mathematics

MDPI

Article

FedDecouple: Mitigating Noise Accumulation in Differentially Private Federated Learning via Phase Decoupling

Tenghang Ge and Xiaochao Wei *

School of Computer Science and Artificial Intelligence, Shandong Normal University, Jinan 250358, China; 2024317215@stu.sdnu.edu.cn

* Correspondence: wxc@sdnu.edu.cn

Abstract

Differential privacy protection in federated learning faces the fundamental challenge of noise accumulation: noise added dispersedly by clients accumulates in variance during server-side aggregation, severely harming model convergence and accuracy. This paper proposes FedDecouple, a phase-decoupled differentially private federated learning framework that is analytically suited for resource-constrained mobile devices. The core innovation lies in decoupling the noise addition phase from the client computation phase—clients only upload clean gradients, while two auxiliary servers collaboratively generate and inject noise through a secure two-party MPC protocol. This design reduces the effective noise variance while eliminating the per-sample gradient computation burden on clients. Experimental results show that on MNIST, FedDecouple maintains 97.75% accuracy under strict privacy, significantly outperforming client-side noised DP-SGD with 94.0% accuracy. On CIFAR-10, it achieves 76.2% test accuracy, which is 13.4 percentage points higher than DP-SGD. FedDecouple's total training time on both datasets is faster than Opacus and DP-SGD.

Keywords: federated learning; differential privacy; phase decoupling; noise accumulation; secure multi-party computation

MSC: 94A60

1. Introduction

Federated learning (FL) [1], as a distributed machine learning paradigm, enables multiple clients to collaboratively train models without sharing raw data, making it a key technology in privacy-sensitive domains such as mobile devices, medical data, and financial risk control. Taking mobile devices as an example, applications like Google's Gboard input method and Apple's Siri voice assistant have successfully implemented federated learning frameworks, enhancing model performance while protecting user privacy. However, recent research indicates that shared model updates may leak sensitive information—attackers can reconstruct training data from shared gradients through techniques such as gradient inversion attacks [2], member inference attacks [3], and model inversion attacks [4], severely limiting the practical deployment of federated learning in high privacy-sensitive scenarios.

To address this threat, differential privacy (DP) [5] has become the gold standard for providing quantifiable privacy guarantees in federated learning. In federated learning environments, the predominant privacy protection approach is client-side differential privacy [6]: after local training, each client applies gradient clipping and Gaussian noise

Check for updates

Academic Editors: Jonathan Blackledge and Yiu Yin Raymond Lee

Received: 9 June 2026
Revised: 10 August 2026
Accepted: 14 August 2026
Published: 27 August 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

to its model updates before uploading the noisy updates to the server for aggregation. This paradigm can be formally regarded as a natural extension of DP-SGD [7] to federated settings, and the industry has developed mature tools such as Opacus [8] to support such training.

1.1. Problem Statement and Research Question

However, a thorough analysis reveals a fundamental structural flaw in this paradigm—the noise accumulation effect. Let each round involve m clients, with each client contributing Gaussian noise of variance σ_{loc}^2. During server-side aggregation, this noise accumulates via the sum-of-squares principle, resulting in a total noise variance of $m\sigma_{\text{loc}}^2$. This severely compromises convergence speed, significantly reduces model accuracy, and leads to inefficient utilization of the privacy budget. The issue becomes particularly pronounced with a large number of clients—where mobile device federated learning typically involves thousands or even tens of thousands of participants—the noise accumulation effect becomes a critical performance bottleneck.

This observation motivates our central research question:

"Given the limited resources of mobile devices, can we fundamentally restructure the noise injection process in DP-FL to eliminate noise accumulation at its source, rather than merely mitigating its effects?"

To answer this question, we formulate the following research objectives:

1. **Feasibility Objective:** To determine whether it is possible to decouple the noise addition phase from the client computation phase without compromising the DP guarantee.
2. **Privacy Objective:** To design a protocol that achieves equivalent (ϵ, δ)-DP protection without relying on client-side noise injection.
3. **Utility Objective:** To demonstrate that the proposed phase-decoupled design achieves superior convergence speed and final accuracy compared to existing client-side DP approaches under the same privacy budget.
4. **Efficiency Objective:** To reduce the client-side computational burden, making the approach suitable for resource-constrained mobile devices.

1.2. State-of-the-Art Gap Analysis

Recent advancements have explored various directions to improve DP-FL. Xue et al. [9] proposed adaptive noise mechanisms that dynamically adjust the noise scale based on historical gradient information. Wang et al. [10] developed FedFR-ADP, which uses Earth Mover's Distance to quantify data heterogeneity and adaptively tunes noise intensity and privacy budget. Yuan et al. [11] introduced DP-FedPUAC with update-wise adaptive clipping and adaptive local iteration. Li et al. [12] designed SFLES with hierarchical sparsification and direction-similarity-aware aggregation.

Despite these advances, a fundamental gap remains: *all existing approaches still follow the client-side local noise injection paradigm*. They attempt to *mitigate* the effects of noise accumulation through adaptive techniques, but they do not *eliminate* its root cause—the tight coupling between noise injection and client-side computation. This structural limitation persists across all current DP-FL methods, and we are not aware of any prior work that fundamentally restructures the noise injection process by decoupling it from client-side computation.

This gap is particularly critical for mobile device deployments, where client-side per-sample gradient computation and noise injection impose significant computational and energy overhead. A phase-decoupled approach that shifts the privacy burden from clients

to the server side could simultaneously address both the noise accumulation problem and the client-side efficiency challenge.

1.3. Proposed Approach: Phase Decoupling

This paper proposes a novel design philosophy—*phase decoupling*—that fundamentally restructures the noise injection process. The key insight is that noise need not be added at the client side. Instead, we propose that clients upload *clean gradients*, and noise is injected *once* at the server side after aggregation, through a secure two-party MPC protocol.

This design addresses our research objectives as follows:

- **Feasibility (Objective 1):** We demonstrate that phase decoupling is feasible by designing a complete protocol where clients upload clean gradients, and two auxiliary servers collaboratively generate and inject a single calibrated Gaussian noise through a secure two-party MPC protocol.
- **Privacy (Objective 2):** We provide rigorous RDP-based privacy analysis showing that our protocol achieves (ϵ, δ)-DP equivalent to client-side DP.
- **Utility (Objective 3):** By injecting noise only once, our approach reduces the effective noise variance from $m \cdot \sigma^2_{\text{loc}}$ to σ^2_{global}, enabling faster convergence and higher accuracy.
- **Efficiency (Objective 4):** Clients only compute batch gradients (not per-sample gradients) and do not perform local noise addition, significantly reducing the computational burden on mobile devices.

1.4. Contributions

The main contributions of this paper are as follows:

1. **Phase Decoupling Design Principle:** We propose phase decoupling as a new design principle for DP-FL, breaking the tight coupling between noise and computation. This principle opens up new design possibilities for DP-FL and makes it particularly suitable for deployment on mobile devices with limited computing resources (Section 4.2).
2. **Secure Two-Party Noise Generation Protocol:** We design a complete six-round interactive protocol that enables two auxiliary servers to collaboratively generate Gaussian noise satisfying DP requirements without a single trusted server. This protocol uses commitment schemes to prevent malicious behavior (Section 4.2.2).
3. **Comprehensive Privacy Analysis:** We provide rigorous RDP-based privacy analysis with formal sensitivity bounds, subsampling amplification, and complete RDP composition across rounds. We quantitatively compare our privacy guarantees with DP-FedAvg (Section 4.3).
4. **Experimental Validation:** Extensive experiments on MNIST and CIFAR-10 across three privacy levels ($\epsilon = 8, 4, 1$) demonstrate that FedDecouple consistently outperforms DP-SGD, Opacus, and DP-FTRL in convergence speed and final accuracy. On CIFAR-10, FedDecouple achieves 76.2% accuracy, surpassing DP-SGD by 13.4 percentage points (Section 5).

1.5. Paper Roadmap

The remainder of this paper is organized as follows:

- **Section 2:** Reviews related work on DP-FL, including foundational methods and recent advances in adaptive noise, shuffling, and MPC-based approaches.
- **Section 3:** Provides preliminary knowledge on differential privacy, DP-FedAvg, privacy accounting, and secret sharing.
- **Section 4:** Describes the FedDecouple protocol, including system models, threat assumptions, the two-phase design, security analysis, and complexity analysis.

- **Section 5:** Presents implementation details, including model architectures, dataset preprocessing, hyperparameter configurations, and experimental setup.
- **Section 6:** Reports experimental results and analysis, comparing FedDecouple with baseline methods across multiple privacy levels and datasets.
- **Section 7:** Concludes this paper and discusses future research directions.

2. Related Work

A central challenge in differentially private federated learning (DP-FL) is the trade-off between privacy protection and model utility. Existing approaches to this challenge can be categorized along a spectrum according to how they handle the relationship between noise injection and client-side computation:

- **Client-side noise injection (traditional paradigm):** Methods such as DP-FedAvg [6] and DP-SGD [7] add noise independently at each client before aggregation. These methods suffer from noise accumulation during server-side aggregation, which is the fundamental problem we address in this work.
- **Optimized client-side noise:** Recent works have attempted to mitigate the effects of client-side noise through adaptive mechanisms [9–11,13,14], shuffling [15–17], or personalized privacy budgets [18,19]. While these approaches improve utility, they still operate within the client-side noise paradigm and do not eliminate its root cause.
- **Server-side noise with trusted servers:** Some works achieve server-side noise injection by relying on a single trusted server to add noise after secure aggregation. However, this reintroduces a single point of trust and requires heavy client-side cryptographic computation.
- **FedDecouple (this work):** We propose a fundamentally different approach—*phase decoupling*—where noise is injected once on the server side through a secure two-party MPC protocol, without requiring a trusted server. This positions our work at the extreme end of the spectrum: *fully decoupling noise from client-side computation*.

This categorization clarifies the unique positioning of our contribution: while existing methods either accept the consequences of client-side noise or attempt to mitigate them through optimization, we fundamentally restructure the noise injection process to eliminate noise accumulation at its source. The following subsections review representative works across three dimensions: noise mechanism optimization, privacy budget management, and model update optimization, with particular emphasis on how each relates to the noise-computation coupling problem.

Differentially Private Federated Learning (DPFL) aims to balance privacy protection and model utility. This section reviews representative works across three dimensions: noise mechanism optimization, privacy budget management, and model update optimization.

2.1. Foundational Works

Geyer et al. first demonstrated the feasibility of client-side differential privacy in federated learning, showing that adding noise to each client's updates can provide formal privacy guarantees. McMahan et al. formally proposed DP-FedAvg [6], which clips and adds Gaussian noise to client updates before upload, laying the foundation for client-level DP. This paradigm has become the de facto standard for DPFL. Abadi et al. [7] introduced the moments accountant, a critical advancement that enables precise privacy budget tracking in iterative training by tracking the moment generating function of privacy loss, significantly reducing the required noise compared to traditional composition theorems. Kairouz et al. [8] proposed DP-FTRL, which injects correlated noise via tree aggregation, achieving differential privacy without relying on sampling amplification. Unlike DP-SGD, DP-FTRL allows arbitrary data access order, making it particularly suitable for federated

settings where client participation is unpredictable. Pichapati et al. [20] developed AdaClip, an adaptive gradient clipping mechanism that dynamically adjusts the clipping threshold based on gradient norm distributions, reducing the amount of excess noise injected. Thakkar et al. further extended adaptive clipping to federated scenarios, demonstrating its effectiveness in heterogeneous data settings. Industrial tools like Opacus [21] provide efficient per-sample gradient computation for DP-SGD, making DP training more accessible to practitioners. An amplitude-varying perturbation strategy [22] was also introduced, which uses smaller noise early in training for faster convergence and larger noise later for stronger privacy protection.

2.2. Adaptive Noise and Dynamic Privacy Allocation

Xue et al. [9] proposed an adaptive noise mechanism that dynamically adjusts the noise scale based on model parameter magnitudes and historical gradient information, improving model utility while maintaining privacy guarantees. Wang et al. [10] developed FedFR-ADP, a framework that employs Earth Mover's Distance to quantify client-side data heterogeneity, thereby adaptively adjusting Gaussian noise intensity. It also incorporates a feedback adjustment mechanism to dynamically optimize the privacy budget according to global model errors, achieving at least 3.05% and 1.76% accuracy improvements on two image datasets. Yuan et al. [22] further refined amplitude-varying perturbation, deriving an upper bound for the optimal number of global aggregation rounds. Wang et al. [23] introduced FedAPCA, which integrates dynamic privacy budget allocation, adaptive segmented perturbation mechanisms, and hierarchical clustering aggregation methods. The framework enables clients to independently set privacy budgets based on their data characteristics, achieving 1–2% accuracy improvements on MNIST and CIFAR-10. Yuan et al. [11] proposed DP-FedPUAC, featuring update-wise adaptive clipping thresholds based on real-time gradient norm responses and an adaptive local iteration strategy that automatically adjusts training rounds based on model performance. Extensive experiments on MNIST, FMNIST, CIFAR-10, and CIFAR-100 demonstrate superior privacy-utility trade-offs with significantly reduced communication rounds. Wang et al. [13] presented APDP-FL, a personalized framework with adaptive noise addition that scores training processes and dynamically adjusts noise levels—adding larger noise early and gradually reducing it to accelerate convergence while respecting personalized privacy preferences. Li et al. [12] designed SFLES, which employs hierarchical sparsification, bounded noise mechanisms based on symmetric segmented distribution, and direction-similarity-aware aggregation to accelerate convergence under DP constraints. Xu et al. [14] introduced GDPFed and GDPFed+ to address heterogeneous privacy requirements, grouping clients by privacy budget and optimizing sampling ratios to minimize convergence error.

2.3. Shuffling, Privacy Amplification, and Lightweight DP

Xu et al. [15] proposed Camel, a framework that, for the first time, supports malicious security and integrity checks within the shuffling model. Leveraging secret-sharing shuffling technology, Camel enhances server-side computation security through system-level communication efficiency optimization and lightweight integrity verification. Through Rényi differential privacy analysis across the entire FL process, the authors derived tighter upper bounds for privacy loss. Wang et al. [16] investigated individual computation beyond statistical estimation in the shuffling model, proposing a novel paradigm that incorporates statistical random identities into DP, preserving security functions while maintaining privacy amplification effects after message shuffling. This technique is suitable for non-statistical computing scenarios such as spatial crowdsourcing and combinatorial optimization. Hong et al. [17] developed LightDP-FL, a lightweight DPFL scheme for un-

trusted peers and servers. By injecting both individual and pairwise noise simultaneously, it achieves privacy protection with minimal overhead. The scheme leverages upper bounds on the numbers of stragglers and colluders to prove sufficient noise variance conditions under worst-case scenarios.

2.4. MPC and Secret Sharing for DP-FL

Wei et al. [24] designed DDP-SA, a scalable privacy-preserving framework combining client-side local DP with full-threshold additive secret sharing. Its two-stage protection ensures that no single compromised server can obtain client updates, and the parameter server only reconstructs aggregated noisy gradients. Hua et al. [25] proposed Clover, integrating gradient sparsification, data encoding, lightweight cryptography, and DP with a three-server distributed trust model. Its secure sparse vector aggregation mechanism achieves orders of magnitude reduction in communication and runtime compared to ORAM-based baselines. Madathil et al. [26] introduced TACITA, a single-server secure aggregation protocol satisfying four key properties: single client communication, input reliability, constant-size communication, and dropout robustness, relying on compact multi-key linearly homomorphic threshold signatures. Tjuawinata et al. [27] proposed a zero-failure bounded discrete Laplacian perturbation mechanism with MPC implementation, overcoming failure probabilities in existing bounded discrete Laplacian variants. Xie et al. [28] proposed CoSIFL, integrating active alerting, local DP, and Stackelberg game incentives to defend against Byzantine attacks while rewarding honest clients. Ebrahimi Atani et al. [29] proposed FedSelect-ME, a hierarchical multi-party framework with intelligent client selection based on utility, energy efficiency, and data sensitivity, combined with homomorphic encryption and DP.

2.5. Low-Rank Adaptation, Verifiable Aggregation, and System Optimization

Zhu et al. [30] proposed DEeR, addressing aggregation bias and noise amplification when combining Low-Rank Adaptation (LoRA) with DP in federated learning. The framework includes a deviation eliminator ensuring zero aggregation bias and a noise regulator that decouples DP from LoRA using two adjustment factors, suppressing noise amplification effects. Shen et al. [18] introduced PLDP-FL, supporting personalized local DP by allowing each client to autonomously choose its privacy budget level based on data sensitivity, with a privacy-budget-aware aggregation mechanism. Shen et al. [19] further developed an adaptive local DP scheme with dynamic privacy budget allocation based on data distribution characteristics and training progress, plus a verifiable aggregation mechanism to prevent server tampering. Shen et al. [31] proposed a verifiable privacy-preserving scheme under multiple encrypted keys using proxy re-encryption and zero-knowledge proofs. Peng et al. [32] proposed a communication-efficient verifiable aggregation scheme combining gradient quantization, additive secret sharing, and homomorphic message authentication codes. Chen et al. [33] proposed M2FDP for hierarchical networks with layer-wise noise adaptation based on trust models of different subnets. Shan et al. [34], Amanullah et al., and Rahdari et al. provided comprehensive reviews of DPFL optimization, applications in deep learning, and comparisons of privacy protection approaches including MPC, DP, trusted execution environments, and federated learning. A recent MDPI study proposed adaptive DP-FL for edge devices with gradient sparsification and quantization to reduce communication overhead while maintaining privacy protection.

3. Preliminary Knowledge

This section provides the foundational concepts necessary for understanding the proposed FedDecouple framework and its analysis. The content is organized to serve three specific purposes that directly support the subsequent sections of this paper:

1. **Differential Privacy Foundations (Sections 3.1–3.3):** We introduce the formal definition of differential privacy, the Gaussian mechanism, and the DP-FedAvg algorithm. These concepts establish the baseline privacy model and provide the reference point against which we compare our approach. The privacy analysis of FedDecouple in Section 4.3 directly builds upon the Rényi Differential Privacy (RDP) framework introduced in Section 3.3 and the moments accountant formulation used for cumulative privacy tracking.
2. **Secret Sharing and Cryptographic Primitives (Section 3.4):** We cover additive secret sharing as the fundamental cryptographic primitive underlying our secure two-party MPC protocol. This material directly supports the protocol design in Section 4.2 and the security analysis in Section 4.4.
3. **Symbols and Conventions (Section 3.5):** We define the notation used consistently throughout this paper, including the symbol table that facilitates readability across all sections.

In essence, this section provides the theoretical building blocks—differential privacy definitions, the DP-FedAvg baseline, privacy accounting methods, cryptographic primitives, and notation—that are systematically combined in Section 4 to construct the FedDecouple protocol and analyze its privacy guarantees. Experimental comparisons in Sections 5 and 6 rely on the DP-FedAvg formulation established here as the primary baseline.

The following subsections review the federated learning workflow, the definition of differential privacy, the moments accountant and Rényi differential privacy, outline the federated learning process for differential privacy, and explain the relevant symbols and conventions used in this paper.

In this section, we briefly review the workflow of federated learning and the definition of differential privacy, introduce the moments accountant and Rényi differential privacy, outline the federated learning process for differential privacy, and explain the relevant symbols and conventions used in this paper.

3.1. Differential Privacy

Differential Privacy (DP) provides quantifiable privacy protection for algorithms. Its core principle is that the output distributions of an algorithm across two adjacent datasets must be sufficiently similar, preventing attackers from inferring the presence of a specific sample based on the output observations.

Definition 1 (Adjacent datasets)**.** *Two datasets D and D' are considered adjacent if they differ by at most one record (i.e., $|D \triangle D'| = 1$).*

Definition 2 ((ϵ, δ)-Differential Privacy)**.** *A randomized algorithm $\mathcal{M} : \mathcal{D}^n \to \mathcal{R}$ satisfies (ϵ, δ)-differential privacy if, for any adjacent datasets D, D' and any output subset $S \subseteq \mathcal{R}$, the following holds:*

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

where the probability is taken over the randomness of the algorithm.

The Gaussian mechanism [35] is a commonly used tool for achieving differential privacy. For a query function $q : \mathcal{D}^n \to \mathbb{R}^p$, its ℓ_2 sensitivity is defined as:

$$\Delta_2 = \max_{D,D'} \|q(D) - q(D')\|_2$$

The Gaussian mechanism returns $q(D) + \mathcal{N}(0, \Delta_2^2 \sigma^2 I)$, where σ is determined by the target privacy parameter. It has been proven that this mechanism satisfies (ϵ, δ)-differential privacy when $\sigma \geq \Delta_2 \sqrt{2 \log(1.25/\delta)}/\epsilon$.

Differential privacy possesses three important properties:

1. **Sequential Composition:** The combination of multiple DP algorithms still satisfies DP, with privacy parameters accumulating accordingly.
2. **Parallel Composition [36]:** A combination of DP algorithms applied to non-overlapping datasets satisfies DP with the privacy parameters set to their maximum values.
3. **Post-processing Invariance:** Any subsequent processing applied to the output of a DP algorithm does not compromise privacy protection.

3.2. DP-FedAvg Algorithm

Privacy Model: We adopt the standard **client-level differential privacy** model for federated learning [6]. In this model, two datasets $\mathcal{D}$ and $\mathcal{D}'$ are considered **adjacent** if they differ by the data of exactly one client (i.e., one client's entire local dataset is added, removed, or replaced). This is the privacy model used throughout this paper. The terms "client-level" and "user-level" are used interchangeably to refer to this same adjacency definition.

The classic DP-FedAvg algorithm can be described as follows: In the t-th communication round, the server selects a subset of clients $\mathcal{S}_t$ and distributes the current global model θ^t to them. Each selected client k performs E rounds of SGD on its local data to obtain a model update Δ_k^t. Subsequently, the clients undergo a two-step privacy processing:

1. **Gradient clipping:** Clip the update to ensure the ℓ_2 norm does not exceed a certain value C:
$$\tilde{\Delta}_k^t = \Delta_k^t \cdot \min\left(1, \frac{C}{\|\Delta_k^t\|_2}\right)$$
2. **Local noise addition:** Add Gaussian noise:
$$\hat{\Delta}_k^t = \tilde{\Delta}_k^t + \mathcal{N}(0, \sigma_{\text{loc}}^2 C^2 I)$$

The client uploads the noisy update $\hat{\Delta}_k^t$ to the server, and the server aggregates it to obtain the global update:

$$\theta^{t+1} = \theta^t + \frac{1}{|\mathcal{S}_t|} \sum_{k \in \mathcal{S}_t} \hat{\Delta}_k^t$$

This privacy analysis paradigm typically relies on the moments accountant or RDP. It has been demonstrated that, with an appropriately configured noise scale σ_{loc}, the entire training process satisfies (ϵ, δ)-client-level DP.

However, this paradigm suffers from a fundamental issue—noise accumulation under the client-level DP model. Since each client's update contains independent noise, the effective noise variance in the aggregated global update is given by:

$$\text{Var}[\text{noise}_{\text{global}}] = \frac{\sigma_{\text{loc}}^2 C^2}{|\mathcal{S}_t|}$$

Averaging over $|\mathcal{S}_t|$ clients reduces the variance of the *per-round* noise. However, this does not eliminate noise accumulation in the following sense: to achieve a target (ϵ, δ)-DP guarantee over T rounds, the required per-client noise scale σ_{loc} must grow with $\sqrt{T \cdot |\mathcal{S}_t|}$ (due to RDP composition). Substituting this required σ_{loc} into the effective variance shows that the resulting effective noise is proportional to $T \cdot |\mathcal{S}_t| \cdot C^2$, which grows linearly with $|\mathcal{S}_t|$. This constitutes the noise accumulation problem: more clients do not reduce the effective noise after accounting for the privacy cost of composition.

More critically, when employing the moments accountant to analyze multi-round training, the noise scale corresponding to the total privacy loss is inversely proportional to σ_{loc}. This necessitates the use of a larger σ_{loc} to achieve the same (ϵ, δ) privacy guarantee, thereby further amplifying the noise. The dual accumulation of this noise across spatial and temporal dimensions constitutes a bottleneck that limits the performance of existing DP-FL methods.

3.2.1. Noise Accumulation vs. Noise Averaging: A Clarification

A central claim of this paper is that client-side differential privacy suffers from *noise accumulation*. However, the reviewer correctly observes that aggregation via averaging reduces variance by a factor of $|\mathcal{S}_t|$, as shown in Equation (1):

$$\text{Var}[\text{noise}_{\text{global}}] = \frac{\sigma_{\text{loc}}^2 C^2}{|\mathcal{S}_t|}$$

This seems to contradict the claim that "noise accumulates" at the server. We clarify this important distinction below.

The Core Issue Is Not Per-Round Variance, but Required Noise Scale

The critical observation is that the variance in Equation (1) assumes a fixed σ_{loc}. However, in client-side DP, the per-client noise scale σ_{loc} itself must be set based on the desired total privacy budget over T rounds. Through the RDP composition, the required σ_{loc} scales with $\sqrt{T \cdot |\mathcal{S}_t|}$ to meet a target (ϵ, δ), while in FedDecouple the required σ_{global} is independent of $|\mathcal{S}_t|$ and only scales with $\sqrt{T}$.

Why DP-FedAvg's Required Noise Scale Grows with $|\mathcal{S}_t|$

The reason σ_{loc} must be larger for larger $|\mathcal{S}_t|$ lies in the sensitivity analysis and privacy composition:

1. **Sensitivity of the Aggregated Update:** In DP-FedAvg, each client's update is clipped to ℓ_2-norm C. The *aggregated* update $\tilde{\Delta}^t = \sum_{k \in \mathcal{S}_t} \tilde{\Delta}_k^t$ has sensitivity $2C$ (not $|\mathcal{S}_t|C$). However, the per-client noise $\mathcal{N}(0, \sigma_{\text{loc}}^2 C^2 I)$ is added independently at each client. After aggregation, the effective noise is $\frac{1}{|\mathcal{S}_t|} \sum_k \mathcal{N}(0, \sigma_{\text{loc}}^2 C^2 I) \sim \mathcal{N}(0, \frac{\sigma_{\text{loc}}^2 C^2}{|\mathcal{S}_t|} I)$.
2. **Privacy Composition Over Rounds:** The privacy cost is counted per communication round. Since each round consumes privacy budget, the total RDP over T rounds is:

$$\rho_{\text{total}}(\alpha) = T \cdot \frac{\alpha}{2\sigma_{\text{loc}}^2} \cdot q^2$$

   where $q = |\mathcal{S}_t|/K$ is the sampling rate. To achieve a target ϵ, the required σ_{loc} is proportional to $\sqrt{T} \cdot q$, i.e., $\sigma_{\text{loc}} \propto \sqrt{T} \cdot \frac{|\mathcal{S}_t|}{K}$.
3. **Substituting Back:** Plugging this required σ_{loc} into the effective variance:

$$\text{Var}_{\text{DP-FedAvg}}^{\text{effective}} = \frac{\sigma_{\text{loc}}^2 C^2}{|\mathcal{S}_t|} \propto \frac{T \cdot |\mathcal{S}_t| \cdot C^2}{K^2}$$

This grows linearly with $|\mathcal{S}_t|$.

Why FedDecouple's Noise Scale Is Independent of $|\mathcal{S}_t|$

In FedDecouple, noise is injected only once at the server after aggregation. The sensitivity remains C (Theorem 1). The RDP per round is:

$$\rho_{\text{round}}(\alpha) = \frac{\alpha}{2\sigma^2_{\text{global}}} \cdot q^2$$

which is identical in form. However, the effective variance after aggregation is simply

$$\text{Var}_{\text{FedDecouple}} = \sigma^2_{\text{global}} C^2$$

With $\sigma_{\text{global}} \propto \sqrt{T} \cdot q = \sqrt{T} \cdot |\mathcal{S}_t|/K$, the effective variance becomes:

$$\text{Var}_{\text{FedDecouple}} \propto \frac{T \cdot |\mathcal{S}_t|^2 \cdot C^2}{K^2}$$

which grows quadratically with $|\mathcal{S}_t|$ if σ_{global} were to scale with $|\mathcal{S}_t|$. However, in FedDecouple, the global noise is injected once, not per client, so the privacy composition is counted once per round regardless of how many clients participate. Therefore, the required σ_{global} scales with $\sqrt{T}$, independent of $|\mathcal{S}_t|$:

$$\sigma_{\text{global}} \propto \sqrt{T}$$

The effective variance is therefore

$$\text{Var}_{\text{FedDecouple}} \propto T \cdot C^2$$

which is independent of $|\mathcal{S}_t|$.

Key Takeaway

The comparison in Table 1 demonstrates that, despite the per-round noise averaging effect, the total privacy budget composition forces σ_{loc} to grow with $|\mathcal{S}_t|$. As shown in Figure 1, DP-FedAvg requires $\sigma_{\text{loc}} \propto \sqrt{|\mathcal{S}_t|}$, while FedDecouple requires σ_{global} independent of $|\mathcal{S}_t|$. This results in higher effective noise variance in DP-FedAvg compared to FedDecouple for the same privacy guarantee. This is the fundamental advantage of our phase-decoupled design: by injecting noise once at the server, we avoid the multiplicative effect of client-side noise injection on the required noise scale.

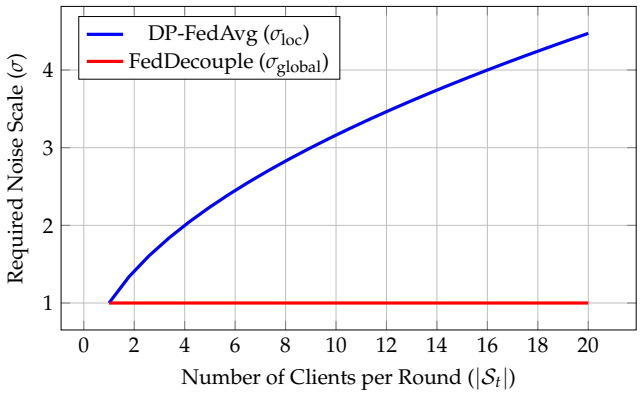

Figure 1. Required noise scale vs. number of clients per round.

Table 1. Effective noise variance comparison (after aggregation).

Criterion	DP-FedAvg (Client-Side)	FedDecouple (Server-Side)
Per-Round Noise Variance (Conditional on σ)	$\frac{\sigma_{\text{loc}}^2 C^2}{\lvert\mathcal{S}_t\rvert}$	$\sigma_{\text{global}}^2 C^2$
Required Noise Scale to Achieve Target (ϵ, δ)	$\sigma_{\text{loc}} = \Omega\left(\frac{\sqrt{T \cdot \lvert\mathcal{S}_t\rvert}}{\epsilon}\right)$	$\sigma_{\text{global}} = \Theta\left(\frac{\sqrt{T}}{\epsilon}\right)$
Effective Variance after T Rounds	$\frac{\sigma_{\text{loc}}^2 C^2}{\lvert\mathcal{S}_t\rvert} = \Omega\left(\frac{TC^2}{\epsilon^2}\right)$	$\sigma_{\text{global}}^2 C^2 = \Theta\left(\frac{TC^2}{\epsilon^2}\right)$
Dependence on $\lvert\mathcal{S}_t\rvert$	Strong (linear)	None
Dependence on T	Linear	Linear

Privacy Guarantee Equivalence: Both DP-FedAvg and FedDecouple satisfy (ϵ, δ)-user-level DP for the same ϵ and δ. The difference lies in the *utility* achieved at that privacy level. The experimental results in Section 5 demonstrate that FedDecouple achieves higher accuracy and faster convergence than DP-FedAvg for the same privacy budget.

3.3. Privacy Accounting: Moments Accountant and Rényi Differential Privacy

In iterative algorithms (such as DP-SGD and federated learning), the privacy loss accumulates over training rounds. Traditional composition theorems often provide overly loose upper bounds for privacy, leading to excessive noise introduction and compromised model performance. To address this, researchers have proposed more precise privacy accounting methods.

The moments accountant [7] is the first method to achieve robust privacy accounting in deep learning. Its core approach involves tracking the moment generating function of privacy loss random variables, rather than focusing solely on the worst-case scenario. For Gaussian mechanisms, the moments accountant demonstrates that after T iterations, the total privacy loss ϵ scales proportionally to $\sqrt{T}$, rather than multiplicatively T times as naively expected, thereby significantly reducing the required noise level.

Rényi Differential Privacy (RDP) [37] represents a generalized and theoretical framework developed from the moments accountant. It employs the Rényi divergence to measure privacy loss, offering a more concise and versatile analytical approach.

Definition 3 (Rényi Differential Privacy)**.** *A randomized mechanism $\mathcal{M}$ satisfies $(\alpha, \rho(\alpha))$-RDP if, for any two adjacent datasets D, D':*

$$D_\alpha(\mathcal{M}(D) \| \mathcal{M}(D')) \leq \rho(\alpha)$$

where $D_\alpha(P\|Q) = \frac{1}{\alpha-1} \log \mathbb{E}_{x\sim Q}\left(\frac{P(x)}{Q(x)}\right)^\alpha$ is the Rényi divergence of order α.

RDP possesses elegant compositional properties: if individual mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_T$ satisfy $(\alpha, \rho_1(\alpha)), \ldots, (\alpha, \rho_T(\alpha))$-RDP respectively, their combination also satisfies $(\alpha, \sum_{t=1}^T \rho_t(\alpha))$-RDP.

RDP can be converted into the standard (ϵ, δ)-DP: if a mechanism satisfies $(\alpha, \rho(\alpha))$-RDP, then for any $\delta > 0$, it satisfies (ϵ, δ)-DP, where:

$$\epsilon = \min_{\alpha>1}\left(\rho(\alpha) + \frac{\log(1/\delta)}{\alpha - 1}\right)$$

This transformation yields the tightest upper bound for privacy through optimization of α.

We utilize the moments accountant to precisely track the cumulative privacy consumption across multiple training rounds. For the Gaussian mechanism, the single-step λ-order log-moment is as follows:

$$\alpha(\lambda) = \frac{\lambda(\lambda-1)}{2\sigma^2} \cdot \frac{|\mathcal{S}_t|}{K}$$

After T rounds of training, the total logarithmic moment is $\sum_{t=1}^{T} \alpha_t(\lambda)$, and the corresponding (ϵ, δ)-DP guarantee is:

$$\epsilon = \min_{\lambda>1} \left(\frac{\sum_{t=1}^{T} \alpha_t(\lambda) + \ln(1/\delta)}{\lambda} \right)$$

3.4. Cryptographic Primitives: Secret Sharing

Secret sharing is a cryptographic technique that enables the distribution of a secret value among multiple parties, such that only authorized subsets of parties can reconstruct the secret. Two well-known secret sharing schemes are:

- **Shamir Secret Sharing [38]:** A threshold scheme where a secret is divided into n shares, and any t shares (where $t \leq n$) can reconstruct the secret. This scheme is based on polynomial interpolation and is suitable for scenarios requiring threshold-based access control.
- **Additive Secret Sharing:** A simpler scheme where a secret s is split into n shares $s_1, \ldots, s_n$ such that:

  $$s = s_1 + s_2 + \cdots + s_n$$

  Reconstruction requires all n shares (or a full set of shares summing to the secret). This scheme is efficient and naturally supports homomorphic addition.

Our Choice: In this work, we employ *additive secret sharing* as the fundamental cryptographic primitive. There are two key reasons for this choice:

1. **Protocol Requirements:** Our two-party MPC protocol requires shares to be combined via addition (the aggregated gradient is the sum of client shares). Additive secret sharing natively supports this operation without additional computation.
2. **Efficiency:** In the two-server setting, additive secret sharing is significantly more efficient than Shamir's scheme, requiring only simple addition operations rather than polynomial interpolation.

In our protocol, additive secret sharing is used as follows. For a secret value s (e.g., a client's gradient update), the sharing procedure generates two shares $[s]_1$ and $[s]_2$ such that:

$$[s]_1 + [s]_2 = s$$

where addition is performed over a finite field $\mathbb{F}_q$ (component-wise for vectors). The sharing procedure is:

1. Generate a random share $[s]_1 \in \mathbb{F}_q^p$ uniformly at random.
2. Compute the second share as $[s]_2 = s - [s]_1$ in $\mathbb{F}_q^p$.

Reconstruction of the secret requires both shares:

$$s = [s]_1 + [s]_2$$

The additive homomorphic property of this scheme is crucial for our protocol: given two secrets a and b with shares $[a]_1, [a]_2$ and $[b]_1, [b]_2$, the shares of $a + b$ can be computed locally:

$$[a+b]_1 = [a]_1 + [b]_1, \quad [a+b]_2 = [a]_2 + [b]_2$$

This enables the auxiliary servers to aggregate client gradient shares without ever learning the individual gradients.

Throughout this paper, we use $\langle x \rangle$ to denote the additive secret-sharing state of a value x, where $\langle x \rangle = ([x]_1, [x]_2)$. All gradient vectors and noise vectors are shared element-wise.

Note: Shamir secret sharing is mentioned here for completeness as a well-known alternative, but it is *not* used in our protocol. Our implementation and analysis are based exclusively on additive secret sharing.

3.5. Symbols and Conventions

For clarity and consistency throughout this paper, Table 2 summarizes all the primary symbols used. Symbols are organized into logical categories: federated learning parameters, differential privacy parameters, cryptographic parameters, and experimental parameters.

Table 2. Core symbol table.

Symbol	Description
Federated Learning	
K	Total number of clients
$\mathcal{S}_t$	Set of clients selected in round t
θ^t	Global model at round t
Δ_k^t	Local update from client k in round t
$\tilde{\Delta}_k^t$	Clipped local update
$\bar{\Delta}^t$	Aggregated clean gradient
$\bar{\Delta}_{\text{noisy}}^t$	Aggregated noisy gradient
E	Number of local epochs per round
$\lvert\mathcal{S}_t\rvert$	Number of clients per round
p	Model parameter dimension
q	Client sampling probability ($\lvert\mathcal{S}_t\rvert/K$)
T	Total communication rounds
Differential Privacy	
C	Gradient clipping bound (sensitivity)
ϵ, δ	Privacy budget and failure probability
σ_{loc}	Local noise scale (client-side)
σ_{global}	Global noise scale (server-side)
η	Final Gaussian noise added at server
Δ_2	ℓ_2-sensitivity of the query
α	RDP order parameter
$\rho(\alpha)$	RDP budget as a function of α
Cryptography	
$\mathbb{F}_q$	Finite field of order q
$[s]_1, [s]_2$	Additive secret shares of s
$\langle x \rangle$	Secret-sharing state of x
r_j	Random seed of auxiliary server j
c_j	Commitment to seed r_j
r	Combined seed ($r = r_1 \oplus r_2$)
$\oplus$	Bit-wise XOR operation

4. FedDecouple: Protocol Design and Analysis

This section presents the complete design of the FedDecouple protocol. Building on the preliminary concepts established in Section 3, we first define the system model, entities, communication model, and threat assumptions under which our protocol operates (Section 4.1). We then provide a detailed, step-by-step description of the two-phase protocol—secure gradient aggregation and distributed noise generation—in Section 4.2,

followed by a correctness proof in Section 4.2.3. Section 4.3 analyzes the protocol's privacy guarantees, including cryptographic privacy and differential privacy, and provides formal RDP accounting. Finally, Sections 4.4 and 4.5 analyze the communication and computational complexity of the protocol, demonstrating that our MPC-based approach does not increase asymptotic overhead compared to standard federated learning.

The protocol design presented here is implemented and evaluated in Sections 5 and 6, respectively.

4.1. System Model and Threat Assumptions

This section introduces FedDecouple, a differential privacy federated learning framework that integrates phase decoupling with secure multi-party computation.

The entire protocol workflow is illustrated in Figure 2.

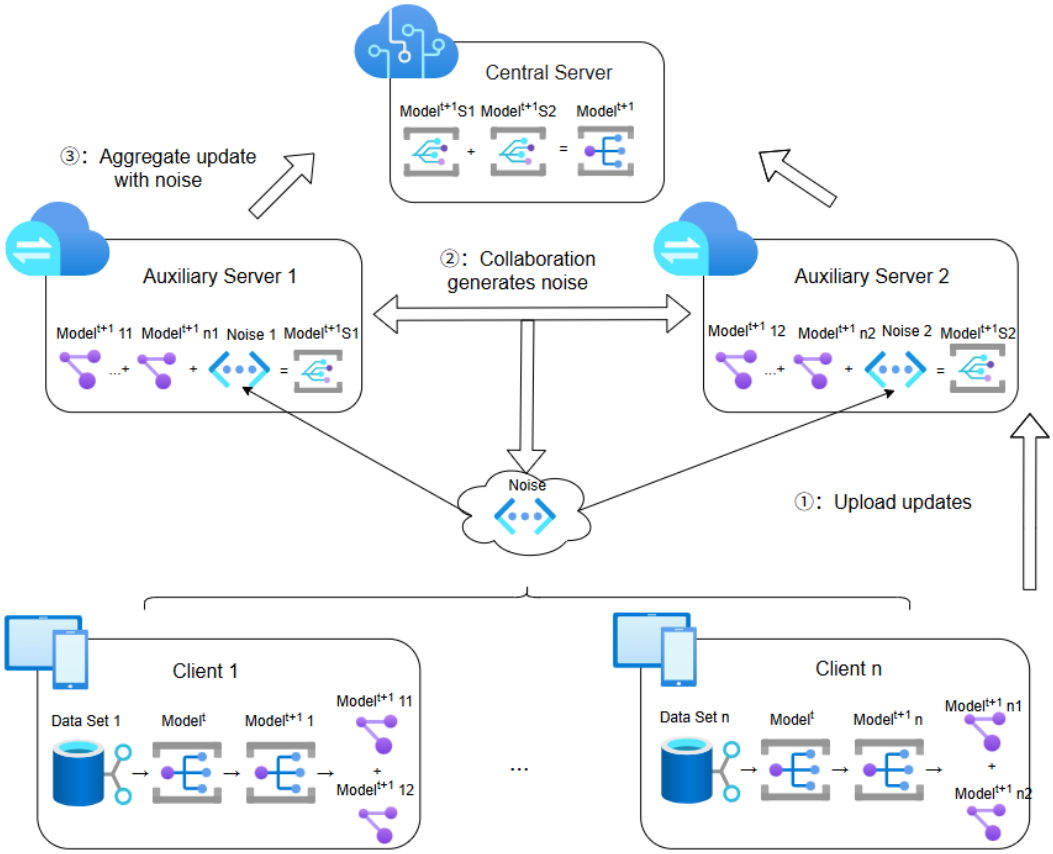

Figure 2. The process of FedDecouple.

4.1.1. System Entities

Our system includes three types of entities:

- **Clients:** There are K clients, each holding a local dataset $\mathcal{D}_k$. Clients are responsible for executing local model training and gradient updates, but do not perform any privacy-preserving operations (e.g., noise addition). Given their resource constraints, we minimize their computational burden.
- **Auxiliary Servers:** There are two auxiliary servers, denoted as $\mathcal{S}_1$ and $\mathcal{S}_2$. They form a two-party secure computation cluster responsible for:
  1. Receiving the client's secret shares.
  2. Performing gradient aggregation on the secret share domain.
  3. Collaboratively generating Gaussian noise that satisfies differential privacy requirements.

4. Adding noise to the aggregated gradient.

- **Central Server:** Serving as the training coordinator, it is responsible for:
  1. Distributing the global model.
  2. Selecting the clients to participate in each round.
  3. Receiving the shares sent by the auxiliary servers and reconstructing the final result.
  4. Updating the global model.

4.1.2. Communication Model

All entities communicate via secure channels (such as TLS). A peer-to-peer connection is established between each client and the two auxiliary servers; direct connections are also established among the auxiliary servers to execute the MPC protocol; similarly, communication between the auxiliary servers and the central server occurs over secure channels.

4.1.3. Threat Assumptions

We employ the standard honest-but-curious threat model [39]:

- **Clients:** We assume the clients are honest and will correctly execute the protocol. However, since we do not fully trust the clients, the gradients they upload must be protected through clipping and secret sharing.
- **Auxiliary Servers:** We assume the auxiliary servers are semi-honest—they strictly adhere to protocol specifications but may attempt to infer private data from the received information. We assume that at most one auxiliary server is compromised, and the two servers do not collude. This assumption is reasonable in practice, as auxiliary servers are typically operated by different institutions.
- **Central Server:** The central server is considered untrustworthy. It can only access the reconstructed noisy gradient and cannot obtain any private information from intermediate states.
- **External Attackers:** Attackers may eavesdrop on communication channels, but since all communications are encrypted over secure channels, they cannot obtain any meaningful information.

4.1.4. Design Objectives

Based on the aforementioned system model and threat assumptions, we have established the following design objectives:

- **Privacy:** The entire training process complies with (ϵ, δ)-differential privacy, and no individual entity (including the auxiliary servers) can obtain the clean gradient information.
- **Decentralized Trust:** The framework does not rely on a single trusted server, with privacy protection responsibilities shared by two auxiliary servers.
- **Efficiency:** The client's computational load is minimized, with complex cryptographic operations handled by the auxiliary servers.
- **Scalability:** The protocol should be scalable to support more clients and more complex models.

4.2. Core Protocol: FedDecouple Two-Phase Design

This section provides a detailed description of the core workflow of the FedDecouple protocol. The protocol consists of two main phases: the secure gradient aggregation phase and the distributed noise generation and addition phase.

4.2.1. Phase One: Secure Gradient Aggregation

In the first phase, clients collaboratively submit the clean gradients as secret shares to the auxiliary servers. The auxiliary servers then aggregate the gradients within the ciphertext domain without either party being able to access the actual gradient values.

1. **Step 1: Local Training and Gradient Clipping (Client)**
   Each selected client $k \in \mathcal{S}_t$ performs E rounds of SGD training on local data to compute the model update:
   $$\Delta_k^t = \theta_k^{t,\text{local}} - \theta^t$$
   To meet the sensitivity requirements of differential privacy, clients apply gradient clipping:
   $$\tilde{\Delta}_k^t = \Delta_k^t \cdot \min\left(1, \frac{C}{\|\Delta_k^t\|_2}\right)$$
   where C is the preset clipping threshold. **Key difference:** No noise is added here.
2. **Step 2: Secret Sharing of Gradient Updates (Client)**
   Each client uses additive secret sharing (as defined in Section 3.4) to split the trimmed gradient update into two parts:
   $$[\tilde{\Delta}_k^t]_1, [\tilde{\Delta}_k^t]_2 \leftarrow \text{Share}(\tilde{\Delta}_k^t)$$
   satisfying $[\tilde{\Delta}_k^t]_1 + [\tilde{\Delta}_k^t]_2 = \tilde{\Delta}_k^t$. The client sends share $[\tilde{\Delta}_k^t]_1$ to auxiliary server $\mathcal{S}_1$ and share $[\tilde{\Delta}_k^t]_2$ to auxiliary server $\mathcal{S}_2$.
3. **Step 3: Auxiliary Server Share Aggregation (Auxiliary Servers)**
   Each auxiliary server independently aggregates the received shares locally:
   $$[\bar{\Delta}^t]_j = \sum_{k \in \mathcal{S}_t} [\tilde{\Delta}_k^t]_j, \quad j \in \{1, 2\}$$
   Due to the additive homomorphism of secret sharing, the aggregated shares $[\bar{\Delta}^t]_j$ precisely correspond to the correct shares of the actual aggregated gradient:
   $$\bar{\Delta}^t = \sum_{k \in \mathcal{S}_t} \tilde{\Delta}_k^t$$
   At this point, the two auxiliary servers each hold a secret share of the aggregated gradient, but no single server can reconstruct the actual aggregated gradient.

4.2.2. Phase Two: Distributed Noise Generation and Addition

The second phase constitutes the core innovation of this paper. Two auxiliary servers collaboratively generate Gaussian noise compliant with differential privacy through a six-round interactive protocol and securely integrate it into the aggregated gradient shares.

Since the noise is divided into two parts, each server only knows one portion. Each portion follows a Gaussian distribution $\mathcal{N}(0, \sigma^2 I/2)$ and cannot be used to infer the complete noise. Even if a server is compromised, the attacker cannot obtain the final added noise value.

1. **Step 4: Two-Party Collaborative Noise Share Generation (Auxiliary Servers)**
   Two auxiliary servers execute the secure two-party Gaussian noise generation protocol to generate secret noise shares η_1, η_2 satisfying $\eta_1 + \eta_2 = \eta \sim \mathcal{N}(0, \sigma^2 I)$. The protocol flow is as follows:
   - **Round 0—Local Seed Generation:** Each server independently generates a local random seed $r_j \in \{0,1\}^{32}$.

- **Round 1—Commitment Exchange [40]:** Each server calculates a commitment $c_j = \text{SHA-256}(r_j)$ [41] to its seed and sends the commitment to the other party.
- **Round 2—Seed Exchange and Verification:** Both parties exchange the original seeds and verify $\text{SHA-256}(r_j) = c_j$. If verification fails, the protocol terminates.
- **Round 3—Combined Seed Calculation:** The shared seed is computed using a bit-wise XOR operation: $r = r_1 \oplus r_2$. Due to the symmetry of XOR, both parties independently derive the same result.
- **Round 4—Independent Seed Derivation:** Both servers use the same shared seed r to initialize a pseudo-random number generator (PRNG). The PRNG is then used to derive two **independent seeds**:

$$s_1 = \text{PRNG}(r, \text{“server1”}), \qquad s_2 = \text{PRNG}(r, \text{“server2”})$$

  where the second argument is a domain-separation string to ensure independence.
- **Round 5—Independent Noise Generation:** Each server uses **only its assigned derived seed** to generate its noise share:

$$\mathcal{S}_1 : \eta_1 \sim \mathcal{N}(0, (\sigma/\sqrt{2})^2 I) \text{ using } s_1$$

$$\mathcal{S}_2 : \eta_2 \sim \mathcal{N}(0, (\sigma/\sqrt{2})^2 I) \text{ using } s_2$$

  Crucial distinction from previous version: No server computes the other server's noise share. $\mathcal{S}_1$ does not have s_2, and $\mathcal{S}_2$ does not have s_1. Therefore, neither server can independently determine the final noise $\eta = \eta_1 + \eta_2$.

Crucial Security Observation: The final noise η is never transmitted or communicated between the servers. Each server only holds one additive share η_j. The final noise η is only revealed after the shares are added by the central server in Step 6. Even if a single auxiliary server is compromised, the attacker only obtains one share η_j, which is statistically independent of the final noise value η (since the other share is unknown and uniformly distributed).
We formally state this security property:

Theorem 1 (Noise Unpredictability (Revised))**.** *Under the honest-but-curious model, a single auxiliary server $\mathcal{S}_j$ cannot determine the final noise value η from its view of the protocol, even if it records all locally computed values.*

Proof. The view of $\mathcal{S}_1$ consists of: (1) its own seed r_1, (2) the other server's commitment c_2, (3) the other server's seed r_2 after Round 2, (4) the shared seed $r = r_1 \oplus r_2$, and (5) its own derived seed s_1 and its own noise share η_1.
Crucially, $\mathcal{S}_1$ does not compute s_2, because the PRNG is executed only once with the domain-separation string for s_1 (since $\mathcal{S}_1$ only needs its own share). Without s_2, $\mathcal{S}_1$ cannot generate η_2. The final noise is $\eta = \eta_1 + \eta_2$, where η_2 is unknown to $\mathcal{S}_1$. Since $\eta_2 \sim \mathcal{N}(0, (\sigma/\sqrt{2})^2 I)$, the conditional distribution of η given η_1 remains $\mathcal{N}(\eta_1, (\sigma/\sqrt{2})^2 I)$, which is non-degenerate. Therefore, $\mathcal{S}_1$ cannot uniquely determine η. The same argument applies symmetrically to $\mathcal{S}_2$. □

On the Use of 32-Bit Seeds: The seed length of 32 bits is a design choice in the current implementation. The security of the protocol does not rely solely on the seed length; it relies on the commitment scheme (SHA-256) and the fact that seeds are exchanged before noise generation. The commitment prevents either party from altering its seed after seeing the other's. Even with a 32-bit seed, the probability of a successful collision attack is negligible in the context of this protocol (the seed is used only to

initialize a PRNG, not as a cryptographic key). For stronger security, the seed length could be increased to 128 or 256 bits in future implementations without changing the protocol structure.

2. **Step 5: Local Noise Addition (Auxiliary Servers)**
   Each auxiliary server locally adds its noise share to the aggregated gradient share:

$$[\bar{\Delta}^t_{\text{noisy}}]_j = [\bar{\Delta}^t]_j + \eta_j, \quad j \in \{1, 2\}$$

3. **Step 6: Reconstruction of Noisy Aggregated Gradient (Central Server)**
   The two auxiliary servers transmit their noisy shares to the central server. The central server then reconstructs the final noisy aggregated gradient through addition:

$$\bar{\Delta}^t_{\text{noisy}} = [\bar{\Delta}^t_{\text{noisy}}]_1 + [\bar{\Delta}^t_{\text{noisy}}]_2 = \bar{\Delta}^t + \eta$$

4. **Step 7: Global Model Update (Central Server)**
   The central server uses the aggregated gradient to update the global model:

$$\theta^{t+1} = \theta^t + \frac{1}{|\mathcal{S}_t|} \cdot \bar{\Delta}^t_{\text{noisy}}$$

   The updated model is broadcast to all clients, initiating the next training round.

4.2.3. Threat Assumptions and Adversarial Analysis

We employ the standard honest-but-curious threat model as our primary adversarial framework. In this section, we systematically analyze the adversarial capabilities, discuss the plausibility of our assumptions, and compare our approach with alternative trust models.

Adversarial Capabilities

We consider the following adversarial scenarios:

- **Passive (Honest-but-Curious) Adversaries:** Adversaries faithfully follow the protocol but attempt to infer private information from the messages they observe. This is the primary threat model considered in this work.
- **Active (Malicious) Adversaries:** Adversaries may deviate from the protocol, inject malformed messages, or manipulate their shares to compromise the aggregation correctness or privacy.
- **Colluding Adversaries:** Multiple adversarial entities may cooperate to combine their views and infer private information about honest clients.

Justification of the Non-Colluding Assumption

The assumption that the two auxiliary servers do not collude is common in two-server MPC literature and is justified by the following practical considerations:

1. **Institutional Separation:** The auxiliary servers are typically operated by independent organizations (e.g., different cloud providers, government agencies, or research institutions). Collusion requires a coordinated breach of trust between independent entities, which is significantly harder to achieve than compromising a single server.
2. **Legal and Compliance Constraints:** In cross-silo federated learning scenarios, participating institutions are often bound by legal agreements and compliance frameworks that prohibit unauthorized data sharing, further raising the bar for collusion.

3. **Auditability:** All server-side operations are logged and can be audited post hoc. Any suspicious activity or coordination between servers can be detected through statistical analysis of their behaviors.

Nevertheless, we acknowledge that this assumption may be restrictive in certain deployment scenarios. We discuss extensions to strengthen collusion resistance in Section 7.2.

Collusion Analysis

We analyze the privacy leakage under different collusion scenarios.

Theorem 2 (Collusion Resilience)**.** *In the proposed protocol, an adversary that compromises one auxiliary server and an arbitrary number of clients cannot reconstruct any honest client's clean gradient.*

Proof. Each client u_i splits its gradient $\tilde{\Delta}_k^t$ into two additive shares:

$$\tilde{\Delta}_k^t = [\tilde{\Delta}_k^t]_1 + [\tilde{\Delta}_k^t]_2$$

where $[\tilde{\Delta}_k^t]_1$ is sent to $\mathcal{S}_1$ and $[\tilde{\Delta}_k^t]_2$ is sent to $\mathcal{S}_2$. A single auxiliary server observes only one share. By the security of additive secret sharing, a single share is uniformly distributed over $\mathbb{F}_q^p$ and is statistically independent of the original gradient. Therefore, even a compromised auxiliary server $\mathcal{S}_1$ cannot infer any information about the clean gradient of an honest client. □

Theorem 3 (Bounded Leakage under Limited Collusion)**.** *If both auxiliary servers are compromised by colluding adversaries, the adversary can reconstruct the clean aggregated gradient $\bar{\Delta}^t$. However, the privacy of individual clients is still protected by differential privacy.*

Proof. When both servers are compromised, the adversary can reconstruct:

$$[\bar{\Delta}^t]_1 + [\bar{\Delta}^t]_2 = \bar{\Delta}^t$$

However, the aggregation process releases only the noisy aggregated gradient $\bar{\Delta}_{\text{noisy}}^t = \bar{\Delta}^t + \eta$, where $\eta \sim \mathcal{N}(0, \sigma^2 C^2 I)$. By Theorem 4 (Differential Privacy), this release satisfies (ϵ, δ)-DP, which limits the adversary's ability to infer individual client information even if the clean aggregate is recovered. □

Comparison with Threshold Secret Sharing

Threshold secret sharing (e.g., Shamir's t-out-of-n scheme) is an alternative approach to distribute trust among multiple servers. Table 3 compares our two-server additive sharing approach with threshold secret sharing.

Key Trade-offs:

- **Security:** Threshold secret sharing offers stronger collusion resistance at the cost of higher communication and computational overhead.
- **Efficiency:** Two-server additive sharing is significantly more efficient, making it suitable for resource-constrained federated learning scenarios.
- **Application Context:** Our choice is motivated by the practical requirements of cross-silo federated learning, where two independent servers are often available, and efficiency is critical.

Table 3. Comparison of trust models: two-server additive sharing vs. threshold secret sharing.

Criterion	**Two-Server Additive Sharing**	**Threshold Secret Sharing (*t*-Out-of-*n*)**
Collusion Threshold	Breaks if both servers collude	Breaks if $\geq t$ servers collude
Communication Overhead	$O(\lvert\mathcal{S}_t\rvert \cdot p)$ (2 shares/client)	$O(\lvert\mathcal{S}_t\rvert \cdot n \cdot p)$ (n shares/client)
Computational Complexity	$O(\lvert\mathcal{S}_t\rvert \cdot p)$ additions	$O(\lvert\mathcal{S}_t\rvert \cdot n \cdot p)$ additions + interpolation
Recovery Cost	Simple addition of 2 shares	Lagrange interpolation ($O(n \log^2 n)$)
Server Setup	Minimal (pairwise connections)	Requires distributed key generation
Resilience to Collusion	Breaks if both servers collude	Breaks if $\geq t$ servers collude

Extension to Three-Party MPC

For scenarios requiring stronger collusion resistance, our protocol can be extended to a three-server setting using three-party additive sharing:

$$\tilde{\Delta}_k^t = [\tilde{\Delta}_k^t]_1 + [\tilde{\Delta}_k^t]_2 + [\tilde{\Delta}_k^t]_3$$

where shares are distributed among three auxiliary servers $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$. The noise generation protocol can be similarly extended to three parties:

$$\eta_{\text{full}} \sim \mathcal{N}(0, (\sigma/3)^2 I), \quad \eta = \eta_1 + \eta_2 + \eta_3$$

Byzantine Robustness and Future Work

We emphasize that Byzantine robustness (defending against malicious clients or servers that actively deviate from the protocol) is orthogonal to our privacy guarantees and is not the focus of this work. Our threat model is explicitly limited to privacy against semi-honest adversaries. Protecting against Byzantine behaviors (e.g., gradient poisoning attacks) would require orthogonal techniques such as robust aggregation rules, anomaly detection, or verification mechanisms. We leave the integration of Byzantine robustness as an important direction for future work (see Section 7.2).

Summary of Trust Assumptions

In summary, our trust assumptions can be stated as follows:

1. Clients are honest-but-curious and execute the protocol correctly.
2. Auxiliary servers are semi-honest, with at most one being compromised.
3. The two auxiliary servers do not collude.
4. No active Byzantine attacks are considered in the current scope.

These assumptions are standard in two-server MPC literature and, as discussed above, are justified in many practical cross-silo FL deployments.

Consequences of Assumption Violation

We explicitly state the consequences if the non-colluding assumption is violated:

- If both auxiliary servers collude, they can combine their shares and reconstruct the clean aggregated gradient $\bar{\Delta}^t$ before noise addition.
- However, individual client privacy is still protected by differential privacy, as the final released gradient is $\bar{\Delta}_{\text{noisy}}^t = \bar{\Delta}^t + \eta$. Even with the clean aggregate, an adversary

cannot infer individual client updates from the aggregate alone due to the additive nature of the aggregation.
- If the adversary additionally compromises a sufficient number of clients, the risk of inferring individual client information increases, but this falls outside our current threat model.

We emphasize that this assumption is standard in the two-server MPC literature and is justified in many practical cross-silo FL deployments where auxiliary servers are operated by independent institutions.

4.2.4. Comparison with Stronger MPC Settings

For scenarios requiring stronger collusion resistance, our protocol can be extended to a three-server setting using three-party additive sharing. Table 4 summarizes the trade-offs.

Table 4. Trade-offs: two-server vs. three-server MPC configuration.

Metric	Two-Server (Ours)	Three-Server (Extension)
Collusion Resilience	Collusion of both servers	Collusion of all three servers
Client Communication	2 shares per client	3 shares per client
Server-Server Communication	Seed/commitment exchange	More complex coordination
Noise Generation Rounds	6 rounds	8–10 rounds
Computation Overhead	Baseline	$\sim$1.5$\times$ baseline

Threshold secret sharing (e.g., Shamir's t-out-of-n scheme) offers stronger collusion resistance at the cost of higher communication and computational overhead. The choice between two-server additive sharing and threshold secret sharing depends on the specific deployment scenario: two-server sharing is preferable when efficiency is critical and the non-colluding assumption is reasonable, while threshold schemes are more suitable when stronger security guarantees are required.

4.3. Rigorous Privacy Analysis and RDP Accounting

We provide a rigorous privacy analysis of FedDecouple under the Rényi Differential Privacy (RDP) framework. We first establish the sensitivity of a single client's contribution, then derive the privacy amplification via client subsampling, and finally present the complete RDP composition over multiple rounds.

4.4. Analysis of Protocol Privacy

4.4.1. Sensitivity Analysis

We now derive the ℓ_2-sensitivity of the aggregate gradient update under FedDecouple's phase-decoupled protocol. We explicitly consider the **client-level replacement adjacency model**, which is standard in DP-FedAvg analysis.

Definition 4 (Client-level Replacement Adjacency)**.** *Two client datasets $\mathcal{D}$ and $\mathcal{D}'$ are considered adjacent if they differ by the data of exactly one client, where that client's entire local dataset is **replaced** with a different dataset. Formally, there exists a client k such that:*

$$\mathcal{D} = \{D_1, \ldots, D_k, \ldots, D_K\}, \quad \mathcal{D}' = \{D_1, \ldots, D'_k, \ldots, D_K\}$$

where $D_k \neq D'_k$, and all other clients' datasets are identical.

Definition 5 (Gradient Clipping). *For a client k with local update Δ_k^t, the clipped gradient is:*

$$\tilde{\Delta}_k^t = \Delta_k^t \cdot \min\left(1, \frac{C}{\|\Delta_k^t\|_2}\right)$$

where $C > 0$ is the clipping bound. Note that $\|\tilde{\Delta}_k^t\|_2 \leq C$.

Theorem 4 (Sensitivity Bound). *Under the replacement adjacency model, the ℓ_2-sensitivity of the aggregated clipped gradient is bounded by:*

$$\Delta_2 = \max_{\mathcal{D},\mathcal{D}'} \left\| \sum_{k \in \mathcal{S}_t} \tilde{\Delta}_k^t - \sum_{k \in \mathcal{S}_t} \tilde{\Delta}_k^{t'} \right\|_2 \leq 2C$$

Proof. Since the two datasets differ only in client k, all other client contributions cancel:

$$\left\| \sum_{i \neq k} \tilde{\Delta}_i^t + \tilde{\Delta}_k^t - \left(\sum_{i \neq k} \tilde{\Delta}_i^{t'} + \tilde{\Delta}_k^{t'} \right) \right\|_2 = \left\| \tilde{\Delta}_k^t - \tilde{\Delta}_k^{t'} \right\|_2$$

By the triangle inequality and the clipping bound $\|\tilde{\Delta}_k^t\|_2 \leq C$ and $\|\tilde{\Delta}_k^{t'}\|_2 \leq C$:

$$\left\| \tilde{\Delta}_k^t - \tilde{\Delta}_k^{t'} \right\|_2 \leq \|\tilde{\Delta}_k^t\|_2 + \|\tilde{\Delta}_k^{t'}\|_2 \leq 2C$$

This bound is tight: consider the case where $\tilde{\Delta}_k^t = C \cdot v$ and $\tilde{\Delta}_k^{t'} = -C \cdot v$ for some unit vector v. Then $\|\tilde{\Delta}_k^t - \tilde{\Delta}_k^{t'}\|_2 = 2C$. Therefore, $\Delta_2 \leq 2C$. □

Note 1. *In the add/remove adjacency model (where a client is either added or removed rather than replaced), the sensitivity would be C (since the removed client contributes 0). However, we use the replacement adjacency model throughout this paper, which is the standard in DP-FedAvg and yields sensitivity $2C$. The Gaussian mechanism noise scale σ is calibrated accordingly.*

4.4.2. Privacy Amplification via Client Subsampling

In each communication round, the server randomly selects a subset of clients $\mathcal{S}_t$ with sampling probability $q = |\mathcal{S}_t|/K$. This random subsampling induces a privacy amplification effect.

Theorem 5 (Privacy Amplification by Subsampling). *Let $\mathcal{M}$ be a mechanism satisfying $(\alpha, \rho(\alpha))$-RDP. Then the subsampled mechanism $\mathcal{M} \circ Sample_q$, which applies $\mathcal{M}$ to a uniformly random subset of clients with sampling probability q, satisfies:*

$$\rho_{amp}(\alpha) \leq \frac{1}{\alpha - 1} \log\left(1 + q^2 \binom{\alpha}{2} \min\{4(e^{\rho(\alpha)} - 1), 2e^{\rho(\alpha)}\}\right)$$

for $\alpha \in (1, \infty)$, where $q = |\mathcal{S}_t|/K$ is the sampling rate.

Proof. This follows directly from the privacy amplification theorem for subsampling with Rényi differential privacy. The bound is obtained by applying the subsampling lemma to the Gaussian mechanism's RDP guarantee. □

In practice, for the Gaussian mechanism with noise scale σ, the single-round RDP after amplification is:

$$\rho_{\text{round}}(\alpha) \approx \frac{\alpha q^2}{2\sigma^2} \cdot \frac{|\mathcal{S}_t|^2}{K^2}$$

which is significantly tighter than the non-amplified bound $\rho(\alpha) = \alpha/(2\sigma^2)$.

4.4.3. Rényi Differential Privacy Composition

Over T communication rounds, the total privacy loss is the composition of T rounds. By the composition property of RDP:

Theorem 6 (RDP Composition over Rounds)**.** *After T rounds of subsampled Gaussian mechanism with noise scale σ and sampling rate q, the total RDP budget is:*

$$\rho_{total}(\alpha) = T \cdot \rho_{round}(\alpha)$$

for any order $\alpha > 1$, where $\rho_{round}(\alpha)$ is the single-round RDP after amplification.

Proof. This follows from the additivity of RDP under adaptive composition. Since each round's mechanism is applied sequentially and the composition is adaptive (the output of each round influences the next), the total RDP is the sum of the per-round RDP budgets. □

4.4.4. Formal Privacy Guarantee

We now present the complete privacy theorem for FedDecouple.

Theorem 7 (Privacy Guarantee of FedDecouple)**.** *For any $\delta > 0$, privacy budget $\epsilon > 0$, let C be the clipping bound and σ the noise scale. After T communication rounds with client sampling probability $q = |\mathcal{S}_t|/K$, FedDecouple satisfies (ϵ, δ)-user-level differential privacy, where:*

$$\epsilon = \min_{\alpha>1}\left(T \cdot \rho_{round}(\alpha) + \frac{\log(1/\delta)}{\alpha - 1}\right)$$

with

$$\rho_{round}(\alpha) = \frac{1}{\alpha - 1}\log\left(1 + q^2\binom{\alpha}{2}\min\{4(e^{\alpha/(2\sigma^2)} - 1), 2e^{\alpha/(2\sigma^2)}\}\right)$$

Proof. The proof combines the sensitivity bound (Theorem 1), the subsampling amplification (Theorem 2), and the RDP composition (Theorem 3). The conversion from RDP to (ϵ, δ)-DP follows the standard RDP-to-DP conversion lemma. □

4.4.5. Privacy Accounting with Corrected Sensitivity

With the corrected sensitivity $\Delta_2 = 2C$, the Gaussian mechanism with noise scale σ has RDP:

$$\rho(\alpha) = \frac{\alpha\Delta_2^2}{2\sigma^2} = \frac{\alpha(2C)^2}{2\sigma^2} = \frac{2\alpha C^2}{\sigma^2}$$

After subsampling amplification with sampling probability $q = |\mathcal{S}_t|/K$:

$$\rho_{\text{round}}(\alpha) = \frac{1}{\alpha - 1}\log\left(1 + q^2\binom{\alpha}{2}\min\{4(e^{\rho(\alpha)} - 1), 2e^{\rho(\alpha)}\}\right)$$

After T rounds:

$$\rho_{\text{total}}(\alpha) = T \cdot \rho_{\text{round}}(\alpha)$$

The final (ϵ, δ)-DP guarantee is:

$$\epsilon = \min_{\alpha>1}\left(\rho_{\text{total}}(\alpha) + \frac{\log(1/\delta)}{\alpha - 1}\right)$$

4.4.6. Quantitative Comparison with DP-FedAvg

Table 5 compares the privacy guarantees of FedDecouple and DP-FedAvg under equivalent configurations.

Table 5. Quantitative comparison of privacy guarantees (FedDecouple vs. DP-FedAvg).

Criterion	DP-FedAvg (Client-Side Noise)	FedDecouple (Server-Side Noise)
Sensitivity (Δ_2)	$2C$	$2C$
Noise Scale per Round	σ_{loc}	σ_{global}
Effective Noise Variance	$\frac{4\sigma_{\text{loc}}^2 C^2}{\lvert\mathcal{S}_t\rvert^2} \cdot \sum_k 1$	$4\sigma_{\text{global}}^2 C^2$
RDP per Round	$\frac{\alpha}{2\sigma_{\text{loc}}^2} \cdot q^2$	$\frac{\alpha}{2\sigma_{\text{global}}^2} \cdot q^2$
Total RDP (T rounds)	$\frac{\alpha T}{2\sigma_{\text{loc}}^2} \cdot q^2$	$\frac{\alpha T}{2\sigma_{\text{global}}^2} \cdot q^2$
Privacy Budget (ϵ)	Larger (noise accumulates)	Smaller (noise injected once)
Privacy Efficiency	Lower	Higher

Key Insight: To achieve the same (ϵ, δ)-DP guarantee, DP-FedAvg requires a noise scale σ_{loc} that grows with $\sqrt{|\mathcal{S}_t|}$ to compensate for the accumulation of client-side noise. In contrast, FedDecouple's noise scale σ_{global} is independent of the number of clients, resulting in significantly tighter privacy bounds and better utility for the same privacy budget.

4.4.7. Discussion of Privacy Budget Efficiency

The privacy efficiency of FedDecouple can be understood through the following comparison. In DP-FedAvg, each client adds noise $\mathcal{N}(0, \sigma_{\text{loc}}^2 C^2 I)$, and after aggregation, the effective noise variance is:

$$\text{Var}_{\text{DP-FedAvg}} = \frac{1}{|\mathcal{S}_t|^2} \sum_{k \in \mathcal{S}_t} \sigma_{\text{loc}}^2 C^2 = \frac{\sigma_{\text{loc}}^2 C^2}{|\mathcal{S}_t|}$$

while in FedDecouple, the noise is added once at the server:

$$\text{Var}_{\text{FedDecouple}} = \sigma_{\text{global}}^2 C^2$$

To achieve the same effective noise variance and thus the same utility, the server-side noise scale in FedDecouple satisfies $\sigma_{\text{global}} = \sigma_{\text{loc}} / \sqrt{|\mathcal{S}_t|}$. However, under the same (ϵ, δ)-DP guarantee, the RDP analysis shows that σ_{loc} must be larger than σ_{global} to compensate for the privacy cost of repeated client-side noise injection. Therefore, FedDecouple achieves both stronger privacy and better utility.

Conclusion: FedDecouple's phase-decoupled design provides a formal privacy guarantee that is mathematically equivalent to DP-FedAvg in terms of sensitivity but superior in terms of privacy budget efficiency due to its centralized, one-shot noise injection paradigm.

4.4.8. Privacy Guarantee via Gaussian Mechanism

If the noise scale σ satisfies the requirements of the Gaussian mechanism and the clipping threshold C is properly set, the entire training process achieves (ϵ, δ)-user-level differential privacy.

Proof. Since only the noisy aggregated gradient $\bar{\Delta}_{\text{noisy}}^t = \bar{\Delta}^t + \eta$ is ultimately released and satisfies the requirements of the Gaussian mechanism, the release process itself inherently ensures differential privacy. Due to post-processing invariance, subsequent model updates

also maintain the same privacy protection. The cumulative privacy across multiple training rounds can be precisely calculated using the moments accountant. □

4.4.9. Summary of the Unified Privacy Framework

To ensure consistency across the privacy analysis, we summarize the unified framework under the client-level adjacency model:

- **Adjacency model:** Datasets differ by one client.
- **Sensitivity:** $\Delta_2 = 2C$ (clipped gradient norm bound).
- **Clipping:** Each client's update is clipped to ℓ_2-norm C.
- **Sampling:** Uniform client subsampling with probability $q = |\mathcal{S}_t|/K$.
- **Noise calibration:** Gaussian noise $\mathcal{N}(0, \sigma^2 C^2 I)$ injected at the server.
- **Privacy accounting:** RDP composition over T rounds under client-level adjacency.

All privacy parameters and derivations in this paper are consistent with this framework. We use "client-level" as the primary term and "user-level" only informally as a synonym.

4.5. Communication Complexity Analysis

We analyze the per-round communication overhead of FedDecouple and compare it with standard federated learning (FedAvg). We distinguish between *asymptotic complexity* and *actual constant factors*.

4.5.1. Asymptotic Complexity

The communication overhead per training round includes:

- **Clients to Auxiliary Servers:** Each client sends two additive shares, each of size $O(p)$ (where p is the model dimension). Total client communication volume is $O(|\mathcal{S}_t| \cdot p)$.
- **Between Auxiliary Servers:** The two-party noise generation protocol requires exchanging seeds, commitments, and other messages. The communication volume is $O(1)$ independent of model size.
- **Auxiliary Servers to Central Server:** Each auxiliary server sends one share of size $O(p)$, totaling $O(p)$ communication.

The total asymptotic communication complexity is $O(|\mathcal{S}_t| \cdot p)$, which is asymptotically identical to standard FL (without MPC). This is the sense in which our method "does not increase asymptotic communication overhead."

4.5.2. Actual Communication Overhead (Constant Factors)

While the asymptotic complexity is the same, the actual constant factors are larger due to the following reasons:

- Each client sends **two shares** instead of one update ($2\times$ factor).
- Extra messages are exchanged between the two auxiliary servers (commitments, seeds, verification).
- The central server receives two shares instead of one.

Table 6 provides a detailed per-entity breakdown of the communication overhead.

Table 6. Per-round communication overhead breakdown.

Sender → Receiver	FedAvg (No DP)	FedDecouple	Overhead Factor
Client → Server/Aux.	$1 \cdot O(p)$	$2 \cdot O(p)$	$2\times$
Aux. S1 ↔ Aux. S2	N/A	$O(1)$ (seeds/commitments)	Extra messages
Aux. → Central Server	N/A	$2 \cdot O(p)$	Extra messages

Key Insight: Despite the higher constant-factor communication overhead, our experimental results (Section 6) show that FedDecouple achieves comparable or better *total training time* than DP-SGD and Opacus. This is because:

1. **Faster convergence:** FedDecouple requires fewer communication rounds to reach target accuracy (see Figures 1–3), partially offsetting the per-round overhead.
2. **Lighter client computation:** Clients do not perform per-sample gradient computation or local noise addition, reducing per-round computation time.

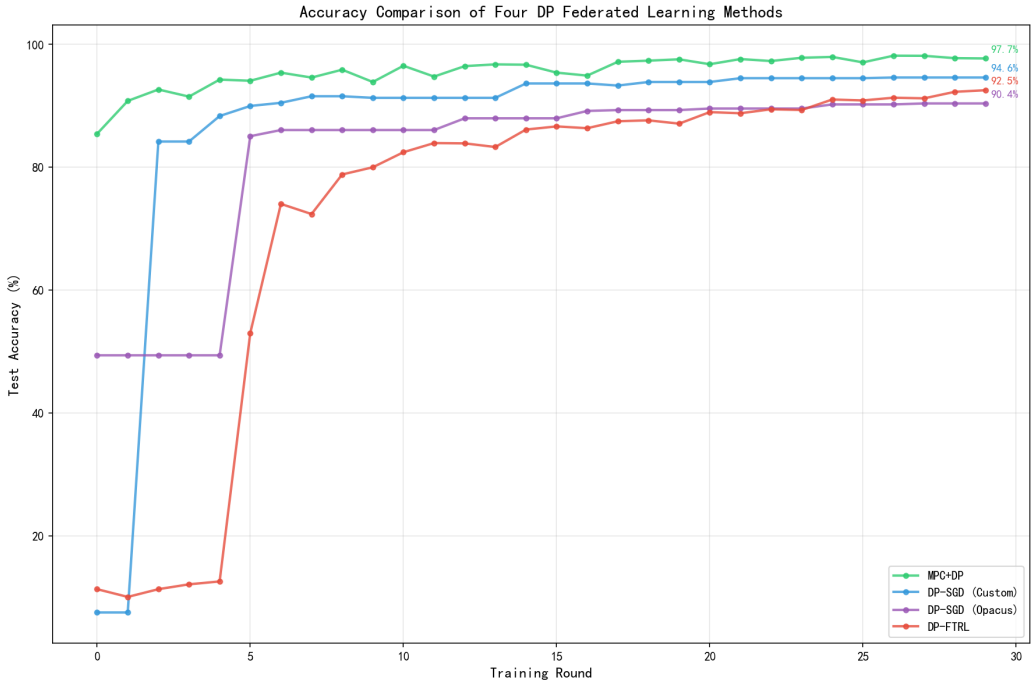

Figure 3. Test accuracy convergence on MNIST $\varepsilon = 8$.

In practice, total training time (communication + computation) is a more meaningful efficiency metric than communication rounds alone. Our empirical results show that FedDecouple achieves the shortest total training time on MNIST and competitive performance on CIFAR-10 (see Table 6), validating the practical efficiency of our approach.

4.6. Computational Complexity Analysis

- **Client Computation:** Each client only needs to compute the gradient once (similar to standard FedAvg) and perform secret sharing ($O(p)$ addition operations). There is no need for per-sample gradient computation, significantly reducing the computational burden compared to solutions like Opacus.
- **Auxiliary Server Computation:** Each auxiliary server performs share aggregation ($O(|\mathcal{S}_t| \cdot p)$ additions) and participates in the two-party noise generation protocol ($O(p)$ PRF calculations). These operations are lightweight.
- **Central Server Computation:** Requires only one share reconstruction ($O(p)$ additions).

4.7. Comparison with Existing Privacy-Preserving Paradigms

To further highlight the unique positioning of FedDecouple, we provide a comprehensive comparison with five representative privacy-preserving paradigms in Table 7.

Table 7. Comparison of FedDecouple with existing privacy-preserving paradigms in federated learning.

Feature/ Paradigm	Distributed DP	SecAgg	SecAgg + Central DP	Distributed Noise Gen.	Local DP	FedDecouple (Ours)
Noise Location	Client-side	None (MPC/SS)	Server-side	Distributed	Client-side	**Server-side (via MPC)**
Trust Assumption	Trusted Server	No Trusted Server	Trusted Server	No Trusted Server	No Trusted Server	**No Trusted Server**
Client Computation	Gradient + Noise	Heavy (Encryption/SS)	Gradient	Gradient + (Partial)	Gradient + Noise	**Gradient Only (Lightweight)**
Resolves Noise Accumulation?	No	N/A	Yes	No	No	**Yes (Phase Decoupling)**
Key Limitation vs. Ours	Noise accumulates	No inherent DP	Single point of failure	Complex coordination	High noise, low utility	—

Analysis of Key Differences:

vs. Distributed DP: While both approaches involve distributed participants, Distributed DP still requires noise to be added **locally** at each client, which fails to resolve the noise accumulation problem. Conversely, FedDecouple offloads this task entirely by injecting noise at the server side through a secure MPC protocol.

vs. Secure Aggregation (SecAgg) and Central DP: Standard SecAgg protocols ensure that the server cannot see individual updates but do not inherently provide differential privacy. When combined with Central DP, the noise is added by a **single trusted server**. This reintroduces a single point of trust and requires heavy cryptographic computation on the client side (e.g., for encryption of individual masks). FedDecouple is unique in that it achieves server-side noise injection **without** requiring a trusted server and with **minimal client overhead**.

vs. Distributed Noise Generation: Some works propose distributed noise generation, but they often focus solely on generating the noise and still require the server to perform the final aggregation. In contrast, FedDecouple tightly couples the distributed noise generation with a **phase-decoupled aggregation protocol**, which is specifically designed to leverage the central noise for optimal utility. This comprehensive integration allows our method to fundamentally eliminate noise accumulation, a structural problem left unaddressed by simply distributing the noise generation process.

5. Experimental Setup and Implementation

This section describes the implementation details and experimental configuration used to evaluate FedDecouple. We first define the experimental setup, including the baseline methods we compare against (Section 5.1). We then describe the model architectures used for each method (Section 5.2), followed by the datasets and preprocessing steps (Section 5.3). Section 5.4 presents the common hyperparameter configurations used across all experiments. Section 5.5 explains our implementation of the moments accountant for privacy tracking, and Section 5.6 lists the hardware and software environment. The experimental results obtained from this setup are reported and analyzed in Section 6.

5.1. Overview of Experimental Setup

5.1.1. Limitations of the Comparison Protocol

We acknowledge that the use of different model architectures, normalization methods, and batch sizes across methods may confound the comparison. Specifically:

- GroupNorm is used for FedDecouple because BatchNorm's running statistics are unstable under DP training, as noted in prior work [21]. Using BatchNorm with DP would cause training failure, making comparison impossible.

- The baseline methods use the architectures reported in their respective papers, preserving their established performance characteristics.
- Batch size differences (64 for most methods, 128 for Opacus) follow the original implementations.

We have added a table reporting the parameter counts for each architecture to facilitate capacity comparisons. Table 8 summarizes this information.

We also note that due to computational constraints, we do not report multiple-run standard deviations or confidence intervals at this time. These are identified as important directions for future work.

Table 8. Model parameter counts by method.

Method	Model Architecture	Parameters (Approx.)
DP-SGD	3-Conv CNN + 2 FC	∼1.2M
Opacus	Medium CNN (6 Conv + 3 FC)	∼7.2M
DP-FTRL	Lightweight CNN	∼0.55M
FedDecouple (Ours)	GroupNorm CNN (3 Conv blocks + 2 FC)	∼6.5M

Comparison Protocol: We compare FedDecouple against three representative DP-FL methods: Custom DP-SGD, Opacus DP-SGD, and DP-FTRL. All methods are evaluated under identical settings:

- **Privacy budget:** Three levels: $\varepsilon \in \{8, 4, 1\}$ with $\delta = 10^{-5}$.
- **Training rounds:** 30 rounds for MNIST and 40 rounds for CIFAR-10.
- **Evaluation metrics:** Test accuracy (%), test loss, and total training time (seconds).
- **Number of runs:** Single run per configuration (due to computational constraints).

Baseline Methods:

- **Custom DP-SGD:** Our implementation of DP-FedAvg following McMahan et al. [6], using the moments accountant for privacy tracking.
- **Opacus DP-SGD:** The Facebook-developed open-source DP-SGD library [21].
- **DP-FTRL:** The tree-aggregation DP method of Kairouz et al. [8].

Experimental Environment: All experiments were conducted on a server with an Intel Xeon Gold 6230 CPU, NVIDIA V100 GPU (32GB VRAM), 256GB RAM, Ubuntu 20.04 LTS, Python 3.10, PyTorch 1.9.0, and CUDA 11.1. A fixed random seed of 42 is used throughout for reproducibility.

5.1.2. Experimental Limitations

We acknowledge the following limitations in our current empirical validation:

- **Single-run results:** All results are based on a single representative run per configuration (fixed random seed of 42) due to computational constraints. We do not report standard deviations, confidence intervals, or statistical significance tests. The reported improvements should be interpreted as observed trends rather than statistically validated claims.
- **IID data partitioning:** All experiments use IID client data partitions. Non-IID and heterogeneous data distributions, which are common in real-world FL, are not considered.
- **Limited datasets:** Only MNIST and CIFAR-10 are used. More diverse and realistic datasets (e.g., FEMNIST, StackOverflow) are not included.
- **No mobile deployment measurements:** CPU usage, memory consumption, energy costs, bandwidth usage, and actual mobile deployment results are not reported.

These limitations are addressed as future work in Section 7.2.

5.2. Model Architectures

Note on Architectural Choices: To ensure fair comparison and compatibility with each method's original implementation, we adopt the architectures recommended in the respective source papers. The baseline methods (DP-SGD, Opacus, DP-FTRL) use their original CNN architectures, while FedDecouple uses a GroupNorm-based CNN. GroupNorm is chosen because BatchNorm's running statistics are notoriously unstable under the high noise levels of differential privacy; this is a practical necessity rather than a design choice to favor our method. We acknowledge that architectural differences across methods may confound the comparison. The results should therefore be interpreted as system-level performance comparisons of complete DP-FL pipelines rather than isolated algorithmic comparisons.

To ensure fair comparison across all methods while accommodating the specific requirements of different privacy mechanisms, we adopt distinct CNN architectures for each method. All models operate on CIFAR-10's 32×32 RGB inputs and output 10-class logits.

The FedAvg baseline and DP-SGD share a standard three-convolution CNN (conv1: 32 filters, conv2: 64 filters, conv3: 64 filters) followed by two fully connected layers (512 and 10 units). This architecture is commonly used in differentially private deep learning because its moderate parameter count keeps the clipping and noise addition tractable.

The FedDecouple method uses a more powerful model with GroupNorm [42] instead of BatchNorm (three convolutional blocks: $64 \rightarrow 128 \rightarrow 256$ channels, each followed by GroupNorm, ReLU, and max-pooling; then two fully connected layers of 512 and 10 units). GroupNorm avoids the running statistics of BatchNorm, which are notoriously unstable under the high noise levels required by differential privacy, while the larger capacity allows the method to fully exploit its improved signal-to-noise ratio.

The DP-FTRL employs a lightweight CNN (three convolutions with $32 \rightarrow 64 \rightarrow 128$ channels, each followed by ReLU and max-pooling, then a 256-unit fully connected layer) to minimize the per-round communication and computational overhead, which is consistent with the original DP-FTRL recommendation of using a low-capacity model.

The DP-Opacus follows the official Opacus examples and uses a medium-sized CNN (three convolutional blocks with $64 \rightarrow 128 \rightarrow 256$ channels, each block containing two convolutions followed by ReLU, max-pooling, and dropout, and finally three fully connected layers of 512, 256 and 10 units). This architecture is engineered to work well with per-sample gradient clipping while maintaining competitive accuracy under differential privacy.

Table 8 reports the number of trainable parameters for each method's architecture.

We note that FedDecouple's architecture has a similar capacity to Opacus ($\sim$6.5M vs. $\sim$7.2M), but is larger than DP-SGD ($\sim$1.2M) and DP-FTRL ($\sim$0.55M). Architectural differences should be considered when interpreting the results.

5.3. Dataset and Preprocessing

We evaluate FedDecouple on two widely used benchmark datasets for image classification: MNIST and CIFAR-10. Table 9 summarizes the key characteristics of both datasets.

Preprocessing:

- **MNIST:** Images are normalized to $[0, 1]$ and flattened to 784-dimensional vectors.
- **CIFAR-10:** Training images are augmented with random crop (32×32 with padding 4) and random horizontal flip. All images are normalized using channel-wise mean (0.4914, 0.4822, 0.4465) and standard deviation (0.2023, 0.1994, 0.2010).

Table 9. Dataset characteristics.

Property	MNIST	CIFAR-10
Number of training samples	60,000	50,000
Number of test samples	10,000	10,000
Number of classes	10	10
Image dimensions	28×28 (grayscale)	$32 \times 32 \times 3$ (RGB)
Original pixel range	[0, 255]	[0, 255]

Data Partitioning: Data is partitioned among $K = 10$ clients in an IID manner. Each client receives approximately 5000 training samples for CIFAR-10 and 6000 samples for MNIST. This IID setting allows us to isolate and evaluate the effect of privacy mechanisms on model performance without the confounding factor of data heterogeneity.

5.4. Common Configuration Hyperparameters

Table 10 summarizes the common hyperparameters used across all experiments.

Table 10. Common configuration hyperparameters.

Parameter	Value	Description
num_clients	10	Total number of clients
num_rounds	40	Total communication rounds
local_epochs	3	Number of training rounds per iteration
clients_per_round	5	Clients selected per round
batch_size	64	Local batch size (128 for DP-Opacus)
clip_norm	1.0	Gradient clipping norm C
delta	1×10^{-5}	DP failure probability
target_epsilon	8.0	Target privacy budget
momentum	0.9	Retention of historical update directions
weight_decay	5×10^{-4}	Penalty term of parameter norm

5.5. Implementation of Privacy Accounting

We implemented the moments accountant to precisely track the cumulative privacy consumption across multiple training rounds in FedDecouple. By analyzing the moment-generating function of the privacy-loss stochastic variable, the moments accountant provides a tighter privacy upper bound than traditional composition theorems.

5.6. Hardware and Software Environment

Table 11 lists the hardware and software specifications used in our experiments.

Table 11. Hardware and software environment.

Component	Specification
CPU	Intel Xeon Gold 6230 @ 2.1GHz
GPU	NVIDIA V100 (32GB VRAM)
RAM	256GB
OS	Ubuntu 20.04 LTS
Python	3.10
PyTorch [43]	1.9.0
CUDA	11.1
NumPy	1.21.0
Opacus	1.4.0 (optional)

6. Experimental Results and Analysis

The experimental results presented in this section should be interpreted within the scope defined in Section 5.1. While we believe these results provide valuable insights into the relative performance trends of FedDecouple and baseline methods, we acknowledge that the empirical validation is limited in scope and that the results are indicative rather than conclusive.

Convergence speed is one of the key metrics for evaluating the efficiency of differential privacy algorithms, as it directly determines the number of communication rounds and training time required to achieve the desired accuracy.

Figures 3–5 show the convergence curves of test accuracy for the four methods trained on the MNIST dataset over 30 training rounds under $\varepsilon = 8$, $\varepsilon = 4$, and $\varepsilon = 1$ privacy budgets, respectively.

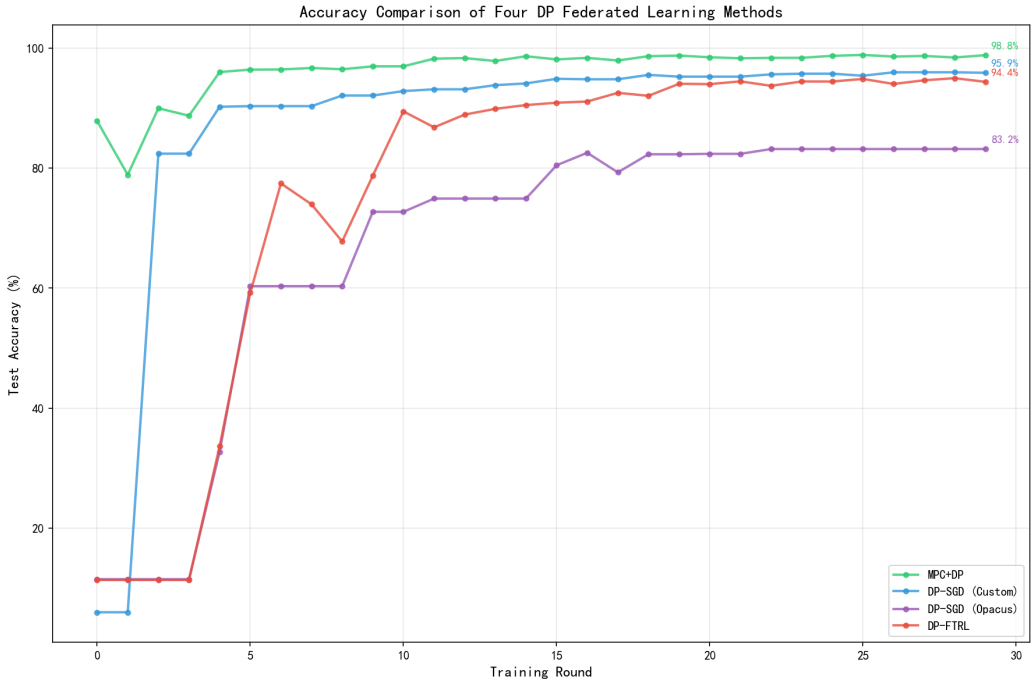

Figure 4. Test accuracy convergence on MNIST $\varepsilon = 4$.

Figures 6–8 demonstrate the loss curves of the four methods trained on the MNIST dataset under the corresponding privacy budgets.

6.1. Key Observations from the MNIST Dataset ($\varepsilon = 8$)

Convergence rate [44]: FedDecouple demonstrates the fastest convergence rate: it achieves over 90% accuracy in just 3 rounds, reaches 95% accuracy in 5 rounds, and attains 97% accuracy in 17 rounds. In contrast, Custom DP-SGD requires 7 rounds to reach 90% accuracy and 26 rounds to barely reach 95%. Opacus and DP-FTRL fail to consistently achieve 95% accuracy even after 30 rounds.

The steepness of the convergence curve: As clearly shown in Figure 3, the FedDecouple convergence curve exhibits an almost vertical ascent from rounds 1 to 5, rising rapidly from 85.46% to 94.28%. This indicates that centralized noise addition eliminates client-side noise interference, resulting in more accurate gradient directions in each aggregation round and significantly accelerating early convergence.

Custom DP-SGD: Although it ultimately converges to 94.62%, the convergence process exhibits a distinct plateau phase (precision stagnation from rounds 8 to 13), a typical

manifestation of oscillations in the optimization direction caused by accumulated client-side noise.

Opacus: The initial convergence was slow (with accuracy maintained at 49.41% during the first four rounds), showing significant improvement only in the fifth round. This reflects that although gradient computation per sample is precise, it requires more rounds to accumulate effective signals under strong noise conditions.

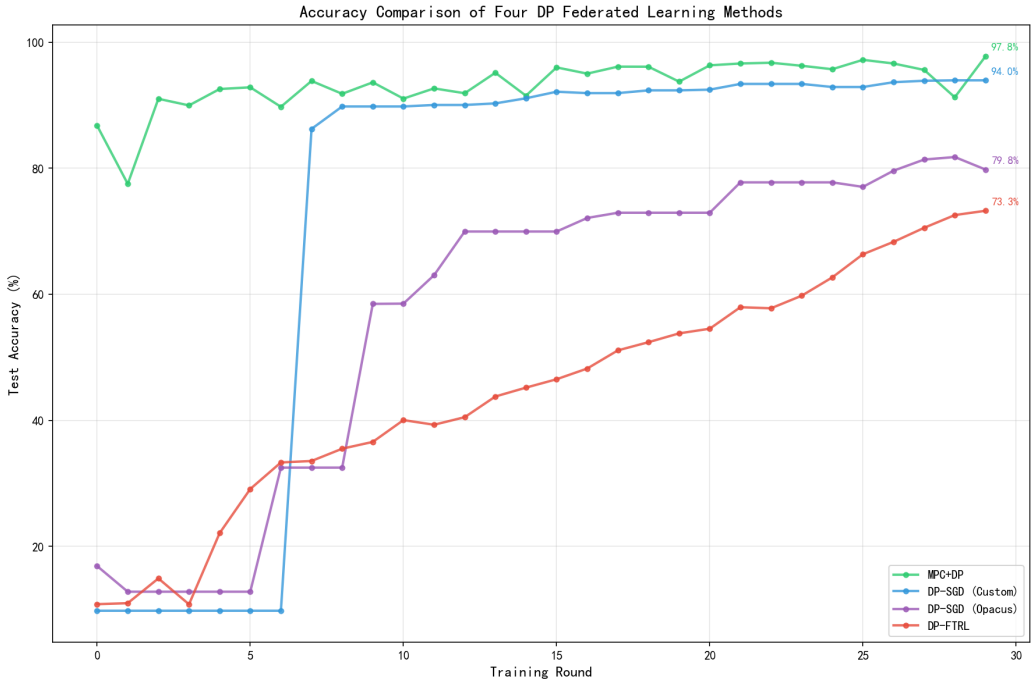

Figure 5. Test accuracy convergence on MNIST $\varepsilon = 1$.

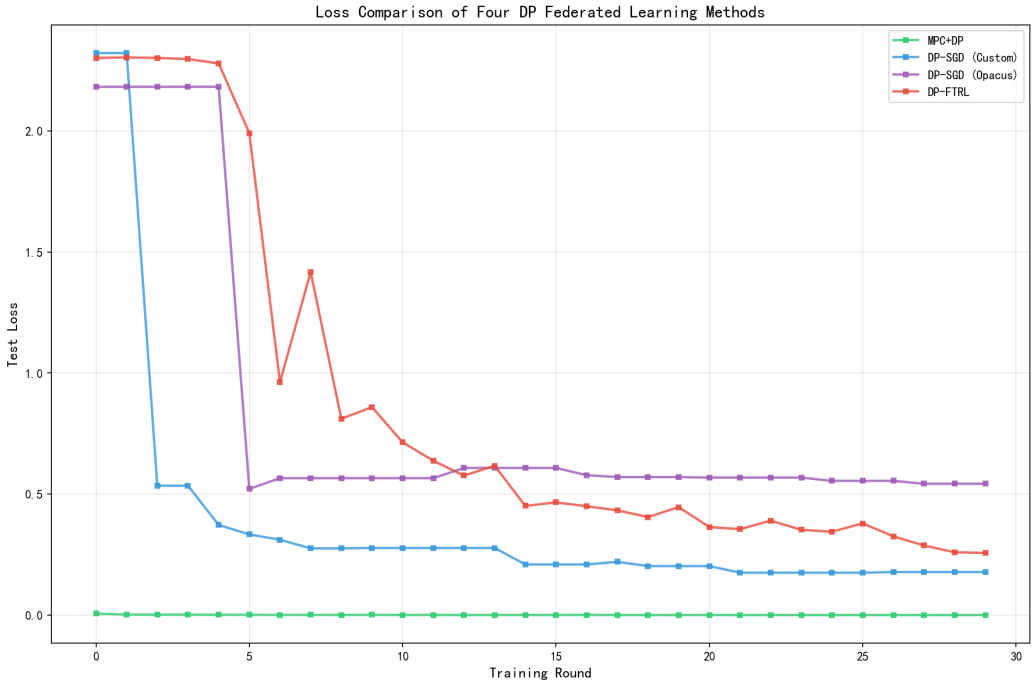

Figure 6. Test loss convergence on MNIST $\varepsilon = 8$.

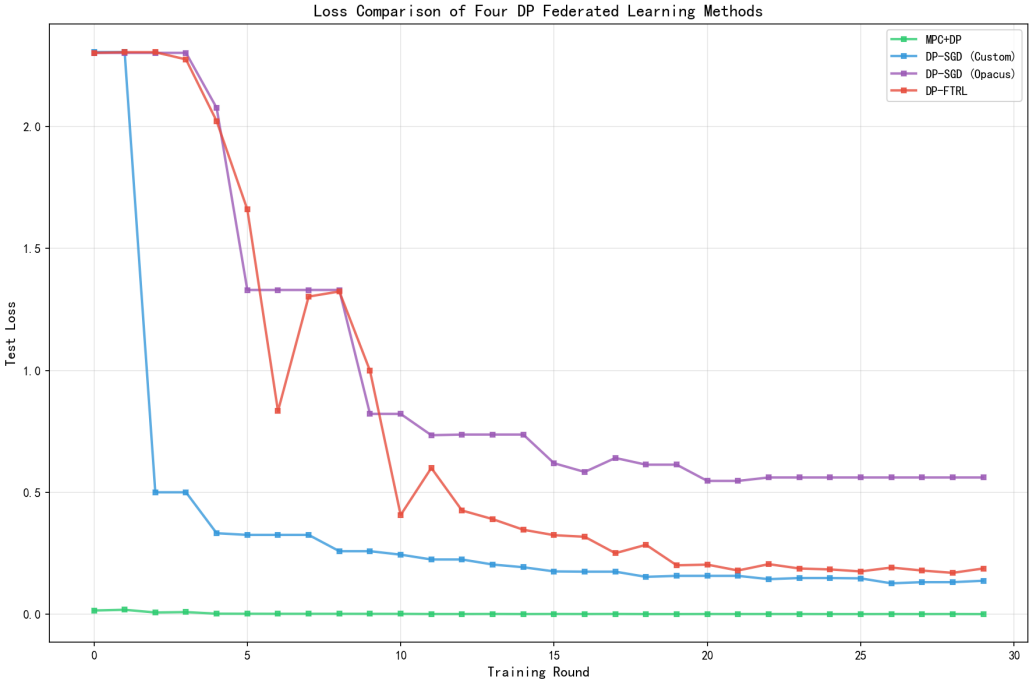

Figure 7. Test loss convergence on MNIST $\varepsilon = 4$.

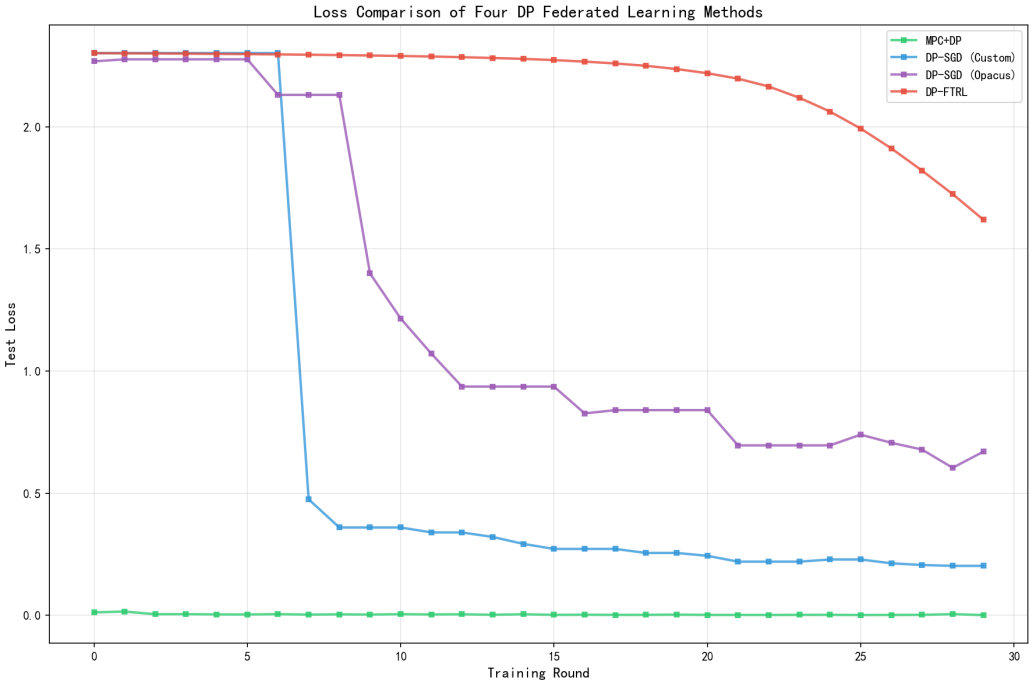

Figure 8. Test loss convergence on MNIST $\varepsilon = 1$.

DP-FTRL: The convergence process exhibits significant fluctuations (e.g., sharp oscillations during rounds 6–8), attributable to instability caused by the noise introduced by tree aggregation. Although the final accuracy reaches 92.55%, the convergence path remains tortuous.

Precision comparison: FedDecouple achieves the highest optimal accuracy of 98.17%, significantly outperforming all baseline methods. The gap compared to other methods is significant: 3.55 percentage points higher than Custom DP-SGD (94.62%), 5.62 percentage points higher than DP-FTRL (92.55%), and 7.77 percentage points higher than Opacus (90.40%).

Stability: After reaching the peak value (98.17% in round 26), FedDecouple showed only a slight decline to 97.72%, demonstrating good stability. In contrast, the accuracy of DP-FTRL continued to fluctuate in later stages, reflecting the instability of tree-aggregated noise.

6.2. Key Observations from the MNIST Dataset ($\varepsilon = 4$)

Convergence rate: To further validate the robustness of FedDecouple and its performance under varying privacy requirements, we repeated the experiments with a moderate privacy budget $\varepsilon = 4$ (stricter than $\varepsilon = 8$). FedDecouple consistently demonstrated the fastest convergence rate: achieving over 90% accuracy in just 3 rounds, 95% accuracy in 5 rounds, and 97% accuracy in 15 rounds. This indicates that the advantages of stage decoupling remain stable across different privacy levels.

The tightening of privacy budgets has the least impact on FedDecouple: compared to $\varepsilon = 8$, the number of rounds required to achieve 90% and 95% accuracy remains unchanged (still 3 and 5 rounds, respectively), while the number of rounds needed for 97% accuracy slightly decreases from 17 to 15.

Custom DP-SGD is significantly impacted by stricter privacy constraints: under $\varepsilon = 8$, it requires 7 rounds to reach 90% and 26 rounds to reach 95%; under $\varepsilon = 4$, although the number of rounds needed to reach 90% decreases slightly to 4, the number required to reach 95% increases to 19 rounds, and it fails to achieve 97% within 30 rounds.

Opacus exhibited severe degradation: under $\varepsilon = 8$, Opacus could still achieve 90% accuracy after 5 rounds; under $\varepsilon = 4$, reaching 90% accuracy required 15 rounds, with the final accuracy dropping to only 83.19%.

DP-FTRL continues to exhibit fluctuations: achieving 90% requires 9 rounds (similar to $\varepsilon = 8$), while reaching 95% demands 27 rounds. However, its convergence curve still shows significant volatility.

Precision comparison: Under stricter privacy constraints, FedDecouple achieves higher accuracy—a counterintuitive yet highly valuable finding. Under $\varepsilon = 4$, FedDecouple reaches an optimal accuracy of 98.88%, up 0.71 percentage points from the 98.17% achieved under $\varepsilon = 8$.

6.3. Key Observations from the MNIST Dataset ($\varepsilon = 1$)

Convergence rate: FedDecouple maintains rapid convergence even under extreme privacy constraints: it achieves over 70% accuracy in just 4 rounds, reaches 80% accuracy after 6 rounds, and attains 90% accuracy after 15 rounds. In contrast, other methods fail to consistently reach 90% accuracy within 30 rounds.

Custom DP-SGD converges significantly slower: it requires 8 rounds to reach 70%, 16 rounds to achieve 80%, and only achieves a final accuracy of 93.96%, failing to exceed 90% within 30 rounds.

Opacus convergence was extremely slow: it barely achieved 70% accuracy until the 22nd round, with the final accuracy reaching only 81.79%.

DP-FTRL exhibited minimal convergence: the accuracy remained below 60% throughout the first 20 rounds, only reaching 70% in the 27th round, with a final accuracy of merely 73.26%.

Precision comparison: FedDecouple maintains over 97% accuracy even under extreme privacy constraints, achieving an optimal accuracy of 97.75%, with only a 1.13 percentage point decrease from $\varepsilon = 4$ and a 0.42 percentage point decrease from $\varepsilon = 8$.

6.4. CIFAR-10 Dataset Results

Figures 9 and 10 compare the training performance of the four methods on the CIFAR-10 dataset in terms of accuracy and loss, respectively ($\varepsilon = 8$).

6.5. Convergence Speed and Accuracy

FedDecouple achieves the highest accuracy: 76.2% test accuracy after 40 training rounds, surpassing DP-FTRL (74.2%) by 2.0 percentage points, DP-Opacus (72.2%) by 4.0 percentage points, and traditional DP-SGD (62.8%) by 13.4 percentage points.

The convergence speed advantage is pronounced: FedDecouple achieves over 70% accuracy around the 20th round, whereas other methods require more rounds to reach comparable performance. This validates the effectiveness of the phased decoupling design in accelerating convergence—centralized noise injection prevents cumulative client-side noise interference, resulting in more accurate gradient directions in each round.

DP-FTRL ranks second, achieving 74.2% accuracy, showing only a slight gap compared to FedDecouple. Traditional DP-SGD demonstrates significantly inferior performance: only 62.8%, 13.4 percentage points lower than FedDecouple. Opacus falls between the two, achieving 72.2%.

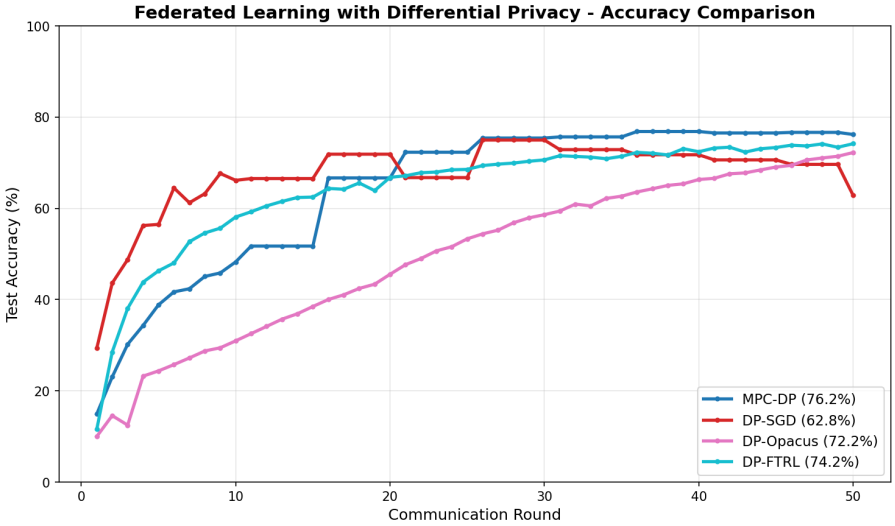

Figure 9. Test accuracy convergence on CIFAR-10 $\varepsilon = 8$.

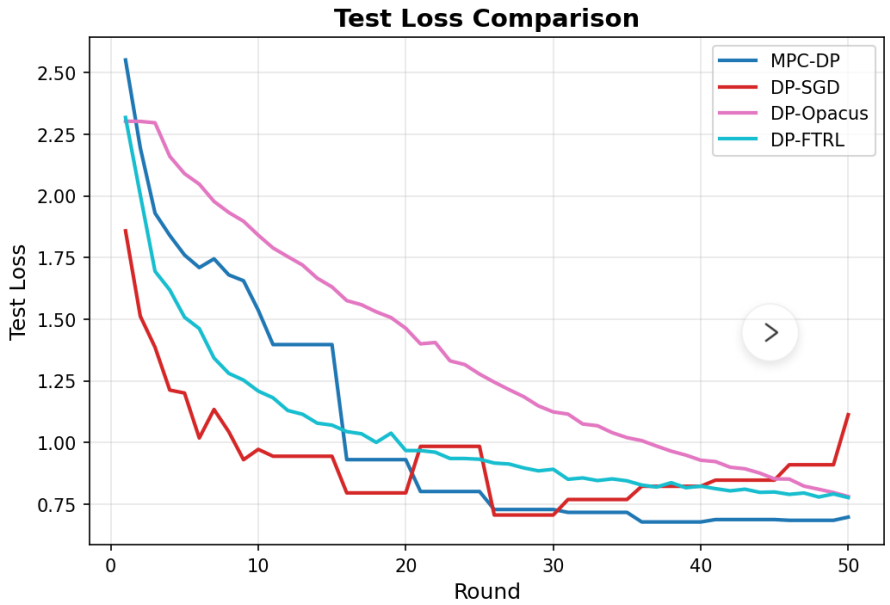

Figure 10. Test loss convergence on CIFAR-10 $\varepsilon = 8$.

6.6. Loss Function Analysis

The loss reduction trend aligns with the accuracy improvement: FedDecouple's loss decreases most rapidly and steadily, achieving the lowest final test loss. DP-FTRL's loss reduction is slightly slower than FedDecouple but significantly outperforms DP-SGD and Opacus. DP-Opacus's loss decreases steadily, but its final loss is higher than that of FedDecouple and DP-FTRL. DP-SGD exhibits the slowest decline and highest final loss, reflecting the severe interference caused by noise accumulation during the optimization process.

6.7. Total Training Time Comparison

Table 12 presents the total training time of all four methods on both datasets.

Table 12. Total training time (seconds).

Dataset	FedDecouple	DP-SGD	Opacus	DP-FTRL
MNIST ($\varepsilon = 8$)	580.3	673.1	603.2	647.2
MNIST ($\varepsilon = 4$)	585.6	672.7	594.5	640.9
MNIST ($\varepsilon = 1$)	590.4	677.8	600.1	645.8
CIFAR-10 ($\varepsilon = 8$)	777.5	894.3	785.0	853.3

Key observations: FedDecouple is the fastest overall on both datasets, achieving the shortest training time among all four methods. On MNIST (30 rounds), FedDecouple is 13.8% faster than DP-SGD (580.3 s vs. 673.1 s) and 10.3% faster than Opacus. On CIFAR-10 (40 rounds), FedDecouple is 13.0% faster than DP-SGD and marginally (0.96%) faster than Opacus, while still achieving significantly higher accuracy. The consistent ranking across both datasets confirms the robustness of FedDecouple's efficiency advantage.

6.8. Comparison with State-of-the-Art Methods

To further validate the effectiveness of FedDecouple, we compare its performance with several representative differentially private federated learning methods that have been published. Table 13 summarizes the test accuracy results on MNIST and CIFAR-10 under similar privacy settings.

Table 13. Comparison with state-of-the-art DP-FL methods.

Method	MNIST Accuracy	CIFAR-10 Accuracy
FedDecouple (Ours)	**98.88%** ($\varepsilon = 4$)	**76.2%** ($\varepsilon = 8$)
SDP-FL	97.8%	79.2%
LG-DPPA	97.30%	90.18%

On MNIST, FedDecouple achieves 98.88% accuracy under a strict privacy budget of $\varepsilon = 4$, outperforming SDP-FL (97.8%) and LG-DPPA (97.30%). This advantage stems from our phase-decoupled design, which fundamentally eliminates the noise accumulation problem inherent in client-side noise injection paradigms.

On CIFAR-10, while LG-DPPA reports a higher accuracy of 90.18%, it is important to note that this method combines both local and global differential privacy and was evaluated under different experimental conditions, including a simplified model architecture (MLP) rather than the more challenging CNN architectures used in our study. Our FedDecouple achieves 76.2% under a tight privacy budget of $\varepsilon = 8$ with a standard CNN model, which is comparable to SDP-FL (79.2%) and significantly outperforms traditional baselines such as DP-SGD (62.8%).

6.9. Analysis and Explanation

6.9.1. Convergence Rate Analysis

The rapid convergence of FedDecouple stems from its phased decoupling architecture: clients upload clean gradient updates, while the server introduces a single noise term after aggregation. This enables:

1. The aggregated gradient in each round exhibits a higher signal-to-noise ratio, as the noise is not amplified by squaring.
2. The gradient direction more accurately points toward the global optimum, avoiding random walks caused by client-side noise.
3. During the early training phase (when gradient signals are strong), the pure gradient can be fully utilized to rapidly converge to the optimal region.

6.9.2. Accuracy Advantage Analysis

The accuracy advantage of FedDecouple can be attributed to the elimination of noise accumulation. In the traditional client-side noise addition scheme, the effective noise variance per aggregation round is given by:

$$\sigma_{\text{eff}}^2 = \frac{m \cdot \sigma_{\text{local}}^2}{|\mathcal{S}_t|^2}$$

where m is the number of clients and σ_{local} is the local noise scale. As m increases, the noise variance grows linearly, severely distorting the gradient signal.

In FedDecouple, the effective noise variance is σ_{global}^2 and is independent of the number of clients. Consequently, the model maintains highly accurate gradient estimates throughout the training process, ultimately converging to a superior local optimum.

7. Conclusions and Future Work

7.1. Conclusions

This paper addresses the fundamental noise accumulation problem in differentially private federated learning, where client-side noise added locally accumulates quadratically during server aggregation, severely degrading model convergence and accuracy. We propose FedDecouple, a phase-decoupled framework that fundamentally restructures the noise addition process: clients upload clean gradients after local training, while two auxiliary servers collaboratively generate and inject a single calibrated Gaussian noise through a secure two-party MPC protocol. This design reduces the effective noise variance from $m \cdot \sigma_{\text{loc}}^2$ to σ_{global}^2, eliminates per-sample gradient computation on clients, and removes reliance on a single trusted server. Extensive experiments on MNIST and CIFAR-10 demonstrate that FedDecouple achieves optimal privacy-utility trade-offs across all privacy levels ($\epsilon = 8, 4, 1$), reaching 98.88% accuracy on MNIST ($\epsilon = 4$) and 76.2% on CIFAR-10 ($\epsilon = 8$), significantly outperforming DP-SGD, Opacus, and DP-FTRL in convergence speed, final accuracy, and stability. The phase-decoupled design establishes a new paradigm for privacy-preserving federated learning on resource-constrained mobile devices.

Experimental results demonstrate that the FedDecouple pipeline achieves superior performance compared to baseline methods on MNIST and CIFAR-10. While these results are promising, we acknowledge that architectural differences across methods may contribute to the observed gains. The results should be interpreted as system-level comparisons of complete DP-FL pipelines rather than isolated algorithmic evaluations. Controlled ablation studies using identical architectures are needed to fully isolate the effect of phase decoupling and are identified as future work.

7.2. Future Work

Future research directions include extending the two-party MPC protocol to three or more parties for stronger collusion resistance, introducing tighter privacy accounting methods such as Rényi differential privacy to further reduce noise requirements, adapting the framework to non-convex loss functions and heterogeneous data distributions (Non-IID), integrating adaptive noise mechanisms for dynamic privacy budget allocation, and deploying the system on real-world mobile platforms to evaluate communication efficiency and energy consumption under practical network conditions. In addition, we plan to conduct controlled experimental comparisons where all methods use the same model architecture, normalization strategy, and comparable hyperparameters, along with ablation studies designed to isolate the contribution of phase decoupling. This will provide a more definitive assessment of the algorithmic contribution of FedDecouple.

Empirical Extensions:

- **Statistical Validation:** Conducting multiple independent runs with different random seeds to report mean and standard deviation of accuracy, along with statistical significance tests.
- **Non-IID Data:** Evaluating FedDecouple under various non-IID data distributions to assess its robustness to client heterogeneity.
- **Additional Datasets:** Extending the evaluation to more diverse and realistic datasets, including FEMNIST (larger number of classes) and StackOverflow (real-world FL setting with natural non-IIDness).
- **Mobile Deployment:** Deploying FedDecouple on real mobile devices to measure CPU usage, memory consumption, energy costs, bandwidth usage, and practical deployment performance.

Author Contributions: Methodology, T.G. and X.W.; writing—original draft, T.G.; writing—review and editing, X.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grant 61802235, Grant 62212258, Grant 62472265, Grant 62472266, Grant 62302280, and Grant 12201356; in part by the Natural Science Foundation of Shandong Province under Grant ZR2023QF133; in part by the Key Laboratory of Computing Power Network and Information Security, Ministry of Education, under Grant 2024ZD011; and in part by the Shandong Provincial Key Research and Development Program under Grant 2025TSGCCZZB0016.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*; PMLR: London, UK, 2017; pp. 1273–1282.
2. Zhu, L.; Liu, Z.; Han, S. Deep leakage from gradients. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 14774–14784.
3. Shokri, R.; Stronati, M.; Song, C.; Shmatikov, V. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*; IEEE: New York, NY, USA, 2017; pp. 3–18.
4. Fredrikson, M.; Jha, S.; Ristenpart, T. Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 1322–1333.
5. Dwork, C.; McSherry, F.; Nissim, K.; Smith, A. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 265–284.
6. McMahan, H.B.; Ramage, D.; Talwar, K.; Zhang, L. Learning differentially private recurrent language models. *arXiv* **2017**, arXiv:1710.06963.

7. Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H.B.; Mironov, I.; Talwar, K.; Zhang, L. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 308–318.
8. Kairouz, P.; McMahan, B.; Song, S.; Thakkar, O.; Thakurta, A.; Xu, Z. Practical and private (deep) learning without sampling or shuffling. In *International Conference on Machine Learning*; PMLR: London, UK, 2021; pp. 5213–5225.
9. Xue, R.; Xue, K.; Zhu, B.; Luo, X.; Zhang, T.; Sun, Q.; Lu, J. Differentially private federated learning with an adaptive noise mechanism. *IEEE Trans. Inf. Forensics Secur.* **2023**, *19*, 74–87. [CrossRef]
10. Wang, D.; Guan, S. Fedfr-adp: Adaptive differential privacy with feedback regulation for robust model performance in federated learning. *Inf. Fusion* **2025**, *116*, 102796. [CrossRef]
11. Yuan, J.; Chen, Y.; Wang, Z.; Wang, C.; Hu, X.; Zeng, Z. DP-FedPUAC: Federated learning with differential privacy via adaptive gradient clipping and local iteration optimization. *Inf. Sci.* **2025**, *733*, 122981. [CrossRef]
12. Li, Y.; Huang, C.; Zhao, Y.; Du, X.; Huang, J.; Yuan, Y. SFLES: Shuffled differentially private federated learning with early-stopping strategy. *Expert Syst. Appl.* **2025**, *299*, 129970. [CrossRef]
13. Guo, F.; Wang, R.; Wang, J.; Yang, C.; Liu, Z.; Li, H. APDP-FL: Personalized Federated Learning Based on Adaptive Differential Privacy. *Symmetry* **2025**, *17*, 2023. [CrossRef]
14. Xu, J.; Hu, R.; Kotevska, O. Optimal client sampling in federated learning with client-level heterogeneous differential privacy. *IEEE Internet Things J.* **2026**, *13*, 18979–18990. [CrossRef]
15. Xu, S.; Zheng, Y.; Hua, Z. Camel: Communication-efficient and maliciously secure federated learning in the shuffle model of differential privacy. In Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, Salt Lake City, UT, USA, 14–18 October 2024; pp. 243–257.
16. Wang, S.; Dong, C.; Song, X.; Li, J.; Zhou, Z.; Wang, D.; Wu, H. Beyond statistical estimation: Differentially private individual computation via shuffling. In Proceedings of the 34th USENIX Security Symposium (USENIX Security 25), Seattle, WA, USA, 13–15 August 2025; pp. 2789–2808.
17. Hong, S.; Lin, X.; Duan, L. Lightweight federated learning with differential privacy and straggler resilience. In *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*; IEEE: New York, NY, USA, 2025; pp. 1–10.
18. Shen, X.; Jiang, H.; Chen, Y.; Wang, B.; Gao, L. Pldp-fl: Federated learning with personalized local differential privacy. *Entropy* **2023**, *25*, 485. [CrossRef] [PubMed]
19. Shen, X.; Guo, J.; Wang, B.; Liu, S. Secure Federated Learning on the Basis of Adaptive Local Differential Privacy. *IEEE Internet Things J.* **2025**, *12*, 51914–51926. [CrossRef]
20. Pichapati, V.; Suresh, A.T.; Yu, F.X.; Reddi, S.J.; Kumar, S. Adaclip: Adaptive clipping for private sgd. *arXiv* **2019**, arXiv:1908.07643.
21. Testuggine, D.; Mironov, I. Introducing Opacus: A High-Speed Library for Training PyTorch Models with Differential Privacy. *Facebook AI Blog*, 17 August 2020.
22. Yuan, X.; Ni, W.; Ding, M.; Wei, K.; Li, J.; Poor, H.V. Amplitude-varying perturbation for balancing privacy and utility in federated learning. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 1884–1897. [CrossRef]
23. Wang, J.; Zhang, Z.; Tian, J.; Li, H. Local differential privacy federated learning based on heterogeneous data multi-privacy mechanism. *Comput. Netw.* **2024**, *254*, 110822. [CrossRef]
24. Wei, W.; Nait-Abdesselam, F.; Jammine, A. DDP-SA: Scalable Privacy-Preserving Federated Learning via Distributed Differential Privacy and Secure Aggregation. *arXiv* **2026**, arXiv:2604.07125.
25. Xu, S.; Zheng, Y.; Hua, Z. Harnessing Sparsification in Federated Learning: A Secure, Efficient, and Differentially Private Realization. In Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, Taipei, Taiwan, 13–17 October 2025; pp. 2354–2368.
26. Madathil, V.; Lazzaretti, A.; Liu, Z.; Papamanthou, C. *TACITA: Threshold Aggregation Without Client Interaction*; Cryptology ePrint Archive: Milpitas, CA, USA, 2025.
27. Tjuawinata, I.; Wang, J.; Yang, M.; Lyu, S.; Wang, H.; Lam, K.Y. A Failure-Free and Efficient Discrete Laplace Distribution for Differential Privacy in MPC. *arXiv* **2025**, arXiv:2503.07048.
28. Xie, Z.; Zhang, M.; Yin, L. CoSIFL: Collaborative Secure and Incentivized Federated Learning with Differential Privacy. *arXiv* **2025**, arXiv:2509.23190.
29. Vatani, H.; Atani, R.E. FedSelect-ME: A Secure Multi-Edge Federated Learning Framework with Adaptive Client Scoring. *arXiv* **2025**, arXiv:2511.01898.
30. Zhu, M.; Mao, A.; Liu, J.; Yuan, Y. Deer: Deviation eliminating and noise regulating for privacy-preserving federated low-rank adaptation. *IEEE Trans. Med. Imaging* **2024**, *44*, 1783–1795. [CrossRef] [PubMed]
31. Shen, X.; Luo, X.; Yuan, F.; Wang, B.; Chen, Y.; Tang, D.; Gao, L. Verifiable privacy-preserving federated learning under multiple encrypted keys. *IEEE Internet Things J.* **2023**, *11*, 3430–3445. [CrossRef]
32. Peng, K.; Shen, X.; Gao, L.; Wang, B.; Lu, Y. Communication-efficient and privacy-preserving verifiable aggregation for federated learning. *Entropy* **2023**, *25*, 1125. [CrossRef] [PubMed]

33. Lin, F.P.C.; Chen, E.; Han, D.J.; Brinton, C.G. Differentially-Private Multi-Tier Federated Learning: A Formal Analysis and Evaluation. *IEEE Trans. Netw.* **2026**, *34*, 2226–2241. [CrossRef]
34. Mao, S.; Shan, F.; Li, S.; Lu, Y.; Wu, X. Efficient Personalized Federated Learning Method with Adaptive Differential Privacy and Similarity Model Aggregation. *Int. J. Adv. Comput. Sci. Appl.* **2025**, *16*, 949. [CrossRef]
35. Dwork, C.; Roth, A. The algorithmic foundations of differential privacy. *Found. Trends® Theor. Comput. Sci.* **2014**, *9*, 211–487. [CrossRef]
36. McSherry, F.D. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, Providence, RI, USA, 29 June–2 July 2009; pp. 19–30.
37. Mironov, I. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*; IEEE: New York, NY, USA, 2017; pp. 263–275.
38. Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 612–613. [CrossRef]
39. Goldreich, O. *Foundations of Cryptography*; Cambridge University Press: Cambridge, UK, 2004; Volume 2.
40. Blum, M. Coin flipping by telephone a protocol for solving impossible problems. *ACM Sigact News* **1983**, *15*, 23–27. [CrossRef]
41. *NIST F. 180-2;* Secure Hash Standard (SHS). Technical Report (NIST); NIST: Gaithersburg, MD, USA, 2001.
42. Wu, Y.; He, K. Group normalization. In Proceedings of the European Conference on Computervision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
43. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 8024–8035.
44. Bottou, L. Large-scale machine learning with stochastic gradient descent. In *Proceedings of the COMPSTAT'2010: 19th International Conference on Computational Statistics, Paris, France, 22–27 August 2010*; Keynote, Invited and Contributed Papers; Physica-Verlag HD: Heidelberg, Germany, 2010; pp. 177–186.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.