In DL, model updates are shared directly among neighboring nodes according to a predefined communication topology, collaboratively training a global model in a fully peer-to-peer fashion.

Several notable DL algorithms have been proposed. Gossip Learning [18] is based on randomized model exchanges. Individual models perform randomized walks across the network. Each time a model visits a node, it is updated with that node's local data before moving on to the next node. Each node initially holds its own model, so there are as many models as there are nodes in the network. Decentralized Parallel Stochastic Gradient Descent (D-PSGD) [8] extends standard SGD to the peer-to-peer setting. Each node performs local SGD updates and communicates the updated parameters with its neighbours which average it into their own models to achieve consensus over time. More recently, de Vos et al. introduced Epidemic Learning [14], a DL algorithm that accelerates convergence and saves communication costs by leveraging randomized, dynamically changing communication networks. In each training round, every node randomly samples a subset of nodes and sends its model updates to the sampled nodes, resulting in a faster diffusion of information.

The scope of these studies lies primarily on the convergence and communication efficiency of algorithms, while the interaction between network topology and privacy leakage is not sufficiently explored, particularly in the case of adversarial inference attacks. This motivates a deeper investigation into how network topology influences vulnerability to privacy leakage and how topology-aware mechanisms can be integrated with existing solutions to enhance privacy.

3.2. Network topology in Decentralized Learning

Network topology plays a crucial role in decentralized learning as it governs how information is propagated across nodes. It directly influences both the vulnerability to privacy attacks and convergence rate. Prior research has shown that utilizing the network topology effectively allows decentralized networks to outperform centralized ones [8]. In particular, well-connected and fast-mixing topologies improve convergence speed while sparse or poorly connected networks can

significantly slow down convergence [19]. Examples of such efficient topologies include expander graphs [20] and exponential graphs [21]. These achieve fast information mixing with relatively low communication overhead.

Beyond static connectivity and mixing speed, recent work shows that the role of topology is more nuanced than older research suggests. For instance, Vogels et al. show that the spectral gap, a graph property that is a good indicator of connectivity and mixing speed, does not fully capture the behavior of DL systems and is not a good predictor for empirical convergence speed [22]. This indicates that other graph properties or external factors may influence convergence.

One such factor is the heterogeneity of the data distribution across nodes. Le Bars et al. investigate the influence of data heterogeneity on convergence in decentralized learning and found that aligning the communication topology with the data distribution can reduce, or even eliminate, the negative effect of data heterogeneity on convergence [23]. Their findings highlight that data distribution and topology should not be considered independently, but rather optimized together for efficient decentralized learning under heterogeneous settings.

The focus of these works lies primarily on the effect of topology on convergence while these same topological properties that govern convergence also have direct implications for privacy. Building on these findings, our work investigates how such properties influence vulnerability to privacy attacks in decentralized learning.

3.3. Privacy Threats in Collaborative Learning

A number of works have demonstrated that model updates exchanged during collaborative learning can leak private information. Shokri et al. [4] showed that trained machine learning models leak information that allows adversaries to determine whether a data record was used in training a model. This type of attack is known as a Membership Inference Attack. MIA can reveal sensitive information about the data owner. For instance, if a model trained to predict disease reveals that an individual's data was used in training, this may infer that this individual suffers from the disease.

The effectiveness of MIA has been shown to be influenced by numerous factors, including the degree of model overfitting, the number of output classes, the amount of training data, and the model architecture [4], [24], [25], [26], [27].

In work closely related to ours, Touat et. al [28], investigated the relationship between the graph mixing properties of decentralized learning topologies and the risk of MIA. They found that fast mixing networks offer better privacy-utility tradeoffs and recommend dynamic topologies to improve model mixing, especially in the case of large and sparse topologies. Our research builds on these findings by exploring how topology-aware obfuscation strategies can optimize the privacy-utility tradeoff across varying decentralized network structures.

There are also other attacks such as Attribute Inference Attacks [5], where attackers exploit shared model updates to infer sensitive attributes of participant's training data. And in the most severe cases, model updates can even be used to reconstruct raw training samples [6], [7]. In our research, the focus lies on Membership Inference Attacks as it is closely related to the other privacy attacks [29]. Therefore, assessing a model's vulnerability to MIA attacks is a great way to obtain insights into the overall vulnerability to other privacy attacks.

These works collectively demonstrate that collaborative learning methods such as federated and decentralized learning are not private by design.

3.4. Privacy-Preserving Techniques

Privacy-preserving techniques have been studied extensively in traditional machine learning settings [17], [30], [31], [32], [33]. Here, data is typically centralized and models are trained on independent data points, where each sample contributes to the model without interacting with others. In such settings, techniques such as differential privacy and secure aggregation are effective because interaction between data points is limited.

In decentralized learning, these techniques are challenging to apply directly, as model updates are exchanged between nodes and information spreads throughout the network through repeated

communication. Additionally, this setting enables stronger adversarial capabilities, as attackers can observe intermediate model updates rather than only having access to black-box model outputs. Because of this, the influence of a single data point can extend beyond its original node, making privacy leakage more complex and harder to control.

To address these challenges, both adaptations of classical privacy-preserving techniques and new approaches specifically designed for decentralized learning have been proposed. A widely used framework to quantify the privacy leakage of algorithms is Differential Privacy [17]. DP introduces formal privacy guarantees by injecting calibrated noise into computations to ensure that the final output does not leak too much about any individual data point. This framework offers exactly the kind of privacy guarantees needed to protect against inference attacks. Cyffers et al. [10], [11] introduced pairwise network DP, a measure that quantifies the privacy loss of a decentralized algorithm for each pair of nodes in a communication network. Furthermore, they show that in their decentralized DP algorithm, Muffliato, the optimal privacy-utility tradeoffs depend on topological properties of the network, like the eigengap and maximum degree of the graph.

Another approach is to utilize state-of-the-art cryptographic secure aggregation protocols [12], [13]. By using, for example, homomorphic encryption, model aggregations can be performed without revealing anything about individual nodes' model updates. However, these approaches are often computationally intensive, introduce significant communication overhead, or rely on additional cryptographic assumptions.

Recent work explores novel, noiseless mitigation techniques such as randomized communication and dynamically changing network topologies [14], as well as decoupling node identity through virtualization or chunking of model updates [15]. These techniques aim to amplify privacy without sacrificing utility. Although some of these approaches employ dynamic communication patterns, they are typically evaluated on a fixed class of network topology. Because of this, a systematic evaluation of hybrid privacy mechanisms across varying decentralized network topologies remains an open research direction.

3.5. Graph Neural Networks

Graph Neural Networks (GNNs) are deep learning models designed to operate on graph-structured data such as social networks or chemical molecules. In these settings, nodes represent entities (e.g., individuals or atoms) with corresponding features (e.g., age, gender, atomic radius), while edges encode relationships between them.

Traditional machine learning models are not well-suited for this because the graph structure is irregular and there is no fixed ordering of nodes. Representations of the graph should be invariant to permutations in node ordering, while still capturing the relationship between them. GNNs address this challenge by using a mechanism called message passing, where each node aggregates the features of its neighboring nodes and applies a neural network transformation to the aggregated representation. In this way, the underlying graph structure is naturally included in the learning process.

These Graph Neural Networks share strong similarities with decentralized learning, as both use local information exchange in a graph topology to approximate global representations.

Recent work has shown that, similar to decentralized learning, GNNs are vulnerable to membership inference attacks. Olatunji et al. [34] demonstrated that adversaries can exploit black-box access to a model, such as its output predictions, to infer whether a node was part of the training set. Further research shows that graph density and feature similarity have a major impact on the attack's success [35]. In response, one proposed defense mechanism is to perturb the original graph structure by adding more edges. This artificially increases graph density and dilutes the influence of individual nodes, making their contributions less distinguishable and reducing membership signal.

Daigavane et al. propose a defense mechanism based on DP-SGD for GNNs [36]. In their approach, the gradients of individual nodes are clipped, and training is done on randomly sampled subsets of the graph to limit the sensitivity of each update. Noise is then added to the aggregated gradient before updating the model parameters.

However, this approach assumes a central trusted setting where aggregation happens before communication. In decentralized learning, nodes communicate gradients directly and each node can be malicious. As a result, noise must be applied locally, which changes the privacy mechanism and typically leads to lowered utility.

These findings highlight that graph structure has a great influence on privacy leakage in GNNs. This reinforces the importance of topology aware privacy mechanisms not only in GNNs, but also in decentralized learning, where information propagation similarly depends on the underlying communication graph.

3.6. Summary and Research Gap

Existing work in decentralized learning has mainly focused on improving convergence and communication efficiency, with the effect of network topology on these properties being of particular interest. More recent studies highlight that topology also affects privacy, as it determines how information spreads across the network. However, the interaction between topology and privacy leakage remains underexplored, especially in adversarial settings like membership inference attacks.

At the same time, existing privacy-preserving techniques in machine learning, such as DP and secure aggregation, are difficult to apply directly in decentralized settings or introduce significant tradeoffs between privacy and utility. Recent noiseless approaches provide promising alternatives, but are typically only evaluated under a fixed number of topology classes.

Furthermore, insights from GNNs show that graph structure can greatly impact privacy leakage, highlighting the importance of topology-aware defense mechanisms.

Therefore, a key research gap remains in understanding how the communication topology influences privacy leakage in decentralized learning, and how topology-aware defense mechanisms can be implemented to improve the privacy-utility tradeoff. This thesis addresses this gap by systematically studying the effect of topology on MIA vulnerability, and exploring topology-aware defense mechanisms.

4

Methodology

Building on the concepts introduced in Section 2, this section describes the methodology to answer the research question: In decentralized learning, how does communication topology influence vulnerability to membership inference attacks and model utility, and can topology-aware defense mechanisms improve the privacy-utility tradeoff compared to topology-agnostic approaches?

This section discusses the decentralized learning framework (topologies, aggregation and training protocol), the threat model and MIA evaluation procedure, the implementation of the privacy mechanisms, and the experimental design.

4.1. Decentralized Learning Framework

4.1.1. System Model and Communication Topology

We adopt the graph-based model of Section 2.1. To study the effect of network structure on privacy and utility, we use two complementary analysis approaches.

- **Topology-Class Analysis:** Experiments are conducted on "tailor-made" graph structures representing extreme or illustrative communication settings. Illustrative examples of these structures are shown in Figure 4.1.
  - **Ring:** Nodes form a cycle, resulting in minimal connectivity and high network diameter.
  - **Star:** A central hub connects to all other nodes, resembling a centralized communication structure.
  - **Grid / Lattice:** Nodes are arranged in a 2D grid with nearest-neighbor connections, yielding moderate degree and a regular structure.
  - **Fully Connected (clique):** Every pair of nodes is connected, resulting in maximal connectivity and fast information mixing speed.
- **Topology-Property Analysis:** Additional experiments use parametric graph families in which structural graph properties can be systematically varied. We first consider random d**-regular graphs**, where each node has the same fixed degree d. These graphs function as a controlled baseline with uniform connectivity. These graphs are used for studying the effect of the overall network degree. Because they do not capture heterogeneity between nodes, they are not suitable for analyzing the impact of degree differences at the node level.

  Particularly important for our experiments are the Erdős–Rényi (ER) graphs, where edges are included independently between any pair of nodes with probability p, allowing control over the graph density. This family naturally produces networks with heterogeneous node degrees, making it suitable for studying how node degree impacts privacy leakage and model performance. By varying p we can control the average degree and density of the network allowing for controlled experiments that isolate the effects of network structure.

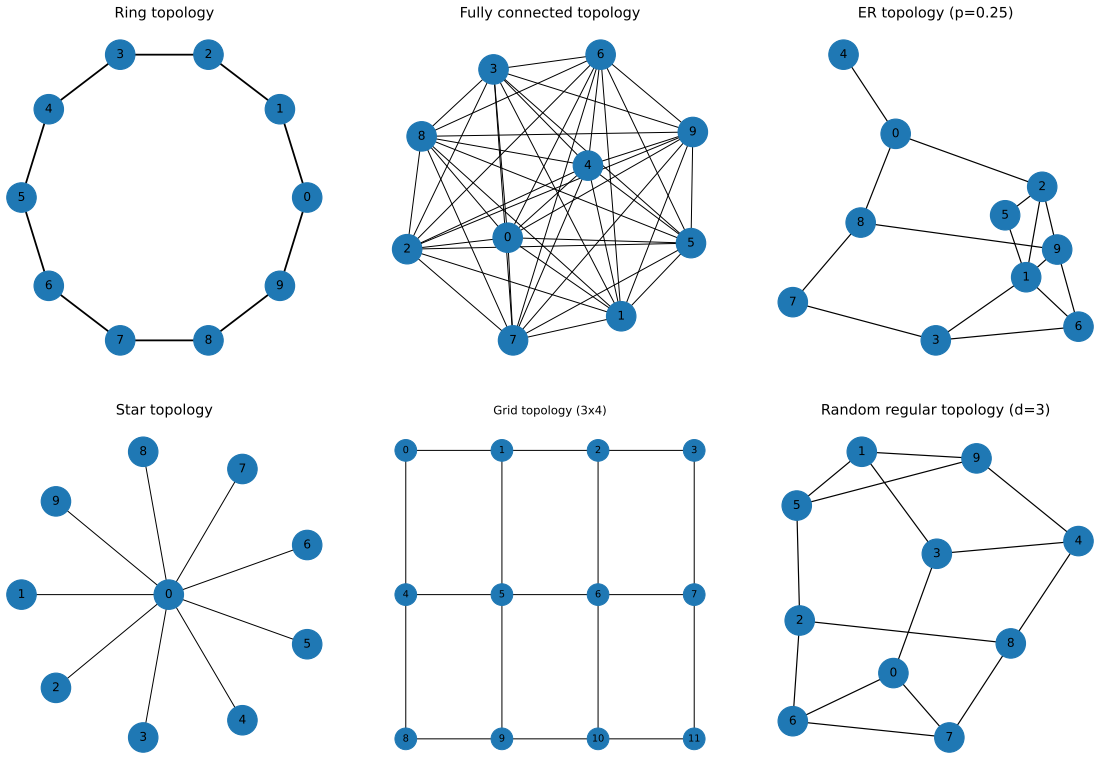


Figure 4.1: Illustrative examples of the communication topologies used in the experiments (8-12 nodes).

Figure 4.1 illustrates the main communication topologies used in the experiments using small example graphs (8-12 nodes).

All graphs are implemented with NetworkX[1] and converted into neighbor lists and aggregation weights as described below.

4.1.2. Aggregation Weights

We use Metropolis-inspired weights (Section 2.1) as the consensus scheme during model aggregation. For each edge (i, j), the mixing weight is defined as

$$w_{ij} = \frac{1}{\max(d_i, d_j)}.$$

In our implementation, these weights are used to compute a weighted average of the received neighbor models. This weighted mix is then combined with the node's own model update, controlled by a global mixing parameter β. The parameter β determines the relative influence of the neighbors' models compared to the node's own model. When β is close to 1, the update becomes dominated by the neighbors' models, whereas when β is close to 0, the node relies mostly on its own model. The model update therefore takes the form:

$$x_i^{(t+1)} = (1 - \beta)x_i^{(t)} + \beta \sum_{j \in \mathcal{N}_i} \tilde{w}_{ij} x_j^{(t)},$$

where $\tilde{w}_{ij}$ are the normalized Metropolis weights over the neighbors of node i, defined as:

$$\tilde{w}_{ij} = \frac{w_{ij}}{\sum_{k \in \mathcal{N}_i} w_{ik}}$$

[1] `https://networkx.org`

This normalization ensures that the neighbor models form a valid weighted average:

$$\sum_{j \in \mathcal{N}_i} \tilde{w}_{ij} = 1,$$

while preserving the relative weighting determined by the consensus scheme. This construction enables the decentralized aggregation of neighbor models, while the parameter β controls the strength of neighbor interaction in the update. This construction is different from the standard Metropolis consensus matrix, where the diagonal self-weights are chosen to make the matrix doubly stochastic. Here, the self-weight is instead controlled explicitly through β.

4.1.3. Training Protocol

The decentralized training process is carried out in rounds, with each training round proceeding as follows.

1. **Local training**: Each node i performs local SGD on its private dataset using a fixed batch size and a fixed number of local epochs. Optionally, local updates are made differentially private using DP-SGD (see 4.3.1).
2. **Message preparation**: Node i constructs messages for each neighbor $j \in \mathcal{N}_i$. The full model parameters obtained after the local training step are sent. In this thesis, we only consider protocols in which the full model parameters (or chunked subsets thereof) are communicated. Protocols based on parameter updates (deltas) are not considered. The rationale for this design choice is discussed in Section 4.2.3. When chunking is enabled, the state is split into chunks and each neighbor receives a subset of chunks (see 4.3.2).
3. **Communication**: Messages are delivered to the corresponding neighbors without the help of a central server.
4. **MIA evaluation (optional)**: Before aggregation, we optionally run the MIA attack on the messages received by each node. This simulates a passive adversary that only observes the messages it received from its neighbors (see 4.2).
5. **Averaging**: Each node updates its model using the Metropolis-inspired weighted averaging step described in 4.1.2. For chunked messages, only the received parameters are updated, the remaining parameters keep their local value. We use a mixing coefficient $\beta \in [0, 1]$ to control how the local model is combined with the weighted average of the received chunks.

Nodes are initialized from a common global initial state, this ensures all nodes start with identical model parameters which ensures more stable convergence of the decentralized training process. All nodes use the same model architecture and the training data is partitioned across nodes. The framework supports both IID and Dirichlet partitioning. However, as explained in Section 4.5.1, the final reported experiments use IID partitioning.

Evaluation of each node's model is performed on a held-out test set that follows the same distribution as that node's training data. The same test set is also used as the non-member dataset when performing the membership inference attack for that node.

4.2. Threat Model and Membership Inference Attack Setup

4.2.1. Threat Model

For our threat model we consider a decentralized learning settings with a single *honest but curious* adversary. This adversary is a malicious peer in the network that follows the protocol correctly but uses information received during the protocol to attempt to infer whether a specific data sample was part of the training set of a neighboring victim node.

Adversary capabilities. The adversary can only access information that it legitimately receives during the training process. In particular, the adversary can see:

- The messages received from the neighboring victim node during the communication phase. This can be full model parameters or chunked subsets thereof.

- Its own local state, including the model parameters before and after training or aggregation.

By combining this information, that adversary can reconstruct a proxy of the victim's model and carry out the membership inference attack.

Adversary knowledge. Following Kerckhoff's principle, which states that a cryptographic system should remain secure even if everything about the protocol except the key is public knowledge, we assume that the adversary has full knowledge of the training protocol. This includes the model architecture, the training algorithm, hyperparameters and communication scheme. In addition, the adversary may know the network topology and dataset partitioning mechanism. However, the adversary does not have access to the victim's private training data.

Adversary limitations. As stated earlier, the adversary does not have access to the victim's raw training data. Furthermore, it cannot observe the internal outputs of the victim's training procedure or any communication not addressed to it. Finally, the adversary follows the protocol and does not alter, drop, or inject messages.

This threat model captures a realistic scenario in which a compromised or curious peer attempts to infer membership information from legitimately observed model updates in a peer-to-peer learning environment. We do not consider collusion between multiple nodes in this work and leave this scenario for future study.

4.2.2. Attack Procedure

For each victim node and each of its neighbors (we consider non-colluding adversaries but assume that each of the neighbors may be a malicious peer), the attack proceeds as follows:

1. **Proxy model construction**: The attacker creates a proxy model to approximate the model state of the victim after local training. The adversary creates this proxy by combining its own model state before communication with the messages received (full model, or chunked subsets) from the victim. When the full model is received, the proxy is exactly the victim's model. When chunking is used, the proxy is only partially updated with the victim's chunk. The remaining parameters remain the attacker's pre-communication state.
2. **Membership inference**: Using this proxy model, the attack runs a membership inference attack against it. During data partitioning, the victim's training set is split into a training set and holdout set (e.g., 80%/20%). The attacker is given a set of candidate samples, some of which are members (sampled from the victim's training data) and some non-members (sampled from the holdout set). For each sample, the attack outputs the classification loss using cross-entropy between the model prediction and the true label. Lower loss indicates that sample is likely a member while higher loss suggests likely a non-member. The attack performance is measured using the area under the ROC curve, as described in Section 2.2.

The intuition behind the loss-based attack is that models are trained to minimize loss on training samples. As a result, training samples often obtain lower loss than unseen samples, which can be exploited to infer membership.

The attack is implemented using the `mia_attacks` library [37]. The library is implemented and optimized for running an attack on a full target model. The attack requires a complete target model and uses its outputs to determine membership and AUC. The library does not work with model updates (delta). Therefore, in a protocol where only deltas are shared with neighbors, an attacker would first have to construct a victim proxy model, based on its own local model or previously received updates, before running the attack.

4.2.3. Choice of Full-State Messaging for MIA Evaluation

We evaluate MIA and our proposed mechanisms under full model communication only. We do not consider protocols in which participants exchange parameter updates as deltas. This choice is deliberate. When the full model, or chunks thereof, are communicated, an attacker can construct a proxy model that preserves a much stronger membership signal. This allows the effectiveness of

the proposed mechanisms to be evaluated more clearly. A higher AUC indicates that membership inference is a realistic threat, and any reduction in AUC can therefore be meaningfully attributed to the defense mechanisms. By contrast, when only deltas are communicated, the attacker must first reconstruct a proxy model from the available information as mentioned before. This reconstruction is typically done by combining the deltas with the attackers own local model, which dilutes the information leakage in the updates and results in weaker membership inference signal.

While attacks specifically designed to operate on parameter updates, can still exploit membership inference effectively, the current `mia_attacks` library does not provide such attacks. For this reason, and to obtain a clearer membership inference signal when evaluating our defenses, we restrict our experiments to the full model communication setting.

4.3. Privacy Mechanisms

Algorithm 1 in Section 2.1 introduced the baseline decentralized learning procedure that serves as the reference point for all methods considered in this thesis.

In practice, we apply aggregation with mixing parameter $\beta \in (0, 1]$, replacing the update $x_i^{r+1,0} \leftarrow x_i^+$ with $x_i^{r+1,0} \leftarrow (1-\beta)x_i^{r,T} + \beta x_i^+$, where x_i^+ denotes the result of the aggregation step. This improves stability and gives more control over the influence of a node's own contribution. This is used consistently across all methods, but is omitted from the algorithms for clarity.

4.3.1. Differential Privacy via DP-SGD

To reduce leakage from gradients, we integrate differential privacy into local training. Each node runs DP-SGD as formally defined in Algorithm 2 in Section 2.3. For each sample, gradients are clipped to a maximum ℓ_2 norm C. All samples in the batch are aggregated and Gaussian noise with scale $\sigma \cdot C$ is added before the optimizer step. We use the Opacus library[2] for the privacy engine and ϵ accounting. The noise multiplier σ (and thus the resulting privacy guarantee) can be set globally together with the clipping threshold C.

We use Opacus' `BatchMemoryManager`[3] to support larger logical batch sizes. Batch size is an important hyperparameter in DP-SGD. This is because larger batches aggregate more clipped per-sample gradients before noise is added. This reduces variance and can improve model utility without lowering the noise multiplier and thus sacrificing privacy guarantees.

4.3.2. Topology-Aware Chunking

Chunking (Section 2.4) reduces the amount of information any single neighbor receives. Instead of receiving the full model state, only a subset of chunks is received. Unlike fixed-K chunking, which uses a fixed global number of chunks independent of the graph structure, the topology-aware chunking variant considered in this thesis varies the number of chunks depending on the sender's degree.

Topology-aware chunking is implemented as follows. For each tensor in the model state, we split it into d_i contiguous row blocks, where d_i is the sender's degree. The order of the blocks is permuted deterministically (with a seed combining round and sender id for reproducibility) so that the chunks each neighbor receives varies per round. Each neighbor is then sent a fixed number of chunks S (S=1 by default). When $S > 1$, each neighbor receives multiple chunks, trading some privacy for better utility. Tensors that are too small to be split are either broadcast to all neighbors or sent to a single randomly chosen neighbor, depending on the configuration. A schematic overview of the row-block chunking is shown in Figure 4.2.

By varying the chunks each neighbor receives per round, neighbors are not restricted to a fixed subset of parameters during training. Over multiple rounds, neighbors will eventually incorporate most parts of the full model in their local model. This setup could allow an attacker to store the chunks received from a victim across rounds and combine them to construct a stronger proxy

[2] `https://opacus.ai/`

[3] Utility provided by Opacus to simulate larger logical batch sizes by splitting them into smaller physical batches during training.

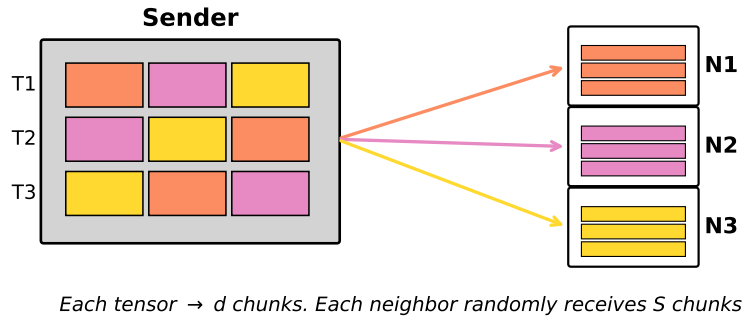


Figure 4.2: Topology-aware chunking: the sender splits each tensor in the model state into d chunks (here $d = 3$); each neighbor receives one chunk per tensor, so no single neighbor sees the full model.

model. We do not consider this type of attacker, as it can be prevented through mechanisms proposed in the literature, such as node identity decoupling through virtualization [15]. If an attacker cannot know which node sent a particular chunk, it becomes difficult to associate and combine chunks belonging to the same victim.

Receivers only update the parameter indices they received. For parameter indices they did not receive, the node's own model update becomes the total update. The metropolis averaging step explained before is therefore applied in a partial way. For each parameter index, the update is a weighted average of the node's own value and the values received from neighbors that sent a chunk containing that index. The Metropolis weights (w_{ij}) remain those defined by the consensus scheme (one weight per edge in the network graph). The difference is that, for each parameter index, the weights are normalized only over the subset of neighbors that contributed a chunk containing that index. This ensures that the neighbor contribution remains a valid weighted average.

The aggregation is therefore applied independently for each parameter index, using only the subset of neighbors that provided that index. The resulting updated model is obtained through the TopologyAwareMerge procedure. This keeps the protocol well-defined when chunking is active. Algorithm 3 in Section 2.4 formally defines the generic decentralized learning procedure with chunked communication using abstract ChunkAndAssign and Merge subroutines. For topology-aware chunking, these subroutines are instantiated as the topology-aware procedures TopologyAwareChunkAndAssign (Algorithm 4) and TopologyAwareMerge (Algorithm 5).

The TopologyAwareChunkAndAssign procedure works as follows. For each tensor θ in the model x, if the size of its leading dimension $n_0(\theta)$ is larger than the degree d_i of node i, then θ is split into $d_i = |\mathcal{N}_i|$ contiguous blocks. These blocks are randomly permuted, and each neighbor is assigned S cyclic blocks. If a tensor is too small to split, it is sent in full to each neighbor or randomly sent to one neighbor depending on the configuration.

Upon receiving the parameter chunks from its neighbors, node i updates its model using the TopologyAwareMerge procedure (Algorithm 5). For each element e of each tensor θ in the model x, let $\mathcal{N}_i(e)$ denote the set of neighbors that provided a value for e. The mixing weights W are then renormalized over the set $S_i(e)$, which includes node i and the neighbors in $\mathcal{N}_i(e)$. The values are then averaged using the renormalized weights. Once this process is completed for each parameter e in x, the updated values define the new model.

These subroutines are called at every communication round within the generic chunked decentralized learning procedure defined in Algorithm 3.

Algorithm 4 TopologyAwareChunkAndAssign($x, \mathcal{N}_i, S$)

Initialize outgoing messages for all neighbors in $\mathcal{N}_i$
for each tensor θ in x **do**
  if $n_0(\theta) \geq d_i$ **then**
    Split θ into d_i nearly equal contiguous row-blocks $B_1, \ldots, B_{d_i}$
    Draw permutation π of $\{1, \ldots, d_i\}$; $\tilde{B}_\ell \leftarrow B_{\pi(\ell)}$
    for $m = 1, \ldots, d_i$ **do**
      Assign neighbor j_m the S cyclic blocks $\left\{\tilde{B}_{((m+\ell-2) \bmod d_i)+1}\right\}_{\ell=1}^{S}$
    end for
  else
    Send θ in full to one randomly chosen neighbor in $\mathcal{N}_i$
  end if
end for
return outgoing messages

Algorithm 5 TopologyAwareMerge($x, \{\text{recv}_j\}_{j \in \mathcal{N}_i}, W$)

for each entry e of each tensor in x **do**
  $\mathcal{N}_i(e) \leftarrow \{j \in \mathcal{N}_i : \text{recv}_j \text{ includes a value for } e\}$ // set of neighbors that contributed e
  $S_i(e) \leftarrow \{i\} \cup \mathcal{N}_i(e)$ // include self
  $\hat{W}_{ij}(e) \leftarrow \frac{W_{ij}}{\sum_{h \in S_i(e)} W_{ih}}$ for $j \in S_i(e)$ // re-normalize weights
  $x^+[e] \leftarrow \hat{W}_{ii}(e)\, x[e] + \sum_{j \in \mathcal{N}_i(e)} \hat{W}_{ij}(e)\, \text{recv}_j[e]$ // $x[e]$ separate because not included in $\text{recv}_j[e]$
end for
return x^+

4.3.3. Graph-independent Fixed-K Chunking

Besides the topology-aware chunking used as the main mechanism in this thesis, we implement a graph-independent chunking baseline for comparison with the topology-aware approach and the proposed ChunkDP. In this baseline, all model parameters are flattened into a single vector in a fixed order and partitioned into K contiguous chunks of approximately equal size, where K denotes the global number of chunks each node splits their model into every round. At each communication round, a node randomly samples S chunks ($S \leq K$), and sends these same S chunks to all neighbors. Here S is the number of chunks transmitted per round to each neighbor. Unlike topology-aware chunking, the chunk size and selection of chunks does not depend on the network structure or individual neighbors. Fixed-K chunking is a topology-independent chunking strategy and serves as a natural baseline for evaluating the benefits of topology-aware mechanisms.

4.4. Hybrid Strategy: ChunkDP

Although topology-aware chunking provides substantial privacy benefits for well-connected nodes, the protection it provides is uneven, leaving poorly-connected nodes highly vulnerable to membership inference attacks (Sections 5.1.1 and 5.1.5). To address this limitation, we propose *ChunkDP*, a hybrid defense mechanism that combines topology-aware chunking and differential privacy. Local training is done with DP-SGD, while communication uses topology-aware chunking.

Chunking reduces information leakage by ensuring an adversary only receives a subset of the model parameters. Because parameters are chunked and distributed across neighbors in topology-aware chunking, the degree of a node determines how much information any single neighbor receives. Low degree nodes cannot split their model across many neighbors and therefore are less protected by topology-aware chunking alone. This effect is particularly visible in heterogeneous topologies such as star networks, where the leaves suffer from almost perfect MIA AUC of approximately 1.00 while the central hub exhibits an AUC of approximately 0.61 (Section 5.1.5). These results

show that topology-aware chunking offers effective privacy benefits for well-connected nodes, but leaves poorly connected nodes highly vulnerable to MIAs. ChunkDP addresses this imbalance by applying more noise specifically to nodes with low degree.

Let d_i denote the degree of node i in the communication graph. The noise multiplier used by node i is defined as

$$\sigma_i = \frac{\sigma_0}{d_i},$$

where σ_0 is the global noise multiplier used by nodes with degree 1. This degree-dependent noise scaling is motivated specifically by the limitations of the topology-aware chunking variant. In fixed-K chunking, the number of chunks is independent of node degree.

This noise scaling mechanism should be interpreted as a heuristic rather than a formally derived privacy guarantee. Differential privacy guarantees are usually defined for a fixed noise multiplier. In this setting however, the noise multiplier varies across nodes depending on their degree. A formal analysis of the resulting privacy guarantees of ChunkDP using node-dependent noise scaling is outside the scope of this work.

As mentioned before, nodes with a low degree are less protected by topology-aware chunking because their parameters are distributed across fewer neighbors. With ChunkDP, these vulnerable nodes use a larger noise multiplier to cover this limitation introduced by topology-aware chunking. On the other hand, nodes with a higher degree benefit more from topology-aware chunking and use a smaller noise multiplier, reducing utility loss caused by adding noise. In short, the proposed ChunkDP applies differential privacy where it is most needed while preserving utility for nodes that are already well protected.

ChunkDP is formally defined in Algorithm 6. It combines the differentially private training procedure (Algorithm 2) with topology-aware chunked communication (Algorithm 3). The main difference is that it uses node-specific noise σ_i instead of a global noise multiplier σ.

Algorithm 6 ChunkDP: Decentralized differentially private stochastic gradient descent at node i with topology-aware chunking

Require: Initial parameters $x_i^{0,0} = x_0$; learning rate γ; batch size B; gradient norm bound C; base noise multiplier σ_0; T local steps per round; R rounds; neighbors $\mathcal{N}_i = \{j_1, \dots, j_{d_i}\}$ with $d_i = |\mathcal{N}_i|$; mixing matrix W;

Set node-specific noise multiplier $\sigma_i \leftarrow \sigma_0/d_i$ // move into for-loop for dynamic topologies
for communication round $r = 0, 1, \dots, R-1$ **do**
  Set node-specific noise multiplier $\sigma_i \leftarrow \sigma_0/d_i$
  for local step $\tau = 1, \dots, T$ **do**
    Randomly sample batch $\mathcal{B}$ of B examples from node i's local data
    for each $\xi \in \mathcal{B}$ **do**
      $g(\xi) \leftarrow \nabla F_i(x_i^{r,\tau}; \xi)$
      Clip per-sample gradient:

$$\bar{g}(\xi) \leftarrow \frac{g(\xi)}{\max\left(1, \frac{\|g(\xi)\|_2}{C}\right)}$$

    end for
    Noisy minibatch gradient: // uses node specific noise multiplier

$$\tilde{g} \leftarrow \frac{1}{B}\left(\sum_{\xi \in \mathcal{B}} \bar{g}(\xi) + \eta\right), \quad \eta \sim \mathcal{N}(0, \sigma_i^2 C^2 I)$$

    Local update: $x_i^{r,\tau+1} \leftarrow x_i^{r,\tau} - \gamma\, \tilde{g}$
  end for
  Let $x \leftarrow x_i^{r,T}$
  $msg \leftarrow$ TopologyAwareChunkAndAssign$(x, \mathcal{N}_i, S)$
  Send msg; receive $\{\text{recv}_j\}_{j \in \mathcal{N}_i}$
  $x_i^{r+1,0} \leftarrow$ TopologyAwareMerge$(x, \{\text{recv}_j\}_{j \in \mathcal{N}_i}, W)$
end for
Output $x_i^{R,0}$, or a network consensus estimate of the parameters

4.5. Datasets, Models, and Data Partitioning

4.5.1. Datasets

We use standard image classification benchmarks available through the PyTorch `torchvision` library[4]: EMNIST (balanced split) and CIFAR-100 (see Table 4.1). Each dataset is partitioned across nodes without duplication. For evaluation, we keep a global test set and, for MIA, a holdout non-member set per node sampled from the same distribution as that node's training data. CIFAR-100 is used for all reported final experiments as it is a more challenging classification task (100 classes) that tends to produce more overfitting and thus a stronger MIA signal, allowing for a clearer evaluation of the proposed defenses. EMNIST is used only for validation, debugging and development as it is computationally cheaper and trains significantly faster than CIFAR-100.

Table 4.1: Datasets used in the experiments.

Dataset	Classes	Train/Test	Image size	Role
EMNIST	47	113k / 19k	28×28	Testing/development
CIFAR-100	100	50k / 10k	32×32	Final experiments

[4] `https://pytorch.org/vision/stable/datasets.html`

4.5.2. Data Partitioning

The framework supports two data partitioning strategies. The final reported experiments use IID partitioning, while Dirichlet partitioning was implemented and used during testing.

- **IID**: The dataset is shuffled and split into equal sized subsets, one per node. Class labels are balanced and equally distributed across nodes.
- **Dirichlet**: The dataset is shuffled and split into non-IID subsets. The Dirichlet distribution is used with concentration parameter α, which controls the class label distribution across nodes. Smaller α yields larger label skew, while larger α approaches IID.

By default, Dirichlet sampling produces datasets of varying sizes across nodes. To control for this, we take the smallest dataset size across all nodes and cap every node's dataset at this size. This ensures that differences in vulnerability are caused by topology changes rather than variations in local dataset size.

With CIFAR-100, local node datasets can get very small. The CIFAR-100 training set contains 50,000 samples across 100 classes. For membership inference evaluation, 20% of the training data is held out as non-members, leaving 40,000 samples for training. Under IID partitioning across 100 nodes without duplication, each node receives approximately 400 training samples. Since CIFAR-100 contains 100 classes, assuming IID data partitioning, this corresponds to roughly 4 samples per class on average for each node.

Under Dirichlet partitioning, the number of samples per node can be even smaller due to the dataset size normalization mentioned earlier, where all nodes are capped at the size of the smallest partition. For this reason, the final experiments are restricted to IID partitioning.

The non-member set that is held out for every node is sampled from the same distribution as that node's training data. This is particularly important when the class distribution differs across nodes, such as in Dirichlet partitioning. If a global non-member set were used for membership inference evaluation instead, the attack could exploit differences in data distribution rather than actual membership information leakage from the victim's model.

4.5.3. Models

We use small Convolutional Neural Networks (CNNs) for both EMNIST and CIFAR-100, all implemented in `PyTorch`. The same model architecture is shared by all nodes and is chosen to match the dataset. Relatively small models are used because local datasets are small in the decentralized setting. When datasets such as CIFAR-100 are distributed across many nodes without duplication, each node only receives a few hundred samples. Since CIFAR-100 contains 100 classes, this corresponds to only 3-5 samples per class on average. Using overly large models in this situation would lead to extreme overparameterization and unstable training. Smaller models are better suited to the limited amount of local training data in our experiments.

4.6. Experimental Design and Evaluation Metrics

4.6.1. Evaluation Metrics

For our analysis, we distinguish between node-level metrics, computed separately for each victim node, and network-level metrics, which aggregate node-level results across the network.

Node-level metrics. For each evaluated round and each victim node we record the following:

- **Average AUC**: since we consider every neighbor as a potential attacker, we record the average AUC of all attackers.
- **Maximum AUC**: we record the AUC of the worst-case neighbor, and the identity of the neighbor achieving the maximum.
- **Training and test accuracy**: we record the classification accuracy on a node's training set and their accuracy on the global test set. In addition to the standard top-1 accuracy, we report top-5 accuracy as the primary utility metric in all experiments. This metric is particularly suited for CIFAR-100, which is a relatively challenging classification task, as it counts a

prediction as correct if the true label is in the top 5 predicted probabilities. These metrics allow us to track convergence and utility of the model.

Running membership inference attacks is computationally expensive. Therefore, we evaluate MIA periodically rather than after every training round. Running the attack for every victim-neighbor pair would scale approximately as

$$O(|V| \cdot \bar{d} \cdot C_{\text{MIA}})$$

where $|V|$ denotes the number of nodes in the network, $\bar{d}$ the average node degree, and C_{MIA} the cost of a single MIA evaluation. Moreover, differences between model states between consecutive rounds are small enough that evaluating MIA at every round would be unlikely to produce substantially different results.

We also record graph-theoretic properties (degree, betweenness, closeness, k-core) per node to study the correlation between these graph properties and MIA vulnerability. In the final analaysis, degree is used as the main explanatory variable as it directly controls the amount of information communicated under topology-aware chunking and showed the strongest and most consistent correlation with privacy leakage. The remaining properties were analyzed but showed less interesting or consistent results, and are therefore not the focus of the research.

Network-level metrics. To summarize privacy leakage and utility at the network level, we aggregate node-level metrics over sets of victim nodes. For example, **Mean Maximum AUC** corresponds to the mean of the node-level Maximum AUC values, while the **Mean Average AUC** corresponds to the mean of the node-level Average AUC values. The **Mean Top-5 Test Accuracy** is obtained by averaging the node-level Top-5 test accuracy across nodes. These aggregations can be computed over the entire network or over subsets of nodes, like degree-based groups.

4.6.2. Scalar privacy-utility score

Privacy and utility are naturally competing objectives. Lower membership leakage (AUC close to 0.5), often coincides with a lower test accuracy, and vice versa: the model with the strongest accuracy is not always the most private. To summarize this tradeoff into a single score that can be compared across different training configurations, we define a lightweight privacy-utility score, scaled by a weight $\lambda \in [0, 1]$.

Let $u \in [0, 1]$ denote mean top-5 test accuracy and let $a \in [0, 1]$ denote the mean maximum MIA AUC across victim nodes. We map AUC to a risk score $r \in [0, 1]$ as

$$r \;=\; \max\{0,\; 2a - 1\},$$

so that $r = 0$ when attacks are no better than random guessing ($a = 0.5$), and r increases toward 1 as the attack approaches perfect separation ($a \to 1$). The privacy-utility score is defined as

$$S_\lambda \;=\; (1 - \lambda)\, u \;-\; \lambda\, r.$$

Larger values of S_λ are better. The two extreme values of λ have intuitive interpretation. $\lambda = 0$ optimizes for pure utility ($S_0 = u$), whereas $\lambda = 1$ ignores accuracy and rewards only low leakage ($S_1 = -r$). Intermediate values blend the two objectives.

This score is intended as a practical, context-dependent metric rather than a one-size-fits-all one. The weight λ should be chosen based on the deployment context. Section 5.3 applies this score across three values of λ for three different narratives, to illustrate how the optimal hyper parameters (σ, C) change as the emphasis moves from utility toward privacy.

- **Utility-focused ($\lambda = 0.25$)**: In some applications, prediction accuracy is the primary concern and moderate privacy leakage may be acceptable. Examples include on-device personalization or keyboard next word prediction. In these settings, a small reduction in privacy risk may not justify a large decrease in utility.
- **Balanced ($\lambda = 0.5$)**: Many practical systems require a compromise between performance and privacy risk. Example include general-purpose recommendation systems or computer vision systems. In these scenarios, both utility and privacy are considered equally important.

- **Privacy-first ($\lambda = 0.75$)**: In high-sensitive applications, minimizing privacy risks is substantially more important than maximizing performance. Examples include models trained on medical records, employee data or financial information. In such settings, a moderate reduction in performance may be acceptable if it significantly reduces risk of membership inference.

4.6.3. Experimental Configurations

The experiments are organized in several stages in order to study the relationship between network structure, the proposed mechanisms, and membership inference vulnerability. Unless stated otherwise, all final experiments are done on CIFAR-100 using a small CNN, $N = 100$ peers, IID data partitioning, and the loss-based MIA evaluated at the final communication round. Results are averaged over five random seeds. For consistency across randomly generated graphs, the topology seed is kept fixed, while the training and data-partitioning seed is varied.

- **Topology analysis.** We first analyze how communication topology influence membership inference vulnerability in decentralized learning. Experiments are done on several illustrative graph families, including ring, star, grid, and fully connected graphs, as mentioned in Section 4.1.1. In addition, d-regular graphs with $d \in \{3, 10, 25\}$ are used to study the effect of uniform node degree, and Erdős-Rényi graphs with $p \in \{0.04, 0.08, 0.16, 0.32\}$. For these graphs, the edge probability p is varied to experiment with different graph density levels. These topology-analysis experiments compare the undefended baseline with topology-aware chunking only, without added noise. This provides insight into how node connectivity and graph structure affect membership inference vulnerability for individual nodes as well as for the network as a whole. They also motivate the design of the proposed ChunkDP.
- **Defense ablation.** Based on the observations from the topology analysis, we evaluate the effect of different privacy mechanisms on utility and MIA vulnerability. The ablation experiments are conducted on Erdős-Rényi graphs with $p \in \{0.08, 0.16\}$. Five configurations are considered:
  1. No defenses (baseline)
  2. Differential privacy only (Section 4.3.1)
  3. Topology-aware chunking only (Section 4.3.2)
  4. Fixed-K chunking only (Section 4.3.3, evaluated for different numbers of chunks $K \in \{8, 16, 32, 64, 128\}$)
  5. ChunkDP combining topology-aware chunking and differential privacy (Section 4.4)

  Unless stated otherwise, DP-based configurations use noise multiplier $\sigma = 0.5$ and clipping threshold $C = 1$. This comparison allows us to isolate the effect of each mechanism and evaluate how they affect the privacy-utility tradeoff. The fixed-K chunking is included to compare the results of the topology-aware mechanisms against an existing topology-independent defense technique.
- **Privacy parameter sweep.** Finally, we evaluate different combinations of differential privacy parameters for ChunkDP. In particular, we sweep the base noise multiplier $\sigma_0 \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$ and the gradient clipping threshold $C \in \{1.0, 1.5, 2.0\}$. The sweep is conducted on Erdős-Rényi graphs with $p \in \{0.08, 0.16\}$. These experiments are used to identify ChunkDP configurations with favorable privacy-utility tradeoffs under different values of the privacy-utility weight λ.

4.6.4. Reproducibility

We use two type of random seeds in our experiments. The first seed is used for data partitioning, training and reproducibility of noise and chunking. The second seed is used for topology generation. The reason for using two separate seeds is that, although we want to vary training, data partitioning and other components to evaluate the robustness of results, we want to keep the topology fixed so that all experiments are evaluated on the same network structure. Otherwise, it would be difficult

to draw conclusions about the vulnerability of a specific node if its graph properties changed every run.

When MIA is run after a round, the training RNG state is saved before the MIA phase and restored afterward. This is necessary because the MIA procedure consumes randomness that changes the RNG stage. By restoring the saved state, the training trajectory is unchanged by the MIA step.

All hyperparameters are set via command-line arguments and logged for reproducibility. These include training parameters (e.g., learning rate, batch size, local epochs, momentum), privacy parameters (e.g., DP noise multiplier and clipping threshold), communication parameters (e.g., chunking options and mixing coefficient β), topology parameters, data partitioning parameters and MIA settings. Results are written to CSV files that include the configuration (dataset, model, topology, DP/chunk flags, noise, seed, etc.) for later analysis and plotting.

All simulation code, published CSV results, plotting scripts, and thesis figures are available in the accompanying GitHub repository [38]. The results directory contains all experiment results used in this thesis. These results, as well as all figures and tables presented in this thesis, can be reproduced by following the instructions documented in the repository.

5

Results

This section presents the experimental results of the evaluation of the research question: In decentralized learning, how does communication topology influence vulnerability to membership inference attacks and model utility, and can topology-aware defense mechanisms improve the privacy-utility tradeoff compared to topology-agnostic approaches? The results are organized in three parts. First, we present a systematic analysis of the effect of communication topology on privacy leakage and utility for both standard decentralized learning and topology-aware chunking. Second, we evaluate ChunkDP and compare it to the undefended baseline, differential privacy, topology-aware chunking, and topology-independent fixed-K chunking. Finally, we perform a parameter sweep to identify ChunkDP configurations that achieve favorable privacy-utility tradeoffs under different settings. Throughout this section, reported accuracy refers to top-5 accuracy.

5.1. Topology Analysis

This section begins with the first building block towards answering the research question on topology-aware privacy amplification: a systematic overview of membership inference attack vulnerability and model utility under different decentralized communication topologies. We contrast standard decentralized learning with the topology-aware chunking mechanism (chunking only, no DP). These experiments follow the topology analysis as described in Section 4.6.3. We use illustrative deterministic graphs (ring, star, grid, fully connected), d-regular graphs with varying degrees, and Erdős-Rényi graphs where the edge probability p controls graph density. Reported metrics are means over 5 random seeds, with error bars showing the deviation across seeds.

5.1.1. Deterministic topologies

Figure 5.1 compares baseline training (no defense mechanism) to training with the topology-aware chunking mechanism across ring, star, grid and fully connected (clique) graphs. Under the baseline, mean maximum MIA AUC is very high across all four graphs (0.97-0.99). This indicates strong membership leakage regardless of whether the graph is dense (clique) or sparse (ring) in this experimental setting.

Topology-aware chunking significantly reduces leakages for the grid and especially for the fully connected graph: mean maximum MIA AUC drops from about 0.98 to 0.87 on the grid and from about 0.97 to 0.61 on the fully connected graph. This also comes with some decrease in mean top-5 test accuracy (grid: $\sim 0.46 \rightarrow 0.44$; full: $\sim 0.49 \rightarrow 0.46$). Thus, for these topologies, topology-aware chunking significantly reduces leakage while largely preserving utility at the network level.

The ring network shows a different result. Leakage remains high even with chunking (MIA AUC $\sim 0.99 \rightarrow 0.97$), so the decrease is modest compared with the grid or fully connected graph. The star graph is an outlier in the opposite direction. Topology-aware chunking severely degrades utility (mean accuracy $\sim 0.42 \rightarrow 0.27$) while increasing MIA AUC ($\sim 0.98 \rightarrow 0.99$).

Together, these results show that the impact of the topology-aware chunking mechanism varies substantially between topologies.

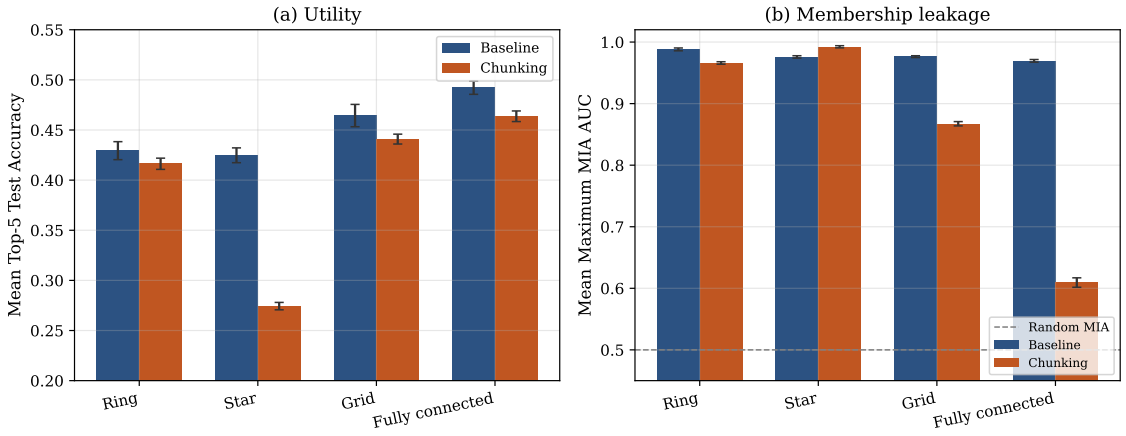


Figure 5.1: Deterministic graphs (ring, star, grid, fully connected): mean top-5 test accuracy (left) and mean maximum MIA AUC (right), comparing baseline decentralized learning and using the topology-aware chunking defense mechanism.

5.1.2. Regular graphs: role of uniform degree

Figure 5.2 shows the effect of degree for d-regular graphs ($d \in \{3, 10, 25\}$) on utility and membership inference vulnerability.

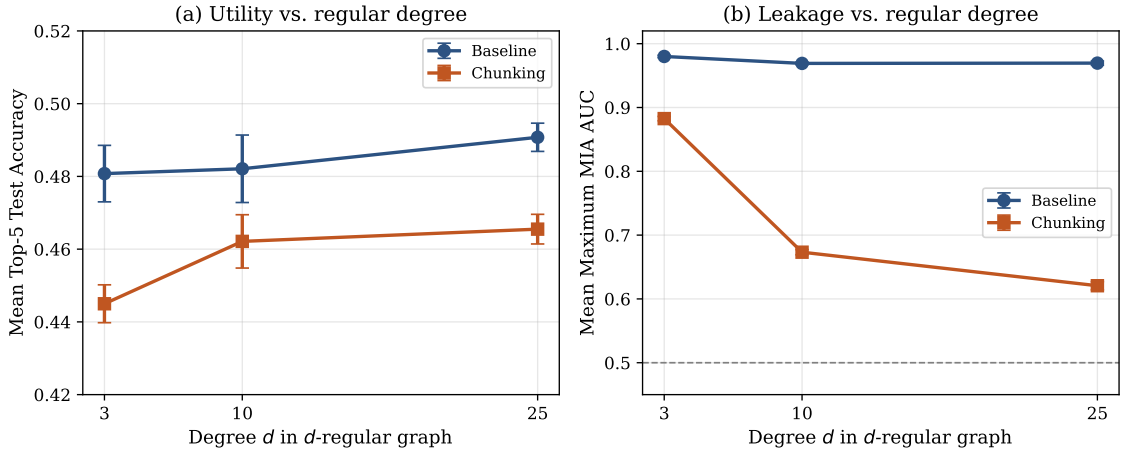


Figure 5.2: d-regular graphs: mean top-5 test accuracy (left) and mean maximum MIA AUC (right) as a function of degree d, comparing baseline decentralized learning and using the topology-aware chunking defense mechanism.

The baseline shows similarly high leakage (AUC $\approx$ 0.97-0.98) and comparable accuracy ($\approx$ 0.48-0.49) for all degrees. This indicates that change in degree alone does not strongly affect baseline privacy risk in this setting.

With topology-aware chunking, both utility and leakage depend on d. Higher degree is correlated with lower MIA AUC (from $\sim$ 0.88 at $d = 3$ to $\sim$ 0.62 at $d = 25$) and slightly higher accuracy ($\sim$ 0.45 to $\sim$ 0.47). In other words, the privacy benefits offered by topology-aware chunking get stronger as the graph gets denser, while the accuracy penalty relative to the baseline gets smaller. This is consistent with the fact that nodes with more neighbors distribute their updates into smaller chunks, reducing the information any single neighbor sees.

5.1.3. Erdős-Rényi graphs

To study graphs with controlled density but heterogeneous node distribution, we use the ER graphs and vary the edge probability $p \in \{0.04, 0.08, 0.16, 0.32\}$. This corresponds to an expected degree

of 4, 8, 16 and 32 per node. Figure 5.3 summarizes network level metrics. This consists of the mean top-5 test accuracy and mean maximum MIA AUC.

Under the baseline, accuracy rises mildly with p (from about 0.47 at p=0.04 to about 0.49 at p=0.32), while MIA AUC stays the about the same (roughly 0.97-0.98) across different values of p. Topology-aware chunking decreases leakage more strongly as p increases, with AUC decreasing from 0.87 at p=0.04 to about 0.62 at p=0.32, while accuracy under chunking moves from 0.41 to 0.46. Thus, as the ER graph becomes denser, the privacy benefit of topology-aware chunking strengthens and the utility gap to the baseline narrows. This is consistent with the trend observed for d-regular graphs (Section 5.1.2).

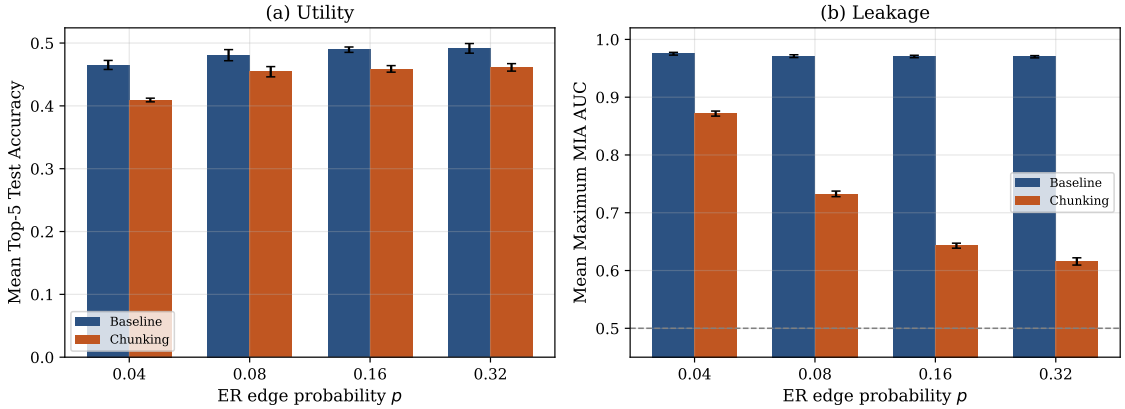


Figure 5.3: Erdős-Rényi graphs: mean top-5 test accuracy (left) and mean maximum MIA AUC (right) vs. edge probability p, comparing baseline decentralized learning and using the topology-aware chunking defense mechanism.

5.1.4. Cross-topology privacy-utility snapshot

Figure 5.4 shows each graph family as a single point per condition (baseline, topology-aware chunking), plotting utility against MIA AUC. Ideal points lie toward the bottom right (high accuracy, low AUC). Baseline collapses in a very narrow band with very high leakage and similar accuracy. Ring and star fall out of this cluster, showing similarly high leakage but slightly lower accuracy.

Topology-aware chunking shows a more scattered picture. Most graphs display significantly lower leakage than their baseline counterparts, with the exception of the star and ring network. In contrast, other graph families like the ER and d-regular graphs show decreasing leakage as density increases. The lowest leakage is observed in the fully connected graph (0.61) while ER (p=0.32) and 25-regular show similar results ($\approx$ 0.62).

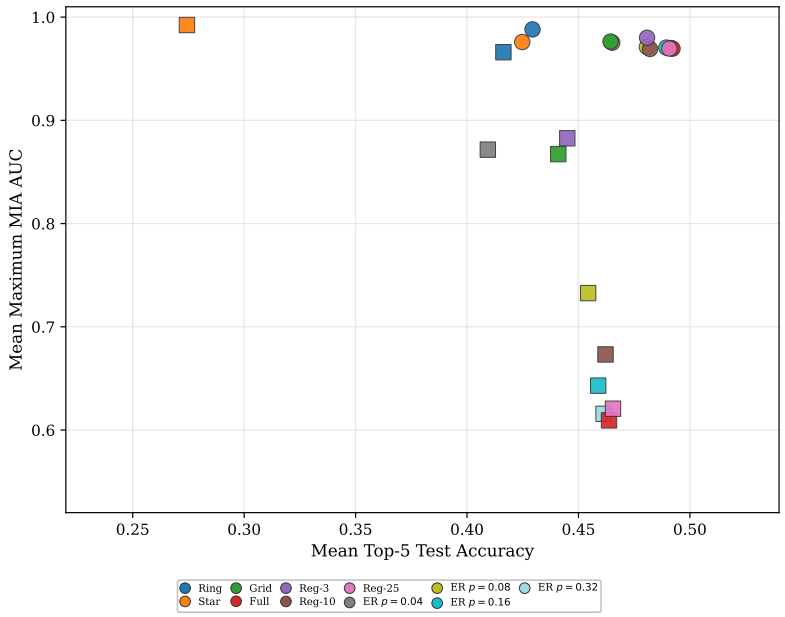


Figure 5.4: Privacy-utility snapshot: each label denotes a graph family. Circles: baseline; squares: topology-aware chunking.

5.1.5. Node-level heterogeneity: hubs, degrees, and roles

Network wide averages can obscure large disparities between peers. We therefore analyze vulnerability at the level of structural roles in topologies with natural partitions (star, grid, ER), and quantify how much spread in vulnerability there is in graphs with uniform degree (ring, clique, d-regular). Table 5.4 shows the corresponding network-level results for quick reference.

Star: hub versus leaves. In the star network, one central node has degree $N-1$ and the remaining $N-1$ nodes (leaves) have degree 1. Under the baseline, hub and leaves show similar membership leakage (MIA AUC ≈ 0.98) but significantly different utility. The hub reaches a mean top-5 test accuracy of around 0.58, while a typical leaf has an accuracy of around 0.42. Topology-aware chunking produces two distinct privacy behaviors that are invisible from the network average alone. The hub's mean maximum MIA AUC drops to roughly 0.61, while leaves remain extremely vulnerable (≈ 1.00) (Table 5.1).

At the same time, topology-aware chunking collapses accuracy for both roles, with the hub near 0.30 and the leaves near 0.27. Thus, the poor result of the topology-aware chunking mechanism for the star topology in Section 5.1.1 does not reflect a uniform effect. Instead, it is an uneven situation in which most nodes (the leaves) remain highly vulnerable, while the hub enjoys significant reduced leakage. This highlights the importance of role-aware analysis, as network-level MIA AUC scores can mask substantial differences between individual nodes.

The star graph demonstrates a fundamental limitation of topology-aware chunking. Although the central node benefits from reduced leakage, the leaves remain highly vulnerable because their degree is too low to effectively distribute their model across neighbors using topology-aware chunking.

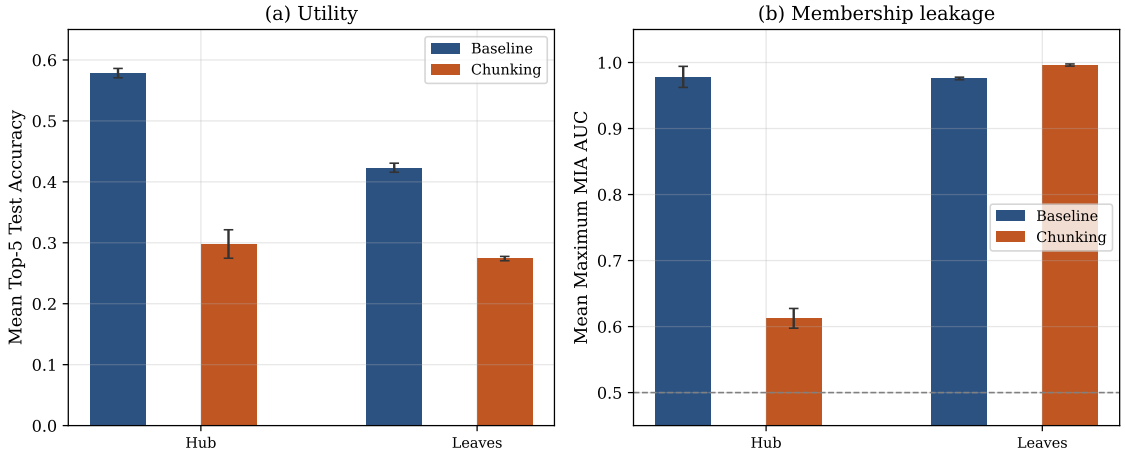


Figure 5.5: Star topology: mean top-5 test accuracy (left) and mean maximum MIA AUC (right) averaged within the hub and within the leaves.

Table 5.1: Star topology: hub (degree $N-1$) vs. leaves (degree 1). Metrics are averaged over nodes in the role.

Mode	Role	Mean Top-5 Accuracy	Mean Average MIA AUC	Mean Maximum MIA AUC
Baseline	Hub	0.578 ± 0.008	0.978 ± 0.016	0.978 ± 0.016
	Leaves	0.423 ± 0.007	0.976 ± 0.002	0.976 ± 0.002
Chunking	Hub	0.298 ± 0.023	0.546 ± 0.009	0.612 ± 0.015
	Leaves	0.274 ± 0.004	0.996 ± 0.002	0.996 ± 0.002

Grid: corners, edges, and interior. On the 10×10 grid, we can distinguish corners (d=2), boundary edges (d=3), and interior nodes (d=4). Under the baseline, leakage is high (AUC ≈ 0.97-0.99), with only small variations between groups. Corners seem to leak slightly more than interior nodes.

Topology-aware chunking preserves this separation in leakage, but amplifies the separation. Interior nodes enjoy the biggest reduction in MIA AUC (≈ 0.97 → 0.84), while corners benefit the least (≈ 0.99 → 0.96). Accuracy drops slightly for all groups. These trends are illustrated in Figure 5.6 and detailed in Table 5.2.

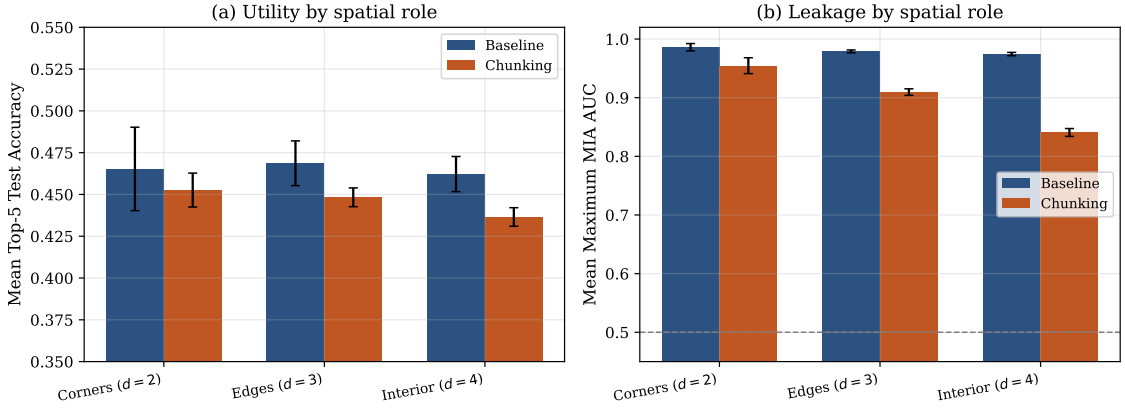


Figure 5.6: Grid topology: metrics stratified by node type (corner / edge / interior).

Table 5.2: Grid topology: nodes grouped by node type (corner / edge / interior on the 10×10 grid).

Mode	Role	Mean Top-5 Accuracy	Mean Average MIA AUC	Mean Maximum MIA AUC
Baseline	Corners (d=2)	0.465 ± 0.025	0.986 ± 0.006	0.986 ± 0.006
	Edges (d=3)	0.469 ± 0.013	0.979 ± 0.002	0.979 ± 0.002
	Interior (d=4)	0.462 ± 0.011	0.974 ± 0.003	0.974 ± 0.003
Chunking	Corners (d=2)	0.453 ± 0.010	0.945 ± 0.018	0.955 ± 0.014
	Edges (d=3)	0.448 ± 0.006	0.878 ± 0.005	0.910 ± 0.006
	Interior (d=4)	0.437 ± 0.006	0.800 ± 0.008	0.841 ± 0.007

Erdős-Rényi graphs: heterogeneity by node degree. For ER graphs, we group nodes by fixed degree bins. Table A.1 in Appendix A and Figure 5.7 show, for each bin, the mean top-5 test accuracy and mean maximum MIA AUC. For each value of p, nodes are grouped into four degree bins to reflect the observed degree distribution. This provides a rough but consistent stratification of node connectivity.

Across values of p, the baseline shows high leakage in every bin, typically having MIA AUC of approximately 0.97-0.98. Under topology-aware chunking, higher degree bins display lower MIA AUC than lower degree bins at the same p. The lowest degree bins at small p often have near-perfect AUC (≈ 1.0 for p=0.04, degree 1, and 0.93 for degree 2). As p increases, nodes fall into higher-degree bins, and leakage correspondingly decreases.

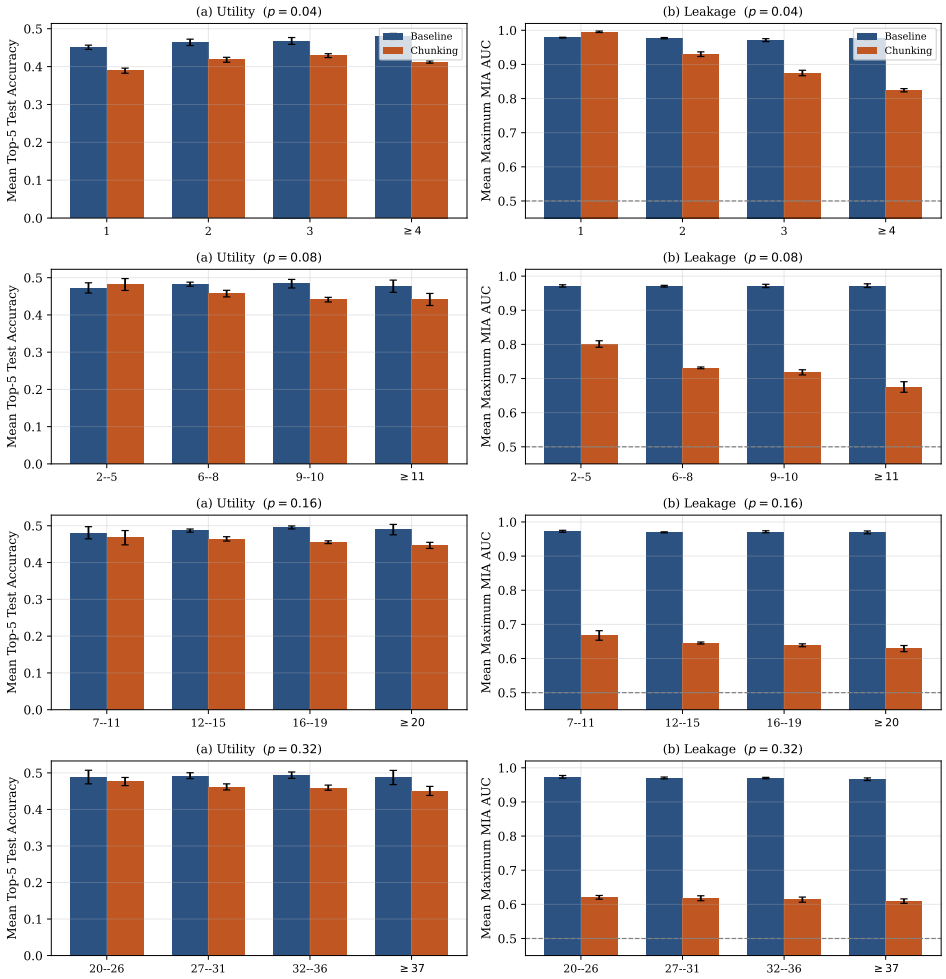


Figure 5.7: Erdős-Rényi graphs: node-level heterogeneity by degree bin (one row per p). Left: mean top-5 test accuracy; right: mean maximum MIA AUC. Each bar aggregates all nodes whose degree falls in the bin for that run.

Ring, clique, and d-regular graphs: uniform roles but unequal spread. When every node has the same degree (ring, fully connected, d-regular), we cannot classify nodes into groups based on degree. Table 5.3 uses node-level maximum MIA AUC and shows the mean for the graph and the standard deviation across nodes. The first metric reflects variation across seeds, while the second captures within-graph variation across nodes.

Under the baseline, within-graph variation is modest on all topologies (0.15-0.19). Topology-aware chunking leaves the ring's variation essentially unchanged but increases it on d-regular graphs (e.g. mean within-run SD rising from ≈ 0.017 to ≈ 0.034 for d=3) and fully connected graphs.

Finally, Figure 5.8 compares the distribution of per-node maximum MIA AUC of the star and the ring. Under topology-aware chunking, the star shows a highly skewed distribution. Most nodes cluster at very high leakage, while there are nodes that achieve significantly less AUC, corresponding to the hub-leaf asymmetry. In contrast, the ring shows a distribution with consistently high leakage across nodes and no outliers, although the spread around the mean is slightly broader than the concentrated mass observed in the star.

Under the baseline, both topologies have high leakage, with a greater variation in the ring.

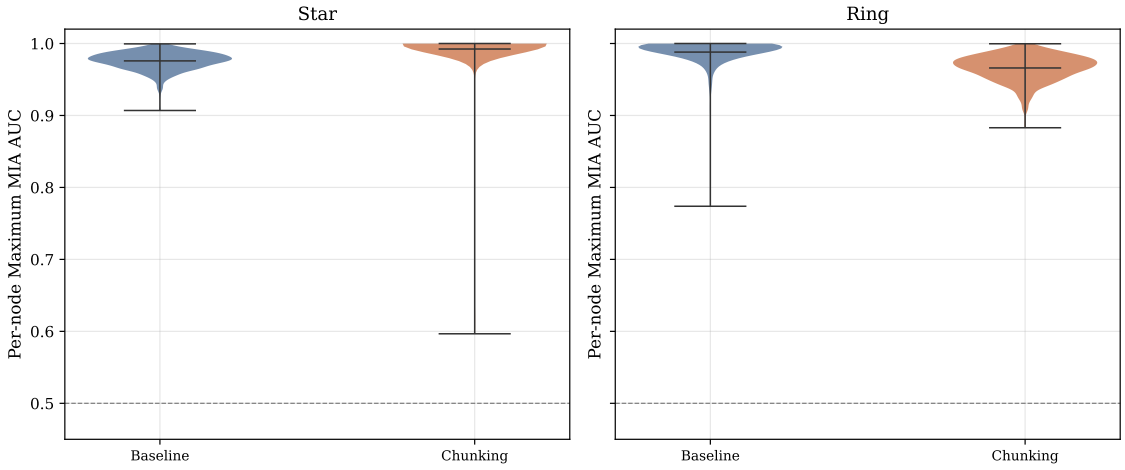


Figure 5.8: Distribution of per-node MIA AUC. Star vs. ring; baseline vs. topology-aware chunking. Violin width reflects density of nodes at each AUC level.

Table 5.3: Graphs with homogenous node distribution: mean maximum MIA AUC and the standard deviation of per-node maximum MIA AUC within each run, summarized as mean ± std across seeds. Low std. across nodes reflects limited variation in node-level vulnerability within each topology.

Topology	Mode	Mean Maximum MIA AUC	Std. across nodes
Ring	Baseline	0.988 ± 0.002	0.019 ± 0.009
	Chunking	0.966 ± 0.002	0.018 ± 0.001
Fully conn.	Baseline	0.969 ± 0.002	0.015 ± 0.002
	Chunking	0.609 ± 0.008	0.021 ± 0.003
3-regular	Baseline	0.980 ± 0.002	0.017 ± 0.002
	Chunking	0.883 ± 0.003	0.034 ± 0.002
10-regular	Baseline	0.969 ± 0.002	0.016 ± 0.002
	Chunking	0.673 ± 0.004	0.030 ± 0.003
25-regular	Baseline	0.970 ± 0.003	0.015 ± 0.001
	Chunking	0.621 ± 0.006	0.025 ± 0.003

Summary. Across the considered topologies, topology-aware chunking is most effective at reducing membership inference leakage in denser graphs, while providing limited protection for low-degree nodes. Heterogeneous structures such as the star graph demonstrate strong fluctuations in protection between different types of nodes. These observations highlight a key limitation of

topology-aware chunking, especially for low degree nodes, and motivate the ChunkDP mechanism studied in the next section.

Table 5.4: Network-level summary: for each topology and training mode, mean top-5 test accuracy and mean maximum/average MIA AUC.

Topology	Mode	Mean Top-5 Accuracy	Mean Maximum MIA AUC	Mean Average MIA AUC
Ring	Baseline	0.429 ± 0.009	0.988 ± 0.002	0.988 ± 0.002
	Chunking	0.416 ± 0.006	0.966 ± 0.002	0.957 ± 0.002
Star	Baseline	0.425 ± 0.007	0.976 ± 0.002	0.976 ± 0.002
	Chunking	0.274 ± 0.004	0.992 ± 0.002	0.992 ± 0.002
Grid	Baseline	0.464 ± 0.011	0.976 ± 0.001	0.976 ± 0.002
	Chunking	0.441 ± 0.005	0.867 ± 0.004	0.831 ± 0.005
Fully connected	Baseline	0.492 ± 0.007	0.969 ± 0.002	0.969 ± 0.002
	Chunking	0.464 ± 0.005	0.609 ± 0.008	0.556 ± 0.007
3-regular	Baseline	0.481 ± 0.008	0.980 ± 0.002	0.980 ± 0.002
	Chunking	0.445 ± 0.005	0.883 ± 0.003	0.851 ± 0.005
10-regular	Baseline	0.482 ± 0.009	0.969 ± 0.002	0.969 ± 0.002
	Chunking	0.462 ± 0.007	0.673 ± 0.004	0.632 ± 0.004
25-regular	Baseline	0.491 ± 0.004	0.970 ± 0.003	0.970 ± 0.003
	Chunking	0.466 ± 0.004	0.621 ± 0.006	0.578 ± 0.006
ER ($p = 0.04$)	Baseline	0.465 ± 0.007	0.975 ± 0.002	0.975 ± 0.002
	Chunking	0.409 ± 0.003	0.872 ± 0.004	0.796 ± 0.005
ER ($p = 0.08$)	Baseline	0.481 ± 0.009	0.971 ± 0.002	0.971 ± 0.002
	Chunking	0.454 ± 0.008	0.733 ± 0.005	0.666 ± 0.005
ER ($p = 0.16$)	Baseline	0.489 ± 0.004	0.970 ± 0.002	0.970 ± 0.002
	Chunking	0.459 ± 0.005	0.643 ± 0.004	0.600 ± 0.005
ER ($p = 0.32$)	Baseline	0.492 ± 0.008	0.970 ± 0.002	0.970 ± 0.002
	Chunking	0.461 ± 0.006	0.616 ± 0.006	0.572 ± 0.007

5.2. ChunkDP Ablation

The topology results in Section 5.1 show a clear relationship between privacy leakage and node degree when using the topology-aware chunking mechanism. Building on these observations, we evaluate the ChunkDP mechanism introduced in Section 4.4. Low-degree nodes, who reveal relatively large portions of their update under topology-aware chunking, receive stronger noise, while the higher degree nodes rely more on the privacy provided by topology-aware chunking.

We evaluate a 2x2 ablation setup (with/without DP-SGD and with/without topology-aware chunking). We additionally study fixed-K chunking in isolation for varying numbers of chunks K (without DP-SGD; Section 4.6.3). For this, we use Erdős–Rényi graphs with p=0.08 and p=0.16, N=100 peers. Unless stated otherwise, DP runs use noise multiplier σ=0.5 and clipping C=1. We run this ablation setup for five different seeds.

5.2.1. Sparse networks: p=0.08

We first consider an Erdős-Rényi graph with edge probability p=0.08. Figure 5.9 shows mean top-5 test accuracy and mean maximum MIA AUC side by side, while Figure 5.10 plots all conditions in the same privacy-utility scatter plot (accuracy on the x-axis, AUC on the y-axis).

The baseline reaches high accuracy ($u \approx 0.48$) but has near perfect membership inference ($a \approx 0.97$). Topology-aware chunking alone significantly reduces leakage ($a \approx 0.73$) at a small accuracy cost ($u \approx 0.46$). Fixed-K chunking demonstrates a clear K-dependent tradeoff: as K increases from 8 to 128, leakage decreases ($a \approx 0.70 \rightarrow 0.56$) but utility also decreases ($u \approx 0.47 \rightarrow 0.32$). DP-only reduces the attack to near-random guessing on average ($a \approx 0.46$) but collapses utility ($u \approx 0.23$), reflecting the noise injected at every local step. ChunkDP recovers some of the accuracy lost by the

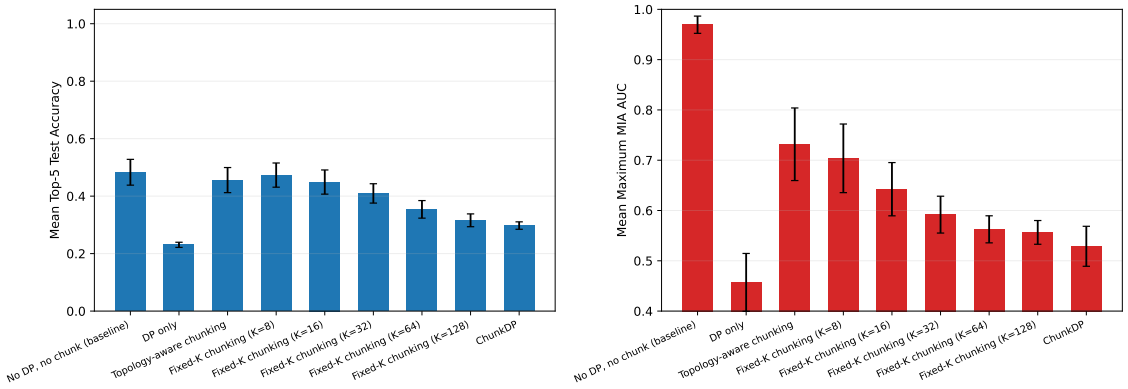


Figure 5.9: ChunkDP ablation: mean top-5 test accuracy (left) and mean maximum MIA AUC (right) (p=0.08). Error bars show the within-run standard deviation of the metric over nodes.

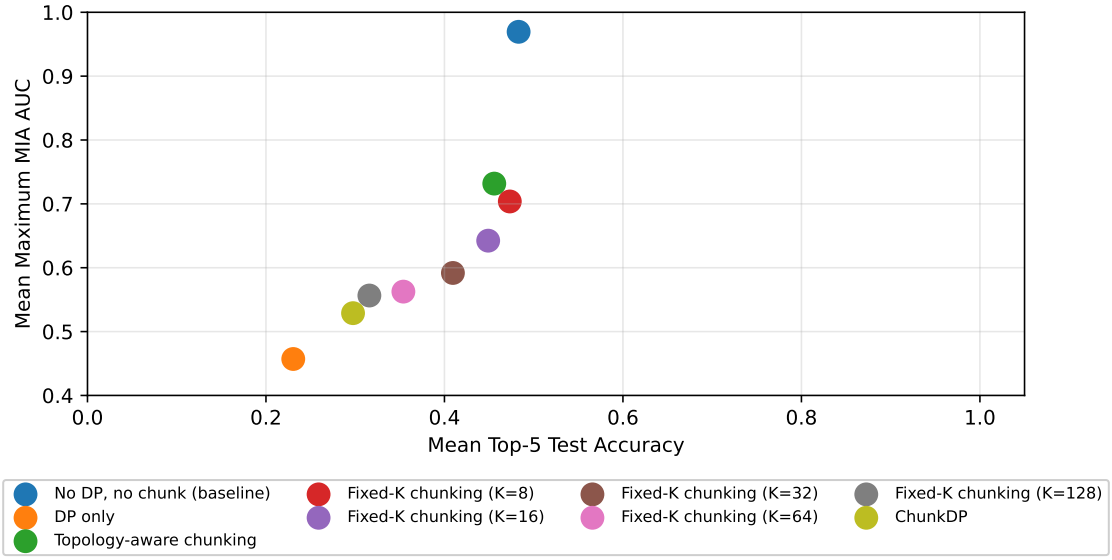


Figure 5.10: Privacy-utility view of the ChunkDP ablation for p=0.08 (ideal toward the bottom-right).

DP-only ($u \approx 0.30$) while leakage is only slightly above random ($a \approx 0.53$).

Interpreting these conditions with the privacy-utility score from Section 4.6.2 demonstrates that different settings favor different configurations. Accuracy-focused settings (λ close to 0) prefer the undefended baseline, while privacy-focused setting (λ close to 1) prefer DP-only. In more balanced settings, ChunkDP ranks ahead of DP-only because it achieves similar privacy for better accuracy.

Figure 5.11 makes this explicit for three fixed tradeoff weights $\lambda \in \{0.25, 0.50, 0.75\}$ corresponding to three different use-cases defined in Section 4.6.2: utility-focused ($\lambda = 0.25$), balanced ($\lambda = 0.50$), and privacy-first ($\lambda = 0.75$).

At λ=0.25 fixed-K chunking with moderate K performs best (highest at K=16, closely followed by K=32). At λ=0.50 ChunkDP narrowly achieves the best score. At λ=0.75 DP-only wins even though accuracy is low. This is because $r \approx 0$. For this setting, ChunkDP ranks second.

None of these λ values reward the undefended baseline, which carries the largest leakage. Even lower values of λ are required for the baseline to be ranked the best.

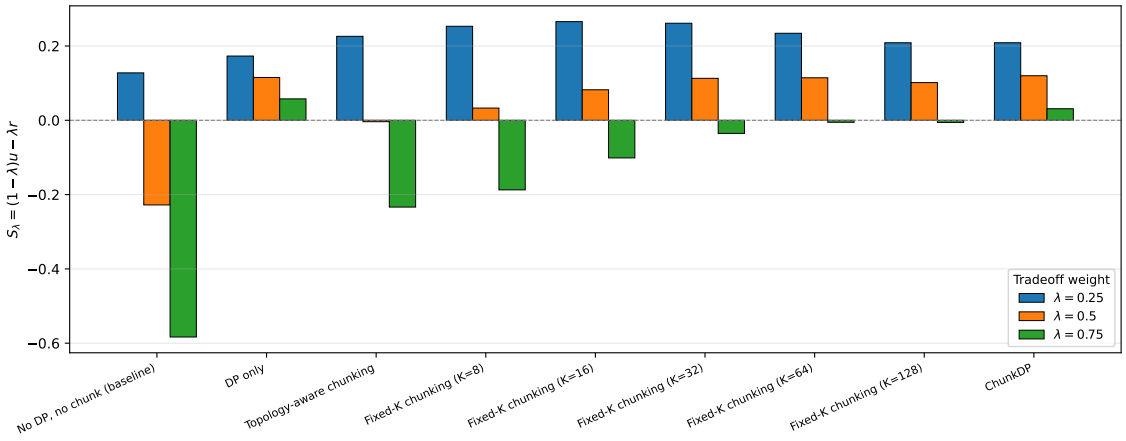


Figure 5.11: ChunkDP ablation (p=0.08): privacy-utility score $S_\lambda = (1-\lambda)u - \lambda r$ for three illustrative tradeoff weights corresponding to three different use-cases.

5.2.2. Denser graph: p=0.16

To get a better understanding of how graph connectivity influences the privacy-utility tradeoff, we repeat the same ablation setup on an ER graph with p=0.16 (expected degree twice as large as for p=0.08 at the same N). Figures 5.12, 5.13, and 5.14 mirror the layout above.

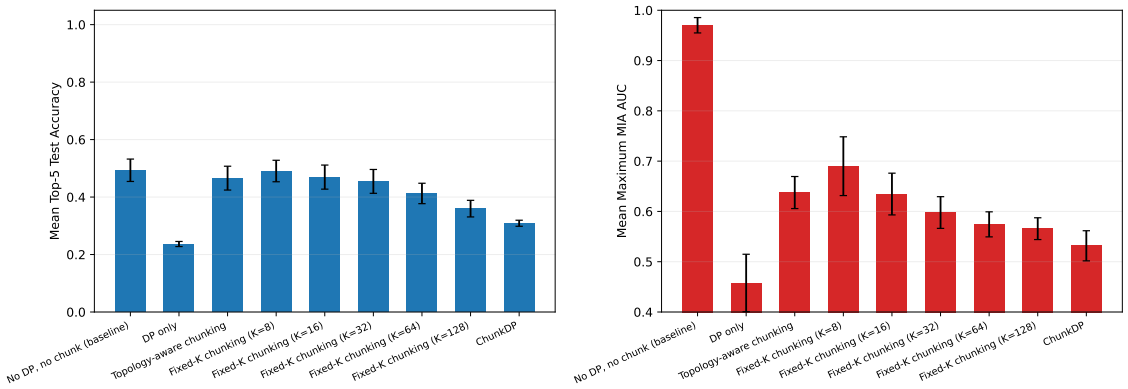


Figure 5.12: ChunkDP ablation: mean top-5 test accuracy (left) and mean maximum MIA AUC (right) (p=0.16). Error bars show the within-run standard deviation of the metric over nodes.

The results look similar, but the denser graph shifts magnitudes slightly. The baseline and chunking options gain a little accuracy, and topology-aware chunking leakage decreases ($a \approx 0.64$ compared with $a \approx 0.73$ at p=0.08). This is consistent with the topology experiments where denser networks resulted in improved benefits of topology-aware chunking. ChunkDP has $u \approx 0.31$ and $a \approx 0.53$, still clearly between topology-aware chunk-only and DP-only.

When comparing the denser graph to the sparser graph for the privacy-utility score at the three tradeoff weights, the results are slightly different. For p=0.16, moderate fixed-K chunking still wins in the utility-focused regime (λ=0.25) with K=32 scoring the highest. At λ=0.50, fixed-K chunking with K=64 scores the highest compared to ChunkDP for p=0.08. Finally, DP-only still leads at λ=0.75 (Figure 5.14).

These results use noise multiplier σ=0.5 and clipping C=1 for ChunkDP. In Section 5.3, we optimize these hyper parameters for each of the three narratives described before.

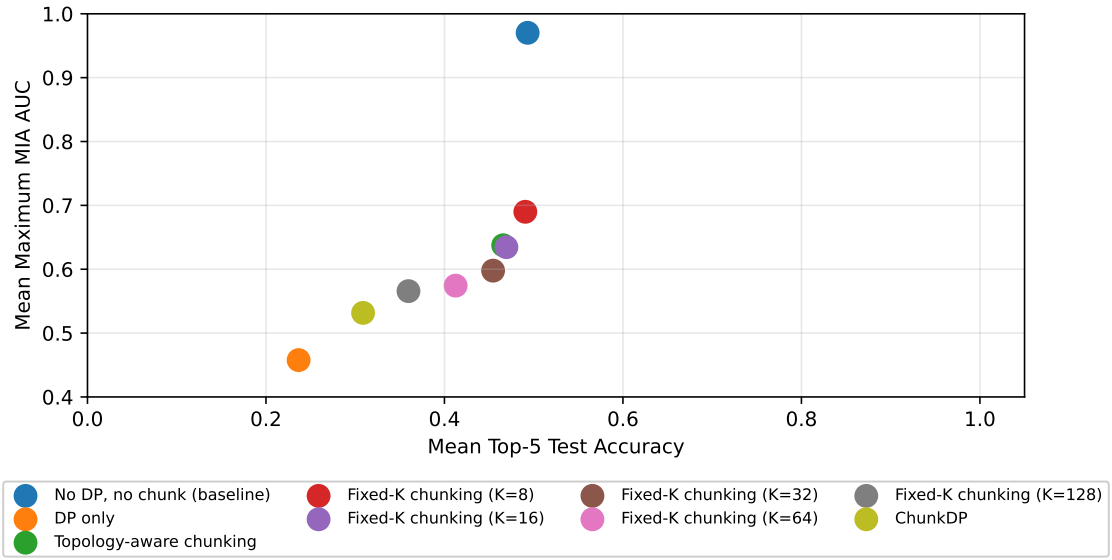


Figure 5.13: Privacy-utility view of the ChunkDP ablation for p=0.16 (ideal toward the bottom-right).

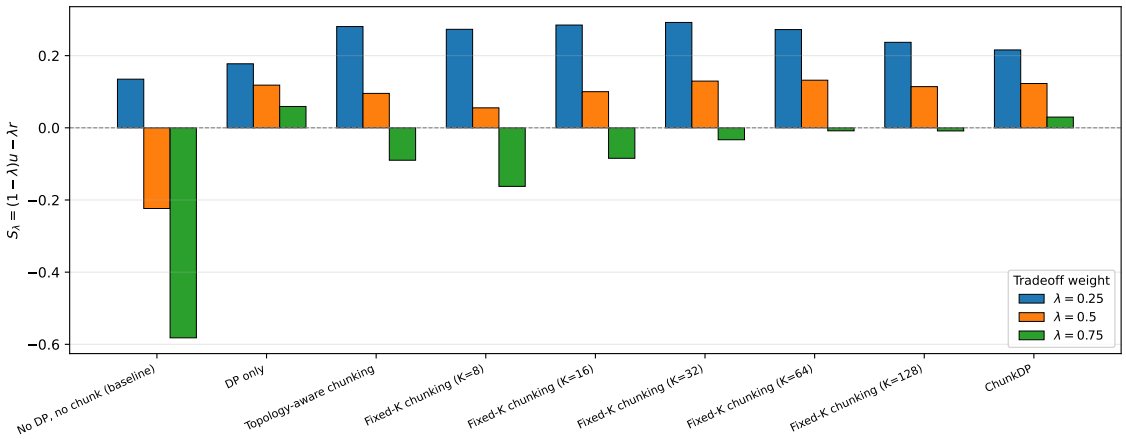


Figure 5.14: ChunkDP ablation (p=0.16): privacy-utility score $S_\lambda = (1-\lambda)u - \lambda r$ for three illustrative tradeoff weights corresponding to three different use-cases.

5.3. Privacy parameter sweep

In this section we optimize ChunkDP by sweeping the DP-SGD base noise multiplier σ_0 together with the per-sample clipping threshold C. Recall that $\sigma_i = \sigma_0/d_i$, so σ_0 directly scales the node-level noise used in ChunkDP.

These parameters directly influence the privacy-utility tradeoff, as σ_0 determines the amount of noise added while C bounds the sensitivity of the gradient of any single sample. We are sweeping these parameters jointly because the influence of these parameters is not independent. Since the noise added has a standard deviation proportional to $\sigma_i C$ (See Algorithm 6), the effects of these parameters are coupled. Thus, they must be tuned jointly to obtain an effective privacy-utility tradeoff.

We keep the other hyperparameters fixed while sweeping σ_0 and C. This isolates how aggressively noise can be traded off against utility while still providing sufficient protection for vulnerable nodes. The sweep follows the configuration described in Section 4.6.3 and sweeps a 7×3 grid $\sigma \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$ and $C \in \{1.0, 1.5, 2.0\}$ (21 cells), each averaged over five seeds.

5.3.1. Sweep at p=0.08

We first evaluate the sweep for a sparse ER graph (p=0.08). Across the 7×3 grid, looser clipping and lower noise generally increases accuracy at the cost of higher MIA AUC, while stricter clipping and higher noise lowers both. Figure 5.15 shows this in a scatter plot and places ChunkDP configurations in context with non-hybrid conditions at the same p: the undefended baseline, DP-only, topology-aware chunking-only, and fixed-K chunking with global chunk counts $K \in \{8, 16, 32, 64, 128\}$.

Figure 5.16 complements this by ranking all (σ, C) combinations by the privacy-utility score S_λ for the three fixed tradeoff weights $\lambda \in \{0.25, 0.50, 0.75\}$ (utility-oriented, balanced, privacy-first) described before. In each panel, the ChunkDP configurations are represented by horizontal bars showing which configuration ranks best for which specific scenario (top=best). For comparison, the listing is extended with three non-hybrid defense mechanisms shown as additional bars in the same panels: DP-only, topology-aware chunking, and and fixed-K chunking for $K \in \{8, 16, 32, 64, 128\}$. For p=0.08, the top-ranked cells are fixed-K chunking with $K = 16$ at λ=0.25, ChunkDP with (0.1, 1.0) at λ=0.50, and DP-only at λ=0.75.

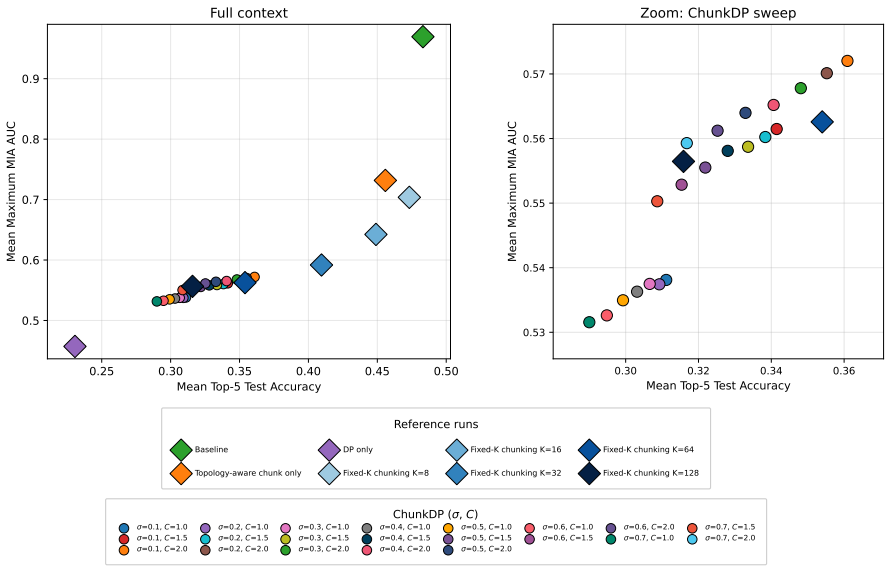


Figure 5.15: ChunkDP noise-clipping sweep (p=0.08). Left: Scatter plot of AUC vs utility for ChunkDP points (circles) with varying noise and clipping threshold, and non-hybrid references (diamonds). Right: zoom on the ChunkDP cluster.

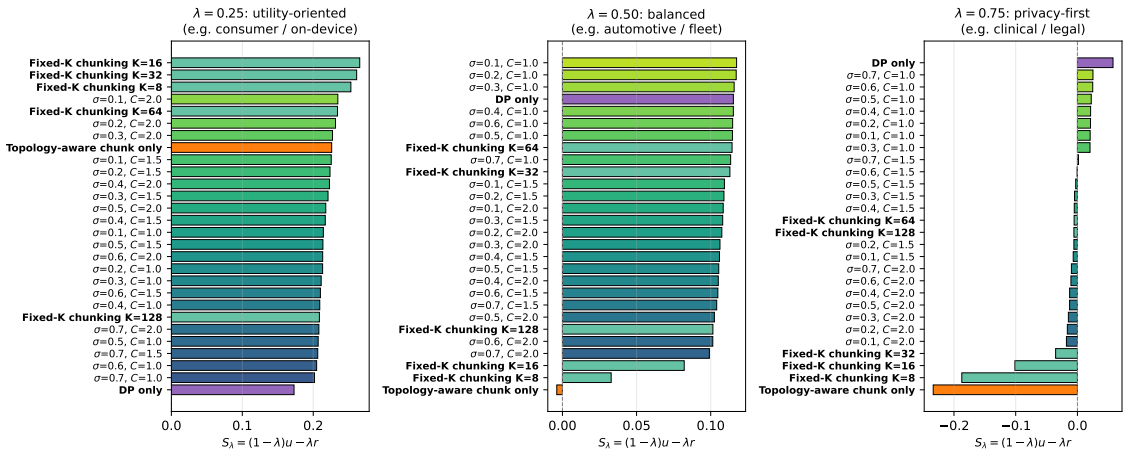


Figure 5.16: ChunkDP noise-clipping sweep (p=0.08). Horizontal bars give privacy-utility score S_λ for each configuration, sorted on highest score first.

5.3.2. Sweep at p=0.16

We repeat the same sweep on a denser ER graph (p=0.16). Figures 5.17 and 5.18 mirror the p=0.08 figures. As in the sparser setting, the ChunkDP points lie between DP only and topology-aware chunking only. However, in the denser setting, the ChunkDP points now have slightly higher accuracy but enjoy less reduction in MIA AUC compared to the chunking options. For p=0.16, the ChunkDP points form visible clusters that are primarily grouped by clipping threshold C, while varying σ mostly moves configurations within each cluster. In the ranking of privacy-utility scores for different scenarios (5.18), utility-focused and balanced scenarios are led by fixed-K chunking (best around K=32 and K=64, respectively), while DP-only remains best for λ=0.75. In the denser setting, all three values of λ prefer less noise compared to the sparser network. Restricting to ChunkDP-only cells, λ=0.25 and λ=0.50 share the same top cell $(0.1, 2.0)$, whereas λ=0.75 prefers $(0.3, 1.0)$.

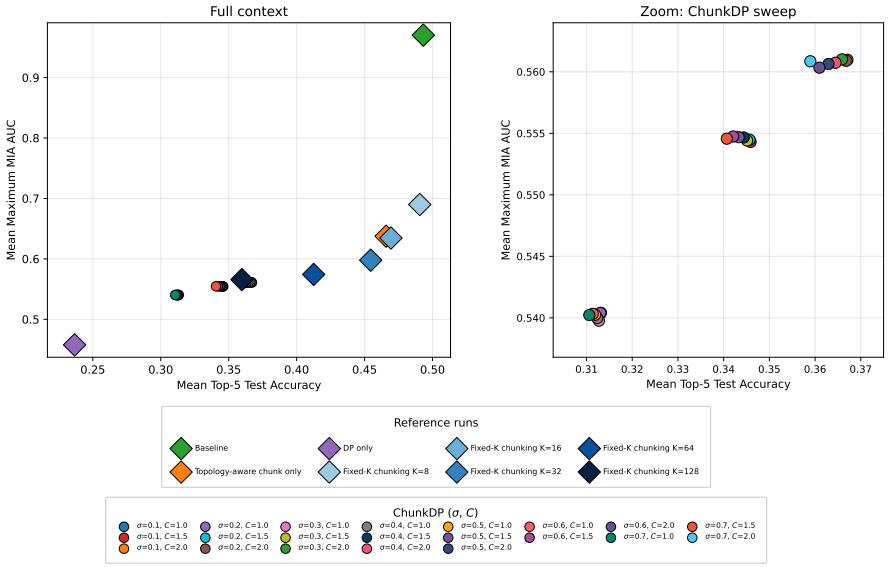


Figure 5.17: ChunkDP noise-clipping sweep (p=0.16). Left: Scatter plot of AUC vs utility for ChunkDP points (circles) with varying noise and clipping threshold, and non-hybrid references (diamonds). Right: zoom on the ChunkDP cluster.

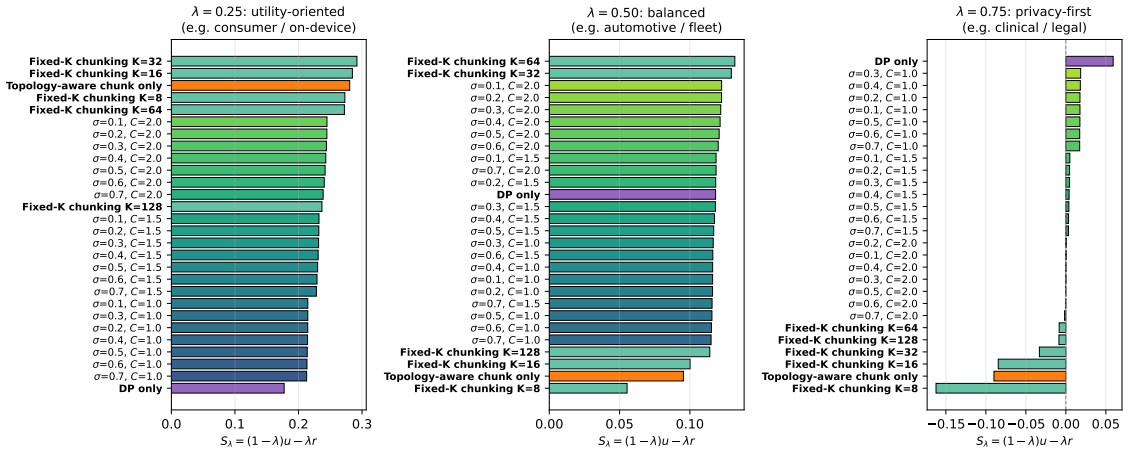


Figure 5.18: ChunkDP noise-clipping sweep (p=0.16). Horizontal bars give privacy-utility score S_λ for each configuration, sorted on highest score first.

6

Discussion

This section interprets the experiment results in relation to the research questions and discusses their broader implications. We first summarize the key findings, then give our interpretations, followed by an answer to the research questions. Subsequently, we discuss the implications the findings have for industry and academia. Finally, we reflect on the limitations of the study and suggest directions for future work.

6.1. Summary of Key Findings

This thesis studied how communication topology affects vulnerability to Membership Inference Attacks in decentralized learning. Building on this, it investigated whether topology-aware chunking can be combined with differential privacy to improve the privacy utility tradeoff compared to topology-independent fixed-K chunking.

The results in Section 5.1.1 to Section 5.1.5 (Figures 5.1-5.4) show that topology is not a minor implementation detail, but a main factor influencing both privacy leakage and utility. Under the baseline setting, where no defenses were applied, MIA AUC remains very high across all evaluated graph families ($\approx$ 0.97-0.99; Figure 5.1). While there seems to be a very slight decline in AUC between the two extremes, mean maximum MIA AUC is 0.988 on the ring versus 0.969 on the fully connected graph (a decline of $\approx$ 0.02), this difference is too small to be practically meaningful.

In contrast, when using topology-aware chunking as the only defense mechanism, privacy leakage becomes strongly topology dependent (Figures 5.1, 5.2, and 5.3). Across experiments, we see a clear pattern: dense or well-connected structures benefit significantly more from topology-aware chunking, while sparse networks remain highly vulnerable. On the fully connected graph, mean maximum MIA AUC drops from 0.969 to 0.609 ($\Delta \approx 0.36$), whereas on the ring it falls only from 0.988 to 0.966 ($\Delta \approx 0.02$). This is further highlighted by heterogeneous networks like the star topology (Figure 5.5), where the central hub benefits from reduced leakage ($\approx$ 0.98 to $\approx$ 0.61) while low-degree leaf nodes remain highly vulnerable ($\approx$ 1.00).

The comparison with topology-independent chunking is important for interpreting the results (Figures 5.9 and 5.10). Fixed-K chunking uses a fixed global number of chunks independent of node degree, meaning that sparse and dense nodes receive the same chunking based privacy amplification. As a result, fixed-K chunking does not introduce the same degree dependent privacy gap observed in topology-aware chunking. In our experiments, fixed-K chunking performed equal or better than the topology-aware chunking variant when using a similar effective number of chunks (e.g., p=0.08 corresponding to K=8). For example, fixed-K chunking with K=8 achieves mean maximum MIA AUC $\approx$ 0.70, slightly outperforming topology-aware chunking-only ($\approx$ 0.73). Combined with the heterogeneity of leakage across nodes for topology-aware chunking, this suggests that the topology-aware variant on its own does not consistently provide stronger protection than fixed-K chunking, and is less suitable in settings with highly variable node degrees.

However, the topology-aware experiments still provide useful insights into the interaction between graph structures and privacy leakage (Figure 5.4). Under topology-aware chunking, the benefits gained by increased density seem to diminish at very high degrees, as highly connected graphs (e.g., 25-regular, or Erdős-Rényi with p=0.32, all with 100 peers) reach mean maximum MIA AUC ≈ 0.62, approaching the fully connected AUC value of 0.61. This suggests that increasing connectivity increases privacy only up to a certain point, beyond which additional graph density yields limited gains. This is consistent with what we find for fixed-K chunking (Figure 5.9): increasing global number of chunks K increases privacy but with diminishing returns as K becomes large.

Section 5.2 presents the MIA AUC and accuracy results of adding noise to both the undefended baseline and to the topology-aware chunking procedure (Figures 5.9, 5.10, 5.12, and 5.13). Across both the evaluated ER densities ($p \in \{0.08, 0.16\}$), the baseline reaches high accuracy ($u \approx 0.48$-0.49) but with substantial privacy leakage ($a \approx 0.97$). Adding DP noise to the baseline reduces leakage towards random guessing ($a \approx 0.46$) at a large accuracy cost ($u \approx 0.23$). Topology-aware chunking has a smaller reduction in MIA AUC ($a \approx 0.73$ at p=0.08; $a \approx 0.64$ at p=0.16) but also a smaller reduction in accuracy ($u \approx 0.46$). ChunkDP falls between the two ($a \approx 0.53$, $u \approx 0.30$ at both densities), recovering part of the accuracy lost to DP-only while keeping leakage clearly below topology-aware chunking-only. The effect of increasing p is consistent with the topology results stated above. At p=0.16, topology-aware chunk-only leakage is lower than at p=0.08 and the benefits of ChunkDP seem to decrease as more privacy is offered by the topology-aware chunking mechanism itself.

Finally, Section 5.3 shows that, under ChunkDP, different values for (σ, C) can shift its position closer towards either DP-only or the topology-aware chunking-only regime (Figures 5.15 and 5.16 for p=0.08; Figures 5.17 and 5.18 for p=0.16). Ranking configurations by the privacy-utility score S_λ gives different winners depending on the privacy-utility scenario: at p=0.08, fixed-K chunking with K=16 ranks best for utility-focused settings (λ=0.25), ChunkDP with (σ_0, C)=(0.1, 1.0) for balanced settings (λ=0.50), and DP-only for privacy-first settings (λ=0.75). At p=0.16, fixed-K chunking with K=32 and K=64 leads for λ=0.25 and λ=0.50, respectively, while DP-only remains best at λ=0.75.

6.2. Interpretation

Taken together, these findings support several important insights.

First, as expected from prior literature, utility improves as the density of the graph structure increases. However, when no defenses are used, all nodes display similarly high levels of privacy leakage and remain highly vulnerable to membership inference attacks. This indicates that topology alone is not sufficient to provide meaningful privacy protection. While denser structures improve utility, they do not meaningfully limit the ability of an adversary to infer node membership. Effective privacy protection requires explicit defense mechanisms rather than relying on the structure of the graph.

Second, privacy leakage is heterogeneous across nodes and graph roles. The network-level MIA AUC can be close to random guessing while a single node exhibits full leakage. These results show that network-level MIA scores can mask substantial variation in vulnerability across nodes and are therefore best complemented with a node-level or role-based analysis.

Third, topology-aware chunking is a structural defense, not a uniform privacy mechanism. Its effectiveness is strongly dependent on node degree and graph density. Dense nodes benefit significantly, while sparser nodes remain vulnerable, so topology-aware chunking does not provide uniform protection across the network. This limitation motivates the design of ChunkDP. By adapting the allocated noise to the graph structure, nodes that remain vulnerable under topology-aware chunking receive more noise, while well-connected nodes receive less noise. In this way, ChunkDP compensates for the uneven protection provided by topology-aware chunking.

Fourth, fixed-K chunking remains a strong graph-independent baseline. Unlike topology-aware chunking, it is not dependent on node degree or underlying network structure. Because of this, it provides more uniform privacy protection across nodes. In our experiments, fixed-K chunking is

preferred in utility-focused settings and also performs strongly in balanced situations, narrowly losing to some configurations of ChunkDP under sparse graphs. Moreover, fixed-K chunking consistently outperforms topology-aware chunking when used in isolation. This suggests that using topology-aware chunking strategies does not necessarily translate into better privacy-utility tradeoffs. The additional complexity introduced by topology-aware chunking may therefore not always be justified, especially in the absence of differential privacy.

The strong performance of fixed-K chunking can be explained by the uniform privacy blanket it provides to all nodes. Every node receives the exact same privacy amplification, resulting in relatively homogeneous privacy protection across the network. In contrast, topology-aware approaches create uneven protection, with low-degree nodes being much more vulnerable than high-degree nodes. As a result, the stronger privacy protection enjoyed by well-connected nodes is insufficient to compensate for the worse protection of poorly-connected nodes, helping explain why the more homogeneous protection provided by fixed-K chunking outperforms topology-aware chunking.

ChunkDP compensates for this uneven protection by allocating more noise to these vulnerable nodes. While this improves the privacy of the most vulnerable nodes and reduces the heterogeneity of membership vulnerability, it also introduces additional noise that inevitably reduces utility. Because of this, the benefits gained from topology-awareness do not always translate into a better overall privacy-utility tradeoff than the simpler fixed-K baseline.

Finally, the privacy-utility tradeoff is inherently topology- and scenario-dependent. No single mechanism scores best in all settings, and optimal configurations of ChunkDP depend on both network structure and the relative importance of privacy versus utility.

6.3. Answers to the Research Questions

This thesis investigated how communication topology influences vulnerability to membership inference attacks and model utility, and whether topology-aware defense mechanisms can improve the privacy-utility tradeoff compared to topology-agnostic approaches.

RQ1: How do different communication topologies affect membership inference vulnerability and model utility in decentralized learning?

The results demonstrate that communication topology has a strong impact on utility, with denser networks achieving higher utility than sparser networks. In contrast, topology alone does not significantly impact vulnerability to MIAs as even well-connected nodes remain highly vulnerable. When using a defense like topology-aware chunking however, the influence of topology becomes much greater. Well-connected nodes enjoy a large reduction in leakage as their model can be spread over many chunks, while sparse nodes still remain highly vulnerable. This effect is most evident in heterogeneous topologies like the star network where the central hub has greatly reduced leakage while the sparse leaves remain highly vulnerable.

RQ2: How does topology-aware chunking influence the privacy-utility tradeoff compared to topology-agnostic fixed-K chunking?

The experiments show that topology-aware chunking does not universally improve the privacy-utility tradeoff over fixed-K chunking in decentralized learning systems. Although topology-aware chunking can significantly reduce leakage for well-connected nodes, its protection is uneven and highly dependent on the network structure. Fixed-K chunking provides more uniform privacy protection and consistently outperforms topology-aware chunking when both are used in isolation. As a result, Fixed-K chunking remains a strong, topology independent baseline, particularly in utility-focused settings.

RQ3: Can ChunkDP improve the privacy-utility tradeoff compared to differential privacy-only and topology-agnostic chunking approaches?

The results show that the proposed ChunkDP can improve the privacy-utility tradeoff in sparse graphs under balanced privacy-utility settings, where both privacy and utility are important. By combining topology-aware chunking with topology-aware noise allocation, ChunkDP compensates

for the uneven protection provided by topology-aware chunking alone. However, the extent of this improvement over fixed-K chunking is limited and depends on the communication graph and privacy requirements of the application. ChunkDP therefore should not be viewed as a universally superior defense mechanism, but rather as a topology-aware approach whose effectiveness depends on the specific privacy requirements and underlying network structure.

Taken together, these findings indicate that topology-awareness can be beneficial for the privacy-utility tradeoff in decentralized learning, but only when incorporated carefully into the defense mechanism. While fixed-K chunking consistently outperforms topology-aware chunking on its own, the proposed ChunkDP shows that the network structure can be leveraged to improve the privacy-utility tradeoff under suitable conditions. However, these benefits are limited and depend strongly on the communication graph and privacy requirements of the application.

6.4. Implications

The results have implications for both research and industry.

From a research perspective, they highlight the importance of studying privacy at the level of individual nodes and structural roles, rather than only using network-level averages. They also demonstrate that communication topology alone does not provide meaningful privacy protection, as nodes remain highly vulnerable to membership inference attacks when no defenses are applied. Furthermore, the results indicate that topology-aware mechanisms can provide benefits in certain settings, but do not universally outperform simpler, topology-independent mechanisms such as fixed-K chunking. This suggests that future work should further investigate when and how topology-aware mechanisms can effectively be combined with other privacy mechanisms.

From a practical perspective, the results show that deploying decentralized learning systems requires careful consideration of both the defense mechanism and the underlying network topology. In particular, when using the topology-aware chunking variant, systems with sparse connectivity or highly heterogeneous degree distributions may require additional protection for vulnerable nodes.

More generally, selecting an appropriate defense mechanism depends on the specific privacy-utility requirements, rather than a one-size-fits-all solution. An important consideration is the distinction between empirical privacy and formal privacy guarantees. While the results showed that ChunkDP can reduce vulnerability to membership inference attacks in practice, they are based on empirical observations under specific empirical settings and are highly dependent on factors such as the attack model, dataset, hyperparameter choices, and threat assumptions. In contrast, a mechanism like differential privacy provides formal and verifiable guarantees. For practitioners operating in regulated domains (e.g., under GDPR requirements), such formal guarantees are preferable because it allows them to verify to an official authority that their model is private using strong guarantees rather than relying only on empirical evidence.

6.5. Ethical Considerations

This thesis investigates privacy risks and defense mechanisms in decentralized learning. The work does not involve sensitive personal data, as all experiments were done using the publicly available CIFAR-100 dataset. CIFAR-100 is a widely used benchmark dataset in machine learning research. As a result, the experiments do not introduce any privacy risk to individuals.

Furthermore, we do not propose any new membership inference attacks or techniques that can be used with malicious intent to increase privacy leakage. Instead, already existing attacks are only used as an evaluation tool to assess the effectiveness of defense mechanisms aimed at reducing privacy leakage in decentralized learning systems.

By improving understanding of how the communication topology influences vulnerability to membership inference attacks and by exploring topology-aware defenses such as ChunkDP, this research contributes to the broader goal of developing machine learning systems that offer stronger privacy protection. Such improvements may be particularly valuable in privacy-sensitive domains where protecting individuals' data is of critical importance.

6.6. Limitations

6.6.1. CIFAR-100 and small local sample sizes

Most of the experiments used CIFAR-100 with 100 peers and no data duplication. This yields only a few samples per class per node on average. While this setting is useful for exposing MIA behavior and increasing topology effects, because nodes overfit on their data and thus have strong leakage, it may not be representative of applications with larger local datasets or different data types.

6.6.2. Large number of hyperparameters

The system contains many interacting hyperparameters (learning rate, local epochs, mixing coefficient, topology parameters, attack settings, chunking, noise). Although controlled sweeps were done for key parameters, full optimization for each parameter was not feasible. Some observed differences may therefore be dependent on specific hyperparameter choices.

6.6.3. Compute constrained seed and attack coverage

As mentioned before, it was not feasible to do sweeps on every single parameter. The main bottleneck is computational cost. This also constrained the breadth of repeated runs and attack coverage in the reported results. The main results rely on a limited number of seeds and only consider loss-based MIA. Because of this, the limited number of seeds reduces statistical confidence in the reported results and the reported privacy leakage may not fully reflect the effectiveness of stronger attack strategies like LiRA or RMIA. Therefore, the results should be interpreted as estimates under a relatively simple attack model.

6.6.4. Restricted threat model

The threat model assumes passive, honest-but-curious, non-colluding adversaries. It does not include collusion, active protocol deviation. Therefore, the reported privacy gains should be interpreted within this threat model and not as guarantees against all decentralized attackers.

In colluding settings, the privacy benefits obtained from topology-aware chunking diminish, whereas noise-based mechanisms retain their privacy. Therefore, accounting for colluding attackers would lead to different optimal hyperparameters for our privacy-utility tradeoff.

6.6.5. Heuristic ChunkDP without formal privacy guarantees

The degree scaled ChunkDP is introduced as a practical heuristic. Because noise is applied heterogeneously at the node level, deriving a complete graph level DP accounting for ChunkDP is outside the scope of this work. As a result, ChunkDP is currently supported by empirical evidence rather than formal privacy guarantees. Deriving such guarantees is left for future work. Such guarantees strengthen the practical applicability of ChunkDP because it makes it easier for practitioners operating in regulated domains to prove to auditors and stakeholders that their model is mathematically guaranteed to be private, rather than relying on empirical evidence.

6.6.6. Final round based privacy utility reporting

The results focus on final-round outcomes. This provides a clear summary but may not fully capture earlier rounds in which leakage or utility evolve differently. Round based dynamics could reveal additional risks or opportunities, such as early leakage peaks.

6.7. Future Work

6.7.1. Formal privacy accounting for topology-aware hybrid defenses

An important next step is to derive formal privacy accounting for topology-aware hybrid defenses. In this thesis, the rule that scales based on degree is chosen as a practical heuristic, so future work should also explore and compare alternative scaling formulas rather than relying on this specific mechanism.

6.7.2. Design variations of topology-aware chunking

Future work could explore different design choices within topology-aware chunking. In our proposed variant, each neighbor receives a different chunk, allowing the full model to be distributed over the network each round. This design prioritizes information diversity and accelerates global mixing. However, an alternative approach would be to align topology-aware chunking more closely with fixed-K chunking by selecting the communicated chunks at random and sending those chunks to all neighbors. While this reduces the diversity of information exchanged each round, it may improve convergence stability by ensuring consistent updates across neighbors. Preliminary experiments with such an alternative partitioning strategy are presented in Appendix **??**, but a more comprehensive evaluation remains future work.

Additionally, other topology-aware chunking strategies could be investigated. The current approach uses degree-dependent partitioning, but alternative mechanisms that use different graph properties or partitioning strategies may further improve performance.

For example, instead of matching the number of chunks to the node degree, the number of chunks could be increased globally (e.g., add multiple chunks per neighbor), this would help sparser nodes significantly and could help improve the privacy-utility tradeoff.

6.7.3. Extending ChunkDP with topology-aware clipping

Future work could also explore additional topology-aware components within ChunkDP. In the current implementation, the amount of noise added to the model updates is adapted to the graph structure, while the clipping threshold remains identical for all nodes. As a result, all nodes are subject to the same clipping behavior, potentially introducing unnecessary utility loss for well-connected nodes that already receive significant privacy benefits from topology-aware chunking.

A promising extension would be to make the clipping threshold topology-aware as well. For example, clipping thresholds could be increased based on node degree as well, allowing vulnerable nodes to receive stronger privacy protection while reducing unnecessary utility loss for well-connected nodes. However, because the final scale of the added noise depends on both the clipping threshold and the noise multiplier, these parameters cannot be considered independently. Designing topology-aware clipping thresholds therefore require jointly optimizing clipping and noise allocation across the network. Investigating the interaction between topology-aware clipping and topology-aware noise allocation is therefore an interesting direction for future work.

6.7.4. Stronger and broader attacker evaluation

Future experiments should systematically explore using different and stronger MIA attacks like LIRA or RMIA across the full topology suite and extend the threat model to colluding attackers. This will provide a more complete estimate of realistic adversarial capability.

6.7.5. Broader generalization studies

Future work should evaluate the generalizability and robustness of ChunkDP across additional datasets, models, peer counts, and hyperparameter settings.

6.7.6. Temporal privacy analysis and round adaptive defense mechanisms

A temporal privacy analysis is needed. Tracking AUC and utility through training rounds can reveal when systems are most vulnerable and whether round adaptive mechanisms, e.g., changing chunking or DP related parameters as rounds change, can improve the privacy-utility tradeoff.

7

Conclusion

This thesis set out to evaluate how communication topology influences vulnerability to membership inference attacks and model utility, and whether topology-aware defense mechanisms can improve the privacy-utility tradeoff compared to topology-independent approaches.

The results show that communication topology has a strong influence on model utility, with dense graphs achieving better performance than sparse graphs. In contrast, topology has only a limited effect on the vulnerability to MIAs, as nodes across different graphs remain highly vulnerable in the absence of defense mechanisms. However, the importance of topology increases considerably when topology-aware chunking is used. With topology-aware chunking, privacy leakage becomes strongly dependent on the communication graph, with well-connected nodes enjoying substantial privacy benefits while poorly-connected nodes remain highly vulnerable.

This uneven protection reveals a key limitation of topology-aware chunking. While it offers substantial protection to well-connected nodes, poorly-connected nodes remain highly vulnerable. As a result, the privacy benefits gained by well-connected nodes do not translate into improved network-wide privacy-utility tradeoffs. In our experiments, topology-independent fixed-K chunking consistently outperforms topology-aware chunking when both are used in isolation. Fixed-K chunking provides more uniform protection across nodes while maintaining competitive utility. This makes fixed-K chunking a lightweight but robust graph-independent baseline.

To address the limitation of topology-aware chunking, we proposed ChunkDP, which combines topology-aware chunking with degree-scaled differential privacy noise. By allocating more noise to vulnerable low-degree nodes while adding less noise to well-protected nodes, ChunkDP compensates for the uneven protection provided by topology-aware chunking. The results show that ChunkDP can improve the privacy-utility tradeoff compared to fixed-K chunking in sparse networks under balanced privacy-utility settings, where both privacy and utility are important. However, these benefits are limited and diminish as network connectivity increases.

Overall, fixed-K chunking remains a highly competitive graph-independent baseline. ChunkDP does not universally outperform it, but it can provide a more favorable privacy-utility tradeoff under suitable conditions. This indicates that topology-awareness in chunking is not automatically beneficial, but can provide value when combined with topology-aware noise allocation under context-specific privacy requirements.

More broadly, the findings highlight that there is no universally optimal defense. The effectiveness of a given defense depends on the underlying network topology, the distribution of node degrees, and the relative importance of privacy versus utility. Rather than relying on a single method, system designers should consider a combination of techniques and optimize their configurations to the specific application setting.

Bibliography

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

[2] P. Kairouz et al., "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1-2, pp. 1–210, 2021, ISSN: 1935-8237. DOI: 10.1561/2200000083. [Online]. Available: http://dx.doi.org/10.1561/2200000083.

[3] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020. DOI: 10.1109/MSP.2020.2975749.

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

[5] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 691–706. DOI: 10.1109/SP.2019.00029.

[6] L. Zhu, Z. Liu, and S. Han, *Deep leakage from gradients*, 2019. arXiv: 1906.08935 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1906.08935.

[7] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, *Inverting gradients – how easy is it to break privacy in federated learning?* 2020. arXiv: 2003.14053 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2003.14053.

[8] 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]. [Online]. Available: https://arxiv.org/abs/1705.09056.

[9] MarcT0K, *Federated learning (centralized vs decentralized)*, https://commons.wikimedia.org/wiki/File:Federated_learning_(centralized_vs_decentralized).png, Wikimedia Commons, CC BY-SA 4.0, 2023.

[10] E. Cyffers, M. Even, A. Bellet, and L. Massoulié, *Muffliato: Peer-to-peer privacy amplification for decentralized optimization and averaging*, Jun. 2022. DOI: 10.48550/arXiv.2206.05091.

[11] E. Cyffers, "Differential Privacy for Decentralized Learning," Theses, Université de Lille, Dec. 2024. [Online]. Available: https://hal.science/tel-04907994.

[12] E. Hosseini, S. Chen, and A. Khisti, *Secure aggregation in federated learning using multiparty homomorphic encryption*, 2025. arXiv: 2503.00581 [cs.CR]. [Online]. Available: https://arxiv.org/abs/2503.00581.

[13] K. A. Bonawitz et al., "Practical secure aggregation for federated learning on user-held data," in *NIPS Workshop on Private Multi-Party Machine Learning*, 2016. [Online]. Available: https://arxiv.org/abs/1611.04482.

[14] 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]. [Online]. Available: https://arxiv.org/abs/2310.01972.

[15] S. Biswas et al., "Noiseless privacy-preserving decentralized learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2025, no. 1, pp. 824–844, Jan. 2025, ISSN: 2299-0984. DOI: 10.56553/popets-2025-0043. [Online]. Available: http://dx.doi.org/10.56553/popets-2025-0043.

[16] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramèr, "Membership inference attacks from first principles," *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 1897–1914, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:244920593.

[17] C. Dwork, "Differential Privacy," in *Automata, languages and programming (ICALP 2006). Part II*, ser. Lecture Notes in Computer Science, vol. 4052, Berlin, Heidelberg: Springer, 2006, pp. 1–12.

[18] R. Ormándi, I. Hegedűs, and M. Jelasity, "Gossip learning with linear models on fully distributed data," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 4, pp. 556–571, May 2012, ISSN: 1532-0634. DOI: 10.1002/cpe.2858. [Online]. Available: http://dx.doi.org/10.1002/cpe.2858.

[19] A. Nedić, A. Olshevsky, and M. G. Rabbat, *Network topology and communication-computation tradeoffs in decentralized optimization*, 2018. arXiv: 1709.08765 [math.OC]. [Online]. Available: https://arxiv.org/abs/1709.08765.

[20] Y.-T. Chow, W. Shi, T. Wu, and W. Yin, *Expander graph and communication-efficient decentralized optimization*, 2016. arXiv: 1612.00903 [math.OC]. [Online]. Available: https://arxiv.org/abs/1612.00903.

[21] B. Ying, K. Yuan, Y. Chen, H. Hu, P. Pan, and W. Yin, *Exponential graph is provably efficient for decentralized deep training*, 2021. arXiv: 2110.13363 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2110.13363.

[22] T. Vogels, H. Hendrikx, and M. Jaggi, *Beyond spectral gap: The role of the topology in decentralized learning*, 2022. arXiv: 2206.03093 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2206.03093.

[23] B. L. Bars, A. Bellet, M. Tommasi, E. Lavoie, and A.-M. Kermarrec, *Refined convergence and topology learning for decentralized sgd with heterogeneous data*, 2022. arXiv: 2204.04452 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2204.04452.

[24] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei, *Towards demystifying membership inference attacks*, 2019. arXiv: 1807.09173 [cs.CR]. [Online]. Available: https://arxiv.org/abs/1807.09173.

[25] X. He, R. Wen, Y. Wu, M. Backes, Y. Shen, and Y. Zhang, *Node-level membership inference attacks against graph neural networks*, 2021. arXiv: 2102.05429 [cs.CR]. [Online]. Available: https://arxiv.org/abs/2102.05429.

[26] J. Hayes, L. Melis, G. Danezis, and E. D. Cristofaro, *Logan: Membership inference attacks against generative models*, 2018. arXiv: 1705.07663 [cs.CR]. [Online]. Available: https://arxiv.org/abs/1705.07663.

[27] N. Koren, A. Goldsteen, G. Amit, and A. Farkash, *Membership inference attacks against time-series models*, 2024. arXiv: 2407.02870 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2407.02870.

[28] O. Touat et al., *Exposing the vulnerability of decentralized learning to membership inference attacks through the lens of graph mixing*, 2025. [Online]. Available: https://arxiv.org/html/2412.12837v3.

[29] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, 2018, pp. 268–282. DOI: 10.1109/CSF.2018.00027.

[30] M. Abadi et al., "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS'16, ACM, Oct. 2016, pp. 308–318. DOI: 10.1145/2976749.2978318. [Online]. Available: http://dx.doi.org/10.1145/2976749.2978318.

[31] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17, Dallas, Texas, USA: Association for Computing Machinery, 2017, pp. 1175–1191, ISBN: 9781450349468. DOI: 10.1145/3133956.3133982. [Online]. Available: https://doi.org/10.1145/3133956.3133982.

[32] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2015, pp. 909–910. DOI: 10.1109/ALLERTON.2015.7447103.

[33] N. Phan, Y. Wang, X. Wu, and D. Dou, "Differential privacy preservation for deep auto-encoders: An application of human behavior prediction," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16, Phoenix, Arizona: AAAI Press, 2016, pp. 1309–1316.

[34] I. E. Olatunji, W. Nejdl, and M. Khosla, *Membership inference attack on graph neural networks*, 2021. arXiv: 2101.06570 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2101.06570.

[35] X. He, R. Wen, Y. Wu, M. Backes, Y. Shen, and Y. Zhang, *Node-level membership inference attacks against graph neural networks*, 2021. arXiv: 2102.05429 [cs.CR]. [Online]. Available: https://arxiv.org/abs/2102.05429.

[36] A. Daigavane, G. Madan, A. Sinha, A. G. Thakurta, G. Aggarwal, and P. Jain, *Node-level differentially private graph neural networks*, 2022. arXiv: 2111.15521 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2111.15521.

[37] Antibloch, *Mia_attacks: Membership inference attack implementations*, https://github.com/antibloch/mia_attacks, GitHub repository, 2024.

[38] N. Tomassen, *Decentralized-learning*, https://github.com/nielstomassen/decentralized-learning, GitHub repository, 2026.

A

Additional Tables

Table A.1: Erdős–Rényi graphs: node groups by degree bin for different edge probabilities p. Each entry averages over nodes in the bin.

p	Mode	Degree bin	Mean Top-5 Accuracy	Mean Average MIA AUC	Mean Maximum MIA AUC
0.04	Baseline	1	0.451 ± 0.006	0.978 ± 0.001	0.978 ± 0.001
	Baseline	2	0.464 ± 0.008	0.977 ± 0.002	0.977 ± 0.002
	Baseline	3	0.468 ± 0.009	0.971 ± 0.004	0.971 ± 0.004
	Baseline	≥ 4	0.480 ± 0.010	0.976 ± 0.004	0.976 ± 0.005
	Chunking	1	0.389 ± 0.007	0.996 ± 0.002	0.996 ± 0.002
	Chunking	2	0.418 ± 0.007	0.894 ± 0.009	0.930 ± 0.007
	Chunking	3	0.429 ± 0.005	0.805 ± 0.009	0.875 ± 0.008
	Chunking	≥ 4	0.412 ± 0.003	0.717 ± 0.004	0.824 ± 0.005
0.08	Baseline	2–5	0.473 ± 0.014	0.971 ± 0.003	0.971 ± 0.003
	Baseline	6–8	0.483 ± 0.005	0.970 ± 0.003	0.970 ± 0.003
	Baseline	9–10	0.484 ± 0.012	0.971 ± 0.005	0.971 ± 0.005
	Baseline	≥ 11	0.477 ± 0.016	0.972 ± 0.006	0.972 ± 0.005
	Chunking	2–5	0.482 ± 0.016	0.747 ± 0.008	0.801 ± 0.010
	Chunking	6–8	0.457 ± 0.009	0.667 ± 0.005	0.731 ± 0.003
	Chunking	9–10	0.441 ± 0.006	0.639 ± 0.004	0.718 ± 0.008
	Chunking	≥ 11	0.442 ± 0.016	0.614 ± 0.012	0.675 ± 0.015
0.16	Baseline	7–11	0.481 ± 0.017	0.973 ± 0.003	0.973 ± 0.003
	Baseline	12–15	0.487 ± 0.004	0.970 ± 0.002	0.970 ± 0.001
	Baseline	16–19	0.495 ± 0.004	0.971 ± 0.003	0.971 ± 0.003
	Baseline	≥ 20	0.489 ± 0.014	0.969 ± 0.004	0.969 ± 0.004
	Chunking	7–11	0.468 ± 0.019	0.622 ± 0.014	0.668 ± 0.014
	Chunking	12–15	0.464 ± 0.006	0.603 ± 0.005	0.645 ± 0.003
	Chunking	16–19	0.455 ± 0.004	0.595 ± 0.004	0.639 ± 0.004
	Chunking	≥ 20	0.447 ± 0.008	0.587 ± 0.006	0.629 ± 0.009
0.32	Baseline	20–26	0.489 ± 0.019	0.973 ± 0.004	0.973 ± 0.004
	Baseline	27–31	0.492 ± 0.008	0.970 ± 0.003	0.970 ± 0.003
	Baseline	32–36	0.494 ± 0.009	0.970 ± 0.002	0.970 ± 0.002
	Baseline	≥ 37	0.487 ± 0.019	0.966 ± 0.004	0.967 ± 0.004
	Chunking	20–26	0.476 ± 0.011	0.578 ± 0.006	0.621 ± 0.005
	Chunking	27–31	0.462 ± 0.008	0.574 ± 0.008	0.618 ± 0.007
	Chunking	32–36	0.460 ± 0.007	0.569 ± 0.007	0.614 ± 0.007
	Chunking	≥ 37	0.451 ± 0.012	0.566 ± 0.006	0.609 ± 0.006

B

Additional Experiments

B.1. Alternative Topology-Aware Chunk Partitioning

We compared the default topology-aware chunking implementation and the corresponding ChunkDP mechanism against an alternative topology-aware chunking strategy, that is more similar to Fixed-K chunking in terms of chunk partitioning. In this variant, the full model is first flattened into a single parameter vector and then partitioned into exactly d chunks, where d denotes the node degree. This differs from the default approach described in Section 4.3.2, which applies chunking separately per tensor. Furthermore, the same chunk is broadcast to all neighbors instead of sending different chunks to different neighbors.

Although the partitioning mechanism is structurally closer to Fixed-K chunking, the number of chunks is still degree dependent making it a topology-aware chunking strategy. The effective communicated model fraction remains identical in both approaches. The results show that the performance difference between the two implementations is negligible for both topology-aware chunking and ChunkDP, with the largest observed deviation being a 2.2% change in test accuracy.

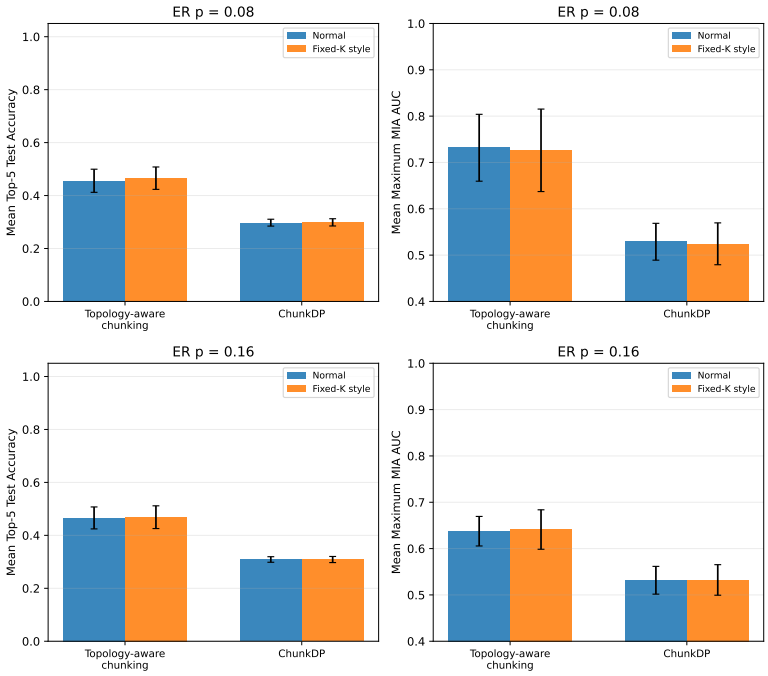
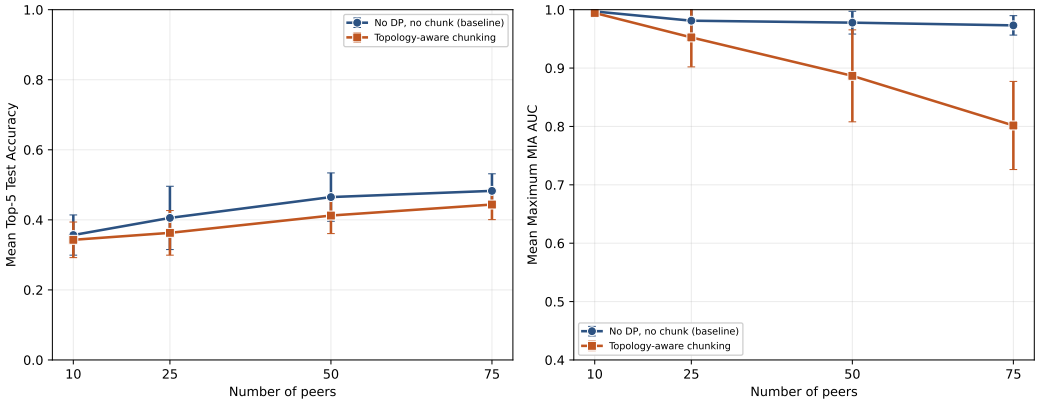


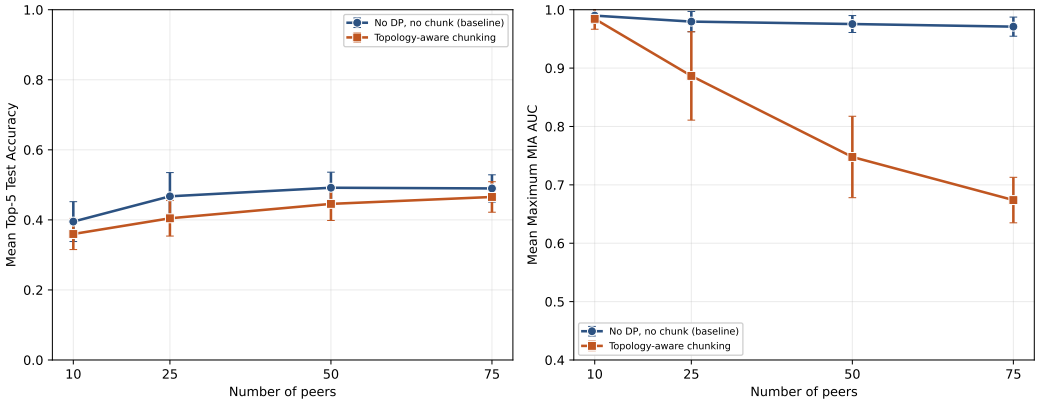
Figure B.1: Comparison between the default topology-aware chunking implementation and a degree-dependent Fixed-K-style partitioning variant for both topology-aware chunking and ChunkDP.

B.2. Effect of the number of peers on privacy and utility

The main topology analysis (Section 5.1) fixes the number of peers at N=100. To assess how sensitive the topology results are to network size, we repeat the same experimental setup, contrasting standard decentralized learning with the topology-aware chunking mechanism on Erdős–Rényi graphs with $p \in \{0.08, 0.16\}$ while varying $N \in \{10, 25, 50, 75\}$. All other settings match the orginal topology analysis. Figure B.2 summarizes this peer-count sweep for p=0.08 (subfigure (a)) and p=0.16 (subfigure (b)). As in our original findings, the undefended baseline remains highly vulnerable



(a) p=0.08.



(b) p=0.16.

Figure B.2: Peer-count sweep: mean top-5 test accuracy (left) and mean maximum MIA AUC (right) versus number of peers, for standard decentralized learning and when topology-aware chunking is applied. Error bars show the within-run standard deviation over nodes.

at every peer count and both densities ($a \approx 0.97$–1.0). Interestingly, its utility rises with N at both densities ($u \approx 0.36 \rightarrow 0.48$ at p=0.08; $u \approx 0.40 \rightarrow 0.49$ at p=0.16), but leakage is essentially unchanged by graph density. Consistent with our earlier experiments, we find that higher graph density increases utility across varying peer counts. For the baseline, u increases by roughly 0.04–0.06 at each peer count (e.g. $u \approx 0.36$ vs. 0.40 at N=10; $u \approx 0.48$ vs. 0.49 at N=75). As in the main topology experiments, topology-aware chunking is where density matters most for privacy. At fixed N, increasing density from p=0.08 to p=0.16 lowers MIA AUC at every peer count (e.g. $a \approx 0.99$ vs. 0.98 at N=10; $a \approx 0.80$ vs. 0.67 at N=75). Within each density, leakage also further decreases as N grows. This makes sense because increased number of peers means an increased number of average neighbors at the same density. This means that the model can be split into more chunks and thus enjoy more protection from topology-aware chunking.

Overall, these results reinforce our main findings, demonstrating that the relationships between topology, utility, and membership inference vulnerability remain robust across varying peer counts.