Noiseless Privacy-Preserving Decentralized Learning

Sayan Biswas
EPFL, Switzerland

Mathieu Even
Inria, DI ENS, PSL University, France

Anne-Marie Kermarrec
EPFL, Switzerland

Laurent Massoulié
Inria, DI ENS, PSL University, France

Rafael Pires
EPFL, Switzerland

Rishi Sharma*
EPFL, Switzerland

Martijn de Vos
EPFL, Switzerland

ABSTRACT

Decentralized learning (DL) enables collaborative learning without a server and without training data leaving the users' devices. However, the models shared in DL can still be used to infer training data. Conventional defenses such as differential privacy and secure aggregation fall short in effectively safeguarding user privacy in DL, either sacrificing model utility or efficiency. We introduce Shatter, a novel DL approach in which nodes create virtual nodes (VNs) to disseminate chunks of their full model on their behalf. This enhances privacy by *(i)* preventing attackers from collecting full models from other nodes, and *(ii)* hiding the identity of the original node that produced a given model chunk. We theoretically prove the convergence of Shatter and provide a formal analysis demonstrating how Shatter reduces the efficacy of attacks compared to when exchanging full models between nodes. We evaluate the convergence and attack resilience of Shatter with existing DL algorithms, with heterogeneous datasets, and against three standard privacy attacks. Our evaluation shows that Shatter not only renders these privacy attacks infeasible when each node operates 16 VNs but also exhibits a positive impact on model utility compared to standard DL. In summary, Shatter enhances the privacy of DL while maintaining the utility and efficiency of the model.

KEYWORDS

Privacy-Preserving Machine Learning, Decentralized Learning, Collaborative Learning, Virtual Nodes

1 INTRODUCTION

In Decentralized learning (DL), nodes collaboratively train a global machine learning (ML) model without sharing their private data with other entities [53]. The nodes are connected to other nodes, called *neighbors*, via a communication topology. In each round, nodes start with their local models and perform training steps with their private datasets. Updated local models are exchanged with neighbors in the communication graph and aggregated. The aggregated model is then used to start the next round, and the process repeats until convergence. Popular DL algorithms include Decentralized parallel stochastic gradient descent (D-PSGD) [53], Gossip learning (GL) [65], and Epidemic learning (EL) [18].

*Corresponding author: first.last@epfl.ch

Proceedings on Privacy Enhancing Technologies YYYY(X), 1–21

https://doi.org/XXXXXXX.XXXXXXX

In DL, the private data of participating nodes never gets shared. While this is a major argument in favor of the privacy of DL solutions, recent works have shown that model updates can be prone to privacy violations in DL [17, 67]. For example, *(i)* membership inference attacks (MIAs) [75] allow attackers to infer if a particular data sample was used during training by a node, and *(ii)* gradient inversion attacks (GIAs) [30, 97] allow the attackers to reconstruct private data samples used for training. These privacy breaches deter the adoption of DL algorithms in domains where privacy is crucial, like healthcare and finance. Although numerous strategies to protect the users' privacy have been proposed in the centralized settings for ML [52], there is a lack of adequate defenses against privacy attacks in DL. For instance, noise-based solutions bring privacy at the cost of convergence [11, 92], and secure computation schemes provide privacy at the cost of intricate node coordination [9, 57, 85], resource overhead [85], or specialized hardware [14, 42].

To address the privacy concerns of DL without compromising its utility and efficiency, we introduce Shatter, a novel DL system that protects shared model updates from these privacy attacks. Shatter consists of three key components: *chunking*, *full sharing*, and *virtualization*. *Chunking* limits the access of receiving nodes to a model chunk, *i.e.*, a subset of all parameters rather than the full model, thus contributing to privacy. *Full sharing* ensures that there is no information loss as the sender shares each model parameter with multiple nodes through one of the chunks, thus ensuring model utility. *Virtualization* decouples identities from model chunks by having each node operate multiple virtual nodes (VNs) that communicate model chunks with other VNs, thus further contributing to privacy. Additionally, we randomize the communication topology in each round to prevent an adversary from structurally attacking a fixed set of nodes and to boost model utility. Compared to standard DL, our approach does not sacrifice efficiency and only comes with a manageable increase in communication volume.

Chunking may resemble sparsification, a communication-efficient technique where nodes share subsets of their full model with others [2]. However, the *full sharing* of Shatter disseminates *all* the model chunks to different VNs, ensuring that all model parameters of each node are shared and aggregated in each communication round. Sparsification does not guarantee that all model parameters are sent to other nodes. While studies have suggested that sharing small model chunks can potentially be privacy-friendly as less information is shared among participants [74, 83, 97], to the best of our knowledge, we are the first to leverage this to defend against privacy-invasive attacks in DL.

We visualize the overall architecture of Shatter in Figure 1. Standard DL algorithms (left) connect nodes directly in a communication topology, and nodes exchange their full model with

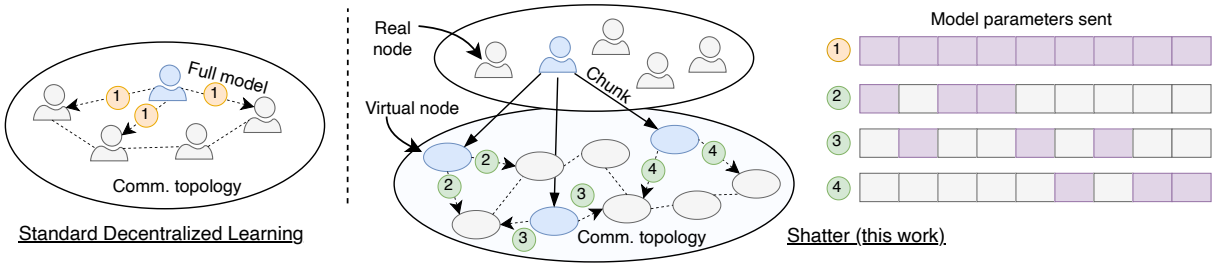

Figure 1: With standard decentralized learning (DL) (left), nodes continuously send their full model to other nodes in the communication topology. With SHATTER, nodes operate multiple VNs (middle) and VNs directly communicate with other VNs. In addition, each VN sends a part of the full model of the real node (RN) to other VNs (right). This hides the identity of the original node that produced a given model chunk.

neighbors every round. In SHATTER (right), each node, which we refer to as a real node (RN) in the context of SHATTER, starts by creating some virtual nodes (VNs) for itself. All the VNs then participate in the communication topology construction. In SHATTER, RNs perform the training and chunking of the models, whereas VNs are responsible for disseminating model chunks through the communication topology. The right part of Figure 1 highlights how different VNs send different model chunks, where grayed-out chunks are not sent. Finally, model chunks received by VNs are forwarded back to their corresponding RN and aggregated there. The next round is then initiated, which repeats until model convergence.

We formally prove the convergence of SHATTER and theoretically demonstrate its privacy guarantees that provide an information-theoretical interpretation of the experimental results, illustrating the comprehensive privacy-preserving properties of SHATTER. We implement SHATTER to empirically evaluate its robustness against *honest-but-curious* adversaries that mount three standard attacks: the linkability, membership inference, and gradient inversion attack. Our experimental results show that SHATTER provides significantly better privacy protection than baseline approaches while improving convergence speed and attainable model accuracy, at the cost of a manageable increase in communication costs.

In summary, we make the following contributions:

- We introduce SHATTER, a novel privacy-preserving DL algorithm where real nodes (RNs) operate multiple virtual nodes (VNs) and VNs share model chunks with each other (Sections 3 and 4).
- We prove the convergence of SHATTER with an arbitrary number of local training steps between each communication round. Our bounds involve regularity properties of local functions, the number of local steps, the number of VNs operated by each RN, and the degree of random graphs sampled at each communication round (Section 5.1).
- We formally show that SHATTER improves the privacy of RNs from an information-theoretical perspective as the number of VNs operated by each RN increases. This offers analytical insight into the diminishing efficacy of attacks exploiting shared model parameters or gradient updates (Section 5.2).
- We implement SHATTER and empirically compare its privacy robustness against state-of-the-art baselines and three privacy-invasive attacks: the linkability attack (LA), MIA and GIA (Section 6). Our results show that SHATTER improves model convergence and exhibits superior privacy defense against the evaluated privacy attacks while also showcasing higher model utility.

2 BACKGROUND AND PRELIMINARIES

In this work, we consider a decentralized setting where a set of nodes $\mathcal{N}$ seek to collaboratively learn an ML model. This is also known as collaborative machine learning (CML) [67, 76]. Each node $i \in \mathcal{N}$ has its private dataset D_i to compute local model updates. The data of each node never leaves the device. The goal of the training process is to find the parameters of the model θ that perform well on the union of the local datasets by minimizing the average loss across all the nodes in the network. The most adopted CML approach is federated learning (FL) which uses a parameter server to coordinate the learning process [58].

Decentralized learning (DL) [5, 53, 64] is a CML algorithm in which each node exchanges model updates with its neighbors through a communication topology comprising of an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ where $\mathcal{N}$ denotes the set of all nodes and $(i, j) \in \mathcal{E}$ denotes an edge or communication channel between nodes i and j (Figure 1, left). Among many variants, D-PSGD [46, 53] is considered a standard algorithm to solve the DL training tasks. At the start of D-PSGD, each node i , with its loss function f_i, identically initializes its local model $\theta_i^{(0)}$ and executes the following:

(1) *Local training.* In each round $0 \leq t \leq T-1$ and each epoch $0 \leq h \leq H-1$, setting $\tilde{\theta}_i^{(t,0)}$ as $\theta_i^{(t)}$, node i independently takes samples ξ_i from its local dataset, computes the stochastic gradient $\nabla f_i(\tilde{\theta}_i^{(t,h)}, \xi_i)$, and updates its local model as $\tilde{\theta}_i^{(t,h+1)} \leftarrow \tilde{\theta}_i^{(t,h)} - \eta \nabla f_i(\tilde{\theta}_i^{(t,h)}, \xi_i)$, where η is the learning rate.
(2) *Model exchange.* Node i sends $\tilde{\theta}_i^{(t,H)}$ to and receives $\tilde{\theta}_j^{(t,H)}$ from each of its neighbors j in $\mathcal{G}$.
(3) *Model aggregation.* Node i mixes the local model parameters that it receives from its neighbors with its own using a weighted aggregation scheme as $\theta_i^{(t+1)} = \sum_{\{j:(i,j)\in\mathcal{E}\}\cup\{i\}} w_{ji}\tilde{\theta}_j^{(t,H)}$, where w_{ji} is the $(j,i)^{\text{th}}$ entry of the mixing matrix W. A common approach is to aggregate all the models with equal weights.

After T rounds, node i adopts the final $\theta_i^{(T)}$, thereby terminating the DL training. The D-PSGD pseudocode is also provided in Algorithm 2 in Appendix B.

2.1 Privacy Attacks in CML

A key property of CML algorithms is that training data never leaves the node's device, therefore providing some form of data privacy. Nonetheless, an adversarial server or node may be able to extract sensitive information from the model updates being shared with them. We next outline three prominent privacy attacks in CML.

Membership inference attack (MIA). The goal of the MIA is to correctly decide whether a particular sample has been part of the training set of a particular node [40]. This is broadly a black-box attack on the model, with the adversary having access to the global training set and samples from the test set, which are never part of the training set of any node. While we assume that the adversary can access actual samples from the global training and test set, this can be relaxed by generating shadow datasets [75]. Many MIA attacks are based on the observation that samples included in model training exhibit relatively low loss values as opposed to samples that the model has never seen. Since the MIA can present a major privacy breach in domains where sensitive information is used for model training, this attack is widely considered a standard benchmark to audit the privacy of ML models [86].

We evaluate the privacy guarantees of Shatter using a loss-based MIA, mainly because of its simplicity, generality, and effectiveness [67]. The adversary queries the received model to obtain the loss values on the data samples. The negative of the loss values can be considered as *confidence scores*, where the adversary can use a threshold to output MIA predictions [87]. To quantify the attack regardless of the threshold, it is common to use the ROC-AUC metric (area under the ROC curve) on the confidence scores [56, 69]. Specifically, membership prediction is positive if the confidence score exceeds a threshold value τ and negative otherwise. Hence, we can compute the corresponding true positive rate (TPR) and false positive rate (FPR) for each τ, resulting in the receiver operating characteristic (ROC) curve.

Gradient inversion attack (GIA). With the GIA, the adversary aims to reconstruct input data samples from the gradients exchanged during the training process in CML [30, 97]. This is a key privacy violation in contexts involving sensitive information, such as personal photographs, medical records, or financial information. Since the success of this attack depends on the information contained in the gradients, this white-box attack in DL is performed at the first communication round or convergence, when the gradient approximation by the adversary is the best [67]. The GIA is an optimization problem where the adversary performs iterative gradient descent to find the input data that results in the input gradients [20, 97]. More sophisticated GIAs on image data include GradInversion [88] when the training is done on a batch of images, and ROG [92] using a fraction of gradients [92]. We use the *state-of-the-art* GIA scheme ROG to evaluate Shatter. In addition, we leverage artifacts and pre-trained weights of reconstruction networks provided with ROG and assume that the adversary knows the ground-truth labels of samples in the batch. These, however, can also be analytically obtained from the gradients of the final neural network layer [80, 95].

Linkability attack (LA). In the context of obfuscated model updates, the LA allows an adversary to link a received obfuscated update with the training set it came from [51]. Similar to the MIA, this is a loss-based black-box attack on the model, but contrary to MIA, the adversary has access to the training sets of each participating node. The LA may also be performed on shadow data representing the training sets of participating nodes instead of actual data. In the context of this work, we assume that the adversary can access the actual training sets. An adversary performs a LA on received model updates by computing the loss of each received model update on each available training set and reporting the training set with the lowest loss. When the obfuscated update does not contain the complete model update vector, *e.g.*, when using model sparsification, the adversary completes the vector with the aggregated model updates from the previous round [51].

2.2 Threat Model

This work focuses on privacy-preserving DL in a *permissioned network setting* where membership is strictly controlled and well-defined. This aligns with the observation that DL is commonly considered and deployed in enterprise settings, where network membership is typically controlled [7]. Such settings include, for example, hospitals collaborating on a DL task [73]. In permissioned networks, all nodes are known entities with verified identities. Consequently, we consider threats commonly found in open networks, such as the Sybil Attack [25], beyond the scope of our work.

Our work focuses on the *honest-but-curious* (HbC) attacker model, *i.e.*, participating nodes faithfully execute the learning protocol but can attempt to retrieve sensitive information about the other nodes from locally-available information. This attacker model is commonly adopted in related work on privacy in CML algorithms [16, 17, 57, 85, 92]. Thus, we adhere to a threat model that is locally privacy-invasive, allowing any update that a node shares with any other node to be potentially exploited to infer information about their personal data used for training their local model in each round. In particular, we do not consider that aggregated models of any node in any round of communication are published or shared with any third-party entity (*e.g.*, a server). We also limit ourselves to a setting where only participating nodes can be adversaries attempting to compromise the privacy of other nodes in the network. Besides received model updates, attacker nodes may also use knowledge of their own model parameters to carry out attacks, and may store historical model updates or other information they received and act on this. Furthermore, all nodes can potentially be attackers. However, we assume that nodes do not collude between them and do not maliciously modify the model parameters. Such local-level privacy risks (*i.e.*, any information that leaves a node can be used against them) are widespread in the literature [62] and in line with some of the recently popularized threat models considered in practice [4, 99] that do not assume the presence of any trusted entity responsible for aggregation or orchestrating the communication.

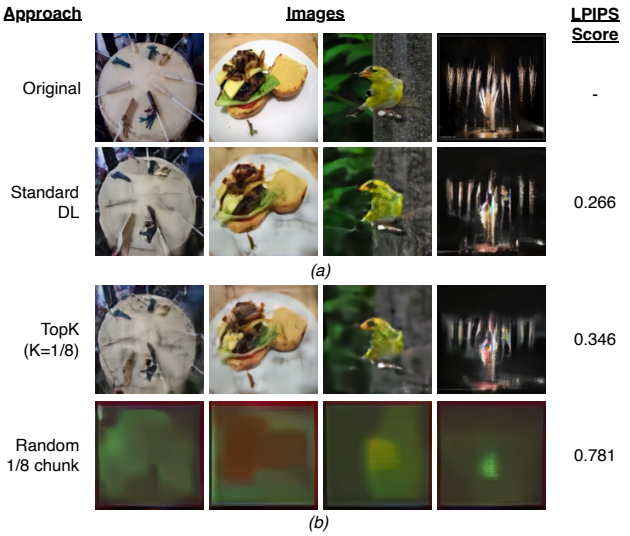

Figure 2: Selected reconstructed images (per row) using the GIA and the average LPIPS score (↑ is more private) for all 1600 images processed in a round, when using standard DL with 100 clients (a), TopK sparsification and random model chunking (b).

3 MOTIVATION BEHIND SHATTER

The design of Shatter is anchored in three key insights into the privacy challenges faced by conventional DL systems.

1) Naive (full) model sharing in DL reveals sensitive data. A key aspect of collaborative ML mechanisms is that private data never leaves the devices of participating users. While this might give a sense of security, Pasquini *et al.* [67] recently demonstrated how model sharing in standard DL settings can reveal sensitive information. To show this empirically, we make each node conduct a GIA during the first training round on the incoming models in a standard DL setting with 100 nodes. Conducting the GIA during the first training round (or close to convergence) is optimal for the attack's success [67]. Each node trains its local model (LeNet) using a batch with 16 images from ImageNet, and adversaries (all other nodes) attempt to reconstruct the images in this batch using ROG [19]. We further elaborate the experiment setup in Section 6.1.

The results of ROG in DL for four random images are shown in Figure 2a, with the original image in the top row and the reconstructed image by the attacker in the second row. For each approach, we also show the average LPIPS score of all 1600 images processed during a training round [93]. This score indicates the perceptual image patch similarity, where higher scores indicate more differences between the original and reconstructed image, *i.e.*, the higher the more private. We observe significant similarities between the original and reconstructed images in DL settings, making it trivial for an attacker to obtain knowledge and semantics of private training data of other users. Empirical findings highlighted by other DL studies [67] and our experiments illustrate that *naive model sharing in DL falls short as a method for safeguarding data privacy*, as they reveal sensitive data even in early training rounds.

2) Partial model sharing can protect privacy. Intuitively, sending only a subset of model parameters to other nodes raises the bar for adversaries to obtain sensitive information, and can be leveraged to increase privacy in DL systems. We note, however, that partial model sharing, also known as sparsification, is typically employed to reduce communication cost [2, 45, 47, 55, 74, 77]. Instead of sharing all model parameters, with sparsification, nodes only send a fraction of the model updates, *i.e.*, each node i constructs and shares a sparsified model $S_i^{(t)} \subseteq \theta_i^{(t)}$ in round t. Typical sparsification approaches retain random parameters (random sharing) or the ones with the highest gradient magnitudes (TopK sharing).

While the work of Yue *et al.* demonstrates how TopK sparsification gives a false sense of privacy [92], there are key differences in privacy guarantees between standard sparsification techniques. To understand the privacy implications of using TopK and random sharing, we conduct the GIA with these sharing methods. Figure 2b shows the reconstructed images with ROG when the network uses TopK sharing with $\frac{1}{8}$th of the model updates being communicated. Even though the LPIPS score related to using TopK sharing is higher than sharing full models (second row in Figure 2a), it is evident that TopK sharing still allows an attacker to reconstruct potentially sensitive information from private training datasets. This is because TopK shares the parameters with the highest gradient magnitudes, and hence, the ones that are the most influential in the training round. Access to these parameters raises the effectiveness of the GIA that relies on gradients to reconstruct private data.

The bottom row in Figure 2b shows the reconstructed images by an adversary when each node randomly sends $\frac{1}{8}$th of its model update to neighboring nodes (random sharing). Random sharing yields blurry images, making it nearly impossible for an adversary to obtain semantic information. This is also demonstrated by the associated LPIPS score, which is nearly three times as high as when using TopK sharing. While it is evident that *random sharing is effective at obfuscating updates in DL*, partial model sharing has been shown to hurt time-to-convergence and final achieved accuracy, therefore prolonging the overall training process [23, 24, 38].

3) Noise-based solutions have drawbacks. Another approach to increase privacy in CML is to add small perturbations (*a.k.a.* noise) to model updates before sharing them with other nodes [82,

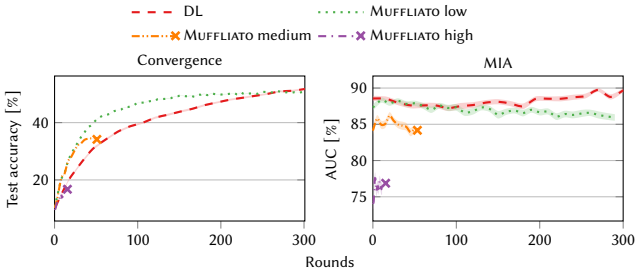

Figure 3: The test accuracy (↑ is better) on the left and MIA attack success (↓ is more private) on the right for DL and Muffliato with three noise levels, using CIFAR-10 as training dataset.

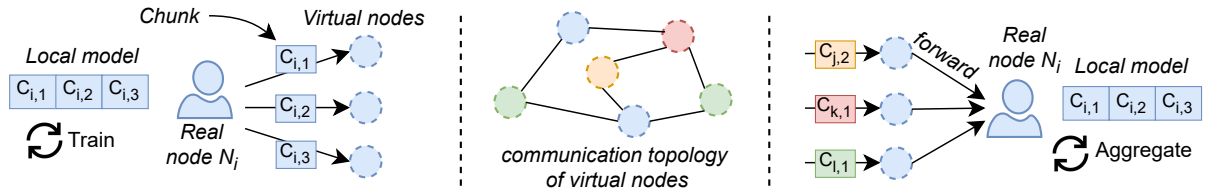

Figure 4: The three-step workflow of SHATTER, showing the operations during a single training round. A RN N_i first splits its local model into k chunks and sends each chunk to one of its VNs (left). We refer to the s^{th} chunk of RN N_i as $C_{i,s}$. VNs are connected into a communication topology and exchange model chunks with other VNs (middle). VNs forward received chunks to the corresponding RN who aggregates the received chunks into its local model and performs a training step (right).

90]. Noise-based mechanisms for DL with differential-privacy foundations like MUFFLIATO [17] have been theoretically proven to converge while providing strong privacy guarantees. In practice, though, adding noise for privacy protection can hurt the utility of the model [67, 92]. We illustrate this by evaluating the convergence and the success of the MIA for both EL, a DL variant where the communication topology is refreshed every round, and MUFFLIATO on a non independent and identically distributed (non-IID) partitioning of the CIFAR-10 dataset with a ResNet-18 model. We comprehensively search for three noise levels in MUFFLIATO, one with an attack success close to DL, one with a low attack success rate, and one in between. We run MUFFLIATO with 10 communication rounds as recommended by the authors of MUFFLIATO [17]. While this increases the communication cost of MUFFLIATO by 10× compared to DL, we consider this fair from a convergence and privacy perspective.

Figure 3 shows our results and demonstrates the pitfalls of the *state-of-the-art* noise-based DL solution MUFFLIATO. The figure shows the test accuracy and MIA attack resilience for DL and MUFFLIATO for three different noise levels (low: $\sigma = 0.025$, medium: $\sigma = 0.05$, high: $\sigma = 0.1$). While MUFFLIATO (*low*) does reduce the MIA success rate when compared to DL, this reduction is marginal. The better convergence of MUFFLIATO (*low*) can be attributed to the 10 communication rounds in MUFFLIATO, as opposed to 1 in DL. Extra communication rounds enable nodes to better aggregate the models across the network. We also observe that model training with MUFFLIATO (*medium*) breaks down after roughly 45 rounds into training. This is because noise introduces numerical instability in the learning process, leading to *nan* (not a number) loss values. On the other hand, MUFFLIATO (*high*) handicaps the model utility, as there is no convergence (Figure 3, left). Its better performance in MIA is, hence, not representative of the learning process. We emphasize that there is no way to predict the noise levels in MUFFLIATO, and in similar solutions. Furthermore, Yue *et al.* empirically show that noise-based solutions hurts the final accuracy when reducing the success of GIA [92]. The results in Figure 3 highlight the *need for practical privacy-preserving solutions that do not adversely affect the convergence of DL tasks.*

4 DESIGN OF SHATTER

Based on the observations in Section 3, we present the design of SHATTER. The essence of SHATTER is to exchange a randomly selected subset of parameters rather than the entire set (*chunking*) while still disseminating all the parameters through the network to not lose convergence (*full sharing*). SHATTER achieves this by having each real node (RN) operate multiple virtual nodes (VNs) (*virtualization*) and the VNs communicate random model chunks with other VNs in the network. Our design enhances the privacy of DL as it significantly complicates the task of adversarial RNs to extract sensitive information from received model chunks.

4.1 System Model

In SHATTER, each participating RN spawns and operates VNs. We call an RN N_i the *parent* of a certain VN if the latter was spawned by N_i. Each VN has a unique identifier that identifies the node in the network. We assume a permissioned network, a standard assumption of many DL approaches. In the t^{th} communication round, let $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_t)$ be the graph on the VNs, where $\mathcal{V}$ is the set of VNs and $\mathcal{E}_t$ is the set of edges (communication) established between the VNs in round t. SHATTER connects VNs using a r-regular communication graph, which is a commonly used topology across DL algorithms due to its simplicity and load balancing. With this topology, each VN has exactly r incoming and outgoing edges, *i.e.*, each VN sends to and receives from r other VNs. The convergence of DL on r-regular graphs has extensively been studied [18]. RNs and VNs can go offline or come back online during model training.

In line with Section 2.2, we consider all RNs to be honest-but-curious, *i.e.*, the participating RNs faithfully execute the protocol but may attempt to retrieve sensitive information about the other RNs from the model updates it receives through its VNs. We assume that every VN works in the best interest of their corresponding RNs, *i.e.*, an adversarial RN can leverage all its VNs to participate in the attack. We consider that the RNs have full control over their respective VNs in a sense that: *i*) every RN knows which other VNs its own VNs have communicated with in a given training round, and *ii*) the information about which parent RN controls a given VN cannot be retrieved by any external VN or RN (*i.e.*, the link between any RN and its VNs is hidden from everyone else in the network).

4.2 SHATTER Workflow

We show the full workflow of SHATTER in Figure 4 and provide pseudocode in Algorithm 1. The key notations used to describe and analyze the working of SHATTER are summarized in Appendix A.

Algorithm 1: SHATTER from the perspective of RN N_i

1 Initialize $\theta_i^{(0)}$
2 Spawn k VNs: $v_i(1), \dots v_i(k)$
3 **for** $t = 0, \dots, T-1$ **do**
4 $\quad \tilde{\theta}_i^{(t,0)} \leftarrow \theta_i^{(t)}$
5 $\quad$ **for** $h = 1, \dots, H$ **do**
6 $\quad\quad \xi_i \leftarrow$ mini-batch sampled from D_i
7 $\quad\quad \tilde{\theta}_i^{(t,h+1)} \leftarrow \tilde{\theta}_i^{(t,h)} - \eta \nabla f_i(\tilde{\theta}_i^{(t,h)}, \xi_i)$
8 $\quad$ **Chunk** $\tilde{\theta}_i^{(t,H)}$ into k chunks
9 $\quad$ **for** $s = 1, \dots, k$ **do**
10 $\quad\quad$ **Forward** chunk s to $v_i(s)$ for every $s \in [k]$
11 $\quad$ Randomize communication topology $\mathcal{G}_t$
12 $\quad$ **Receive** r chunks from each of the k VNs
13 $\quad$ **Aggregate** the received chunks to produce $\theta_i^{(t+1)}$
14 **return** $\theta_i^{(T)}$

In SHATTER, each participating real node (RN) creates a set of k VNs for itself (Line 2). These VNs are, in practice, implemented as processes on remote systems. To streamline our analysis, we work under the assumption that each RN operates the same number of VNs. We show in Section 5.2 that increasing k, *i.e.*, spawning more VNs, reduces the vulnerability against privacy attacks.

At the start of each training round t, each RN first updates its local model using its private dataset (Lines 5–7). The RN then segments its local model into k smaller model chunks (*chunking*-specifications are given later) and forwards these to its VNs that multicast the model on the RN's behalf (*full sharing* – Lines 8–10). We also visualize this in Figure 4 (left) where some real node N_i segments its local model into three chunks $C_{i,1}, C_{i,2}$, and $C_{i,3}$, each of which is forwarded to a VN of N_i. In standard DL algorithms, RNs directly communicate their model updates. SHATTER instead connects VNs in a communication topology $\mathcal{G}_t$ that is randomized every round (Line 11), also see Figure 4 (middle). In SHATTER, VNs thus act as communication proxy for model chunks created by RNs. All model chunks received by a VN during a training round are forwarded back to its parent RN and aggregated into the local model (Line 13). This is also visualized in Figure 4 (right) where an RN N_i receives model chunks from the VNs operated by RNs N_j, N_k, and N_l. We note that RN N_i is oblivious to the identity of the RNs behind the received model chunks. In essence, from the perspective of a single real node, it sends out the same model parameters as in standard DL. While a real node sends a smaller subset of random parameters to one virtual node, it also sends the remaining parameters in chunks to other virtual nodes, which ultimately enhances privacy without adverse effects on convergence.

Model chunking strategy. Each node randomly samples parameters to chunk its local model without replacement. With this strategy, a RN sends distinct parameters to its VNs. Thus, in one round, two chunks from the same RN will always be disjoint. and, hence, an adversary cannot put together two chunks originating from the same real node based on an intersection of parameters.

Additionally, we use *static* model chunking (*i.e.*, fixed across rounds) where all RNs follow an identical chunking strategy (*i.e.*, they choose the exact same partitions by having a shared seed). That is, each VN is responsible for the same set of parameter indices across rounds. We acknowledge that alternative model chunking strategies are possible, and describe some of these strategies and their impact on privacy in Appendix C.

Dynamic topologies. To enhance the privacy and efficiency of SHATTER, we build SHATTER over EL, the *state-of-the-art* DL algorithm where the communication topology $\mathcal{G}_t$ is refreshed in every training round t. The benefits of having dynamic topologies over static topologies in SHATTER are twofold. Firstly, it restricts any attacker from receiving model chunks consistently from a VN of the same victim node. This reduces the chances of attacks that target specific nodes as an attacker is unable to reliably obtain model chunks from the victim across training rounds. Secondly, dynamic topologies converge faster than static ones thanks to the better mixing of the models [18]. We argue, however, that the components of SHATTER are generic enough to be compatible with other DL baselines, such as D-PSGD or variants.

Refreshing $\mathcal{G}_t$ each round can be achieved by having VNs participate in topology construction using a decentralized peer-sampling service [43, 63, 79]. In this paper, we stick to the *EL-Oracle* model [18] in which a central coordinator agnostic of the identities of RNs creates a random r-regular topology of only VNs every training round. We assume that nodes faithfully participate in the construction of the communication topology, which can be achieved by using accountable peer sampling services [3, 89].

Model aggregation. After the VNs forward the received model parameters back to their parent RN, each RN receives the same total number of parameters in a r-regular topology, but the count of incoming parameters at a particular index may differ. SHATTER uses a parameter-wise weighted averaging where the weight is proportional to the frequency of how often this parameter is received. For example, if a parameter p is received two times, p_1 and p_2 referring to these individual parameters, a RN will average p_1, p_2 and the same parameter in its local model while using an averaging weight of $\frac{1}{3}$ for each parameter. Moreover, it might occur that a RN does not receive a particular parameter at all, in which case it will simply adopt the parameter in its local model.

5 THEORETICAL ANALYSIS

Notations. Let $\mathcal{N} = \{N_1, \dots, N_n\}$ be the RNs in the network. Setting $d \in \mathbb{N}$ as the dimension of the parameter space, let $\theta_i^{(t)} = (\theta_i^{(t)}(1), \dots, \theta_i^{(t)}(d)) \in \mathbb{R}^d$ be the model held by N_i in round t for every $i \in [n]$. Each RN is assumed to have k VNs, with $v_i(s)$ denoting the s^{th} VN of N_i for all $i \in [n]$ and $s \in [k]$, and $\mathcal{V} = \{v_i(s), i \in [n], s \in [k]\}$ being the set of all VNs. Let $\theta_{i,s}^{(t)}$ be the chunk of N_i's model that is forwarded to $v_i(s)$ in round t. For every $i, j \in [n]$ and $p \in [d]$, let $\theta_{i \leftarrow j}^{(t)}(p)$ denote the p^{th} parameter of N_j's model that is shared with N_i and, correspondingly, let $\theta_{i \leftarrow j}^{(t)}$ be the entire contribution that N_i receives from N_j's model updates (via the interaction of their VNs) for aggregation leading up to the next communication round. Analogously, let $\theta_{i \nleftarrow j}^{(t)}$ be the part of N_j's model that N_i has *not* received in round t via their VNs. Let

$\left|\theta^{(t)}_{i\leftarrow j}\right|$ and $\left|\theta^{(t)}_{i\nleftarrow j}\right|$ denote the number of model parameters of $\theta^{(t)}_j$ that are shared and *not* shared with N_i, respectively, in round t (*i.e.*, $\left|\theta^{(t)}_{i\leftarrow j}\right| + \left|\theta^{(t)}_{i\nleftarrow j}\right| = d$). In the subsequent analysis, we assume that:

i. the number of model parameters held by each VN is the same (*i.e.*, $d = kc$ for some $c \in \mathbb{N}$). Thus, the chunking partitions the model of the corresponding RN into k equal parts, each with c parameters).

ii. the kn total VNs form an r-regular dynamic topology, for some $1 \le r \le nk-1$, facilitating interaction among them for the exchange of the model chunks they individually hold.

5.1 Convergence of Shatter

For $i \in [n]$, let $f_i : \mathbb{R}^d \to \mathbb{R}$ be the loss of node N_i with respect to its own dataset, and let $f = \frac{1}{n}\sum_{i=1}^n f_i$ the function that the nodes seek to minimize. Nodes will alternate between rounds of $H \ge 1$ local stochastic gradient steps and communication rounds, that consist of communications through VNs.

We recall that $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_t)$ is the r-regular communication graph on the VNs in round t and let $\mathcal{E}_t$ (the set of edges of $\mathcal{G}_t$) be sampled on $\mathcal{V}$ uniformly at random and independently from the past. A communication round consists of a simple gossip averaging in the VN graph – this could be refined with accelerated gossip averaging steps [27] or compressed communications [47] for instance. Note that for computation steps, *no* noise is added, as opposed to other decentralized privacy-preserving approaches, such as [17] for gossip averaging or [16, 26] for token algorithms. As with communications, computations are also assumed to be synchronous, and employing asynchronous local steps, such as asynchronous SGD, would require a more complex and involved analysis [60, 96]. We can now write the updates of the training algorithm as follows.

(1) *Local steps.* For all $i \in [n]$, let $\tilde{\theta}^{(t,0)}_i = \theta^{(t)}_i$ and for $h \le H-1$,
$$\tilde{\theta}^{(t,h+1)}_i = \tilde{\theta}^{(t,h)}_i - \eta g^{(t,h)}_i ,$$
where $\eta > 0$ is the learning rate and $g^{(t,h)}_i = \nabla f_i(\tilde{\theta}^{(t,h)}_i, \xi_i)$ a stochastic gradient of function f_i computed on the mini-batch ξ_i sampled independently from the past.

(2) *Communication step.* Let $\hat{\theta}^{(t)}_i = \tilde{\theta}^{(t,H)}_i$ and for a VN $v_i(s)$, let $\hat{\theta}^{(t)}_{v_i(s)}$ be its model chunk. Then, an averaging operation is performed on the graph of virtual nodes:
$$\theta^{(t+1)}_i = \sum_{s=1}^{k} \frac{1}{r} \sum_{w\in\mathcal{V}:\{v_i(s),w\}\in\mathcal{E}_t} \hat{\theta}^{(t)}_w .$$

Theorem 1. Assume that functions f_i are $L-$smooth, f is lower bounded and minimized at some $\theta^\star \in \mathbb{R}^d$, the stochastic gradients are unbiased (*i.e.*, $\mathbb{E}\left[g^{(t,h)}_i\right] = \nabla f_i\left(\tilde{\theta}^{(t,h)}_i\right)\ \forall\, i,t,h$) of variance upper bounded by σ^2: $\frac{1}{n}\sum_{i=1}^n \mathbb{E}\left[\left\|g^{(t,h)}_i - \nabla f_i(\tilde{\theta}^{(t,h)}_i)\right\|^2\right] \le \sigma^2$, and let ζ^2 be the population variance, that satisfies:
$$\forall \theta \in \mathbb{R}^d, \quad \frac{1}{n}\sum_{i=1}^n \|\nabla f_i(\theta) - \nabla f(\theta)\|^2 \le \zeta^2 .$$
Finally, assume that $\rho < 1$, where ρ is defined in Equation (3) in the proof of Lemma 5. Then, for all $T > 0$, setting $F_0 = f(\bar{\theta}^{(0)}) - f(\theta^\star)$, there exists a constant stepsize $\eta > 0$ such that:
$$\frac{1}{T}\sum_{t<T}\mathbb{E}\left[\left\|\nabla f(\bar{\theta}^{(t)})\right\|^2\right] = O\left(\sqrt{\frac{LF_0\sigma^2}{nHT}} + \frac{LF_0}{T(1-\rho)} + \left[\frac{LF_0\left(H\zeta + \sigma\sqrt{(1-\rho)H}\right)}{(1-\rho)HT}\right]^{\frac{2}{3}}\right),$$

The proof is postponed to Appendix F.1 and involves an intermediate result given as Lemma 5 preceding the proof of the main result. The convergence bound shows that Shatter finds first-order stationary points of f and. As f is non-convex, we fall back to show that the algorithm will find an approximate first-order stationary point of the objective [12]. The first term $\sqrt{\frac{LF_0\sigma^2}{nHT}}$ is the *statistical rate* and is the largest as T increases. nHT is precisely the number of stochastic gradients computed up to iteration T, and this term cannot be improved [12]. The second and third terms are then lower order terms: for $T = \Omega(nH(1-\rho)^{-2})$, the first term dominates.

Finally, the assumption $\rho < 1$ needs to be verified. For large n, it will always be satisfied for $r > 1$. In the n-large regime, ρ scales as $\rho \sim \frac{1}{r}$ and, thus, increasing r will always lead to faster communications. However, it appears that ρ only needs to be bounded away from 1, as only the $1/(1-\rho)$ factor appears in the rate. Having $\rho \le 1/2$ is sufficient to obtain the best rates, and this only requires $r = O(1)$. There is no need to scale r with n, and the graph of virtual nodes can have bounded degrees.

5.2 Privacy Guarantees

Under the assumed threat model (see Section 4.1) and staying consistent with the experimental evaluation (see Section 6.3), we have that for any given RN N_j in the network, every other RN who receives any part of N_j's model in any given round could independently act as a potential HbC adversary trying to compromise N_j's privacy. It is important to recall that, as we do not consider collusion, the privacy-invasive attacks are *pairwise independent* between the RNs. In other words, if $\mathcal{N}(j,t) \subset \mathcal{N}$ is the set of all the RNs that receive some parts of N_j's model in a certain round t through the interaction between their corresponding VNs, we assume that each $N_i \in \mathcal{N}(j,t)$ can *independently* use the information at their disposal to compromise the privacy of N_j. However, the neighbors of N_j *do not* exchange information between them to impose a colluded attack against N_j. Therefore, in order to examine the privacy guarantees for any RN N_j, it suffices to carry out the analysis from the perspective of any $N_i \in \mathcal{N}(j,t)$. Hence, in the subsequent analysis, without loss of generality, we fix a certain RN N_j as a potential victim and, correspondingly, an HbC adversarial RN $N_i \in \mathcal{N}(j,t)$ trying to compromise the privacy of N_j, thus proceeding to study how Shatter improves the privacy of any RN present in the network.

Setting $\theta^{(t)}_{i\leftarrow v_i(s)}(p)$ as the model parameter $p \in [d]$ that N_i receives from its VN $v_i(s)$ for any $s \in [k]$ after any communication round t, for every $j, j' \in [n]$, $j \ne j'$, we have:
$$\mathbb{P}\left[\theta^{(t)}_{i\leftarrow v_i(s)}(p) = \theta^{(t)}_{i\leftarrow j}(p)\right] = \mathbb{P}\left[\theta^{(t)}_{i\leftarrow v_i(s)}(p) = \theta^{(t)}_{i\leftarrow j'}(p)\right].$$
This is because the system model of Shatter (as described in Section 4.1) assumes that the VNs that communicate with $v_i(s)$ do

not reveal any information about their respective parent RNs and, thus, the model parameters transmitted by them to $v_i(s)$ (which, in turn, N_i receives) are equiprobable to come from any of the other participating RNs. Hence, by achieving perfect indistinguishability among the RNs w.r.t. the model parameters received in any communication round, we analyze the probabilistic characteristics of the received model parameters to establish a fundamental understanding of how empirical risks are influenced in the face of attacks relying on shared model updates (*e.g.*, GIA).

Theorem 2. For any $i, j \in [n]$ with $i \neq j$ and $t > 0$, $\mathbb{P}\left[\left|\theta^{(t)}_{i \nleftarrow j}\right| = 0\right]$, is a decreasing function of k.

Remark 1. Theorem 2 implies that under Shatter, the probability of the entire model of any RN being shared with another RN in any communication round decreases with an increase in the number of VNs. The proof is postponed to Appendix F.2.

Theorem 3. For any $i, j \in [n]$ with $i \neq j$ and $t > 0$, $\mathbb{E}\left[\left|\theta^{(t)}_{i \nleftarrow j}\right|\right]$ is an increasing function of k.

Remark 2. Theorem 3 ensures that under Shatter, the expected number of model parameters received by any RN from any other RN decreases with an increase in the number of VNs. The proof is postponed to Appendix F.3.

In order to formalize the impact of Shatter on the privacy of the shared model parameters between the RNs from an information theoretical perspective, we now analyze the mutual information (MI) [72] (the definition is provided in Appendix F.4) between the model parameters that are shared (observation of an attacker) and *not* shared (secrets) between any pair of RNs. MI and its close variants (*e.g.*, conditional entropy) have been shown to nurture a compatible relationship with formal privacy guarantees like differential privacy (DP) [15] and have been shown to capture an operational interpretation of an attacker model [48]. MI measures the correlation between observations and secrets and its use as a metric to provide an information theoretical understanding of privacy is widespread in the literature. Some noteworthy examples include: gauging anonymity [13, 98], estimating privacy in training ML models with a cross-entropy loss function [1, 41, 70, 78], and assessing location-privacy [8, 66].

Assumption 1. For any $i, j \in [n]$ and $t > 0$, $\theta^{(t)}_j \sim \mathcal{N}\left(\boldsymbol{\mu}^{(t)}_j, \Sigma^{(t)}_j\right)$ for some mean $\boldsymbol{\mu}^{(t)}_j \in \mathbb{R}^d$ and covariance matrix $\Sigma^{(t)}_j \in \mathbb{R}^{d \times d}$.

Assumption 2. There exists $B \in \mathbb{R}$ such that $\left(\Sigma^{(t)}_j\right)_{pp'} \leq B$ for all $p, p' \in [d]$.

Assumption 3. Let $d(t)$ be the number of model parameters of N_j that N_i receives through the interaction of their respective VNs in round t. Setting $X_{ij} = \theta^{(t)}_{i \leftarrow j}$ and $Y_{ij} = \theta^{(t)}_{i \nleftarrow j}$, the eigenvalues of the Schur complement $\Sigma^{(t)}_j / \Sigma^{(t)}_{Y_{ij}}$ of the block $\Sigma^{(t)}_{Y_{ij}}$ in $\Sigma^{(t)}_j = \begin{pmatrix} \Sigma^{(t)}_{X_{ij}} & \Sigma^{(t)}_{X_{ij}Y_{ij}} \\ \Sigma^{(t)}_{Y_{ij}X_{ij}} & \Sigma^{(t)}_{Y_{ij}} \end{pmatrix}$ are bounded below and above by α and β, respectively.

Assumption 4. $\Sigma^{(t)}_j$, $\Sigma^{(t)}_{X_{ij}}$, and $\Sigma^{(t)}_{Y_{ij}}$ are positive-definite.

Theorem 4. If Assumptions 1 to 4 hold, for any $i, j \in [n]$ with $i \neq j$ and $t > 0$, we have $I\left(\theta^{(t)}_{i \leftarrow j}; \theta^{(t)}_{i \nleftarrow j}\right) \leq \Gamma \hat{B}^{d(t)} d(t)^{d(t)/2}$, where $\Gamma = \left(\frac{\alpha}{\beta}\right)^{\frac{\mathrm{tr}\left(\Sigma^{(t)}_j / \Sigma^{(t)}_{Y_{ij}}\right)}{\beta - \alpha}}$ and $\hat{B} = B\left(\frac{\beta^\alpha}{\alpha^\beta}\right)^{\frac{1}{\beta - \alpha}}$.

Remark 3. Due to Theorem 4, Shatter guarantees that, for a certain pair of RNs N_i and N_j, MI between the model parameters of N_j that are shared and *not* shared with N_i in a certain round of communication of their respective VNs is bounded above by an increasing function of the number of parameters of N_j's model that is shared with N_i and bounded below by 0 by the definition of MI. Furthermore, recalling that Theorem 3 ensures that the expected number of model parameters shared between any pair of RNs decreases with an increase in the number of VNs, we essentially guarantee that in every round of Shatter, MI decreases in expectation and, thus, the information-theoretical privacy guarantees of the RNs formally get stronger as the number of VNs increases. The proof of Theorem 4 is postponed to Appendix F.4.

One of the important implications jointly posed by Theorems 2, 3, and 4, as highlighted by Remark 3, is that with an increase in the number of VNs, the chances of an attacker to receive the full model of any user and the average number of model parameters exchanged between the RNs will both decrease and, consequently, the information-theoretical privacy guarantees (parameterized by MI) for the RNs will improve. These results, in turn, provide analytical insight into how Shatter is more resilient towards attacks that rely on exploiting the shared model parameters or gradient updates in their entirety. In particular, the conclusions drawn from Theorems 2, 3, 4, and Remark 3, provide an information-theoretical understanding of how Shatter effectively safeguards against the gradient inversion attack, as evidenced by the experimental results in Figure 8. The findings indicate the diminished effectiveness of gradient inversion attacks with an increasing number of VNs.

Remark 4. Although the privacy of Shatter is studied under the assumption of no collusion in the network, it is noteworthy that the preceding analysis can be extended to scenarios where HbC RNs collude to compromise the privacy of another RN. Colluding RNs targeting a victim RN N_j can be modelled as N_j making larger chunks of its model. As shown above, Shatter enhances N_j's privacy as the number of VNs increases. Thus, Shatter would continue to offer better privacy guarantees compared to EL as long as some level of chunking is maintained, with the worst-case (where all RNs collude against N_j) being equivalent to that of EL.

5.3 Overhead of Shatter

Compared to standard DL algorithms such as D-PSGD, Shatter incurs some communication and operational overhead. This can be considered as the price one has to pay for the privacy protection offered by Shatter. We now analyze these costs.

Communication Costs. We first compare the communication cost of Shatter with that of D-PSGD, in terms of parameters sent. In line with the rest of the paper, we use r to refer to the topology

degree, k indicates the number of VNs each RN deploys, and d indicates the number of parameters in a full model.

For simplicity, we derive the communication cost from the perspective of a single node. In D-PSGD, a node sends d parameters to r nodes, incurring a per-node communication cost of dr. Thus, the communication cost of D-PSGD (and other standard DL algorithms) scales linearly in the model size and topology degree.

In SHATTER, VNs are controlled and hosted by their corresponding RN. Therefore, the communication perspective of RN should include the cost of RN–VN communication as well. We separately analyze the communication cost in each of the following three phases of the SHATTER workflow.

(1) **RN → VNs.** In SHATTER, each RN first sends a model chunk of size $\frac{d}{k}$ to each of its k VNs, resulting in a communication cost of $k\frac{d}{k} = d$ for this step.
(2) **VNs → VNs.** Next, the k VNs operated by an RN send a model chunk of size $\frac{d}{k}$ to r other VNs, incurring a communication cost of $k\frac{d}{k}r = dr$.
(3) **VNs → RN.** Finally, all r VNs who received a chunk in (2) forward these model chunks back to their associated RN. This incurs the same communication cost as step (2): dr.

Therefore, the communication cost of SHATTER is: $d + 2dr$.

The overhead of SHATTER compared to D-PSGD, in number of parameters sent, is as follows:

$$\frac{\text{comm. cost SHATTER}}{\text{comm. cost D-PSGD}} = \frac{d + 2dr}{dr} = 2 + \frac{1}{r}$$

The communication cost of SHATTER is between 2 and 2.5× that of D-PSGD. We do not account for the case where $r = 1$, as such a communication graph would not be connected.

Number of Messages. We now analyze the total number of messages sent in a single round, from the perspective of a single RN. In a round of D-PSGD, a RN sends r messages, one to each of its neighbors. In a round of SHATTER, an RN first sends a single message to its k VNs, VNs exchange kr messages and VNs finally forward kr received messages back to their RN, resulting in a total of $k + 2kr$ messages sent. The overhead of SHATTER compared to D-PSGD, in number of messages sent, is as follows:

$$\frac{\text{num. msgs. SHATTER}}{\text{num. msgs. D-PSGD}} = \frac{k + 2kr}{r} = 2k + \frac{k}{r}$$

The VNs in SHATTER introduce a level of indirection for model transfers, affecting the total communication time in each round. While in D-PSGD models are directly exchanged between RNs, SHATTER routes model chunks through VNs, resulting in additional communication latency. This latency becomes more pronounced if VNs are geographically distant from their parent RNs. The exact increase in latency depends on the underlying network infrastructure and bandwidth capabilities. However, we remark that the time to convergence in DL is dominated by the gradient updates and the training is not latency-critical. Furthermore, this increase in the communication cost and number of messages is an upper bound and can be reduced through optimization techniques such as having VNs aggregate chunks before forwarding them to their RN.

Operational Costs. There are also operational costs involved with hosting and operating VNs. Since the VNs only need to forward messages and partial models, they can be hosted as lightweight containers that do not require heavy computational resources. Nonetheless, they require an active network connection and sufficient memory to ensure correct operation.

6 EVALUATION

Our evaluation answers the following questions: (1) How does the convergence and MIA resilience of SHATTER evolve across training rounds, compared to our baselines (Section 6.2)? (2) How resilient is SHATTER against the privacy attacks in DL (Section 6.3)? (3) How does increasing the number of VNs affect the model convergence and attack resilience (Section 6.4)? (4) What is the contribution of each SHATTER component to privacy and convergence (Section 6.5)?

6.1 Experimental Setup

Implementation and compute infrastructure. We implement SHATTER using the DecentralizePy framework [23] in approximately 4300 lines of Python 3.8.10, relying on PyTorch 2.1.1 [68] to train and implement ML models[1]. Each node (RN and VN) is executed on a separate process, responsible for executing tasks independently of other nodes.

All our experiments are executed on AWS infrastructure, and our compute infrastructure consists of 25 g5.2x large instances. Each instance is equipped with one NVIDIA A10G Tensor Core GPU and 8 second-generation AMD EPYC 7R32 processors. Each instance hosts between 2-4 RNs and 1-16 VNs for each experiment.

DL algorithm and baselines. SHATTER uses EL as the underlying learning algorithm as EL randomizes its communication topology every round. As such, we compare the privacy guarantees offered by SHATTER against those of EL, *i.e.*, in a setting without additional privacy protection. While many privacy-preserving mechanisms are designed for FL, only a few are compatible with DL. We use MUFFLIATO as a baseline for privacy-preserving DL [17]. MUFFLIATO is a noise-based privacy-amplification mechanism in which nodes inject local Gaussian noise into their model updates and then conduct 10 gossiping rounds to average the model updates in each training round. For MUFFLIATO, we randomize the communication topology in each gossiping round. While this results in 10× more topology refreshes in MUFFLIATO compared to EL and SHATTER, we consider this advantage as a fair comparison. For all experiments, unless otherwise stated, we set the number of VNs per RN $k = 8$ and generate a random regular graph with degree $r = 8$ every communication round. We assume that all nodes are online and available throughout the training. Appendix E contains additional experiments with node churn, *i.e.*, intermittent node availabilities.

Privacy attacks. We evaluate SHATTER against *(i)* Loss-based MIA, *(ii)* GIA, and *(iii)* LA. MIA is a representative inference attack, quantified using the negative of the loss values on the training samples of a node and the test set of the given dataset [87]. Since SHATTER works with chunks of the model, LA measures how much a VN links to the training distribution [51] by evaluating each received model update and evaluating the loss on the training

[1] Source code available at https://github.com/sacs-epfl/shatter.

sets of each node. For both MIA and LA, one requires the full model to compute the loss. Therefore, in SHATTER, the adversary complements each chunk of the received model update with the average model of the previous round to approximate the full model before attacking it. The GIA over the sparsified model updates is performed using the *state-of-the-art* ROG [92].

Learning tasks and hyperparameters. In our testbed, we use three datasets: CIFAR-10 [50], Twitter Sent-140 [10, 31], and MovieLens [34] for convergence, MIA, and LA. Additionally, we perform GIA (ROG) on 1600 images from the ImageNet [19] validation set.

The CIFAR-10 dataset is a standard image classification task. For CIFAR-10, we adopt a non-IID data partitioning scheme, assigning training images to 100 DL nodes with a Dirichlet distribution. This is a common scheme to model data heterogeneity in CML [18, 33, 39, 81]. This distribution is parameterized by α that controls the level of non-IIDness (higher values of α result in evenly balanced data distributions). For CIFAR-10, we use $\alpha = 0.1$, which corresponds to a high non-IIDness. This simulates a setting where the convergence is slower in DL and the MIA and LA adversary have an advantage because the local distributions are dissimilar. This task uses a ResNet-18 model (with ≈11.5 million parameters) [37].

Our second dataset is Twitter Sent-140, a sentiment classification task containing tweets partitioned by users [10, 31]. We preprocess the dataset such that each user has at least 24 tweets. All tweets have been annotated with a score between 0 (negative) and 4 (positive). We randomly assign the different authors behind the tweets to 50 DL nodes, a partitioning scheme that is also used in related work on DL [23, 24]. For this task, we initialize a pre-trained BERT-BASE-UNCASED Transformer model (with ≈110 million parameters) [21] from Hugging Face with a new final layer with 2 outputs. We then fine-tune the full model on the Twitter Sent-140 dataset.

We also evaluate SHATTER using a recommendation model based on matrix factorization [49] on the MovieLens 100K dataset, comprising user ratings from 0.5 to 5 for several movies. This task models a scenario where individual participants may wish to learn from the movie preferences of other users with similar interests. Like the Twitter Sent-140 task, we consider the multiple-user-one-node setup where we randomly assign multiple users to a DL node.

Lastly, to present the vulnerability of DL and SHATTER against GIA, we instantiate 100 DL nodes, each with 16 images from the ImageNet validation set. We use the LeNet model to perform one DL training round. ROG is performed on the model updates exchanged after the first training round. Since the reconstructed images are the best after the first training round, we do not continue further DL training and evaluate the strongest version of this attack.

We perform 1 epoch of training before exchanging models for all convergence tasks and 5 epochs of training for the ImageNet GIA. Learning rates were tuned on a grid of varying values. For MUFFLIATO, we comprehensively use two noise levels: *(i)* low: with good convergence, which results in low privacy against MIA, and *(ii)* high: with higher attack resilience to MIA, but at the cost of model utility. Lower noise values present no privacy benefits against EL, and higher noise values adversely affect convergence. More details about the used datasets and hyperparameters can be found in Appendix D. In summary, our experimental setup covers datasets with differing tasks, data distributions, and model sizes, and the hyperparameters were carefully tuned.

Metrics and reproducibility. All reported test accuracies are top-1 accuracies obtained when evaluating the model on the respective test datasets. For MovieLens, we only present the RMSE-loss, as it is not a classification task. To quantify the attack resilience of SHATTER and its baselines, we use the ROC-AUC metric when experimenting with the MIA, representing the area-under-curve when plotting the true positive rate against the false positive rate. For the LA, we report the success rate in percentage, *i.e.*, for each adversary, the success rate is the percentage of *correctly linked* model chunks out of the *total attacked* model chunks. As the attacks are extremely computationally expensive, each RN across all baselines randomly evaluates 8 received model updates for CIFAR-10 and MovieLens, and 4 for the Twitter Sent-140 dataset every training round. Finally, we report the LPIPS score when evaluating ROG, a score between 0 and 1, which indicates the similarity between two batches of images (more similar images correspond to lower LPIPS scores). We run all experiments three times, using different seeds, and present averaged results with confidence intervals.

6.2 Convergence and MIA Across Rounds

We first evaluate how the convergence and resilience against the MIA evolve with the training for SHATTER and the baselines for all three datasets. Each RN operates 8 VNs, and we train until EL convergence. An ideal algorithm would have a low MIA AUC and a high test accuracy (or low test loss for MovieLens) close to EL with no privacy-preserving solution.

Figure 5 shows the convergence (left) and MIA attack success (right) for each dataset (row-wise). EL achieves 51.74%, 83.11%, and 1.10 [RMSE] model utility for CIFAR-10, Twitter Sent-140, and MovieLens, respectively. On the other hand, EL is vulnerable to the loss-based MIA, reaching close to 89.7% AUC on CIFAR-10, 56.1% AUC on Twitter, and 65.8% AUC on MovieLens respectively, where the chance of a random guess is 50%. MUFFLIATO (*low*) achieves similar convergence to EL for all datasets but is also not very effective at reducing vulnerability against the MIA. MUFFLIATO (*high*) instead lowers the MIA vulnerability but at the cost of convergence, finally reaching 2.61% lower test accuracy for Twitter Sent-140, and 1.0 [RMSE] higher test loss for MovieLens (considering the best values). For the CIFAR-10 dataset, convergence is so unstable that there is no learning beyond 12 training rounds. In contrast, SHATTER consistently outperforms the baselines in terms of the resilience to MIA while having no adverse impact on convergence for Twitter Sent-140 and MovieLens, and positively raises the top-1 test accuracy by 3.21% for CIFAR-10. For a given iteration, SHATTER reduces MIA AUC (%) by 4.9-25.8% on CIFAR-10, 3.4-4.0% on Twitter Sent-140, and 0.8-15.1% on MovieLens.

These experiments cover a range of settings, *e.g.*, we find that the extreme non-IIDness in CIFAR-10 adversely affects MUFFLIATO, whereas it positively affects SHATTER. Furthermore, the pre-training in Twitter Sent-140 experiments makes it more challenging for the adversary to perform a MIA as models become less personalized. In CIFAR-10, SHATTER has a higher vulnerability to MIA at the start, decreasing sharply afterward. We notice the accuracy of the models at this point is as good as a random guess (10%), and hence

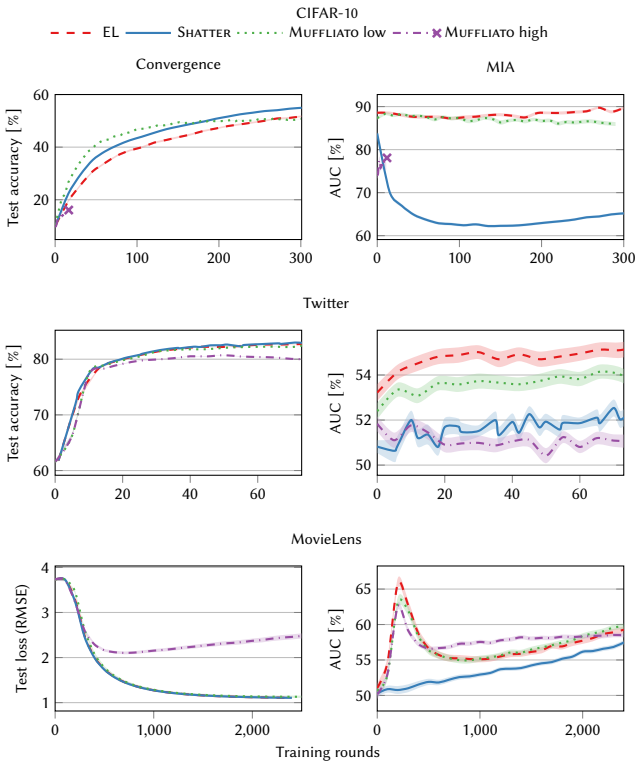

Figure 5: The test accuracy (↑ is better) and MIA AUC (↓ is better) across training rounds for CIFAR-10 (top), Twitter Sent-140 (middle) and the test loss (↓ is better) for MovieLens (bottom), for EL, SHATTER and MUFFLIATO. An AUC of 50% corresponds to an attacker randomly guessing.

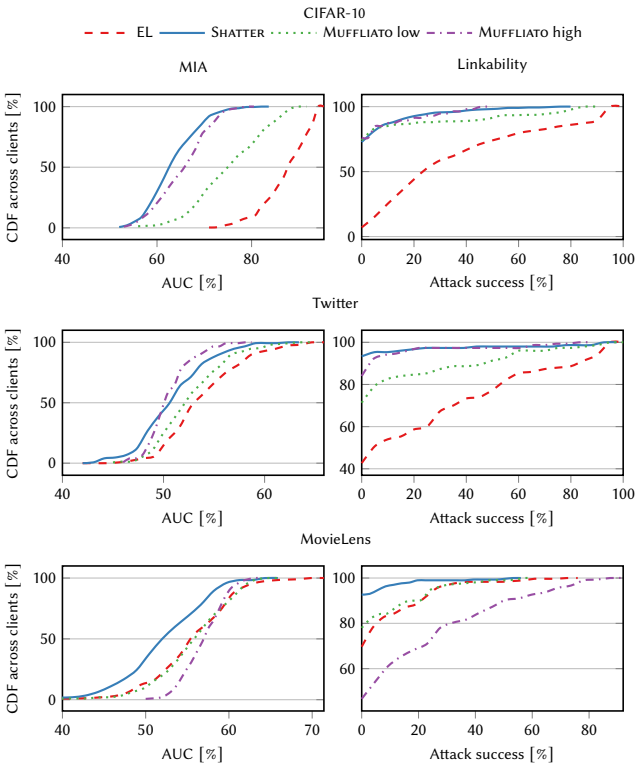

Figure 6: The distribution of MIA AUC (↓ is better) and attack success of the LA across clients for CIFAR-10 (top), Twitter Sent-140 (middle) and MovieLens (bottom) , for EL, SHATTER and MUFFLIATO. An attack success of 2% for the LA corresponds to an attacker randomly guessing on Twitter Sent-140 and 1% on other datasets.

training techniques like regularization can fix it. However, we do not evaluate with regularization and show the worst case from the attack's perspective. Another seemingly unexpected behavior is the higher vulnerability of MUFFLIATO (*high*) against MUFFLIATO (*low*) for the MovieLens dataset. This can be explained by the loss curve on the left, which clearly shows significantly worse generalization for MUFFLIATO (*high*) than MUFFLIATO (*low*).

6.3 Privacy Guarantees of Individual Nodes

We now analyze the vulnerability of individual nodes to the MIA and LA. Ideally, we aim for most nodes to have a low MIA AUC and LA success. Figure 6 shows the CDF of victim nodes against the average MIA AUC (left) and LA success (right) for the baselines. We observe that, consistently across datasets, SHATTER outperforms MUFFLIATO (*low*) and EL in defending against both attacks. MUFFLIATO (*high*) appears to beat SHATTER on the Twitter Sent-140 dataset and appears to be a better defense against LA for CIFAR-10. However, we need to keep in mind that MUFFLIATO (*high*) significantly hurts convergence (see Figure 5).

LA also assesses the vulnerability of the real identities of the nodes that can be leaked through their local data distributions, even when the real identities and model updates are obfuscated. SHATTER preserves the privacy of the nodes by having near-zero linkability for at least 70% across our experiments, up from only 7% nodes with near-zero linkability in EL. Hence, Figure 6 (right) demonstrates that just obfuscating identities, *e.g.*, by using onion routing techniques [32], would not defend against LA. While MUFFLIATO (*high*) is also decent in the defense against LA, this needs to be looked at in conjunction with the convergence. To conclude, SHATTER offers individual clients better privacy protection compared to the baselines and across a range of varying learning tasks.

6.4 Varying the Number of VNs

We next analyze the impact of an increased number of VNs on convergence and resilience against our three privacy attacks. As we increase the number of VNs per RN (k), we keep the degree of the topology constant ($r = 6$), consequently keeping the total bytes transferred constant for SHATTER. We establish the communication and operational costs of SHATTER in Section 5.3.

Convergence. To observe the impact of k on convergence, we run EL and SHATTER with increasing values of k for 300 training rounds with the CIFAR-10 dataset. The convergence bar plot in Figure 7a shows the average test accuracy during a run for different values of k. EL achieves an average test accuracy of 51.7%, which

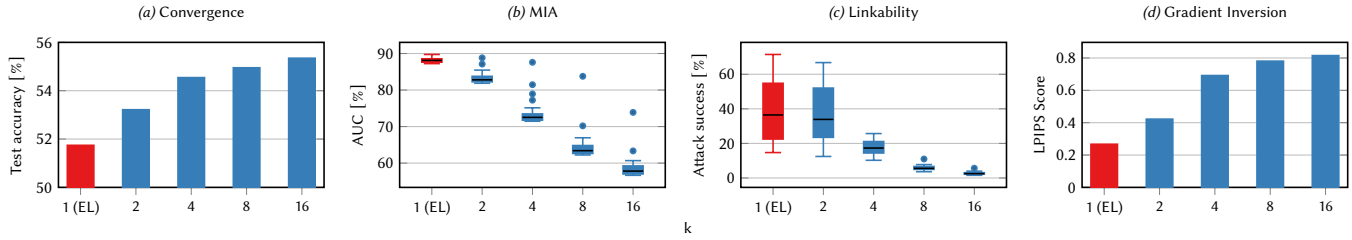

Figure 7: The test accuracy (a, ↑ is better), MIA AUC (b, ↓ is better), attack success rate for LA (c, ↓ is better) on CIFAR-10, and GIA LPIPS score (d, ↑ is better) on ImageNet for increasing values of k (with $k = 1$ corresponding to an EL setting).

increases to 53.2% with $k = 2$. Further increasing k positively affects the test accuracy: increasing k from 2 to 16 increases the attained test accuracy from 53.2% to 55.3%. We attribute this increase in test accuracy to the superior propagation and mixing of model chunks. Furthermore, this increase in test accuracy and faster convergence compensates for the increased communication cost of Shatter.

Attack resilience. Next, we compare the attack resilience of Shatter with different values of k against that of EL, for all three privacy attacks. Figures 7b-d show the LPIPS score, AUC, and attack success percentage for the MIA, LA, and GIA respectively, for increasing values of k. The values for a fixed k are averaged across clients and training rounds. Figure 7b shows that $k = 2$ already defends better against MIA and reduces the median AUC from 88% to 83%. Shatter with $k = 16$ provides significant privacy guarantees, showing a median AUC of only 58%. A similar trend is visible for the LA, as shown in Figure 7c, where an attacker has a success rate of 36.5% in EL and it decreases rapidly with an increase in k. For $k = 16$, a median attack success of merely 2.5% (and at most 4.5%) is observed, underlining the superior privacy benefits of Shatter.

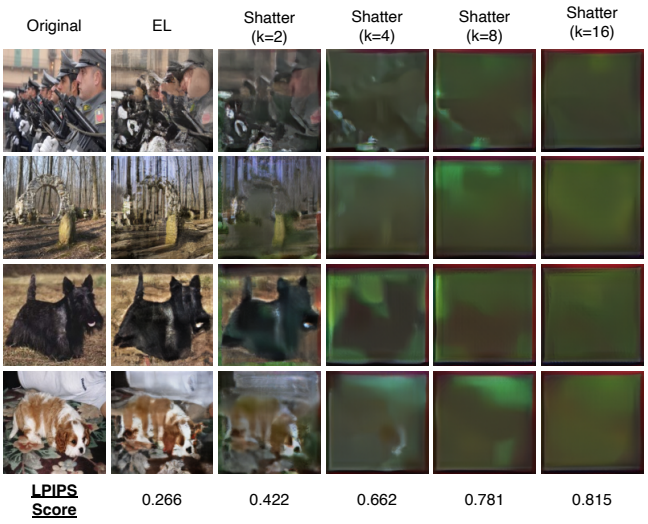

Figure 8: Selected reconstructed images using the GIA, for our baselines and different numbers of VNs (k). We also show the average LPIPS scores (↑ is better) for all 1600 reconstructed images for each setting at the bottom of the figure.

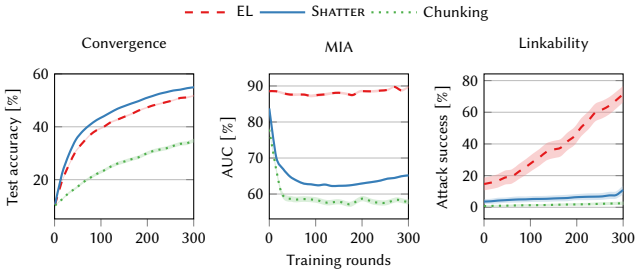

Figure 9: Ablation study: convergence and resistance against MIA and LA with different components of Shatter enabled.

We conduct the GIA on ImageNet when the network learns using a LeNet model, using the experimental setup described in Section 6.1. Four random images from a set of 1600 reconstructed images are shown in Figure 8, with the original image in the left-most column and the images for EL and Shatter for different values of k in the other columns. For each approach, we also show the average LPIPS score [93] over all the images in Figure 7d. For EL, we observe significant similarities between the original and reconstructed images with a low LPIPS score. For $k \geq 4$, the reconstructed images increasingly become too blurry to visually obtain any meaningful information, as evidenced by an increase in the LPIPS score. For $k = 16$, we cannot identify any semantic features in the reconstructed image with a 3.1× higher LPIPS score compared to EL. Employing more VNs, *e.g.*, increasing k, thus results in additional privacy. We discuss the choice of k in more detail in Section 8.

6.5 Ablation Study

We now evaluate the effectiveness of the components of Shatter, *i.e.*, *chunking*, *full sharing* and *virtualization*, in terms of convergence and resistance against the MIA and LA. Figure 9 shows the performance of *chunking*, *i.e.*, sharing one chunk with $1/8^{\text{th}}$ of model parameters and Shatter (*full sharing* + *chunking*) against that of EL. We omit *virtualization* (unlinking identities) in this experiment as it does not affect convergence. Furthermore, the privacy benefits of *chunking* diminish without *virtualization* because an attacker would trivially be able to put together two chunks received from the same source. We observe that *chunking* improves privacy w.r.t. both MIA and LA at the cost of accuracy. With only *chunking*, the nodes reach 16.9 % lower accuracy points than EL and 19.8 %

lower accuracy points than SHATTER with 8 VNs. When comparing SHATTER and *chunking* for privacy, the lower attack success for *chunking* is due to the restrictive learning rate. We tuned the learning rate for optimal convergence and the optimal learning rate for *chunking* is much lower than that of SHATTER. This results in smaller model updates with *chunking* as opposed to SHATTER, making MIA and LA more difficult to carry out. In summary, we elucidate the contributions of the components of SHATTER: *chunking* and *virtualization* bring privacy, and *full sharing* brings utility without compromising the efficiency compared to EL.

7 RELATED WORK

Trusted hardware. Trusted execution environments (TEE) create secure environments on the processors to safeguard data and calculations from untrusted administrative domains [71]. Systems such as ShuffleFL [94], Flatee [61], and Papaya [42] use secure hardware for private averaging to prevent the server from inspecting model updates. GradSec [59] uses ARM TrustZone to prevent inference attacks in FL. ReX [22] uses Intel SGX for secure data sharing in decentralized learning. While these systems effectively hide model updates from the server operator or nodes, they require specialized hardware that is not always available. SHATTER, however, enhances privacy without needing any specialized hardware.

Secure aggregation. Secure aggregation is a privacy-enhancing method to securely share models using masks and has been deployed in the context of FL [9, 29]. These masks are agreed upon before training and cancel out upon aggregation. This scheme requires interactivity and sometimes extensive coordination among participants, requiring all nodes to apply masks, be connected and available. SHATTER avoids using cryptographic techniques and has less strict requirements on coordination among nodes.

Differential privacy. Several differential privacy schemes obtain provable privacy guarantees in DL [17, 54]. These works add noise to model weights before sharing them with the neighbors. While they provide strong theoretical privacy guarantees, we have demonstrated that they often significantly deteriorate model utility. Nevertheless, privacy-critical applications benefit from differential privacy guarantees. In contrast, SHATTER does not introduce such noise and, therefore, avoids the negative effect on model utility.

Onion routing and VNs. The notion of VNs in SHATTER somewhat resembles onion routing, which hides a sender's identity by routing traffic through intermediate nodes [32]. While onion routing techniques can be applied to DL, it would not defend against MIA and GIA, and would be innocuous against LA (*cf.*, Figure 7). VNs were also used in P2P networks, *e.g.*, to ensure load balancing in DHTs [6]. On the other hand, VNs in SHATTER exchange models on behalf of RNs improving the privacy and accuracy of DL.

8 FINAL REMARKS

SHATTER is a novel approach to privacy-preserving DL. It partitions models into chunks and distributes them across a dynamic communication topology of virtual nodes that significantly enhances privacy by preventing adversaries from reconstructing complete models and from identifying the original nodes responsible for specific contributions. We have theoretically and empirically demonstrated the convergence of SHATTER, its formal privacy guarantees, and its resilience against three state-of-the-art privacy attacks across a variety of learning tasks.

Choosing k. The number of VNs per RN, k, is a free parameter in SHATTER. The appropriate value of k in a deployment setting depends on the specifications of the available infrastructures and the sensitivity of the training data as the choice of k directly impacts privacy guarantees and overhead. While environments with powerful infrastructures can use higher values of k to enhance privacy, a lower k may be more appropriate in managing the overhead for settings with limited computational resources. Additionally, highly sensitive data benefits from a higher k, even if this results in more overhead. The choice of k being highly domain- and application-specific, our experiments show that $8 \leq k \leq 16$ strikes a reasonable balance between privacy guarantees and overhead.

SHATTER and differential privacy. SHATTER protects the privacy of individual model updates shared in DL. However, the final model may still leak some information. While this threat model is widely adopted in related work [9, 29, 42], to obtain DP guarantees for extra privacy on the final model, one may inject noise to model parameters before *chunking*. This is, however, orthogonal to SHATTER and comes at the cost of utility.

SHATTER and sparsification. The communication between two VNs resembles random sparsification. SHATTER leverages the fact that it is more difficult to attack a partial set of parameters than the full model. From the perspective of a single RN, SHATTER still disseminates as much information as DL (see Figure 10, Appendix B for details). We remark that SHATTER is compatible with sparsification for improving communication efficiency. For instance, the RN may still use the state-of-the-art sparsification schemes on the trained model parameters and perform *chunking* only on the sparsified parameters. We believe such an analysis would be interesting as a potential future avenue to generalize SHATTER.

Active attacks. While this work focuses on a HbC threat model with a passive adversary, some networks may contain active adversaries that deviate from the specified protocol, *e.g.*, by modifying model parameters or colluding with other nodes. A general defense mechanism against active attacks is difficult to realize due to the large design space of such attacks.

Nevertheless, the components of SHATTER raise the bar to mount particular active attacks. Firstly, SHATTER randomizes the communication topology in each round, which makes it difficult for an attacker to consistently target a specific set of nodes. The latter is a requirement for the successful execution of many active attacks [67]. Secondly, the chunking process involves dividing the model into smaller parts (chunks) and distributing them among VNs. Model chunking limits the exposure of any single part of the model, thereby reducing the potential impact of an active attack. Thanks to the randomization in SHATTER, it is not guaranteed that the chunks by an active attacker end up in the model of a particular victim. The combined effect of these components is that an attacker's ability to mount privacy attacks is significantly crippled. The security of SHATTER can be further extended with auxiliary defense mechanisms to protect against advanced active and Sybil attacks [17, 35, 91]. We aim to explore this in future work.

ACKNOWLEDGMENTS

This work has been funded by the Swiss National Science Foundation, under the project "FRIDAY: Frugal, Privacy-Aware and Practical Decentralized Learning", SNSF proposal No. 10.001.796. Mathieu Even and Laurent Massoulié were supported by the French government under management of the "Agence Nationale de la Recherche" as part of the "Investissements d'avenir" program, reference ANR19-P3IA-0001 (PRAIRIE 3IA Institute). We extend our gratitude to Amazon Web Services (AWS) and the "Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG)", Germany, for providing the compute credits used to run our experiments.

REFERENCES

[1] Martín Abadi and David G. Andersen. 2016. Learning to Protect Communications with Adversarial Neural Cryptography. (2016). arXiv:1610.06918

[2] Dan Alistarh, Torsten Hoefler, Mikael Johansson, Sarit Khirirat, Nikola Konstantinov, and Cédric Renggli. 2018. The Convergence of Sparsified Gradient Methods. In *NeurIPS* (Montréal, Canada). https://proceedings.neurips.cc/paper_files/paper/2018/file/314450613369e0ee72d0da7f6fee773c-Paper.pdf

[3] Alexandros Antonov and Spyros Voulgaris. 2023. SecureCyclon: Dependable Peer Sampling. In *ICDCS*. https://doi.org/10.1109/ICDCS57875.2023.00041

[4] Differential Privacy Team Apple. 2017. Learning with Privacy at Scale. https://machinelearning.apple.com/research/learning-with-privacy-at-scale

[5] Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Mike Rabbat. 2019. Stochastic gradient push for distributed deep learning. In *ICML*. https://proceedings.mlr.press/v97/assran19a.html

[6] Abdalkarim Awad, Christoph Sommer, Reinhard German, and Falko Dressler. 2008. Virtual cord protocol (VCP): A flexible DHT-like routing service for sensor networks. In *2008 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems*. IEEE, 133–142. https://doi.org/10.1109/MAHSS.2008.4660079

[7] Enrique Tomás Martínez Beltrán, Mario Quiles Pérez, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, and Alberto Huertas Celdrán. 2023. Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. *IEEE Communications Surveys & Tutorials* (2023). https://doi.org/10.1109/COMST.2023.3315746

[8] Sayan Biswas and Catuscia Palamidessi. 2023. PRIVIC: A privacy-preserving method for incremental collection of location data. *PoPETs* 2024, 1 (2023). https://doi.org/10.56553/popets-2024-0033 arXiv:2206.10525

[9] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1175–1191. https://doi.org/10.1145/3133956.3133982

[10] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2019. LEAF: A Benchmark for Federated Settings. arXiv:1812.01097 [cs.LG]

[11] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. 2022. Membership Inference Attacks From First Principles. In *2022 IEEE Symposium on Security and Privacy (SP '22)*. https://doi.org/10.1109/SP46214.2022.9833649

[12] Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. 2017. Lower bounds for finding stationary points I. (2017). arXiv:1710.11606

[13] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. 2008. Anonymity Protocols as Noisy Channels. *Information and Computation* 206, 2–4 (2008), 378–401. https://doi.org/10.1016/j.ic.2007.07.003

[14] Pau-Chen Cheng, Kevin Eykholt, Zhongshu Gu, Hani Jamjoom, KR Jayaram, Enriquillo Valdez, and Ashish Verma. 2021. Separation of powers in federated learning (poster paper). In *Proceedings of the First Workshop on Systems Challenges in Reliable and Secure Federated Learning*. https://doi.org/10.1145/3477114.3488765

[15] Paul Cuff and Lanqing Yu. 2016. Differential Privacy As a Mutual Information Constraint. In *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security (CCS)* (Vienna, Austria) *(CCS '16)*. ACM, New York, NY, USA, 43–54. https://doi.org/10.1145/2976749.2978308

[16] Edwige Cyffers and Aurélien Bellet. 2022. Privacy Amplification by Decentralization. In *AISTATS*, Vol. 151. https://proceedings.mlr.press/v151/cyffers22a.html

[17] Edwige Cyffers, Mathieu Even, Aurélien Bellet, and Laurent Massoulié. 2022. Muffliato: Peer-to-Peer Privacy Amplification for Decentralized Optimization and Averaging. In *NeurIPS*. arXiv:2206.05091 https://proceedings.neurips.cc/paper/2022/hash/65d32185f73cbf4535449a792c63926f-Abstract-Conference.html

[18] Martijn de Vos, Sadegh Farhadkhani, Rachid Guerraoui, Anne-Marie Kermarrec, Rafael Pires, and Rishi Sharma. 2023. Epidemic Learning: Boosting Decentralized Learning with Randomized Communication. In *NeurIPS*. arXiv:2310.01972 https://proceedings.neurips.cc/paper_files/paper/2023/hash/7172e147d916eef4cb1eb30016ce725f-Abstract-Conference.html

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*. https://doi.org/10.1109/CVPR.2009.5206848

[20] Jieren Deng, Yijue Wang, Ji Li, Chao Shang, Hang Liu, Sanguthevar Rajasekaran, and Caiwen Ding. 2021. Tag: Gradient attack on transformer-based language models. (2021). arXiv:2103.06819

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2018). arXiv:1810.04805

[22] Akash Dhasade, Nevena Dresevic, Anne-Marie Kermarrec, and Rafael Pires. 2022. TEE-based decentralized recommender systems: The raw data sharing redemption. In *IPDPS*. https://doi.org/10.1109/IPDPS53621.2022.00050 arXiv:2202.11655

[23] Akash Dhasade, Anne-Marie Kermarrec, Rafael Pires, Rishi Sharma, and Milos Vujasinovic. 2023. Decentralized learning made easy with DecentralizePy. In *EuroMLSys*. https://doi.org/10.1145/3578356.3592587 arXiv:2304.08322

[24] Akash Dhasade, Anne-Marie Kermarrec, Rafael Pires, Rishi Sharma, Milos Vujasinovic, and Jeffrey Wigger. 2023. Get More for Less in Decentralized Learning Systems. In *ICDCS*. https://doi.org/10.1109/ICDCS57875.2023.00067 arXiv:2306.04377

[25] John R Douceur. 2002. The sybil attack. In *International workshop on peer-to-peer systems*. Springer. https://doi.org/10.1007/3-540-45748-8_24

[26] Mathieu Even. 2023. Stochastic Gradient Descent under Markovian Sampling Schemes. In *ICML*, Vol. 202. https://proceedings.mlr.press/v202/even23a.html

[27] Mathieu Even, Raphaël Berthier, Francis Bach, Nicolas Flammarion, Hadrien Hendrikx, Pierre Gaillard, Laurent Massoulié, and Adrien Taylor. 2021. Continuized Accelerations of Deterministic and Stochastic Gradient Descents, and of Gossip Algorithms. In *NeurIPS*. https://papers.nips.cc/paper_files/paper/2021/hash/ec26fc2eb2b75aece19c70392dc744c2-Abstract.html

[28] Mathieu Even, Anastasia Koloskova, and Laurent Massoulié. 2024. Asynchronous SGD on Graphs: a Unified Framework for Asynchronous Decentralized and Federated Optimization. In *AISTATS*. arXiv:2311.00465 https://proceedings.mlr.press/v238/even24a.html

[29] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, et al. 2021. SAFELearn: Secure aggregation for private federated learning. In *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE. https://doi.org/10.1109/SPW53761.2021.00017

[30] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting Gradients - How Easy Is It to Break Privacy in Federated Learning?. In *NeurIPS*, Vol. 33. https://proceedings.neurips.cc/paper/2020/file/c4ede56bbd98819ae6112b20ac6bf145-Paper.pdf

[31] Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford* 1, 12 (2009). https://www-cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf

[32] David Goldschlag, Michael Reed, and Paul Syverson. 1999. Onion routing. *Commun. ACM* 42, 2 (1999). https://doi.org/10.1145/293411.293443

[33] Xuan Gong, Abhishek Sharma, Srikrishna Karanam, Ziyan Wu, Terrence Chen, David Doermann, and Arun Innanje. 2022. Preserving Privacy in Federated Learning with Ensemble Cross-Domain Knowledge Distillation. *AAAI* (2022). https://doi.org/10.1609/aaai.v36i11.21446

[34] Grouplens. 2021. MovieLens Datasets. https://grouplens.org/datasets/movielens/

[35] Nirupam Gupta, Thinh T Doan, and Nitin Vaidya. 2023. Byzantine Fault-Tolerance in Federated Local SGD Under $2f$-Redundancy. *IEEE Transactions on Control of Network Systems* 10, 4 (2023). https://doi.org/10.1109/TCNS.2023.3237489

[36] J. HADAMARD. 1893. Resolution d'une question relative aux determinants. *Bull. des Sciences Math.* 2 (1893), 240–246. https://cir.nii.ac.jp/crid/1572824501240738944

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. https://doi.org/10.1109/CVPR.2016.90

[38] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip B. Gibbons. 2020. The Non-IID Data Quagmire of Decentralized Machine Learning. In *ICML*. http://proceedings.mlr.press/v119/hsieh20a/hsieh20a.pdf

[39] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. 2019. Measuring the effects of non-identical data distribution for federated visual classification. (2019). arXiv:1909.06335

[40] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. 2022. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)* 54, 11s (2022). https://doi.org/10.1145/3523273

[41] Chong Huang, Peter Kairouz, Xiao Chen, Lalitha Sankar, and Ram Rajagopal. 2017. Context-aware generative adversarial privacy. *Entropy* 19, 12 (1 12 2017). https://doi.org/10.3390/e19120656

[42] Dzmitry Huba, John Nguyen, Kshitiz Malik, Ruiyu Zhu, Mike Rabbat, Ashkan Yousefpour, Carole-Jean Wu, Hongyuan Zhan, Pavel Ustinov, Harish Srinivas, et al. 2022. Papaya: Practical, private, and scalable federated learning.

MLSys (2022). https://proceedings.mlsys.org/paper_files/paper/2022/hash/a8bc4cb14a20f20d1f96188bd61eec87-Abstract.html
[43] Márk Jelasity, Spyros Voulgaris, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten Van Steen. 2007. Gossip-based peer sampling. *ACM TOCS* 25, 3 (2007). https://doi.org/10.1145/1275517.1275520
[44] Bahman Kalantari and Thomas H. Pate. 2001. A determinantal lower bound. *Linear Algebra Appl.* 326, 1 (2001), 151–159. https://doi.org/10.1016/S0024-3795(00)00287-1
[45] Anastasia Koloskova, Tao Lin, Sebastian U Stich, and Martin Jaggi. 2020. Decentralized Deep Learning with Arbitrary Communication Compression. In *ICLR*. https://openreview.net/pdf?id=SkgGCkrKvH
[46] Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. 2020. A Unified Theory of Decentralized SGD with Changing Topology and Local Updates. In *ICML*, Vol. 119. https://proceedings.mlr.press/v119/koloskova20a.html
[47] Anastasia Koloskova, Sebastian Stich, and Martin Jaggi. 2019. Decentralized stochastic optimization and gossip algorithms with compressed communication. In *ICML*. https://proceedings.mlr.press/v97/koloskova19a.html
[48] Boris Köpf and David A. Basin. 2007. An information-theoretic model for adaptive side-channel attacks. In *CCS*. https://doi.org/10.1145/1315245.1315282
[49] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009). https://doi.org/10.1109/MC.2009.263
[50] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2014. The CIFAR-10 dataset. 55, 5 (2014). https://www.cs.toronto.edu/~kriz/cifar.html
[51] Thomas Lebrun, Antoine Boutet, Jan Aalmoes, and Adrien Baud. 2022. MixNN: protection of federated learning against inference attacks by mixing neural network layers. In *Middleware*. https://doi.org/10.1145/3528535.3565240
[52] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. 2021. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering* (2021). https://doi.org/10.1109/TKDE.2021.3124599
[53] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. 2017. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *NIPS* (2017). arXiv:1705.09056 https://proceedings.neurips.cc/paper_files/paper/2017/file/f75526659f31040afeb61cb7133e4e6d-Paper.pdf
[54] Wanyu Lin, Baochun Li, and Cong Wang. 2022. Towards private learning on decentralized graphs with local differential privacy. *IEEE Transactions on Information Forensics and Security* 17 (2022), 2936–2946. https://doi.org/10.1109/TIFS.2022.3198283
[55] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. 2018. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. In *ICLR*. https://openreview.net/forum?id=SkhQHMW0W
[56] Yugeng Liu, Rui Wen, Xinlei He, Ahmed Salem, Zhikun Zhang, Michael Backes, Emiliano De Cristofaro, Mario Fritz, and Yang Zhang. 2022. ML-Doctor: Holistic Risk Assessment of Inference Attacks Against Machine Learning Models. In *USENIX Security*. https://www.usenix.org/conference/usenixsecurity22/presentation/liu-yugeng
[57] Yang Lu, Zhengxin Yu, and Neeraj Suri. 2023. Privacy-preserving decentralized federated learning over time-varying communication graph. *ACM Transactions on Privacy and Security* 26, 3 (2023). https://doi.org/10.1145/3591354
[58] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*. PMLR. https://proceedings.mlr.press/v54/mcmahan17a/mcmahan17a.pdf
[59] Aghiles Ait Messaoud, Sonia Ben Mokhtar, Vlad Nitu, and Valerio Schiavoni. 2022. Shielding federated learning systems against inference attacks with ARM TrustZone. In *Middleware*. https://doi.org/10.1145/3528535.3565255
[60] Konstantin Mishchenko, Francis Bach, Mathieu Even, and Blake Woodworth. 2022. Asynchronous SGD Beats Minibatch SGD Under Arbitrary Delays. In *NeurIPS*. https://openreview.net/forum?id=4XP0ZuQKXmV
[61] Arup Mondal, Yash More, Ruthu Hulikal Rooparaghunath, and Debayan Gupta. 2021. Flatee: Federated learning across trusted execution environments. (2021). arXiv:2111.06867
[62] Joseph Near, , and David Darais. 2020. Threat models for Differential Privacy. https://www.nist.gov/blogs/cybersecurity-insights/threat-models-differential-privacy
[63] Brice Nédelec, Julian Tanke, Davide Frey, Pascal Molli, and Achour Mostéfaoui. 2018. An adaptive peer-sampling protocol for building networks of browsers. *World Wide Web* 21, 3 (2018), 629–661. https://doi.org/10.1007/s11280-017-0478-5
[64] Angelia Nedić and Alex Olshevsky. 2016. Stochastic gradient-push for strongly convex functions on time-varying directed graphs. *IEEE Trans. Automat. Control* 61, 12 (2016). https://doi.org/10.1109/TAC.2016.2529285
[65] Róbert Ormándi, István Hegedűs, and Márk Jelasity. 2013. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience* 25, 4 (2013), 556–571. https://doi.org/10.1002/cpe.2858
[66] Simon Oya, Carmela Troncoso, and Fernando Pérez-González. 2017. Back to the Drawing Board: Revisiting the Design of Optimal Location Privacy-preserving Mechanisms. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA). ACM, 1959–1972. https://doi.org/10.1145/3133956.3134004
[67] Dario Pasquini, Mathilde Raynal, and Carmela Troncoso. 2023. On the (In) security of Peer-to-Peer Decentralized Machine Learning. In *2023 IEEE Symposium on Security and Privacy (SP)*. 418–436. https://doi.org/10.1109/SP46215.2023.10179291
[68] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS* 32 (2019). arXiv:1912.01703
[69] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. 2017. Knock knock, who's there? Membership inference on aggregate location data. (2017). https://doi.org/10.14722/ndss.2018.23183 arXiv:1708.06145
[70] Marco Romanelli, Kostantinos Chatzikokolakis, and Catuscia Palamidessi. 2020. Optimal Obfuscation Mechanisms via Machine Learning. In *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*. 153–168. https://doi.org/10.1109/CSF49147.2020.00019
[71] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. 2015. Trusted execution environment: what it is, and what it is not. In *2015 IEEE Trustcom/BigDataSE/Ispa*, Vol. 1. 57–64. https://doi.org/10.1109/Trustcom.2015.357
[72] C. E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal* 27, 3 (1948), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x
[73] Chamani Shiranthika, Parvaneh Saeedi, and Ivan V Bajić. 2023. Decentralized learning in healthcare: a review of emerging techniques. *IEEE Access* 11 (2023). https://doi.org/10.1109/ACCESS.2023.3281832
[74] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1310–1321. https://doi.org/10.1145/2810103.2813687
[75] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*. 3–18. https://doi.org/10.1109/SP.2017.41
[76] Elif Ustundag Soykan, Leyli Karaçay, Ferhat Karakoç, and Emrah Tomur. 2022. A survey and guideline on privacy enhancing technologies for collaborative machine learning. *IEEE Access* 10 (2022). https://doi.org/10.1109/ACCESS.2022.3204037
[77] Nikko Strom. 2015. Scalable distributed DNN training using commodity GPU cloud computing. In *16th Annual Conference of the International Speech Communication Association*. https://www.amazon.science/publications/scalable-distributed-dnn-training-using-commodity-gpu-cloud-computing
[78] Ardhendu Tripathy, Ye Wang, and Prakash Ishwar. 2019. Privacy-Preserving Adversarial Networks. In *57th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2019, Monticello, IL, USA, September 24-27, 2019*. IEEE, 495–505. https://doi.org/10.1109/ALLERTON.2019.8919758
[79] Spyros Voulgaris, Daniela Gavidia, and Maarten Van Steen. 2005. Cyclon: Inexpensive membership management for unstructured p2p overlays. *Journal of Network and systems Management* 13 (2005), 197–217. https://doi.org/10.1007/s10922-005-4441-x
[80] Aidmar Wainakh, Till Müßig, Tim Grube, and Max Mühlhäuser. 2021. Label leakage from gradients in distributed machine learning. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE. https://doi.org/10.1109/CCNC49032.2021.9369498
[81] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. *NeurIPS* 33 (2020). https://proceedings.neurips.cc/paper/2020/hash/564127c03caab942e503ee6f810f54fd-Abstract.html
[82] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Hang Su, Bo Zhang, and H Vincent Poor. 2021. User-level privacy-preserving federated learning: Analysis and performance optimization. *IEEE Transactions on Mobile Computing* 21, 9 (2021). https://doi.org/10.1109/TMC.2021.3056991
[83] Wenqi Wei, Ling Liu, Margaret Loper, Ka-Ho Chow, Mehmet Emre Gursoy, Stacey Truex, and Yanzhao Wu. 2020. A framework for evaluating client privacy leakages in federated learning. In *ESORICS*. Springer, 545–566. https://doi.org/10.1007/978-3-030-58951-6_27
[84] Wikipedia contributors. 2024. Hadamard's inequality — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Hadamard%27s_inequality&oldid=1218031906 [Online; accessed 23-April-2024].
[85] Guowen Xu, Guanlin Li, Shangwei Guo, Tianwei Zhang, and Hongwei Li. 2023. Secure decentralized image classification with multiparty homomorphic encryption. *IEEE Transactions on Circuits and Systems for Video Technology* (2023). https://doi.org/10.1109/TCSVT.2023.3234278
[86] Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, Vincent Bindschaedler, and Reza Shokri. 2022. Enhanced membership inference attacks against machine learning models. In *SIGSAC Conference on Computer and Communications Security*.

https://doi.org/10.1145/3548606.3560675
[87] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *IEEE Computer security foundations symposium (CSF)*. IEEE. https://doi.org/10.1109/CSF.2018.00027
[88] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. 2021. See through gradients: Image batch recovery via gradinversion. In *CVPR*. https://doi.org/10.1109/CVPR46437.2021.01607
[89] Man-Ki Yoon. 2023. AccountNet: Accountable Data Propagation Using Verifiable Peer Shuffling. In *ICDCS*. IEEE. https://doi.org/10.1109/ICDCS57875.2023.00050
[90] Da Yu, Huishuai Zhang, Wei Chen, and Tie-Yan Liu. 2021. Do not let privacy overbill utility: Gradient embedding perturbation for private learning. In *ICLR*. arXiv:2102.12677 https://openreview.net/pdf?id=7aogOj_VYO0
[91] Guangsheng Yu, Xu Wang, Caijun Sun, Qin Wang, Ping Yu, Wei Ni, and Ren Ping Liu. 2023. IronForge: an open, secure, fair, decentralized federated learning. *IEEE Transactions on Neural Networks and Learning Systems* (2023). https://doi.org/10.1109/TNNLS.2023.3329249
[92] Kai Yue, Richeng Jin, Chau-Wai Wong, Dror Baron, and Huaiyu Dai. 2023. Gradient obfuscation gives a false sense of security in federated learning. In *USENIX Security*. arXiv:2206.04055 https://www.usenix.org/system/files/sec23summer_372-yue-prepub.pdf
[93] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*. https://doi.org/10.1109/CVPR.2018.00068
[94] Yuhui Zhang, Zhiwei Wang, Jiangfeng Cao, Rui Hou, and Dan Meng. 2021. ShuffleFL: Gradient-preserving federated learning using trusted execution environment. In *Proceedings of the 18th ACM international conference on computing frontiers*. 161–168. https://doi.org/10.1145/3457388.3458665
[95] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2020. iDLG: Improved deep leakage from gradients. (2020). arXiv:2001.02610
[96] Shuxin Zheng, Qi Meng, Taifeng Wang, Wei Chen, Nenghai Yu, Zhi-Ming Ma, and Tie-Yan Liu. 2017. Asynchronous Stochastic Gradient Descent with Delay Compensation. In *ICML*. https://proceedings.mlr.press/v70/zheng17b.html
[97] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *NeurIPS* (2019). https://papers.nips.cc/paper_files/paper/2019/file/60a6c4002cc7b29142def8871531281a-Paper.pdf
[98] Ye Zhu and Riccardo Bettati. 2005. Anonymity vs. Information Leakage in Anonymity Systems. In *ICDCS*. https://doi.org/10.1109/ICDCS.2005.13
[99] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *CCS*. Scottsdale, Arizona. https://doi.org/10.1145/2660267.2660348 arXiv:1407.6981

A TABLE OF NOTATIONS

Notation	Description
	D-PSGD
$\mathcal{N}$	Set of all nodes
$\mathcal{G}$	Graph topology
$\mathcal{E}$	Edges (communication) between the nodes
D_i	Local dataset of node i
ξ_i	Mini-batch sampled from D_i
$\theta_i^{(t)}$	Local model of node i in round t
f_i	Loss function of node i
H	Number of local training steps
	SHATTER
$\mathcal{N}$	Set of all RNs
n	Number of RNs (= $\lvert\mathcal{N}\rvert$)
N_i	i^{th} RN
k	Number of VNs per RN
r	Number of neighbors of each VN
d	Dimension of the parameter space of the models
$\theta_i^{(t)}$	model held by N_i in round t
$v_i(s)$	s^{th} VN of N_i for all $i \in [n]$ and $s \in [k]$
$\theta_{i,s}^{(t)}$	chunk of N_i's model held by $v_i(s)$ in round t
$\theta_{i\leftarrow j}^{(t)}$	Part of N_j's model received by N_i in round t
$\theta_{i\nleftarrow j}^{(t)}$	Part of N_j's model *not* received by N_i in round t
$\mathcal{G}_t$	Communication graph of the VNs in round t
$\mathcal{V}$	Set of all VNs
$\mathcal{E}_t$	Edges (communication) between the VNs in round t

B THE D-PSGD ALGORITHM

We show in Algorithm 2 the standard D-PSGD procedure, from the perspective of node i. Figure 10 illustrates the differences between D-PSGD, Sparsification and SHATTER.

Algorithm 2: The D-PSGD procedure, from the perspective of node i.

Initialize $\theta_i^{(0)}$
for $t = 0, \ldots, T-1$ **do**
  $\tilde{\theta}_i^{(t,0)} \leftarrow \theta_i^{(t)}$
  for $h = 0, \ldots, H-1$ **do**
    $\xi_i \leftarrow$ mini-batch sampled from D_i
    $\tilde{\theta}_i^{(t,h+1)} \leftarrow \tilde{\theta}_i^{(t,h)} - \eta \nabla f_i(\tilde{\theta}_i^{(t,h)}, \xi_i)$
  Send $\tilde{\theta}_i^{(t,H)}$ to the neighbors in topology $\mathcal{G}$
  Receive $\tilde{\theta}_j^{(t,H)}$ from each neighbor j in $\mathcal{G}$
  Aggregate the received models to produce $\theta_i^{(t+1)}$
return $\theta_i^{(T)}$

Table 1: Summary of datasets used for convergence evaluation. For each dataset, the tuned values of learning rate (η), batch size ($\mathcal{B}$), the number of training and testing samples, total model parameters and noise-levels for MUFFLIATO (σ) are presented.

Task	Dataset	Model	η	$\mathcal{B}$	Training Samples	Testing Samples	Total Parameters	MUFFLIATO Noise (σ) Low	Medium	High
Image Classification	CIFAR-10	ResNet-18	0.050	32	50 000	10 000	11 494 592	0.025	0.050	0.100
Sentiment Analysis	Twitter Sent-140	BERT	0.005	32	32 299	8484	109 483 778	0.009	-	0.015
Recommendation	MovieLens	Matrix Factorization	0.075	32	70 000	30 000	206 680	0.025	-	0.250

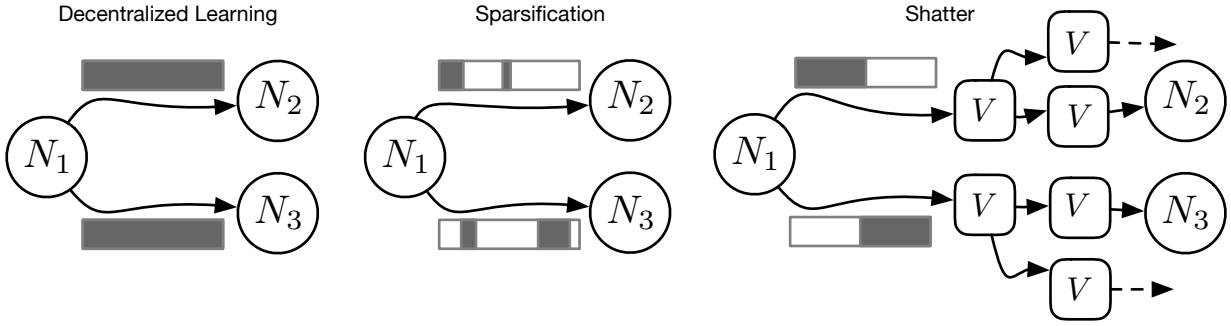

Figure 10: Distinguishing SHATTER from DL and Sparsification.

C ALTERNATIVE MODEL CHUNKING STRATEGY

RNs in SHATTER break down their local model into smaller chunks that are propagated by their VNs. The strategy for model chunking can affect both the privacy guarantees and the convergence of the training process and should therefore be carefully chosen. As discussed in Section 4.2, each RN randomly samples weights from its local model, without replacement. Furthermore, all RNs create chunks with the same indices. We acknowledge that other chunking strategies are possible; we discuss a few alternative model chunking strategies and analyze their trade-offs in terms of privacy and convergence.

1) Linear chunking. With linear chunking, the model parameters are flattened to a one-dimensional array of weights and linearly partitioned into k chunks. Thus, the first $\frac{1}{k}$ parameters are given to the first VN, the next $\frac{1}{k}$ parameters to the second VN etc. However, we experimentally found this strategy to violate privacy since different model chunks carry different amounts of information, *i.e.*, they contain information from different layers. For example, the final chunk, containing the weights associated with the final layer, can leak substantial gradient information [80, 95].

2) Random weight sampling (with replacement). Another strategy involves a RN sampling $\frac{1}{k}$ random weights, with replacement. With this strategy, a RN might send the same weight to multiple of its VNs during a round. However, this risks an adversarial RN inferring that two model chunks originate from the same RN. Specifically, a RN receiving two model chunks that share a common weight at the same position can now, with a high likelihood, conclude that those chunks originate from the local model of the same RN. Furthermore, this strategy might result in a situation where important weights are not sampled at all during a particular training round, which negatively affects performance.

3) Dynamic chunking. While we stick to *static* chunking in SHATTER, another approach would be to change the parameter assignments for each VN over time. While combining this with a random chunking strategy increases the stochasticity of the system, this can potentially be worse for privacy closer to convergence. The parameters do not change much closer to convergence. Therefore, an adversary may continue to listen to the same VN across rounds sending different sets of parameters and may potentially collect all the model parameters of RN. Combining this with the fact that GIA is highly effective close to convergence [67], we may end up with the privacy vulnerability of EL.

D EXPERIMENTAL SETUP DETAILS

Table 1 provides a summary of the datasets along with the hyperparameters used in our testbed. Sentiment analysis over Twitter Sent-140 is a fine-tuning task, whereas we train the models from scratch in the other tasks. In addition to these datasets, we use ImageNet validation set for the GIA using ROG. Two nodes are assigned 16 images each and they train with the batch size $b = 16$ for 5 epochs using the LeNet model architecture. We perturb each node's initial model randomly for increased stochasticity.

E EXPERIMENTS WITH NODE DROPOUTS

We present the convergence of SHATTER in the setting where DL nodes have intermittent connectivity. We simulate different dropout rates for DL nodes and RNs in SHATTER. In each round, nodes decide to participate or not based on the dropout rate. In addition to the dropout rate, we set a dropout correlation factor of 10%, *i.e.*, a node is 10% more likely to drop out if it did not participate in the last

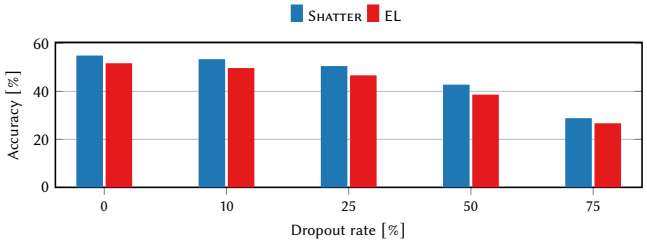

Figure 11: Accuracy achieved after 300 rounds at different dropout rates.

round. When a node does not participate in a round, it does not perform training, sharing, and aggregation. When a RN in Shatter drops out, we assume that its VNs also drop out. While we assumed an $r-regular$ topology in our system model (Section 4.1), this does not hold when nodes have intermittent availabilities. We assume that the communication topology refreshes are agnostic of the availabilities of the nodes.

Figure 11 shows the highest test accuracy reached after 300 rounds of CIFAR-10 training for varying dropout rates. In Shatter, we assume $k = 8$, *i.e.*, 8 VNs per RN. Shatter consistently performs better than EL even with the dropout rate of as high as 75%. Shatter with a 10% dropout rate performs even better than EL without any node dropouts. We notice a drop in the accuracy as the drop rate goes up for both Shatter and EL. This is expected as fewer gradients are computed and fewer parameters are aggregated because some nodes are unavailable for training and aggregation in each round. These results demonstrate that Shatter is not hampered more by node dropouts when compared to the *state-of-the-art* DL baseline of EL. Furthermore, the achieved model utility in Shatter is not significantly affected for topologies slightly deviating from r-regular as seen from the results with the dropout rate of 10%.

F POSTPONED PROOFS

F.1 Proof of Theorem 1

We start by proving the following Lemma.

Lemma 5. Let $C_t : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ be the operator defined as

$$\forall \theta = (\theta_i)_{i\in[n]} \in \mathbb{R}^{d\times n}, \quad C_t(\theta)_i = \sum_{s=1}^{k} \frac{1}{r} \sum_{w\in\mathcal{V}:\{v_i(s),w\}\in\mathcal{E}_t} \theta_w,$$

where for $w = v_i(s)$ a virtual node of N_i, $\theta_w \in \mathbb{R}^d$ contains the subset of coordinates of θ_i associated to the model chunks of $v_i(s)$. We have, for all $\theta = (\theta_i)_{i\in[n]} \in \mathbb{R}^{d\times n}$:

$$\frac{1}{n}\sum_{i=1}^{n} C_t(\theta)_i = \bar{\theta}, \tag{1}$$

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\left\|C_t(\theta)_i - \bar{\theta}\right\|^2\right] \le \frac{\rho}{n}\sum_{i=1}^{n}\left\|\theta_i - \bar{\theta}\right\|^2, \tag{2}$$

where $\bar{\theta} = \frac{1}{n}\sum_{i=1}^{n}\theta_i$ and

$$\rho = \frac{nk}{r(nk-1)}\left(1 + \frac{k-r}{nk-3} + \frac{(k-1)(r-1)}{(nk-2)(nk-3)}\right). \tag{3}$$

Proof of Lemma 5. Let $\theta \in \mathbb{R}^{d\times n}$ and $\ell \in [d]$. For the first part of the lemma, we have:

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n} C_t(\theta)_i &= \frac{1}{n}\sum_{i=1}^{n}\sum_{s=1}^{k}\frac{1}{r}\sum_{w\in\mathcal{V}:\{v_i(s),w\}\in\mathcal{E}_t}\theta_w \\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{s=1}^{k}\frac{1}{r}\sum_{j\in[n],s'\in[k]:\{v_i(s),v_j(s')\}\in\mathcal{E}_t}\theta_{v_j(s')} \\
&= \frac{1}{n}\sum_{j=1}^{n}\sum_{s'=1}^{k}\frac{1}{r}\sum_{i\in[n],s\in[k]:\{v_i(s),v_j(s)\}\in\mathcal{E}_t}\theta_{v_j(s')} \\
&= \frac{1}{n}\sum_{j=1}^{n}\sum_{s'=1}^{k}\theta_{v_j(s')}\frac{1}{r}\sum_{i\in[n],s\in[k]:\{v_i(s),v_j(s)\}\in\mathcal{E}_t} \\
&= \frac{1}{n}\sum_{j=1}^{n}\sum_{s'=1}^{k}\theta_{v_j(s')} = \frac{1}{n}\sum_{j=1}^{n}\theta_j = \bar{\theta}(\ell)
\end{aligned}$$

since $\forall\, j \in [n], s' \in [k]$ we have $\sum_{i\in[n],s\in[k]:\{v_i(s),v_j(s)\}\in\mathcal{E}_t} = \text{degree}(v_j(s')) = r$.

For the second part, we can assume without loss of generality that $\bar{\theta} = 0$, since this quantity is preserved by C_t. In that case,

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\left\|C_t(\theta)_i - \bar{\theta}\right\|^2\right] = \frac{1}{n}\sum_{i=1}^{n}\sum_{\ell=1}^{d}\mathbb{E}\left[(C_t(\theta)_i(\ell))^2\right].$$

Let $\ell \in [d]$ be fixed. For all $i \in [n]$, let $v_i(s_\ell)$ be the chunk that includes coordinate ℓ of model chunks, for some $s_\ell \in [k]$ (without loss of generality, we can assume that this s_ℓ is the same for all i). We have:

$$\begin{aligned}
\mathbb{E}\left[C_t(\theta)_i(\ell)^2\right] &= \mathbb{E}\left[\left(\frac{1}{r}\sum_{s\in[k],j\in[n]:\{v_i(s),v_j(s_\ell)\}\in\mathcal{E}_t}\theta_j(\ell)\right)^2\right] \\
&= \frac{1}{r^2}\sum_{j,j'\in[n]}\sum_{s,s'\in[k]}\theta_j(\ell)\theta_{j'}(\ell) \\
&\quad\times\mathbb{P}\left(\{v_j(s_\ell),v_i(s)\},\{v_{j'}(s_\ell),v_i(s')\}\in\mathcal{E}_t\right).
\end{aligned}$$

For $j = j'$ and $s = s'$,

$$\begin{aligned}
&\mathbb{P}\left(\{v_j(s_\ell),v_i(s)\},\{v_{j'}(s_\ell),v_i(s')\}\in\mathcal{E}_t\right) \\
&\quad= \mathbb{P}\left(\{v_j(s_\ell),v_i(s)\}\in\mathcal{E}_t\right) \\
&\quad= \frac{r}{nk-1},
\end{aligned}$$

which is the probability for a given edge of the virtual nodes graph to be in $\mathcal{E}_t$, a graph sampled uniformly at random from all $r-$regular graphs. Let $j, j' \in [n]\setminus\{i\}$ and $s, s' \in [k]\setminus\{s_\ell\}$. If $j = j'$ but $s \neq s'$,

$$\begin{aligned}
&\mathbb{P}\left(\{v_j(s_\ell),v_i(s)\},\{v_{j'}(s_\ell),v_i(s')\}\in\mathcal{E}_t\right) \\
&= \mathbb{P}\left(\{v_j(s_\ell),v_i(s)\}\in\mathcal{E}_t \middle| \{v_j(s_\ell),v_i(s')\}\in\mathcal{E}_t\right)\times\frac{r}{nk-1},
\end{aligned}$$

and we thus need to compute $\mathbb{P}\left(\{v,w\}\in\mathcal{E}_t \middle| \{v,u\}\in\mathcal{E}_t\right)$, for virtual nodes u, v, w such that $u \neq w$. For all $w, w' \neq u$, by symmetry, we have $\mathbb{P}\left(\{v,w\}\in\mathcal{E}_t \middle| \{v,u\}\in\mathcal{E}_t\right) = \mathbb{P}\left(\{v,w'\}\in\mathcal{E}_t \middle| \{v,u\}\in\mathcal{E}_t\right)$. Then,

$$r = \mathbb{E}\left[\text{degree}(v) \middle| \{v,u\}\in\mathcal{E}_t\right]$$

$$1+\sum_{w'\neq u,v}\mathbb{P}\left(\{v,w'\}\in\mathcal{E}_t\middle|\{v,u\}\in\mathcal{E}_t\right)$$
$$1+(nk-2)\mathbb{P}\left(\{v,w\}\in\mathcal{E}_t\middle|\{v,u\}\in\mathcal{E}_t\right)\,.$$

Thus,

$$\mathbb{P}\left(\{v,w\}\in\mathcal{E}_t\middle|\{v,u\}\in\mathcal{E}_t\right)=\frac{r-1}{nk-2}\,.$$

By symmetry, we obtain the same result for $j\neq j'$, $s\neq s'$. Finally, we need to handle the case $j\neq j'$ and $s\neq s'$. This amounts to computing $\mathbb{P}\left(\{v,w\}\in\mathcal{E}_t\middle|\{v',w'\}\in\mathcal{E}_t\right)$ for four virtual nodes v,v',w,w' that are all different. We have:

$$\begin{aligned}r&=\mathbb{E}\left[\text{degree}(v)\middle|\{v',w'\}\in\mathcal{E}_t\right]\\&=(nk-3)\mathbb{P}\left(\{v,w\}\in\mathcal{E}_t\middle|\{v',w'\}\in\mathcal{E}_t\right)\\&\quad+2\mathbb{P}\left(\{v,v'\}\in\mathcal{E}_t\middle|\{v',w'\}\in\mathcal{E}_t\right)\,.\end{aligned}$$

From what we have already done, $\mathbb{P}\left(\{v,v'\}\in\mathcal{E}_t\middle|\{v',w'\}\in\mathcal{E}_t\right)=\frac{r-1}{nk-2}$, leading to:

$$\mathbb{P}\left(\{v,w\}\in\mathcal{E}_t\middle|\{v',w'\}\in\mathcal{E}_t\right)=\frac{r-2-\frac{r-1}{nk-2}}{nk-3}\,.$$

Thus,

$$\begin{aligned}\mathbb{E}\left[C_t(\theta)_i(\ell)^2\right]&=\frac{1}{r(nk-1)}\sum_{j\in[n],s\in[k]}\theta_j(\ell)^2\\&+\frac{1}{r(nk-1)}\sum_{j\in[n],s\neq s'\in[k]}\theta_j(\ell)^2\frac{r-1}{nk-2}\\&+\frac{1}{r(nk-1)}\sum_{j\neq j'\in[n],s\in[k]}\theta_j(\ell)\theta_{j'}(\ell)\frac{r-1}{nk-2}\\&+\frac{1}{r(nk-1)}\sum_{j\neq j'\in[n],s\neq s'\in[k]}\theta_j(\ell)\theta_{j'}(\ell)\frac{r-2-\frac{r-1}{nk-2}}{nk-3}\\&=\frac{k}{r(nk-1)}\left(1+\frac{(k-1)(r-1)}{(nk-2)}\right)\sum_{j\in[n]}\theta_j(\ell)^2\\&+\frac{k}{r(nk-1)}\frac{r-1}{nk-2}\sum_{j\neq j'\in[n]}\theta_j(\ell)\theta_{j'}(\ell)\\&+\frac{k(k-1)}{r(nk-1)}\frac{r-2-\frac{r-1}{nk-2}}{nk-3}\sum_{j\neq j'\in[n]}\theta_j(\ell)\theta_{j'}(\ell)\,.\end{aligned}$$

Then,

$$\begin{aligned}\sum_{j\neq j'\in[n]}\theta_j(\ell)\theta_{j'}(\ell)&=\left(\sum_{j\in[n]}\theta_j(\ell)\right)^2-\sum_{j\in[n]}\theta_j(\ell)^2\\&=-\sum_{j\in[n]}\theta_j(\ell)^2\,,\end{aligned}$$

leading to:

$$\mathbb{E}\left[C_t(\theta)_i(\ell)^2\right]=\frac{\rho}{n}\sum_{j\in[n]}\theta_j(\ell)^2\,,$$

where

$$\rho=\frac{nk}{r(nk-1)}\left(1+\frac{(k-2)(r-1)}{nk-2}-\frac{(k-1)(r-2-\frac{r-1}{nk-2})}{nk-3}\right)\,,$$

that satisfies

$$\rho\leq\frac{nk}{r(nk-1)}\left(1+\frac{k-r}{nk-3}+\frac{(k-1)(r-1)}{(nk-2)(nk-3)}\right)\,.$$

Importantly, the inequality in Equation (2) for ρ as in Equation (3) is *almost tight*, up to lower order terms in the expression of ρ.

Using this and adapting existing proofs for decentralized SGD with changing topologies and local updates [28, 46] we enable ourself to prove the main result of Theorem 1 to derive the formal convergence guarantees for SHATTER.

PROOF OF THEOREM 1. The proof is based on the proofs of [46, Theorem 2] and [28, Theorem 4.2]: thanks to their unified analyses, we only need to prove that their assumptions are verified for our algorithm. Our regularity assumptions are the same as theirs, we only need to satisfy the ergodic mixing assumption ([46, Assumption 4] or [28, Assumption 2]). For $W_k\equiv C_{k/H}$ if $k\equiv 0[H]$ and $W_k=I_n$ otherwise, we recover their formalism, and [46, Assumption 4] is verified for $\tau=H$ and $p=1-\rho$. However, the proofs of [46, Theorem 2] and [28, Theorem 4.2] require W_k to be gossip matrices: this is not the case for our operators C_t. Still, C_t can in fact be seen as gossip matrices on the set of *virtual nodes*, and the only two properties required to prove [46, Theorem 2] and [28, Theorem 4.2] are that C_t conserves the mean, and the contraction proved in Lemma 5. □

F.2 Proof of Theorem 2

PROOF. For notational convenience, let us define the following:

Definition F.1 (Distance between models). Let $\sigma:\mathbb{R}\mapsto\{0,1\}$ denote the *discrete metric*, *i.e.*, $\forall z,z'\in\mathbb{R}$, $\sigma(z,z')=\begin{cases}0, \text{if } z=z',\\1, \text{otherwise}\end{cases}$.
Then, for a pair of models $\theta,\theta'\in\mathbb{R}^d$, let the *distance* between them be $\delta(\theta,\theta')$ such that $\delta(\theta,\theta')=\sum_{p=1}^{d}\sigma(\theta(p),\theta'(p))$.

We note that $\delta\left(\theta_j^{(t)},\theta_{i\leftarrow j}^{(t)}\right)=\left|\theta_{i\leftrightarrow j}^{(t)}\right|$. Let us fix RNs N_i and N_j where $i,j\in[n]$ and $i\neq j$. Assuming that each VN has a degree of r, when there are k VNs per RN, $\delta\left(\theta_j^{(t)},\theta_{i\leftarrow j}^{(t)}\right)=0$ can occur only when all k VNs of N_j independently communicate with at least one of k VNs of N_i in the given communication round. Therefore, we have:

$$\mathbb{P}\left[\delta\left(\theta_j^{(t)},\theta_{i\leftarrow j}^{(t)}\right)=0\right]=\left(1-\left(1-\frac{r}{nk-1}\right)^k\right)^k$$

In order to prove the desired result, we need to show:

$$\begin{aligned}&\left(1-\left(1-\frac{r}{n(k+1)-1}\right)^{k+1}\right)^{k+1}\\&\leq\left(1-\left(1-\frac{r}{nk-1}\right)^k\right)^k\end{aligned}\tag{4}$$

In order to show (4), we will first show

$$\left(1-\frac{r}{nk-1}\right)^k\leq\left(1-\frac{r}{n(k+1)-1}\right)^{k+1}\tag{5}$$

To this, we define an auxiliary function $F\colon [1,\infty) \mapsto \mathbb{R}$ such that $F(x) = \left(1 - \frac{r}{nx-1}\right)^x$. Hence, for (4) to hold, it suffices to show that $F(.)$ is a non-decreasing function of x. In other words, for every $x \in [1,\infty)$, we wish to show:

$$\begin{aligned}
&\frac{d}{dx}F(x) \geq 0 \\
\iff &\left(\frac{nx-1-r}{nx-1}\right)^x \left(\frac{nrx}{(nx-1)(nx-r-1)} \right. \\
&\left. + \ln\left(\frac{nx-1-r}{nx-1}\right)\right) \geq 0
\end{aligned}$$

Noting that $\left(1 - \frac{r}{nx-1}\right)^x > 0$, it suffices to show:

$$\begin{aligned}
&\frac{nrx}{(nx-1)(nx-r-1)} + \ln\left(1 - \frac{r}{nx-1}\right) \geq 0 \\
\iff &\frac{nrx}{(nx-1)(nx-r-1)} - \ln\left(\frac{nx-1}{nx-r-1}\right) \geq 0 \\
\iff &\frac{nrx}{(nx-1)(nx-r-1)} \geq \ln\left(\frac{nx-1}{nx-r-1}\right) \qquad (6)
\end{aligned}$$

Recalling that *i)* $\ln(z) \leq z-1\ \forall z \geq 0$, *ii)* $n > 1$, $n > 1$, and *iii)* $r < nx - 1$, we get:

$$\ln\left(\frac{nx-1}{nx-r-1}\right) \leq \frac{nx-1}{nx-r-1} - 1 = \frac{r}{nx-r-1} \qquad (7)$$

Thus, incorporating (7) into (6), it suffices to show that:

$$\begin{aligned}
&\frac{nrx}{(nx-1)(nx-r-1)} \geq \frac{r}{nx-r-1} \\
\iff &\frac{nx}{nx-1} \geq 1, \text{ which trivially holds.}
\end{aligned}$$

Hence, we establish that:

$$\begin{aligned}
&\left(1 - \frac{r}{nk-1}\right)^k \leq \left(1 - \frac{r}{n(k+1)-1}\right)^{k+1} \\
\implies &1 - \left(1 - \frac{r}{n(k+1)-1}\right)^{k+1} \\
&\leq 1 - \left(1 - \frac{r}{nk-1}\right)^k \\
\implies &\left(1 - \left(1 - \frac{r}{n(k+1)-1}\right)^{k+1}\right)^k \\
&\leq \left(1 - \left(1 - \frac{r}{nk-1}\right)^k\right)^k \qquad (8)
\end{aligned}$$

But $1 - \left(1 - \frac{r}{n(k+1)-1}\right)^{k+1} \leq 1$ implies:

$$\begin{aligned}
&\left(1 - \left(1 - \frac{r}{n(k+1)-1}\right)^{k+1}\right)^{k+1} \\
&\leq \left(1 - \left(1 - \frac{r}{n(k+1)-1}\right)^{k+1}\right)^k \qquad (9)
\end{aligned}$$

Therefore, combining (8) and (9), we obtain (4), as desired. □

F.3 Proof of Theorem 3

Proof. Setting S as the random variable denoting the number of VNs of N_j that connect with at least one of the VNs of N_i (i.e., the VNs of N_j that are responsible for sharing the corresponding chunks of N_j's model they hold with N_i) and, hence, noting that the number of parameters of N_j's model that are shared with N_i is $\frac{d}{k}S$, we essentially need to show that $\mathbb{E}\left[\delta\left(\theta_j^{(t)}, \theta_{i\leftarrow j}^{(t)}\right)\right] = \mathbb{E}\left[\frac{d}{k}S\right]$ is a decreasing function of k, where $\delta(.)$ is same as has been defined in Definition F.1. By the *law of the unconscious statistician*, we have:

$$\begin{aligned}
&\mathbb{E}\left[\frac{dS}{k}\right] = \sum_{s=0}^{k} \frac{ds}{k}\mathbb{P}\left[S = s\right] \\
&= \sum_{s=0}^{k} \frac{ds}{k}\binom{k}{s}\left(1-(1-p)^k\right)^s (1-p)^{k(k-s)} \\
&\left[\text{where } p = \frac{r}{nk-1}\right] \\
&= \frac{d}{k}\sum_{s=0}^{k} s\binom{k}{s}\left(1-(1-p)^k\right)^s (1-p)^{k(k-s)} \\
&= \frac{d}{k}\mathbb{E}_{X\sim \text{Bin}(k,\pi_k)}(X) \\
&\left[\text{where } \pi_k = 1 - \left(1 - \frac{r}{nk-1}\right)^k\right] \\
&= \frac{d}{k}k\pi_k = d\left(1 - \left(1 - \frac{r}{nk-1}\right)^k\right) \qquad (10)
\end{aligned}$$

By (8) in Theorem 2, we know that $\pi_k = 1 - \left(1 - \frac{r}{nk-1}\right)^k$ is a decreasing function of k. Hence, for a fixed n, d, and r, $\mathbb{E}\left[\frac{dS}{k}\right] = d\pi_k$ is a decreasing function of k. □

F.4 Proof of Theorem 4

Definition F.2 (Mutual information[72])**.** Let (X, Y) be a pair of random variables defined over the discrete space $\mathcal{X} \times \mathcal{Y}$ such that p_{XY} is the joint PMF of X and Y, p_X and p_Y are the corresponding marginal PMFs, and $p_{X|Y}$ is the conditional probability of X given Y. Then the (Shannon) *entropy* of X, $H(X)$, is defined as $H(X) = -\sum_{x\in\mathcal{X}} p_X(x)\log p_X(x)$. The *residual entropy* of X given Y is defined as $H(X|Y) = \sum_{y\in\mathcal{Y}} p_Y(y)H(X|Y=y) = -\sum_{y\in\mathcal{Y}} p_Y(y)\sum_{x\in\mathcal{X}} p_{X|Y}p(x|y)\log p_{X|Y}(x|y)$, and, finally, the *MI* is given by:

$$I(X;Y) = H(X) - H(X|Y) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{XY}(x,y)\ln\frac{p_{XY}(x,y)}{p_X(x)p_Y(y)}$$

Proof. Assumption 1 implies that $X \sim \mathcal{N}\left(\boldsymbol{\mu}_{X_{ij}}^{(t)}, \Sigma_{X_{ij}}^{(t)}\right)$ and $Y \sim \mathcal{N}\left(\boldsymbol{\mu}_{Y_{ij}}^{(t)}, \Sigma_{Y_{ij}}^{(t)}\right)$, where $\boldsymbol{\mu}^{(t)} = \left(\boldsymbol{\mu}_{X_{ij}}^{(t)}, \boldsymbol{\mu}_{Y_{ij}}^{(t)}\right)^{\mathrm{T}}$. Hence,

$$\begin{aligned}
&I\left(X_{ij}; Y_{ij}\right) = D_{\mathrm{KL}}\left(p(X,Y)\,||\,p(X)p(Y)\right) \\
&= D_{\mathrm{KL}}\left(\mathcal{N}\left(\boldsymbol{\mu}_j^{(t)}, \Sigma_j^{(t)}\right)||\,\mathcal{N}\left(\hat{\boldsymbol{\mu}}_j^{(t)}, \hat{\Sigma}_j^{(t)}\right)\right)
\end{aligned}$$

$$\left[\text{where } \hat{\boldsymbol{\mu}}_j^{(t)} = \boldsymbol{\mu}_j^{(t)},\ \hat{\Sigma}_j^{(t)} = \begin{pmatrix} \Sigma_{X_{ij}}^{(t)} & \mathbf{0} \\ \mathbf{0} & \Sigma_{Y_{ij}}^{(t)} \end{pmatrix}\right]$$

$$= \frac{1}{2}\ln\left(\frac{\det\left(\Sigma_{X_{ij}}^{(t)}\right)\det\left(\Sigma_{Y_{ij}}^{(t)}\right)}{\det\left(\Sigma_j^{(t)}\right)}\right)$$

$$= \frac{1}{2}\ln\left(\frac{\det\left(\Sigma_{X_{ij}}^{(t)}\right)}{\det\left(\Sigma_{X_{ij}}^{(t)} - \Sigma_{X_{ij}Y_{ij}}^{(t)}\Sigma_{Y_{ij}}^{(t)-1}\Sigma_{Y_{ij}X_{ij}}^{(t)}\right)}\right) \tag{11}$$

$$\left[\because \det\left(\Sigma_j^{(t)}\right)\right.$$

$$\left.= \det\left(\Sigma_{Y_{ij}}^{(t)}\right)\det\left(\Sigma_{X_{ij}}^{(t)} - \Sigma_{X_{ij}Y_{ij}}^{(t)}\Sigma_{Y_{ij}}^{(t)-1}\Sigma_{Y_{ij}X_{ij}}^{(t)}\right)\right]$$

Recalling Assumption 2, by Hadamard's theorem on determinants [36, 84], we note that

$$\det\left(\Sigma_{X_{ij}}^{(t)}\right) \leq B^{d(t)}d(t)^{d(t)/2} \tag{12}$$

Moreover, observing that $\Sigma_{X_{ij}}^{(t)} - \Sigma_{X_{ij}Y_{ij}}^{(t)}\Sigma_{Y_{ij}}^{(t)-1}\Sigma_{Y_{ij}X_{ij}}^{(t)}$ is the Schur complement $\Sigma_j^{(t)}/\Sigma_{Y_{ij}}^{(t)}$ of the block $\Sigma_{Y_{ij}}^{(t)}$ in $\Sigma_j^{(t)}$, under Assumptions 3 and 4 we use the determinantal lower bounds derived by Kalantari and Pate (Corollary 2 of [44]) to obtain

$$\det\left(\Sigma_j^{(t)}/\Sigma_{Y_{ij}}^{(t)}\right) \geq \alpha^{\kappa}\beta^{d(t)-\kappa} \tag{13}$$

where

$$\kappa = \frac{\beta d(t) - \operatorname{tr}\left(\Sigma_j^{(t)}/\Sigma_{Y_{ij}}^{(t)}\right)}{\beta - \alpha}.$$

Incorporating (12) and (13) into (11), and setting Γ and $\hat{B}$ as defined in the statement of the theorem, we conclude $I(X;Y) \leq \Gamma\hat{B}^{d(t)}d(t)^{d(t)/2}$, as required.

□