Communication-Efficient Secure Aggregation in Decentralized Learning

Sayan Biswas, Anne-Marie Kermarrec, Rafael Pires, Rishi Sharma, and Milos Vujasinovic
EPFL
Lausanne, Switzerland
{sayan.biswas, anne-marie.kermarrec, rafael.pires, rishi.sharma, milos.vujasinovic}@epfl.ch

***Abstract*—Decentralized learning (DL) enables participants to collaboratively train models without a central server, yet it faces significant scalability challenges that demand sparsification to reduce the prohibitive communication costs of peer-to-peer exchange. While secure aggregation effectively mitigates privacy risks in standard settings, it has remained fundamentally incompatible with sparsification in decentralized networks due to the mismatch of indices across local updates, forcing a trade-off between communication efficiency and privacy. This paper introduces CESAR, a novel protocol that resolves this incompatibility by integrating secure aggregation and sparsification to provide provable defense against honest-but-curious and colluding adversaries. By coordinating masks over parameter intersections, CESAR supports node dropouts and robust privacy without central aggregation. Empirical evaluations on models up to 124 million parameters demonstrate that CESAR matches the accuracy of non-private baselines under equal sparsification while cutting total data exchange by $66.7\,\%$ compared to a standard full-parameter decentralized protocol (D-PSGD). With TopK sparsification on IID data, CESAR even exceeds by $0.3\,\%$ the accuracy achieved by D-PSGD with sparsification. Collectively, these results establish CESAR as the first decentralized protocol to achieve both privacy and communication efficiency through secure aggregation in DL.**

***Index Terms*—decentralized machine learning, sparsification, secure aggregation.**

I. Introduction

The performance of modern machine learning (ML) models relies heavily on large, high-quality datasets [1], creating a strong incentive for collaborative learning to pool diverse data from multiple sources [2]. However, direct data sharing poses significant privacy and security challenges, particularly in sensitive domains governed by regulations like general data protection regulation (GDPR) and health insurance portability and accountability act (HIPAA). Decentralized learning (DL) [3] addresses these concerns by keeping data distributed across the network, allowing learning to proceed through the exchange of local model parameters rather than raw data. Although similar approaches have been deployed in centralized settings for applications like text prediction [4] and healthcare [5], DL remains essential when a coordinating entity is infeasible or undesirable. By eliminating the central server to avoid a single point of failure and authority, DL empowers local control and enables distributed decision-making. Yet, despite its growing popularity, this architecture introduces two pressing challenges: communication efficiency and privacy.

Communication Efficiency. Standard DL requires nodes to exchange high-dimensional model parameters with all neighbors every round [6]. As models grow in size, this exchange becomes prohibitively expensive. Sparsification techniques mitigate this cost by transmitting only a selected subset of parameters [7], [8], dramatically reducing communication while largely preserving model accuracy [9].

Privacy. Even without sharing raw data, exchanged model updates can leak sensitive information about local datasets [10], [11], preventing standalone DL deployment in privacy-critical domains. Existing privacy-preserving approaches for DL focus primarily on noise-based [12], [13] and randomized model sharing techniques [14], which degrade utility and slow convergence. By contrast, secure aggregation [15], widely adopted in settings with a centralized aggregator, offers a noise-free alternative: each participant masks its update with coordinated random values that cancel upon summation, revealing only the aggregate. In ML, secure aggregation primarily targets single, large-scale aggregations involving many participants. It remains underexplored in fully decentralized networks where training proceeds through numerous smaller aggregations among few peers each round, and where no central aggregator exists. Adapting secure aggregation to this setting introduces distinct constraints that make protocol design challenging. Tab. I summarizes various approaches for privacy-preserving and communication-efficient DL.

Approach	Comm. eff.	Privacy
Standard DL [6]	✗	✗
Standard DL + sparsification	✓	✗
Noise-based [12], [13]	✗	✓
Secure aggregation	✗	✓
CESAR (ours)	✓	✓

TABLE I: Comparison of training approaches in DL.

While sparsification and secure aggregation individually address communication and privacy, combining them in DL introduces a critical conflict. Secure aggregation requires each shared parameter to be masked and transmitted by multiple

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. The authors thank Sami Abuzakuk, Akash Dhasade, and Martijn de Vos for constructive discussions, and the anonymous reviewers of SRDS 2026 for their valuable feedback and time.

nodes. Yet, sparsification techniques such as TopK [7] and random subsampling [8] select different parameter sets across participants. Consequently, a parameter shared by only one participant creates a dilemma: it must be shared unmasked, aggregated without mask cancellation, or dropped. Each option compromises either privacy or convergence, making this interplay the key obstacle to achieving both communication efficiency and strong privacy in DL.

This paper addresses this gap by introducing CESAR (*communication-efficient secure aggregation*), a secure aggregation protocol for fully decentralized ML. CESAR adapts secure aggregation techniques from centralized frameworks for use with decentralized parallel stochastic gradient descent (D-PSGD), the canonical non-private algorithm for DL, while natively supporting sparsification. Prior work on secure aggregation with sparsification targets federated learning (FL): SparseSecAgg [16] integrates sparsification directly into the aggregation protocol but limits compatibility with other sparsification techniques, while Lu *et al*. [17] incorporate TopK by masking the union of indices selected across nodes. Both approaches rely on a central server and cannot be trivially adapted to DL. By contrast, CESAR operates in a fully serverless setting, coordinates masks over two-hop neighborhoods, and resolves index mismatches by masking only the intersection of locally selected indices. The protocol defends against honest-but-curious (HbC) adversaries, resists collusion through a configurable masking requirement, and gracefully handles node dropouts with minimal communication overhead and minor accuracy impact. Experiments on models up to 124M parameters demonstrate that CESAR matches D-PSGD accuracy while reducing data exchange by $66.7\,\%$ compared to full-parameter sharing, and with TopK on IID data, exceeds D-PSGD accuracy by 0.3%.

Contributions. We make the following contributions:

- We introduce CESAR (Sec. III-C), the first secure aggregation protocol for DL that integrates sparsification and dropout handling without requiring additional servers.
- We formally prove security against HbC adversaries and provide theoretical analysis enabling provable resilience against bounded collusion (Sec. III-D).
- We derive the expected fraction of shared parameters under colluding adversaries (Sec. III-D), enabling principled parameterization of CESAR.
- We evaluate CESAR across five datasets and models up to 124M parameters (Sec. IV-B1), demonstrating accuracy matching D-PSGD with 7–35% communication overhead. For fixed communication budgets, CESAR achieves higher accuracy than both full-parameter secure aggregation and unsparsified D-PSGD (Sec. IV-B3).
- We show that CESAR with dropout support loses only 0.36% accuracy at 10% dropout rate while reducing total data exchange (Sec. IV-B7).

II. Preliminaries

A. Decentralized Learning

In DL nodes $\mathcal{N}$ are organized in a network topology G. Each node $N_i \in \mathcal{N}$ holds a local dataset D_i with distribution φ_i. These datasets remain private throughout training. If local label distributions are homogeneous across nodes, datasets are termed independent and identically distributed (IID), and non-IID otherwise. The collective objective is to find optimal parameters θ^* minimizing a loss function $\mathcal{L}$ across the aggregated dataset, *i.e.*, $\theta^* = \arg\min_\theta \mathcal{L}(D;\theta)$ s.t. $D = \bigcup_{N_j \in \mathcal{N}} D_j$.

Training proceeds over multiple rounds, each comprising three steps: training, sharing, and aggregation. First, each node performs one or more iterations of stochastic gradient descent (SGD) on mini-batches of local data. Next, nodes share updated parameters with all neighbors, as in D-PSGD [6]. Finally, each node averages received parameters with its own model, producing the starting point for the next round. This process iterates until convergence.

B. Sparsification

As models grow increasingly large, transferring them over a network incurs significant costs. Sparsification addresses this through lossy compression, transmitting only a selected subset of parameter indices rather than the entire model.

Two popular approaches are random subsampling [8] and TopK [7]. Both take as input a selection fraction α and a model of size d. Random subsampling selects each index independently with probability α, expecting αd indices. TopK sorts parameters by magnitude and selects the top αd values.

C. Secure Aggregation

Secure aggregation [15] is a multiparty computation technique allowing mutually distrustful parties to jointly compute a linear function of their private inputs, canonically the sum. Given $n \geq 3$ nodes $\mathcal{N} = \{N_1, \ldots, N_n\}$, each node N_i holds a private value v_i and seeks to compute $f(v_1, \ldots, v_n) = \sum_{i=1}^{n} v_i =: \hat{v}$ such that individual values remain private while the aggregate $\hat{v}$ becomes public.

Pairwise masking [15], [18] achieves this by imposing an ordering $N_1 < \ldots < N_n$. Each pair N_i, N_j (s.t. $N_i < N_j$) generates a private random mask $M_{i,j}$. Node N_i adds $M_{i,j}$ to its value while N_j subtracts it, ensuring masks sum to zero across all pairs. Each node N_i computes a masked value $v_i^{'} = v_i - \sum_{j=1}^{i-1} M_{j,i} + \sum_{j=i+1}^{n} M_{i,j}$. This $v_i^{'}$ can be shared publicly, revealing no information about the original v_i.

III. CESAR: Secure aggregation for sparsified decentralized learning

A. System model

We consider a synchronous network of n nodes $\mathcal{N} = \{N_1, \ldots, N_n\}$, interconnected in a communication graph with bidirectional, reliable, and encrypted channels (*e.g.*, TLS over TCP). No central server exists: all nodes are equal in hierarchy. A peer sampling service [19], [20] generates an *aggregation graph* $G(\mathcal{N}, E)$ as an overlay network of the communication graph; all subsequent references to *graph* denote this overlay.

CESAR requires each node to communicate with immediate and second-degree neighbors. We denote immediate neighbors of $N \in \mathcal{N}$ as $\mathrm{View}(N) = \{M \in \mathcal{N} : \{N, M\} \in E\}$ and second-degree neighbors as $\mathrm{View}_2(N) = (\cup_{M \in \mathrm{View}(N)} \mathrm{View}(M)) \setminus \{N\}$. For $N_r, N_i \in \mathcal{N}$ with $\{N_r, N_i\} \in E$, let $\mathrm{Comm}(N_r, N_i) = \mathrm{View}(N_r) \setminus \{N_i\}$ denote nodes that N_i coordinates with to determine index intersections and agree on masks before sharing masked parameters to *receiving node* N_r.

All nodes train models with shared architecture of d parameters enumerated by indices $\mathcal{I} = \{1, \ldots, d\}$. Local parameters of N_i are $v_i \in \mathbb{R}^d$, where $v_i^{(p)}$ denotes the value at index $p \in \mathcal{I}$. The indices selected by N_i for sharing are $I_i \subseteq \mathcal{I}$.

Pseudorandom Mask Generation. Letting Σ denote the space of all seeds, we assume all nodes have access to a pseudorandom number generator $G_\sigma : \mathbb{N} \to \mathbb{Z}_q$. For a given index i, the function $G_\sigma(i)$ returns the i-th element of the pseudorandom sequence determined by $\sigma \in \Sigma$. With $\mathcal{P}(\mathcal{X})$ representing the power set of any set $\mathcal{X}$, we define a *random mask function* $\mathsf{RM}_d : \Sigma \times \Sigma \times \mathcal{P}([d]) \to \mathbb{Z}_q^d$. This function generates a d-dimensional vector where coordinates *not* in $I \subseteq [d]$ are zero. We require this function to satisfy the antisymmetry requirement, $\mathsf{RM}_d(\sigma_i, \sigma_j, I) = -\mathsf{RM}_d(\sigma_j, \sigma_i, I)$, ensuring that pairwise masks cancel upon summation. For the k-th component of the mask, we define:

$$[\mathsf{RM}_d(\sigma_i, \sigma_j, I)]_k = \begin{cases} G_{\sigma_i}(k) - G_{\sigma_j}(k), & \text{if } k \in I, \\ 0, & \text{otherwise.} \end{cases}$$

B. Threat model

CESAR targets permissioned networks with verifiable identities, such as cross-silo collaborative learning. We focus on adversaries who adhere to the protocol, active manipulations are out of scope.

Our primary threat model assumes passive honest-but-curious (HbC) adversaries seeking to extract raw parameters from victim nodes. Any node may act as an HbC adversary using all available information to infer other nodes' models. This threat model is consistent with the standard DL literature [12]–[14]. We consider CESAR in settings both with and without collusion among the adversaries. For the former, we assume the existence of a single colluding set of size ≥ 2 where all retrieved private information is shared among members.

Privacy concerns in this work pertain only to local models. Because secure aggregation does not modify the aggregated model, protecting it falls outside the scope of secure aggregation: noise-based solutions (orthogonal to this work) may address this at reduced utility. We also consider leakage from sparsification metadata out of scope, as such metadata is method-specific. However, sparsification provides privacy benefits over full model sharing (demonstrated in Sec. IV-B6), which we argue outweigh potential metadata leakage.

C. CESAR algorithm

Intuition and Key Challenges. Standard secure aggregation relies on pairwise masks that cancel upon summation: if node N_a holding v_a adds a mask M and node N_b holding v_b subtracts M, then any party that receives and sums *both* contributions observes only $v_a + v_b$. In centralized settings, this condition is trivially satisfied because a single server receives all masked updates and performs the aggregation.

In DL, however, aggregation is local. Each node N aggregates updates only from its neighbors $\mathrm{View}(N)$, and different nodes generally have different neighborhoods. If N_a masks its update with $M_{a,b}$ (agreed with N_b), the corresponding $-M_{a,b}$ must be included in the *same aggregation* for cancellation to occur. That is, every node receiving a masked update from N_a must also receive the counter-masked update from N_b, a condition not guaranteed in general decentralized graphs.

CESAR addresses this by generating *recipient-specific* masks. When node N_i sends an update to a neighbor N_r, it coordinates with all other neighbors of N_r, $\mathrm{Comm}(N_r, N_i)$, to establish pairwise masks. As a result, all nodes contributing to N_r apply masks that cancel when N_r performs aggregation, regardless of differences in their local neighborhoods.

Sparsification introduces a second challenge: independent selection causes nodes N_i and N_j to choose differing index sets I_i and I_j. For indices in the symmetric difference $I_i \triangle I_j$, nodes cannot form cancelling mask pairs, resulting in unmasked transmissions or incomplete cancellation. CESAR addresses this by masking only the shared indices $I_i \cap I_j$ and discarding parameters without sufficient masks, ensuring both cancellation and privacy.

Protocol Walkthrough. Fig. 1 illustrates the execution of CESAR, where nodes N_1, N_2, and N_3 prepare masked updates for their common neighbor N_r. We narrate from the perspective of N_1, though the protocol runs in parallel on all nodes. Each node repeats this process independently for every neighbor, but sparsification is performed only once per round. The masking requirement s is a tunable hyperparameter defining the minimum number of pairwise masks a parameter must carry to be shared. We require s to be consistent across the system (formalized in Assump. 2). Here we set $s = 1$.

Step 1: Sparsification. Each node independently selects indices: N_1 selects $I_1 = \{1, 2, 4\}$, N_2 selects $I_2 = \{1, 2\}$, and N_3 selects $I_3 = \{1, 3\}$.

Step 2: Exchanging Indices and Seeds. To send to N_r, node N_1 coordinates with its mask partners $\mathrm{Comm}(N_r, N_1) = \{N_2, N_3\}$. With each partner N_j, node N_1 exchanges a partial seed σ_{1j} and its index set I_1, receiving σ_{j1} and I_j in return. These index sets determine where masks can be applied: only indices in $I_1 \cap I_j$ can carry a mask agreed with N_j.

Step 3: Masking. Using the exchanged seeds, N_1 computes masks via RM_d: $M_{1,2} = \mathrm{RM}_d(\sigma_{12}, \sigma_{21}, I_1 \cap I_2)$ covering $\{1, 2\}$, and $M_{1,3} = \mathrm{RM}_d(\sigma_{13}, \sigma_{31}, I_1 \cap I_3)$ covering $\{1\}$. By antisymmetry of RM_d, partners N_2 and N_3 independently compute $M_{2,1} = -M_{1,2}$ and $M_{3,1} = -M_{1,3}$ from the same seeds. Node N_1 counts masks per index: index 1 receives two, index 2 receives one, index 4 receives none.

Step 4: Discarding Unmasked Parameters. Any index with fewer than s masks is discarded. With $s = 1$, index 4 fails this threshold and is dropped from I_1.

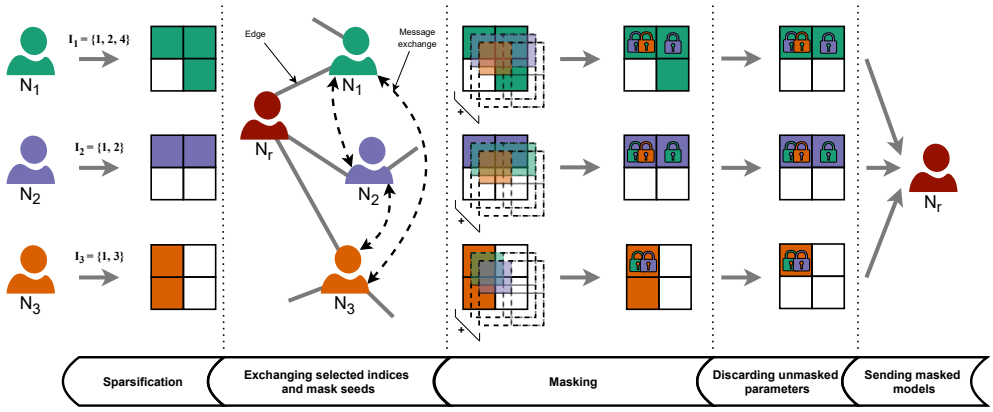

Fig. 1. Overview of CESAR with nodes N_1, N_2, and N_3 sending to common neighbor N_r for $s = 1$. (1) Each node independently sparsifies its local model; (2) Nodes exchange selected indices and partial seeds with mask partners; (3) Each node masks indices in the pairwise intersections; (4) Indices with insufficient masks are discarded; (5) Masked sparse models are sent to N_r.

Algorithm 1: CESAR on node N_i

Input: Model $v_i \in \mathbb{R}^d$, selected indices I_i, masking requirement s
Output: Aggregated model v_i'

for $N_j \in \text{View}_2(N_i)$ **do**
  Send (σ_{ij}, I_i) to N_j; receive (σ_{ji}, I_j)
  $M_{i,j} \leftarrow \text{RM}_d(\sigma_{ij}, \sigma_{ji}, I_i \cap I_j)$
for $N_j \in \text{View}(N_i) \setminus \text{View}_2(N_i)$ **do**
  Send I_i to N_j, receive I_j
for $N_r \in \text{View}(N_i)$ **do**
  $\mathcal{P} \leftarrow \text{Comm}(N_r, N_i) \setminus$ crashed
  $I_{i\to r} \leftarrow \emptyset$
  for $p \in I_i$ **do**
    $c_p \leftarrow$ number of nodes in $\mathcal{P}$ that selected p
    if $c_p \geq s$ **then** add p to $I_{i\to r}$
  $\tilde{v}_{i\to r} \leftarrow (v_i + \sum_{N_j \in \mathcal{P}} M_{i,j})[I_{i\to r}]$
  Send $(I_{i\to r}, \tilde{v}_{i\to r})$ to N_r; receive $(I_{r\to i}, \tilde{v}_{r\to i})$
for $p \in [d]$ **do**
  if *a crashed neighbor had selected* p **then**
    $v_i'^{(p)} \leftarrow v_i^{(p)}$
  else
    $v_i'^{(p)} \leftarrow$ mean of $v_i^{(p)}$ and received values at p
return v_i'

Step 5: Sending Masked Model. Let $I_{1\to r} = \{1, 2\}$ be the surviving indices. N_1 computes masked values $\tilde{v}_{1\to r}^{(p)} = v_1^{(p)} + M_{1,2}^{(p)} + M_{1,3}^{(p)}$ for each $p \in I_{1\to r}$, then transmits $(I_{1\to r}, \tilde{v}_{1\to r})$ to N_r. Upon receiving masked models from all neighbors, N_r aggregates them. Masks cancel since $M_{1,2} + M_{2,1} = 0$ and $M_{1,3} + M_{3,1} = 0$, recovering the true sum of local parameters.

Algorithm Specification. Alg. 1 formalizes CESAR. The notation $v[I]$ extracts the subvector of v at indices I.

Handling Dropouts. In real-world deployments, nodes may disconnect, making robust dropout handling essential. A straightforward approach is as follows: if a receiving node N_r detects that a neighbor has crashed before delivering its masked model, N_r aborts the current round and reuses its local model for the next round.

A more refined strategy is possible. Rather than aborting entirely, N_r can identify and discard only those parameters that the failed node would have contributed, integrating partial contributions from remaining neighbors and preserving most of the aggregation round.

We refer to this method as *parameter-wise dropout handling*, integrated into CESAR as highlighted in blue in Alg. 1. To enable this, each node must share its selected indices not only with second-degree neighbors but also with direct neighbors during mask coordination. Detection itself relies on timeouts: under the synchronous model of Sec. III-A, these realize a perfect failure detector [21].

D. Theoretical analysis

In the interest of space, all proofs are deferred to the extended version of this paper [22].

1) Privacy guarantees: In this section we study the privacy guarantees provided by CESAR. We strictly consider the privacy of parameter values of local models. Consequently, any information leakage from the final aggregated model or leakage caused by sparsification is beyond the scope of this theoretical analysis. Empirical leakage of aggregated models and sparsification is evaluated in Sec. IV-B6.

We consider a parameter private if its exact value cannot be inferred by any other node in the network. A node is considered private if all its parameters are private. In the remainder of this section, we derive a lower bound on the worst-case privacy of CESAR, considering secure aggregation and sparsification.

Theorem 1. *CESAR ensures that no HbC adversary can infer the exact value of any parameter received from its neighbors.*

2) Collusion: DL with colluding adversaries poses significant challenges. Here, we demonstrate how CESAR can be adapted to resist such collusion. In Sec. III-C, we introduced Alg. 1 and explained its operation with a masking requirement of $s = 1$. Increasing s strengthens collusion resistance in a

precise sense: by Th. 5, a masking requirement of s prevents any coalition of at most s colluding adversaries from inferring the exact value of any parameter of the local model of a trusted node. However, increasing s raises concerns about the potential for some masks to be discarded while their corresponding opposite masks are not. Below, we show how the correctness of CESAR remains unaffected by this.

Let us consider a setting where n_t trusted nodes honestly comply with CESAR and k adversarial nodes collude to share all the information they have with each other aiming to violate the privacy of the trusted nodes. Let $\mathcal{N}_T$ and $\mathcal{N}_A$, respectively, denote the sets of the trusted and the colluding adversarial nodes partitioning $\mathcal{N}$. In this setting, studying the privacy guarantees stays relevant only for the trusted nodes. In the subsequent analysis, we make the following assumption.

Assumption 2. *All trusted nodes have the same masking requirement when communicating with a fixed neighbor.*

Theorem 3. *For a fixed $p \in I$ and receiving node N_j, all $N_k \in \text{View}(N_j)$, if $p \in I_k$, apply the same number of masks to the parameter value at index p before discarding.*

Lemma 4. *If the masking requirement of the nodes is k, without worsening the privacy of the trusted nodes, any graph under CESAR, where $|\mathcal{N}_A| \leq k$, can be transformed into a graph where:*

i. $|\mathcal{N}_A| = k$
ii. All trusted nodes have a degree of at least 1
iii. the nodes in $\mathcal{N}_A$ form a clique
iv. $\{N_i, N_j\} \notin E$ *for any* $N_i, N_j \in \mathcal{N}_T$

Th. 3 demonstrates that all neighbors of a node, which have selected the parameter at a predetermined position for transmission to the given node, have an identical number of masks applied to the parameter. Under Assump. 2, it is guaranteed that all such neighbors will either transmit the parameter, thereby cancelling out the masks through aggregation, or, *all* of them will discard the parameter. This ensures that all masks cancel out and the correctness of CESAR remains unaffected.

Theorem 5. *If k is the masking requirement of every trusted node in the network, the true values of their model parameters will remain obfuscated from every node in the network when there are k colluding adversarial nodes.*

As CESAR discards unmasked indices, thus reducing the percentage of model parameters intended for sharing in sparsification, understanding how this varies based on system parameters, masking requirements, and training configuration is vital to evaluate the practical application of CESAR.

Given the difficulty of exhaustively modeling every sparsification approach, we focus on a simplified yet realistic scenario. Here, each index is equally likely to be selected in sparsification, and the selection is mutually independent between nodes. While this assumption does not hold for many sparsification methods, it perfectly characterizes random subsampling. We assume that every node chooses α fraction of their indices in sparsification. Since all nodes select the same fraction and the overall model sizes are identical, the probability $\pi_i^{(p)}$ for any node i to select a given index p is α. This enables derivation of the expected number of model parameters shared by each node in the network under CESAR capturing an environment when we have k colluding nodes.

Theorem 6. *CESAR with random subsampling, where nodes have a masking requirement of s, if $\beta(\alpha, \delta, s)$ is the expected proportion of parameters shared by one node to another in a round, we have $\beta(\alpha, \delta, s) = \sum_{i=s}^{\delta-1} \binom{\delta-1}{i} \alpha^{i+1}(1-\alpha)^{\delta-1-i}$, where α is the probability of independently sampling any parameter by any node and $\delta = |\text{View}(\textit{Receiver})|$.*

Corollary 7. *In the absence of colluding nodes, CESAR with random subsampling ensures that the expected proportion of parameters shared by any node (sender) to another (receiver) in one round is given by $\beta(\alpha, \delta, 1) = \alpha\left(1 - (1-\alpha)^{\delta-1}\right)$, where α is the probability of (independently) sampling any parameter by any node and $\delta = |\text{View}(\textit{Receiver})|$.*

By Corr. 7, as δ increases, β converges to α, *i.e.*, the expected proportion of shared parameters becomes equal to the expected proportion of parameters selected in sparsification. Moreover, this convergence is faster for larger values of α.

3) Communication overhead: In this analysis, α fraction of indices of a model of size d is selected in sparsification. The maximum degree of any node in the graph is $\delta_{\max}$.

The communication overhead incurred by CESAR occurs primarily in the prestep stage. During this phase, each node dispatches a single message to the neighbors of its neighbors. The number of these message transmissions is bounded by $O(\delta_{\max}^2)$. Each of these messages comprises a partial seed of $O(1)$, and a set of indices selected in sparsification of size $O(\alpha d)$. Consequently, the total communication overhead for the prestep in CESAR amounts to $O(\alpha d \delta_{\max}^2)$.

IV. Evaluation

Our evaluation validates CESAR's performance across three critical dimensions: utility, efficiency, and privacy. Specifically, we structure our experiments to answer the following: **RQ1 (Utility & Efficiency):** Can CESAR match the accuracy of non-private baselines (D-PSGD) while significantly reducing communication costs? (Sec. IV-B1 and IV-B3) **RQ2 (Robustness):** How does the masking requirement s affect convergence and resilience against collusion? (Sec. IV-B4 and IV-B5) **RQ3 (Privacy):** Does the protocol effectively limit information leakage against inference attacks compared to standard baselines? (Sec. IV-B6) **RQ4 (Resilience):** Can CESAR maintain model performance in volatile networks with frequent node dropouts? (Sec. IV-B7). We evaluate CESAR against D-PSGD across multiple graph topologies, sparsification fractions, and five datasets with models ranging from 30K to 124M parameters. The implementation is available at https://github.com/sacs-epfl/cesar. It uses Python 3.6 with the *decentralizepy* library [23], built on PyTorch.

A. Experimental Setup

Datasets and Hyperparameters. Experiments span CIFAR-10, CelebA, QASC, MovieLens, and XSum. QASC prompts use the public pool of prompts (P3) [24]. These cover image classification (CIFAR-10, CelebA), question answering (QASC), recommendation (MovieLens), and text summarization (XSum) tasks.

Training data is partitioned so no two nodes share samples and the full test set is used to evaluate performance at each test step (XSum uses the first 5000 of $11\,334$ test samples). For IID distribution, datasets are shuffled and split into equal chunks per node. MovieLens is inherently non-IID. For non-IID partitioning performed only on CIFAR-10, labels are sorted and split into $2n$ chunks, with each node receiving 2 chunks (limiting each node to at most 4 of 10 classes).

Training uses SGD without momentum and hyperparameters are tuned via grid search on D-PSGD with full sharing. Tab. II lists all hyperparameters, dataset sizes, and models.

Hardware and topology. We use two hardware configurations. CIFAR-10, CelebA, and MovieLens run on three machines, each with a 32-core Intel Xeon E5-2630 v3 (2.40 GHz). These experiments use a 48-node network (16 nodes per machine). XSum and QASC run on a cluster with 128 AMD EPYC 7543 cores (2.80 GHz), one NVIDIA A100-80GB (XSum) or three NVIDIA H100-80GB (QASC). These simulate a 32-node network. Aggregation runs entirely on CPU, henceforth GPUs affect only local training speed. All experiments use randomly generated δ-regular graphs, with $\delta \approx \log_2(n)$, a common choice in decentralized networks. Node count remains fixed throughout. Sec. III-D3 confirms that protocol overhead scales with node degree rather than total nodes. All results are averaged across 5 runs with different seeds. Unless stated otherwise, masking requirement $s = 1$. Sec. IV-B5 examines varying s.

Metrics. We evaluate test loss on all datasets and top-1 accuracy on CIFAR-10, CelebA, and QASC. Models trained on XSum are evaluated at training completion using ROUGE-L [29], standard for text summarization tasks [30]. We also measure total data transferred per node, categorized in: (i) mask coordination overhead, (ii) masked model parameters, and (iii) metadata (indices).

Baselines. D-PSGD with Metropolis-Hastings weights [31] is used as the baseline for CESAR. CESAR supports sparsification methods that do not carry information from one round to the next. Therefore, to test the capabilities of CESAR, we run experiments over two prominent sparsification schemes: random subsampling and TopK sparsification. Both are run for different fractions of selected indices. We use TopK only on IID data since previous work has shown that TopK performs poorly in non-IID settings [32].

In all experiments, we invert Corr. 7 using bisection to set sparsification fractions (α) such that the fraction of parameters shared with neighbors (β) in CESAR closely matches the $30\,\%$ and $50\,\%$ marks. Specifically with masking requirement $s = 1$, for nodes with a degree 3, corresponding αs are $43.83\,\%$ and $59.70\,\%$ for $30\,\%$ and $50\,\%$ parameter sharing, respectively. For nodes of degree 6, these values are $34.22\,\%$ and $51.39\,\%$. For nodes of degree 5, a sparsification fraction of $36.03\,\%$ gives $30\,\%$ parameter sharing. We do not use the $50\,\%$ setting for degree 5. Furthermore, to ensure that both algorithms share the exact same fraction of parameters, we first run CESAR and measure the mean fraction of shared weights across all nodes and runs and use the obtained value as the fraction of parameters shared in D-PSGD with the same settings.

Because CESAR works on the basis of obfuscating parameters using masks, we cannot compress the masked parameters. To keep the evaluation consistent, the same is done for D-PSGD. On the other hand, Elias gamma compression is used for compressing indices in both algorithms.

B. Performance

1) Evaluation Across Multiple Datasets: We compare CESAR against D-PSGD across CIFAR-10, CelebA, MovieLens, and XSum on a 6-regular graph with 30% parameter sharing. QASC uses a 5-regular graph under the same conditions. XSum and QASC run on 32 nodes, the remaining datasets use 48 nodes.

Fig. 2 presents results with random subsampling (CIFAR-10 and MovieLens non-IID; CelebA, QASC, and XSum IID). Fig. 3 presents results with TopK (CIFAR-10 IID; CelebA non-IID). On XSum, T5-Efficient-TINY achieves ROUGE-L scores of 14.73 with D-PSGD and 14.72 with CESAR, both starting from 0.00.

CESAR matches D-PSGD performance while incurring at most 35% additional data transfer with TopK and at most 11% with random subsampling. These results confirm that CESAR can replace D-PSGD without needing hyperparameter change.

We observe that overhead data is distributed differently between random subsampling and TopK. In random subsampling, the mask coordination overhead is negligible, as only a seed for generating a random sparsification mask is transmitted. However, after intersecting these masks, the result must be represented as a full set of indices, requiring more metadata later. In contrast, TopK always transfers the selection as a full set of indices, which are sent to all second-degree neighbors during the mask coordination, and this scales with the square of the node degree in the network.

2) In-Depth Analysis on CIFAR-10: We analyze CESAR on CIFAR-10 across 3- and 6-regular graphs at 30% and 50% parameter sharing, using TopK for IID data and random subsampling for non-IID.

Tab. III and Fig. 4 present the results. These empirical findings validate theoretical predictions from Corr. 7.

CESAR accuracy is comparable to D-PSGD across all configurations. On non-IID data with random subsampling, accuracy decreases by up to 0.46% on 3-regular and 0.18% on 6-regular graphs. This reduction stems from precision loss during integer conversion: parameters are converted to integers retaining six decimal places to fit the mask data type.

On IID data with TopK, CESAR outperforms D-PSGD by up to 0.3% on 3-regular and 0.03% on 6-regular graphs. We hypothesize this improvement arises because each parameter

	CIFAR-10	CelebA	QASC	MovieLens	XSum
Learning rate	0.01	0.001	0.001	0.05	0.01
Batch size	8	8	4	64	8
Comm. rounds/epoch	20	10	2	22	15
SGD steps/round	6	10	31	1	53
Training samples	50K	63.7K	8.1K	70K	204K
Task	Img. class.	Img. class.	QA	Recommend.	Text summ.
Model	4 Conv2D	GN-LeNet [25]	GPT2-small [26]	Matrix Fact. [27]	T5-Eff.-TINY [28]
Parameters	89.8K	30.2K	124.4M	206.7K	15.6M

TABLE II: Dataset-specific training hyperparameters, properties, and models.

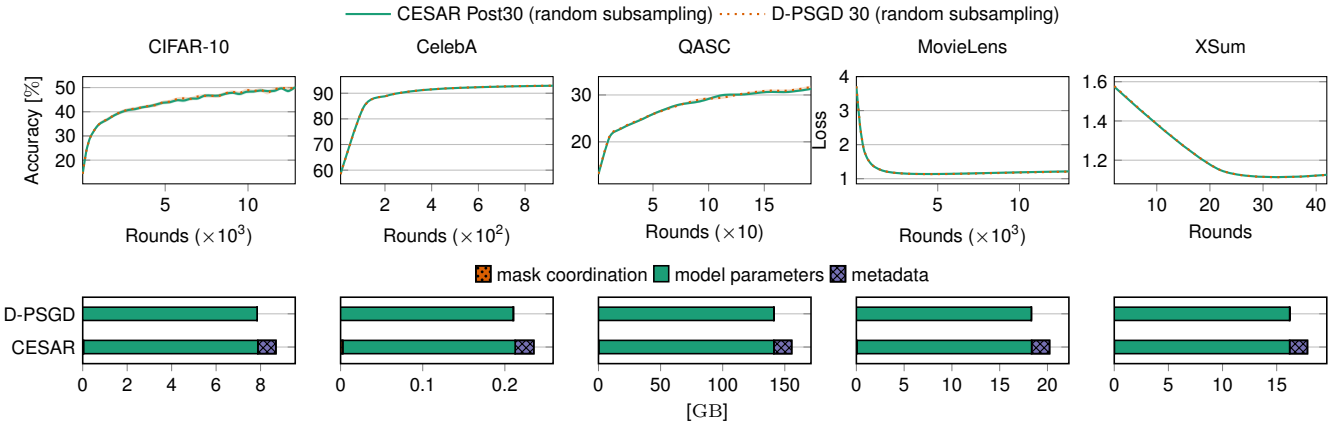

Fig. 2. Comparison of performance and per node communication cost of CESAR against D-PSGD with matching configuration over multiple datasets using random subsampling. CESAR is overlapping D-PSGD in the performance plots.

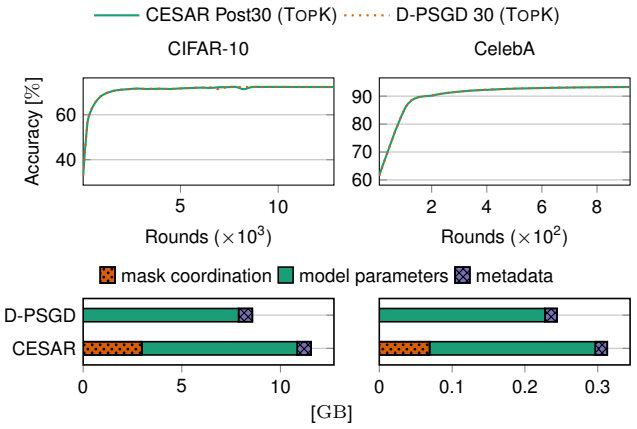

Fig. 3. Comparison of performance and per node communication cost of CESAR against D-PSGD with matching configuration over multiple datasets using TopK. CESAR is overlapping D-PSGD in the performance plots.

in CESAR is received from at least two neighbors improving model generalization.

Primarily due to mask coordination and the sharing of local sparsification results with neighbors, CESAR incurs a data overhead compared to D-PSGD with sparsification, ranging between 12 % and 35 % for TopK, and between 7% and 11% for random subsampling. Furthermore, this overhead varies with the degree of the graph. In random subsampling, it remains nearly constant regardless of the degree since transferring pseudo-random number generator seeds suffices for calculating intersections. With TopK, however, we observe a significant increase as we move from degree 3 to 6 due to transfer of the selected indices.

3) Comparison with secure aggregation: In this section, we examine how the amount of data transferred in CESAR compares directly with secure aggregation. This analysis is performed on IID CIFAR-10 dataset in a 6-regular 48-node network. We consider scenarios where 30% of indices are shared using CESAR and D-PSGD, as well as cases where all indices are shared with D-PSGD and secure aggregation. The results are shown in Fig. 5a.

The results reveal that models trained using CESAR not only achieve higher accuracy for the same amount of data compared to secure aggregation, but they also outperform D-PSGD with full sharing. However, this advantage is observed only up to the point where the models trained with CESAR converge. These findings indicate that despite the overheads incurred during CESAR, it retains the benefits of integrating sparsification, allowing it to achieve same accuracy at lower cost in comparison to secure aggregation.

4) Configuring for Minimized Collusion Risks: Sec. III-D2 discusses formal privacy guarantees in systems with known

Distribution	Sparsification	Degree	% shared weights	Max. Acc. [%] CESAR	Max. Acc. [%] D-PSGD	Data [GB] CESAR	Data [GB] D-PSGD	Time [HH:MM] CESAR	Time [HH:MM] D-PSGD
IID	TopK	3	31.02	71.88	71.58	5.22	4.41	03:19	02:49
			50.48	72.50	72.29	7.90	7.03	03:30	02:53
		6	30.13	72.42	72.40	11.56	8.58	03:44	02:35
			49.67	73.12	73.09	17.28	13.83	04:17	02:44
non-IID	random subsamp.	3	30.00	48.64	49.06	4.34	3.92	02:58	02:40
			50.00	52.54	53.00	7.01	6.53	03:04	02:41
		6	30.00	50.24	50.38	8.69	7.85	03:06	02:44
			50.00	55.44	55.62	14.04	13.07	03:21	02:45

TABLE III: Performance comparison of CESAR against D-PSGD over regular graphs of various degrees and fractions of model shared. Time format: HH:MM (hours:minutes). In all experiments, standard error for elapsed time < 1 minute.

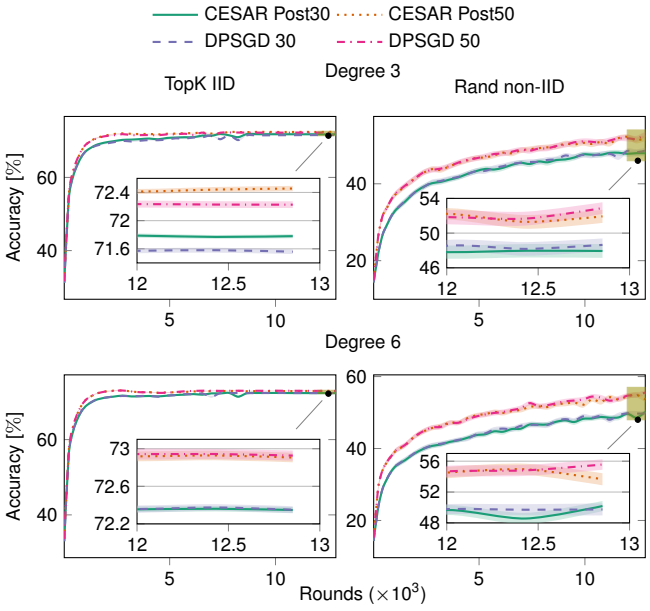

Fig. 4. The accuracy comparison of CESAR against D-PSGD over various configurations, data distributions and network degrees.

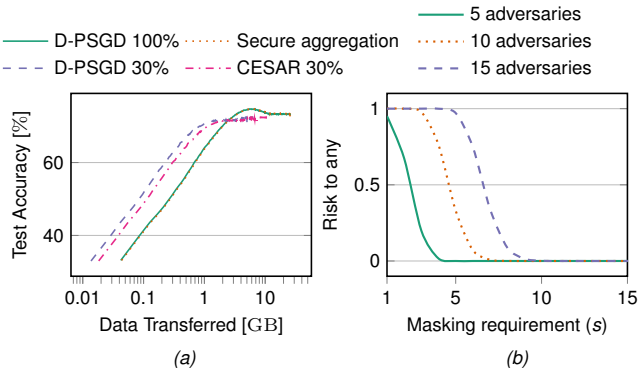

Fig. 5. *(a)* Data transferred to reach a certain accuracy for secure aggregation and full sharing against D-PSGD and CESAR for 30% sharing. *(b)* Estimated attack risk to any honest node in randomly generated 100 node graph with 25-regular topology and a fixed number of colluding adversaries.

number of adversaries. However, this is often not the case, and the number of adversaries has to be estimated. Therefore, we study how the privacy risk changes as we increase the masking requirement (s) in CESAR for a 25-regular graph with 100 nodes in settings with different number of adversaries.

This is performed by running a Monte Carlo simulation 250 000 times for each masking requirement. Each simulation involves generating a random 25-regular graph and assigning each node to be either adversarial or honest. All adversarial nodes are assumed to collude with each other. Over the simulation, we measure the probability that any honest node in the network is at risk of adversaries. A node is considered to be at risk if any of their parameters can be inferred by the adversaries. This can only happen when the node is connected to an adversary who has at least s adversarial neighbors.

The simulation results are shown in Fig. 5b. The results reveal that even for masking requirement lower than the total number of adversaries, yet sufficiently high, the risk can be practically nonexistent to any node in the network. For instance, for the setting with 15 adversaries we do not come across any graphs under risk for masking requirement ≥ 13 in any of our simulations. Furthermore, with masking requirement 9 the risk is 1.45% in this setting.

5) Effects of Masking Requirement: Having seen how increasing s significantly reduces collusion risk, we now examine its impact on convergence and communication overhead. All experiments use a 6-regular 48-node network sharing $30\,\%$ of model parameters, on CIFAR-10 with both IID and non-IID splits. We compare two sparsification strategies: TopK and random subsampling. To maintain a constant $30\,\%$ of parameter sharing, raising s requires increasing the percentage of parameters selected in local sparsification α (from $34.21\,\%$ at $s = 1$, to $42.53\,\%$ at $s = 2$, and $53.34\,\%$ at $s = 3$).

Fig. 6 summarizes convergence and total data exchanged of CESAR against D-PSGD with 30% sharing. With random subsampling the actual percentage of shared parameters stays within $0.001\,\%$ of the $30\,\%$ target for all s, and total data exchanged is within $0.003\,\%$ of D-PSGD. Maximum accuracy

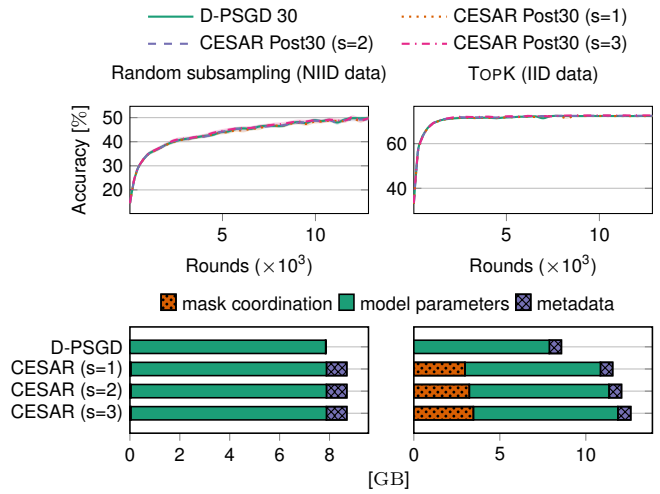

Fig. 6. Comparison of the convergence of CESAR without collusion protection (s=1) and with protection against two (s=2) and three (s=3) colluding nodes, against D-PSGD for the same intended percentage of shared parameters.

varies by at most $0.4\,\%$ points relative to D-PSGD, indicating that increasing s has a negligible effect on both learning and communication when using random subsampling, while significantly strengthening the privacy guarantees of CESAR.

Meanwhile, with TopK, as s grows, maintaining a $30\,\%$ share becomes more challenging: at $s = 3$, we end up sharing $32.13\,\%$. This higher share naturally results in up to $0.3\,\%$ points better accuracy than D-PSGD. However, mask-coordination overhead also rises, since each node shares a larger set of indices selected in local sparsification. In fact, moving from $s = 1$ to $s = 3$ increases total data exchanged by $9.1\,\%$. Thus, as s is increased, alongside improving privacy guarantees, CESAR incurs a marginal communication overhead under TopK sparsification.

6) Privacy leakage from partial sums: We have seen how CESAR keeps local models private in the HbC setting and how these guarantees extend to the presence of colluding nodes. However, even if the exact parameters of a local model remain private, adversaries can collude to learn aggregated information. For example, consider node N_r with local model θ_r and neighbors N_1, N_2, and N_3, with models θ_1, θ_2, and θ_3, respectively. Suppose N_r colludes with N_1 to maximize information extraction. While θ_2 and θ_3 individually remain hidden, the adversaries can infer the sum $\theta_2 + \theta_3$ by subtracting their known models θ_r and θ_1 from the collective sum $\theta_r + \theta_1 + \theta_2 + \theta_3$.

In this section, we analyze privacy leakage as a function of the number of models in a partial sum. We evaluate on a 32-node 5-regular graph using non-IID CIFAR-10. This partitioning ensures that local training distributions are both heterogeneous across nodes and distant from the test distribution, thereby amplifying any leakage and enabling clear comparisons. At the end of each global epoch, every node selects a random neighbor and launches a threshold membership inference attack (MIA) [33] on the last model received from them, measuring the area under the ROC curve (ROC-AUC). It then adds a model of another neighbor (from the same communication round) to the model of the victim, averages the two, and reruns the attack. This process is repeated, cumulatively adding one neighbor at a time, until models of all neighbors have been incorporated.

We evaluate leakage only for the initially selected victim: *member* dataset is the entire local training set of the victim, and *non-member* dataset is the global test set. Although in realistic scenarios an adversary would not access the full training set, this setup provides an upper bound on privacy leakage.

We train using D-PSGD and repeat the experiment under three sharing strategies: full parameter sharing, random subsampling, and TopK sparsification with $30\,\%$ parameter selection. For each fixed partial-sum size, we average results over all nodes and communication rounds. Fig. 7 shows the privacy leakage (ROC-AUC) as a function of the number of models aggregated in the sum.

These results offer several insights: (i) Applying sparsification already reduces leakage compared to full sharing, although different sparsification methods offer varying privacy benefits. (ii) Partial sums leak more than fully aggregated models but less than individual local models, indicating that secure aggregation still mitigates leakage under collusion. (iii) There is a sharp privacy gain as soon as a partial sum includes more than the model of the victim alone. (iv) Compared to a single local model, the fully aggregated model leaks substantially less: its ROC-AUC decreases by 0.24 under full sharing, 0.17 under TopK, and 0.11 under random subsampling.

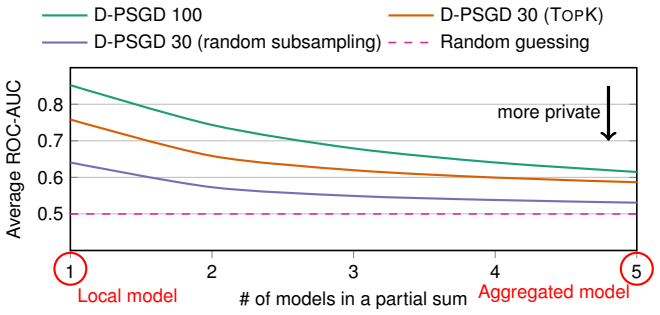

Fig. 7. Privacy leakage from partial sums under a threshold attack on non-IID CIFAR-10.

7) Effect of dropouts: Sec. III-C modifies CESAR to handle node dropouts. We evaluate this on a 48-node 6-regular network training on IID CIFAR-10 with TopK sparsification. Each node selects 34.22% of parameters, yielding approximately 30% sharing after discarding unmasked parameters; the masking requirement is $s = 1$.

Each round, nodes crash independently with fixed probability and rejoin the next round. Crashes are simulated, with neighbors notified directly rather than by timeout. We test 10% and 20% crash rates under worst-case timing: dropouts occur after mask coordination but before masked models are sent.

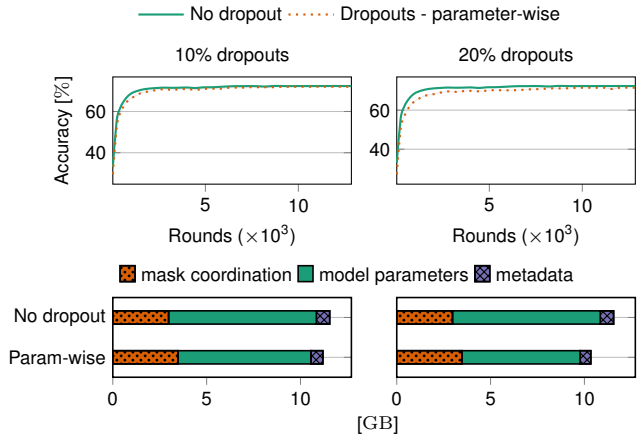

Fig. 8. Comparison of model accuracy and data exchanged by CESAR without dropouts, and with the parameter-wise dropout handling add-on under 10 % and 20 % dropouts.

The results of this experiment are shown in Fig. 8. In this setting, CESAR achieves a maximum accuracy of 72.42% without dropouts and reaches 72.06% and 71.51% accuracy for 10% and 20% dropouts, respectively, when the parameter-wise dropout handling is used. These results demonstrate even with dropouts, the maximum achieved accuracy and convergence rate remain relatively stable, although higher dropout rates result in lower final accuracy.

Additionally, without dropouts, CESAR exchanges 11.58 GB of data. With parameter-wise dropout handling, the system exchanges 11.19 GB and 10.34 GB of data for 10% and 20% dropouts, respectively. This shows that while the add-on requires additional communication (due to sharing indices selected during local sparsification with immediate neighbors), the overall data exchanged is still reduced, as crashed nodes do not share their masked models.

Finally, without the add-on and without dropouts, the experiment takes an average of 4 hours and 23 minutes to complete with CESAR. In contrast, using the add-on increases the runtime to 5 hours and 4 minutes for 10% dropouts and 5 hours and 18 minutes for 20% dropouts, representing an increase in execution time of 16% and 21%, respectively.

C. Summary of Findings

Our systematic evaluation confirms that CESAR successfully reconciles the trade-offs inherent to DL. Regarding **RQ1**, we demonstrated that CESAR matches D-PSGD accuracy across all tasks while reducing data exchange by up to 66 % compared to full-parameter schemes. For **RQ2** and **RQ3**, our results show that increasing the masking requirement s effectively neutralizes collusion risks with negligible impact on convergence, while providing stronger defense against MIA than standard aggregation. Finally, addressing **RQ4**, we showed that our parameter-wise dropout handling maintains high accuracy even under 20 % failure rates. Collectively, these results validate CESAR as a robust, general-purpose protocol for private and efficient DL.

V. Related Work

Privacy attacks. Gradient inversion [34], MIA [10], and attribute inference [35] attacks extract training-set information from exchanged models, including in D-PSGD [36]. Although sparsification has been proposed as a defense [37], Yue *et al.* [38] demonstrate gradient inversion on TopK-sparsified gradients. These attacks remain poorly understood under secure aggregation. Because attacks on averaged models affect CESAR and conventional secure aggregation equally, we consider them orthogonal to this work.

Secure aggregation. Secure aggregation conceals individual updates while revealing only their aggregate, and has become a standard privacy mechanism in FL. Most protocols build on Bonawitz *et al.* [15], using pairwise additive masks and Shamir secret sharing [39] for dropout recovery, typically coordinated by an untrusted server. Subsequent work reduces communication: BBGLR [40] limits each client's mask partners to a small subset, and Flamingo [41] amortizes mask agreement across rounds. These optimizations target FL's single large-scale aggregation per round and they do not transfer directly to DL, which performs many small aggregations among few peers. In comparison prior secure-aggregation work in DL is limited [42]. CESAR is the first protocol to integrate sparsification without additional servers, preserving the fully decentralized model.

Secure aggregation with sparsification. Model-size reduction via compression [43], sparsification [7], and quantization [44] is well studied, yet its integration with secure aggregation remains limited to FL. SparseSecAgg [16] embeds a fixed sparsification mechanism into the aggregation protocol, restricting compatibility with other methods. Lu *et al.* [17] incorporate TopK by masking the *union* of selected indices across clients. Both approaches rely on a central server and therefore cannot be trivially adapted to DL. In contrast, CESAR masks only the *intersection* of locally selected indices and operates without any server.

VI. Conclusion

CESAR is the first secure aggregation protocol for DL that natively supports sparsification and node dropouts without requiring any server. CESAR reconciles the tension between privacy, communication efficiency, and utility by masking only the intersection of locally selected indices, ensuring that pairwise masks cancel upon aggregation. Formal analysis establishes privacy guarantees against HbC and colluding adversaries. Experiments on models with up to 124M parameters show that CESAR matches the accuracy of state-of-the-art baselines while reducing data exchange by 66% compared to full-parameter sharing, and incurring only a 7–35% communication overhead compared to plain (unsecure) sparsification. Further optimization of CESAR for various network topologies and more complex models is a promising direction for future work. Integrating CESAR into real-world applications holds great potential to advance secure, efficient, and privacy-preserving DL systems significantly.

REFERENCES

[1] C. A. Ramezan, T. A. Warner, A. E. Maxwell, and B. S. Price, "Effects of training set size on supervised machine-learning land-cover classification of large-area high-resolution remotely sensed data," *Remote Sensing*, vol. 13, no. 3, p. 368, 2021.

[2] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, and S. Bakas, "Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation," 2018.

[3] A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar, "Fully decentralized federated learning," in *Third workshop on bayesian deep learning (NeurIPS)*, vol. 2, 2018.

[4] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," 2018.

[5] D. Jarrett, E. Stride, K. Vallis, and M. J. Gooding, "Applications and limitations of machine learning in radiation oncology," *The British journal of radiology*, vol. 92, no. 1100, p. 20190001, 2019.

[6] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *NIPS*, 2017.

[7] D. Alistarh, T. Hoefler, M. Johansson, S. Khirirat, N. Konstantinov, and C. Renggli, "The convergence of sparsified gradient methods," in *NeurIPS*, 2018.

[8] Z. Tang, S. Shi, and X. Chu, "Communication-efficient decentralized learning with sparsification and adaptive peer selection," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, 2020.

[9] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," 2020.

[10] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "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*, 2017, pp. 3–18.

[11] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer, "Membership inference attacks from first principles," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1897–1914.

[12] E. Cyffers, M. Even, A. Bellet, and L. Massoulié, "Muffliato: Peer-to-peer privacy amplification for decentralized optimization and averaging," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 15 889–15 902.

[13] S. Biswas, D. Frey, R. Gaudel, A.-M. Kermarrec, D. Lerévérend, R. Pires, R. Sharma, and F. Taïani, "Low-cost privacy-preserving decentralized learning," *Proceedings on Privacy Enhancing Technologies*, 2025.

[14] S. Biswas, M. Even, A.-M. Kermarrec, L. Massoulié, R. Pires, R. Sharma, and M. de Vos, "Noiseless privacy-preserving decentralized learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2025, no. 1, p. 824–844, Jan. 2025.

[15] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[16] I. Ergun, H. U. Sami, and B. Guler, "Sparsified secure aggregation for privacy-preserving federated learning," *arXiv preprint arXiv:2112.12872*, 2021.

[17] S. Lu, R. Li, W. Liu, C. Guan, and X. Yang, "Top-k sparsification with secure aggregation for privacy-preserving federated learning," *Computers & Security*, vol. 124, p. 102993, 2023.

[18] G. Ács and C. Castelluccia, "I have a dream!(differentially private smart metering)," in *International Workshop on Information Hiding*. Springer, 2011, pp. 118–132.

[19] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. Van Steen, "Gossip-based peer sampling," *ACM Transactions on Computer Systems (TOCS)*, vol. 25, no. 3, pp. 8–es, 2007.

[20] R. Guerraoui, A.-M. Kermarrec, A. Kucherenko, R. Pinot, and M. de Vos, "Peerswap: A peer-sampler with randomness guarantees," in *Proceedings of the 43rd International Symposium on Reliable Distributed Systems (SRDS 2024)*, 2024.

[21] T. D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," *Journal of the ACM (JACM)*, vol. 43, no. 2, pp. 225–267, 1996.

[22] S. Biswas, A.-M. Kermarrec, R. Pires, R. Sharma, and M. Vujasinovic, "Communication-efficient secure aggregation in decentralized learning," 2026. [Online]. Available: https://arxiv.org/abs/2405.07708

[23] A. Dhasade, A.-M. Kermarrec, R. Pires, R. Sharma, and M. Vujasinovic, "Decentralized learning made easy with decentralizepy," in *Proceedings of the 3rd Workshop on Machine Learning and Systems*, 2023, pp. 34–41.

[24] S. H. Bach, V. Sanh, Z.-X. Yong, A. Webson, C. Raffel, N. V. Nayak, A. Sharma, T. Kim, M. S. Bari, T. Fevry *et al.*, "Promptsource: An integrated development environment and repository for natural language prompts," *arXiv preprint arXiv:2202.01279*, 2022.

[25] K. Hsieh, A. Phanishayee, O. Mutlu, and P. B. Gibbons, "The non-IID data quagmire of decentralized machine learning," in *ICML*, 2020.

[26] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[27] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, aug 2009.

[28] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[29] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.

[30] W. Yu, C. Zhu, Z. Li, Z. Hu, Q. Wang, H. Ji, and M. Jiang, "A survey of knowledge-enhanced text generation," *ACM Comput. Surv.*, vol. 54, no. 11s, nov 2022.

[31] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[32] A. Dhasade, A.-M. Kermarrec, R. Pires, R. Sharma, M. Vujasinovic, and J. Wigger, "Get more for less in decentralized learning systems," in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS '23)*, 2023, pp. 463–474.

[33] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," 2018.

[34] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in neural information processing systems (NeurIPS '19)*, vol. 32, 2019.

[35] B. Z. H. Zhao, A. Agrawal, C. Coburn, H. J. Asghar, R. Bhaskar, M. A. Kaafar, D. Webb, and P. Dickinson, "On the (in) feasibility of attribute inference attacks on machine learning models," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2021, pp. 232–251.

[36] D. Pasquini, M. Raynal, and C. Troncoso, "On the (in) security of peer-to-peer decentralized machine learning," in *2023 IEEE Symposium on Security and Privacy (SP)*, 2023, pp. 418–436.

[37] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1310–1321.

[38] K. Yue, R. Jin, C.-W. Wong, D. Baron, and H. Dai, "Gradient obfuscation gives a false sense of security in federated learning," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 6381–6398.

[39] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[40] J. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly)logarithmic overhead," Cryptology ePrint Archive, Paper 2020/704, 2020.

[41] Y. Ma, J. Woods, S. Angel, A. Polychroniadou, and T. Rabin, "Flamingo: Multi-round single-server secure aggregation with applications to private federated learning," Cryptology ePrint Archive, Paper 2023/486, 2023.

[42] N. Gupta, J. Katz, and N. Chopra, "Privacy in distributed average consensus," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 9515–9520, 2017, 20th IFAC World Congress.

[43] Y. Collet, "LZ4," 2022.

[44] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and application to data-parallel distributed training of speech DNNs," in *Interspeech 2014*, September 2014.

APPENDIX A
SCALING WITH NETWORK SIZE

In this section, we examine how the communication cost of CESAR scales with an increasing number of nodes in the network. Measurements were conducted for networks comprising 48, 96, 192, and 288 nodes. Each experiment was performed within a 5-regular network using the CIFAR-10 dataset. The dataset was divided into 288 equally-sized segments, corresponding to the maximum node count, with each node receiving one segment. In cases where the number of segments surpassed the number of nodes, the excess segments remained unassigned and were not utilized in the training process.

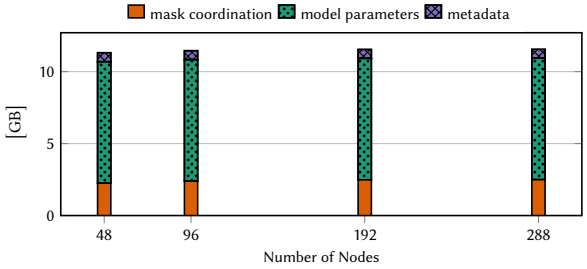

Fig. 9. Amount of data transferred per node during training for different number of nodes

Number of nodes	Parameter data [GB]	CESAR overhead [GB]
48	9.07	2.25
96	9.07	2.39
192	9.06	2.48
288	9.05	2.49

TABLE IV: Amount of data transferred per node during training using CESAR for a fixed setup in networks of different sizes

Fig. 9 illustrates that the communication overhead of CESAR remains practically independent of the number of nodes in the network. However, a more detailed examination in Tab. IV reveals a marginally higher CESAR overhead for larger networks. This phenomenon can be attributed to the experimental setup. As discussed in Sec. III-D3, the number of messages sent during the prestep phase of CESAR is proportional to the number of second degree neighbors. With fewer nodes in the network, while maintaining a fixed topology, it is more likely for two neighbors to share a common neighbor. This overlap effectively reduces the average second degree neighborhood size, thereby lowering the overhead in the prestep.

APPENDIX B
SCALING WITH NODE DEGREE

In this section, we examine how the communication overhead of CESAR scales in practice. As observed in Sec. IV-B1 and Sec. IV-B2, the overhead with random subsampling remains constant regardless of the network topology. In contrast, it increases when TOPK is applied.

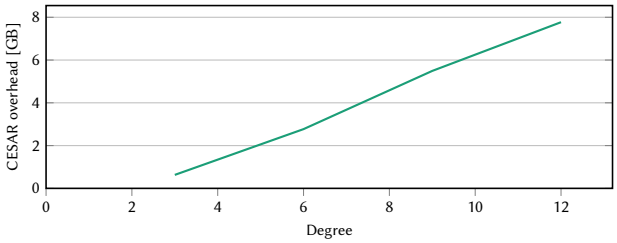

Fig. 10. Amount of data transferred per node in mask coordination of CESAR during training for different graph degrees.

Sec. III-D3 derives that the communication overhead of the mask coordination in CESAR is $(\alpha d \delta_{\max}^2)$, as a node sends their indices to each second-degree neighbor, and their number typically scales with the square of the degree in the network. To verify this, we ran CESAR with TOPK on a 96-node δ-regular network with δ values of 3, 6, 9, and 12. In each run, 40% of parameters were selected for sparsification, and we recorded only the overhead caused by the mask coordination phase.

The results for the given degrees are shown in Fig. 10. They indicate that the overhead scales linearly, contrary to the quadratic scaling we expected. However, Tab. V also reveals that the overhead of the mask coordination actually scales with the size of the second-degree neighborhood. Because the network size is limited, it is common for two nodes to share a neighbor, hence reducing the size of the second-degree neighborhood compared to a theoretical model where the network size is infinite and nodes sharing a neighbor are uncommon. This also implies that the total overhead of the mask coordination is limited by the size of the network to $(\alpha d|\mathcal{N}|)$. This overlap of neighbors highlights the importance of the ability of CESAR to reuse the same mask that two nodes agreed upon over many aggregation with multiple neighbors these two nodes may have in common.

Degree	Overhead [GB]	Avg. 2^{nd} deg. neighborhood size	Overhead per 2^{nd} deg. neighbor [GB]
3	0.63	5.93	0.1064
6	2.77	26.87	0.1031
9	5.49	53.48	0.1027
12	7.77	75.16	0.1034

TABLE V: Amount of data transferred per node in mask coordination of CESAR during training for different graph degrees and their relation to the size of second-degree neighborhood.