FedSAN: Replayable DP Accounting for Asynchronous Federated Instruction Tuning under Out-of-Order Arrivals

Anonymous ACL submission

Abstract

Asynchronous federated learning (FL) is a practical choice for instruction tuning large language models (LLMs), but out-of-order (OOO) client arrivals break deployed differential privacy (DP) accounting. We show that arrival-order accounting can mis-align snapshot provenance with the released-round execution trace, yielding a DP ledger that is not replayable from execution logs and can fail under strict privacy budgets. We propose FedSAN, which uses authenticated snapshot provenance and deterministic reconstructed-order DP accounting (R-RDP) to produce an auditable ledger under OOO delivery. FedSAN can further compress privatized LoRA uploads via post-DP random projection without changing the privacy guarantee. On Llama-3-8B LoRA tuning for PubMedQA and FiQA, FedSAN achieves near-synchronous accuracy, a 2.6× wall-clock speedup over synchronized training, and small audit deviation under severe disorder.

1 Introduction

Federated learning (FL) is increasingly used to instruction-tune large language models (LLMs) across organizations in privacy-sensitive domains such as healthcare and finance, where raw text cannot be centralized (Ning et al., 2025). At LLM scale, however, synchronization latency becomes a major systems bottleneck: heterogeneous compute and network conditions create stragglers, leaving accelerators idle under synchronous rounds (McMahan et al., 2016). Asynchronous FL is therefore an attractive engineering choice, continuously incorporating delayed and out-of-order (OOO) client arrivals to improve throughput (Alasmari et al., 2025).

Problem. Privacy-sensitive FL typically relies on differential privacy (DP) with clipping/noise and a stateful accountant (e.g., RDP) that composes privacy over a sequence of subsampled mechanisms (Shi et al., 2025; Abadi et al., 2016). In asynchronous training, each client update is generated from a server-issued snapshot but may arrive OOO (Li et al., 2025b; Xie et al., 2025). The prevailing implementation—arrival-order accounting—advances the accountant by arrival index (Zhao et al., 2025). Under OOO delivery, this can misattribute snapshot-tied events: a stale update generated from an old snapshot may be charged as a fresh event, so the reported ledger no longer matches the realized released-round execution trace. This mismatch is invisible unless the ledger is replayed from logs, yet it becomes critical under strict privacy budgets ($\varepsilon_{\text{target}} \leq 2$), where even small accounting errors can lead to silent budget violations or substantial utility loss when corrected. We quantify this issue by the audit deviation

$$\Delta\varepsilon_{\text{audit}} := \left|\varepsilon_{\text{replay}} - \varepsilon_{\text{ledger}}\right|,$$

where $\varepsilon_{\text{ledger}}$ is the privacy loss reported online and $\varepsilon_{\text{replay}}$ is recomputed by replaying the logged released-round sequence under the declared accountant. A run fails if either $\varepsilon_{\text{replay}} > \varepsilon_{\text{target}}$ or $\Delta\varepsilon_{\text{audit}} > \tau$.

Approach. We propose FEDSAN, an asynchronous federated instruction-tuning framework that makes DP accounting *provenance-aligned* under OOO delivery. Each client upload carries authenticated snapshot provenance; the server buffers arrivals and deterministically reconstructs a snapshot-consistent released-round sequence, then performs reconstructed-order accounting (R-RDP) on that sequence. This yields a replayable ledger that an auditor can reconstruct from immutable logs. To reduce LLM-scale uplink cost, FEDSAN optionally applies random projection *after* the DP mechanism; by post-processing, this changes only payload size and not the privacy guarantee.

Contributions.

- **Audit failure mode under OOO.** We show that arrival-indexed DP accounting can mis-align snapshot provenance with the released-round execution trace, yielding non-replayable ledgers and degraded utility under strict privacy budgets.
- **Replayable accounting for asynchronous LLM tuning.** We introduce FEDSAN, which reconstructs a snapshot-consistent released-round sequence from authenticated provenance and performs reconstructed-order accounting (R-RDP), producing an auditable DP ledger under asynchronous execution.
- **LLM evidence under strict privacy.** On Llama-3-8B LoRA tuning for PubMedQA and FiQA with $(\varepsilon{=}2, \delta{=}10^{-5})$, FEDSAN achieves near-synchronous accuracy, a $2.6\times$ wall-clock speedup over synchronized training, and a small audit deviation under severe disorder.

2 Related Work

Federated PEFT and DP adaptation for LLMs. Recent federated learning research has moved from small models to adapting foundation models on decentralized text, including federated instruction tuning and parameter-efficient fine-tuning (PEFT) with LoRA/adapters(Liu et al., 2025; Qin et al., 2025; Gao et al., 2025). A related line of work studies *differentially private* federated adaptation for large models, focusing on clipping/noise design, privacy–utility trade-offs, and PEFT-specific privatization strategies(Liu et al., 2025; Zhu et al., 2024; Huang et al., 2025; Wang et al., 2025b). These methods address how to privatize federated LLM adaptation itself, but they typically assume synchronous or otherwise round-consistent execution, where the training log already induces a well-defined sequence of server updates. Our contribution is orthogonal: we study whether the *accounting ledger remains replayable and auditable* when asynchronous LoRA updates are generated from stale snapshots and arrive out of order.

DP accounting under asynchronous federated execution. DP in FL is commonly realized with clipping and Gaussian noise, with privacy tracked by stateful accountants such as RDP under a prescribed sampling model(Koskela and Mohammadi, 2025; Wang et al., 2025a). These accountants compose privacy over a *logical event stream*, not merely over raw message arrivals. Under asynchronous execution, however, many implementations effectively advance the accountant in arrival order(Zhao et al., 2025; Ghiasvand et al., 2025; Li et al., 2025c). The key gap is that, under OOO delivery, the arrival stream can diverge from the released-round execution trace implied by snapshot provenance, making the resulting ledger non-replayable from execution logs(Price, 2025; Ma et al., 2025). Our method addresses this accounting-integrity problem by reconstructing a deterministic, snapshot-consistent released-round stream for replayable DP accounting.

Auditable and trustworthy federated learning. Prior work on trustworthy FL uses mechanisms such as Trusted Execution Environments (TEEs) and cryptographic proofs to attest that clients executed prescribed computations or produced valid updates(Li et al., 2025a; Xing et al., 2025; Guan et al., 2025). These methods primarily verify *computation integrity*, which is complementary to our goal. FedSAN instead targets *accounting integrity*: whether the server charged privacy against the same logical released rounds that were actually applied. By combining authenticated provenance, append-only logging, and deterministic replay, our approach yields a verifier-replayable DP ledger under OOO arrivals and is naturally composable with TEE/ZK-style computation attestation(Chen et al., 2025).

3 Method

We present FEDSAN, an asynchronous federated instruction-tuning framework that yields a *replayable* and *auditable* DP ledger under out-of-order (OOO) client arrivals. The key point is that DP accounting must follow the *released training rounds that actually modify the global model*, rather than the raw arrival stream of client messages.

3.1 Preliminaries: Private Federated Instruction Tuning

We consider K clients. Client i holds a private instruction dataset $\mathcal{D}_i = \{(x, y)\}$. The server keeps the LLM backbone Θ_{frozen} frozen and optimizes only LoRA adapters and the task head, denoted by $\theta \in \mathbb{R}^{d'}$ with $d' \ll d$. The local objective of client

i is

$$\ell_i(\theta) := \mathbb{E}_{(x,y)\sim\mathcal{D}_i}\Big[-\log P\big(y \mid x; \Theta_{\text{frozen}}, \theta\big)\Big], \tag{1}$$

where $P(\cdot \mid x; \Theta_{\text{frozen}}, \theta)$ is the predictive distribution of the frozen backbone and trainable adapters/head. The global objective is

$$\min_{\theta} \ \mathcal{L}(\theta) = \sum_{i=1}^{K} p_i \, \ell_i(\theta), \tag{2}$$

where p_i is a client weight.

Training proceeds in logical rounds indexed by s. At round s, the server samples $\mathcal{S}_s$ with rate $q = |\mathcal{S}_s|/K$ and broadcasts snapshot (θ_s, s). Each selected client $i \in \mathcal{S}_s$ computes a local update $\Delta_{i\to s}$ and applies client-side clipping and Gaussian perturbation:

$$\widetilde{\Delta}_{i\to s} = \mathrm{Clip}(\Delta_{i\to s}, C) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I}), \tag{3}$$

where C is the ℓ_2 clipping threshold and σ is the Gaussian noise multiplier.

Because training is asynchronous, privatized uploads associated with snapshot s may arrive late, out of order, or never be used. Let $\mathcal{R}_s \subseteq \mathcal{S}_s$ denote the subset of uploads that are actually *released* for snapshot s. The server forms the released-round aggregate

$$\bar{\Delta}_s = \frac{1}{|\mathcal{R}_s|} \sum_{i\in\mathcal{R}_s} \widetilde{\Delta}_{i\to s}, \tag{4}$$

and applies it once to the model. **In this paper, one DP event is one released round.** Thus, the accountant advances exactly once when a released-round aggregate is applied. Individual uploads are never charged as standalone events; buffered, stale, timed-out, or otherwise unreleased uploads are logged but do not contribute separate DP events.

3.2 Failure Mode: Arrival-Indexed Accounting Is Not Auditable under OOO

The accountant should consume the ordered sequence of *released rounds actually applied to the model*, not the raw network arrival stream. Under synchronous execution these two orders coincide, but under asynchronous execution they can diverge because uploads are generated from server-issued snapshots and may arrive OOO.

A common asynchronous implementation advances the accountant whenever an upload arrives, effectively composing privacy in arrival order:

$$\varepsilon_{\text{arr}}(T) = \mathrm{Compose}\big(\mathcal{A}_{\pi(1)}, \ldots, \mathcal{A}_{\pi(T)}\big), \tag{5}$$

where $\pi(\cdot)$ is the arrival permutation. Let

$$\Pi_{\text{rel}} = (s_1, \ldots, s_m)$$

denote the released-round sequence actually applied by the server, and let

$$\Pi_{\text{arr}} = (a_1, \ldots, a_T)$$

denote the raw arrival sequence. Under OOO delivery, Π_{arr} and Π_{rel} are generally different objects. If the accountant consumes Π_{arr} while model updates follow Π_{rel}, the resulting ledger is tied to the wrong event stream and is not replayable from the executed training trace.

Toy example. Consider two logical rounds, $s = 10$ and $s = 11$. Suppose clients A, B are selected from snapshot 10 and client C from snapshot 11. Due to asynchronous delays, the server receives

$$C(11) \ \to \ A(10) \ \to \ B(10).$$

An arrival-indexed implementation charges the order

$$11 \to 10,$$

whereas a snapshot-consistent released-round trace may be

$$10 \to 11.$$

Hence, the arrival-indexed ledger records a different event sequence from the one that actually governs model updates. The issue is not that DP composition itself fails, but that the accountant has consumed the wrong event stream.

An auditable privacy ledger must be reproducible from immutable execution logs and the public accounting configuration. Let $\varepsilon_{\text{ledger}}$ be the privacy loss reported online and $\varepsilon_{\text{replay}}$ the privacy loss recomputed offline by replaying the released-round sequence under the same public parameters (q, C, σ, δ) and the same accountant. We define

$$\Delta\varepsilon_{\text{audit}} := \big|\varepsilon_{\text{replay}} - \varepsilon_{\text{ledger}}\big|. \tag{6}$$

A run is marked FAIL if either

$$\varepsilon_{\text{replay}} > \varepsilon_{\text{target}} \quad \text{or} \quad \Delta\varepsilon_{\text{audit}} > \tau.$$

Our audit assumes an append-only log that records, for each upload, authenticated provenance, a payload hash, and the deterministic server disposition (buffer/release/drop), so that a verifier can reconstruct the released-round sequence and replay the accountant.

3.3 FedSAN: Reconstructed-Order Accounting with Projected Updates

FEDSAN closes the auditability gap with two decoupled components: (i) provenance-driven reconstructed-order accounting (R-RDP), which reconstructs the released-round sequence used for DP accounting, and (ii) optional random projection of already-privatized LoRA uploads, which reduces uplink communication. Accounting correctness depends only on provenance and deterministic release/drop logic, whereas projection is pure DP post-processing.

3.3.1 R-RDP: Provenance-Driven Reconstruction of Released Rounds

When a client downloads the current model snapshot, the server attaches a monotone snapshot identifier s and an authentication tag

$$\tau = \mathrm{Auth}(s, \mathtt{cid}, \mathtt{ctr}),$$

and the client later uploads

$$(u_i, s, \tau, \mathtt{cid}, \mathtt{ctr}),$$

where u_i is either the raw privatized update or a projected sketch. Authenticated provenance prevents relabeling and provides the metadata needed for deterministic audit replay.

The server maintains a per-round buffer $\mathcal{B}[\cdot]$, a pointer $\hat{s}$ to the latest snapshot whose fate has been decided, and an accountant $\mathcal{A}$. Upon receiving an authenticated upload tagged with snapshot s, the server inserts it into $\mathcal{B}[s]$ unless it is already stale or outside the configured reordering window. It then repeatedly considers the *next undecided snapshot* $\hat{s}+1$. If the release condition is met, the server forms one released-round aggregate from $\mathcal{B}[\hat{s}+1]$, logs the release, advances the accountant exactly once, applies the round update, and sets $\hat{s} \leftarrow \hat{s}+1$. If the release condition is not met within a fixed public timeout, the server deterministically marks snapshot $\hat{s}+1$ as dropped, logs the drop, advances $\hat{s}$, and does *not* advance the accountant. Thus, a snapshot contributes one DP event if and only if it is released as a round and applied to the model.

Replayability follows from authenticated provenance, deterministic public release/drop rules, and append-only execution logs. Given these ingredients, a third party can reconstruct the released-round sequence

$$\Pi^{\star}_{\mathrm{rel}} = (s^{\star}_1, s^{\star}_2, \dots)$$

Algorithm 1 FEDSAN server pipeline

Require: $(\varepsilon_{\text{target}}, \delta)$, DP parameters (q, C, σ), step size η, accepted uploads per round m, reorder window W, timeout policy TIMEOUT

Initialize: $\theta \leftarrow \theta_0$, $\hat{s} \leftarrow 0$, $\mathcal{B}[\cdot] \leftarrow \emptyset$, $\mathcal{A} \leftarrow \textsc{Init}(\delta)$
Broadcast: public seed defining projection operator Φ if projection is enabled
while training is active **do**
  Receive $(u_{i\to s}, s, \tau, \mathtt{cid}_i, \mathtt{ctr})$
  if $\textsc{VerifyAuth}(s, \tau, \mathtt{cid}_i, \mathtt{ctr})$ fails **then**
    Log invalid upload; continue
  end if
  if $s \le \hat{s}$ or $s > \hat{s} + W$ **then**
    Log stale/out-of-window upload; continue
  end if
  Insert $u_{i\to s}$ into $\mathcal{B}[s]$ and log arrival metadata
  while true do
    $s_{\text{next}} \leftarrow \hat{s} + 1$
    if $\textsc{ReadyToRelease}(\mathcal{B}[s_{\text{next}}], m)$ **then**
      $\bar{u}_{s_{\text{next}}} \leftarrow \textsc{Aggregate}(\mathcal{B}[s_{\text{next}}])$
      Log $\textsc{Release}(s_{\text{next}})$
      $\mathcal{A} \leftarrow \textsc{Accumulate}(\mathcal{A}; q, C, \sigma)$
      $\widehat{\Delta}_{s_{\text{next}}} \leftarrow \textsc{Decode}(\bar{u}_{s_{\text{next}}}; \Phi)$
      $\theta \leftarrow \theta + \eta\, \widehat{\Delta}_{s_{\text{next}}}$; $\mathcal{B}[s_{\text{next}}] \leftarrow \emptyset$; $\hat{s} \leftarrow s_{\text{next}}$
      if $\varepsilon_{\text{effective}}(\mathcal{A}) \ge \varepsilon_{\text{target}}$ **then**
        stop releasing further rounds
      end if
    else if $\textsc{Timeout}(s_{\text{next}}, \mathcal{B}, \text{clock})$ **then**
      Log $\textsc{Drop}(s_{\text{next}})$; $\mathcal{B}[s_{\text{next}}] \leftarrow \emptyset$; $\hat{s} \leftarrow s_{\text{next}}$
    else
      break
    end if
  end while
end while

and reproduce the accountant state under the public parameters (q, C, σ, δ).

3.3.2 Projected Privatized LoRA Updates

Let $\widetilde{\Delta}_{i\to s} \in \mathbb{R}^{d'}$ be the already-privatized client update from Eq. (3). To reduce uplink communication, the client may transmit a low-dimensional sketch

$$z_{i\to s} = \Phi\, \widetilde{\Delta}_{i\to s} \in \mathbb{R}^{m_{\text{proj}}}, \qquad m_{\text{proj}} \ll d', \quad (7)$$

where Φ is a random projection matrix generated from a public server seed. When snapshot s is released, the server decodes the aggregate if projection is enabled:

$$\widehat{\Delta}_s = \textsc{Decode}(\bar{u}_s; \Phi),$$

with DECODE equal to the identity map when projection is disabled. Because projection and decoding are applied strictly after the client-side DP mechanism, they are post-processing and do not affect the privacy guarantee or the accountant inputs.

4 Experiments

We evaluate FEDSAN on privacy-sensitive NLP tasks with LLM-based federated LoRA fine-tuning

in biomedical question answering and financial sentiment analysis. Our experiments address two questions: (1) under a strict privacy budget ($\varepsilon \leq 2$) and severe out-of-order (OOO) arrivals, can FEDSAN preserve both task utility and a replayable DP ledger? (2) can random projection reduce LLM-scale uplink cost without changing the accounting protocol or audit outcome?

4.1 Experimental Setup

Tasks, model, and federated setting. We evaluate on two privacy-sensitive NLP tasks: PubMedQA (biomedical 3-way QA with labels `Yes`/`No`/`Maybe`) and FiQA (financial sentiment with labels `Positive`/`Negative`/`Neutral`). Both tasks are cast as instruction-following generation, and we report label EM/Accuracy together with format adherence. We use Llama-3-8B with LoRA, updating only the adapters and task head while keeping the backbone frozen. Unless otherwise stated, LoRA uses rank r=16 and α=32. We simulate cross-silo FL with K=200 clients, sample m=10 clients per logical round (q=0.05), and run for up to T_{rel}=500 released rounds. Training data are partitioned across clients using a Dirichlet non-IID split with concentration α_{Dir}=0.5. Detailed prompts, decoding, hyperparameters, and partitioning details are deferred to Appendix.

Asynchrony, privacy, and audit protocol. We emulate asynchronous execution by assigning each client upload a virtual delay and constructing the resulting server arrival stream. Unless otherwise stated, we use disorder ratio ρ=0.5 (severe OOO delivery). We enforce client-level DP with clipping threshold C, Gaussian perturbation, and an RDP accountant with δ=10^{-5}, targeting $\varepsilon_{\text{target}}$=2.0. For auditability, each method reports its online privacy state $\varepsilon_{\text{ledger}}$, and we recompute $\varepsilon_{\text{replay}}$ from the logged execution trace under the same public parameters (q, C, σ, δ) and the same accountant. For FEDSAN, replay reconstructs the released-round sequence from authenticated provenance and deterministic server decisions; for arrival-order baselines, replay reconstructs the claimed arrival-indexed accounting stream. We report

$$\Delta\varepsilon_{\text{audit}} = \left|\varepsilon_{\text{replay}} - \varepsilon_{\text{ledger}}\right|,$$

and mark a run as FAIL if either $\varepsilon_{\text{replay}} > \varepsilon_{\text{target}}$ or $\Delta\varepsilon_{\text{audit}} > \tau$.

Timing and communication. All methods run on the same cluster with identical client-side training, serialization, and DP code paths. Wall-clock time includes computation, communication, buffering/release, and accounting, and is normalized by the same number of released rounds. We report uplink communication; unless otherwise stated, the main text uses *effective uplink per released round*, while realized uplink, payload-size calculations, and detailed timing/communication definitions are deferred to Appendix.

Baselines. We compare against: (i) **Sync DP-FedAvg**, a synchronous upper bound with round-consistent accounting; (ii) **Async DP-FedAvg (Arrival)**, which applies updates and advances the accountant in raw arrival order; and (iii) **DP-FedBuff**, a buffered asynchronous baseline that accounts when a buffer is released. All methods use the same client-side DP mechanism and optimizer settings; they differ only in their arrival/release policy, accounting basis, and optional projection.

4.2 Main Results: Utility, Efficiency, and Auditability under Strict Privacy

We evaluate Llama-3-8B with LoRA under a strict privacy budget ($\varepsilon_{\text{target}} = 2.0, \delta = 10^{-5}$) and severe out-of-order (OOO) arrivals. Table 1 reports task utility, wall-clock efficiency, and audit deviation. As defined in Section 3.2, the audit deviation is

$$\Delta\varepsilon_{\text{audit}} = \left|\varepsilon_{\text{replay}} - \varepsilon_{\text{ledger}}\right|,$$

where $\varepsilon_{\text{ledger}}$ is the privacy loss reported online during training and $\varepsilon_{\text{replay}}$ is recomputed offline by replaying the method-defined accounting stream from the execution log under the same public DP parameters and the same accountant. A method is marked FAIL if it exceeds the target budget ($\varepsilon_{\text{replay}} > \varepsilon_{\text{target}}$) or if its published ledger is not reproducible from logs ($\Delta\varepsilon_{\text{audit}} > \tau$).

FedSAN is the only method that simultaneously attains strong utility, substantial async speedup, and audit pass under strict DP. In Table 1, FEDSAN reaches $73.8\% \pm 0.4$ on PubMedQA and $70.2\% \pm 0.5$ on FiQA, which is within one point of the synchronous upper bound ($74.5\% \pm 0.3$ and $71.0\% \pm 0.4$, respectively), while maintaining a small audit deviation of 0.06 and passing the audit criterion. At the same time, it retains most of the efficiency benefit of asynchronous training, reducing wall-clock time from 82.0h to 31.2h (2.6× speedup). This makes FEDSAN the only method in the strict-budget regime that is simultaneously

Table 1: **Main results on PubMedQA and FiQA (Llama-3-8B + LoRA).** We report Accuracy, wall-clock time, and audit deviation $\Delta\varepsilon_{\text{audit}} := \left|\varepsilon_{\text{replay}} - \varepsilon_{\text{ledger}}\right|$. Here $\varepsilon_{\text{replay}}$ is recomputed by replaying the *method-defined accounting stream* from logs under the declared accountant: for FEDSAN, this stream is the released-round sequence reconstructed from authenticated provenance and deterministic server decisions; for arrival-order baselines, it is the arrival-indexed stream claimed by the implementation. A run is marked FAIL if $\varepsilon_{\text{replay}} > \varepsilon_{\text{target}}$ or $\Delta\varepsilon_{\text{audit}} > \tau$. Wall-clock time is measured on the same hardware and normalized by the same number of released rounds.

Method	**Accounting Basis**	**Accuracy (%)**		**Efficiency**		**Audit $\Delta\varepsilon$**
		PubMedQA	**FiQA**	**Time (h)**	**Speedup**	
Target Privacy Budget $\varepsilon_{target} = 2.0$ (Strict Regime)						
Sync DP-FedAvg (Upper Bound)	Round-Consistent	74.5 ± 0.3	71.0 ± 0.4	82.0	1.0×	0.01
Async DP-FedAvg (Arrival)	Arrival Order	58.2 ± 1.2	55.0 ± 1.5	**24.5**	**3.3**×	3.82 (Fail)
DP-FedBuff ($K = 10$)	Arrival Order	65.1 ± 0.8	62.4 ± 0.9	26.8	3.0×	2.15 (Fail)
FedSAN (Ours)	**R-RDP**	**73.8** $\pm$ **0.4**	**70.2** $\pm$ **0.5**	31.2	2.6×	**0.06** (Pass)

Table 2: **Instruction-following evaluation under strict DP ($\varepsilon = 2.0$).** We cast PubMedQA and FiQA as constrained label generation and report label exact match (EM, %) and format adherence (%). Format adherence is the fraction of outputs whose first non-whitespace token belongs to the required label set ({Yes, No, Maybe} for PubMedQA; {Positive, Negative, Neutral} for FiQA), case-insensitive.

Method	**PubMedQA**		**FiQA**	
	EM	**Format Adh.**	**EM**	**Format Adh.**
Sync DP-FedAvg	74.5	99.8	71.0	99.5
Async (Arrival)	58.2	64.2	55.0	59.5
DP-FedBuff	65.1	81.7	62.4	76.3
FedSAN (Ours)	**73.8**	**99.1**	**70.2**	**98.8**

accurate, substantially faster than synchronization, and replayably accountable.

Arrival-order baselines are fast, but their ledgers are not replayably valid under OOO delivery. The naive arrival-order asynchronous baseline is the fastest method (24.5h), but it suffers severe utility degradation (58.2 on PubMedQA and 55.0 on FiQA) together with a very large audit deviation (3.82, FAIL). DP-FedBuff improves utility relative to naive arrival-order asynchrony (65.1 / 62.4), but it still fails audit with $\Delta\varepsilon_{\text{audit}} = 2.15$. These results show that buffering alone is not sufficient: if the accountant is still driven by an arrival-indexed stream rather than the correct released-round execution trace, the resulting ledger remains non-replayable under OOO arrivals.

Under strict privacy, accounting integrity is reflected not only in accuracy but also in instruction-format stability. Table 2 shows that the gap is not limited to aggregate classification accuracy. Under severe OOO delivery, arrival-order training produces frequent failures to emit a valid label token at the first decoding position, yielding format adherence of only 64.2%/59.5% on PubMedQA/FiQA. DP-FedBuff partially mitigates this effect but remains noticeably unstable (81.7%/76.3%). In contrast, FEDSAN preserves near-synchronous instruction-format reliability (99.1%/98.8%), closely matching the synchronous upper bound. This pattern is consistent with Table 1: once the accounting stream is aligned with the actual released-round training trace, strict-budget training remains both utility-preserving and operationally auditable.

Why the baselines fail, and why FedSAN does not. The failure mode is not simply "asynchrony hurts accuracy." Under OOO delivery, the arrival-order baselines define accounting events by raw message arrivals rather than by the released-round sequence that actually governs model updates. Consequently, the accountant state is tied to an event stream that does not faithfully represent the executed training trace, which manifests as large replay mismatch and audit failure. FEDSAN, by contrast, reconstructs a snapshot-consistent released-round sequence from authenticated provenance and deterministic server decisions, and advances the accountant exactly once per released round. This is why it achieves a small audit deviation while preserving most of the utility and efficiency advantages of asynchronous execution.

4.3 System Cost and Component Analysis

We next examine three complementary questions beyond the main results: (1) what systems overhead is required to obtain a replayable released-

round ledger under severe out-of-order (OOO) delivery, (2) whether the advantage of replayable released-round accounting persists under tighter privacy regimes and stronger OOO severity, and (3) which components of FEDSAN are essential for auditability versus communication efficiency. All results in this section are reported under the same strict-DP setting as in Section 4.2, unless otherwise stated.

4.3.1 The Cost of Integrity: Reordering-Window Trade-off

A natural concern is that enforcing released-round consistency may erase the throughput advantage of asynchronous training. To quantify this trade-off, we vary the reordering window W under severe disorder ($\rho = 0.5$). Table 3 reports: (i) the fraction of uploads that are eventually dropped before release, (ii) wall-clock time relative to the $W = 0$ setting under the same arrival realization, (iii) test accuracy, and (iv) audit pass/fail.

Table 3: **Impact of reordering-window size W on efficiency and auditability.** Llama-3-8B under severe disorder ($\rho = 0.5$). Dropped uploads are logged but do not form released rounds and do not contribute DP events. Relative time is normalized to $W = 0$ under the same disorder process.

Window W	Dropped (%)	Rel. Time	Accuracy (%)	Audit
0 (Pure Async)	0.0%	**1.00**×	58.2	✗Fail
5	12.4%	1.15×	69.5	✓Pass
15 (Ours)	**2.1%**	**1.28**×	**73.8**	**✓Pass**
50	0.1%	2.50×	74.2	✓Pass
∞ (Sync-like)	0.0%	3.30×	74.5	✓Pass

A practical sweet spot for auditable asynchrony. The results show a clear trade-off. At $W = 0$, the system maximizes throughput but fails audit because the accountant is effectively tied to an unsafe arrival-indexed execution trace under OOO delivery. As W increases, the server can recover more snapshot-consistent released rounds, which restores replayability and improves utility, but at the cost of additional waiting and buffering latency. In our setting, $W = 15$ is the best operating point: it passes audit, incurs only 1.28× wall-clock overhead relative to unsafe pure async, and recovers near-synchronous utility (73.8% versus 74.5%). Larger windows reduce the drop rate further, but their latency rapidly approaches synchronization, indicating diminishing returns once released-round consistency is already reliably enforced. This trade-off is important because it shows that replayability is not obtained for free, but neither does it require fully reverting to synchronous execution.

4.3.2 Robustness across Privacy Regimes

We next test whether the advantage of replayable released-round accounting persists as the privacy budget tightens. On FiQA, we sweep $\varepsilon_{\text{target}} \in \{0.5, 1.0, 2.0, 4.0\}$ with fixed $\delta = 10^{-5}$. For each target budget, all methods tune the noise multiplier to match their own declared accounting protocol, and we report test accuracy.

Table 4: **Utility across privacy regimes on FiQA.** We sweep $\varepsilon_{\text{target}} \in \{0.5, 1.0, 2.0, 4.0\}$ with $\delta = 10^{-5}$.

Privacy Budget	Accuracy (%) on FiQA		
	Sync (Upper)	Async (Arrival)	FedSAN (Ours)
$\varepsilon = 0.5$ (Extreme)	52.0	28.5 (Random)	**50.8**
$\varepsilon = 1.0$ (Strict)	64.5	41.2	**63.1**
$\varepsilon = 2.0$ (Standard)	71.0	55.0	**70.2**
$\varepsilon = 4.0$ (Relaxed)	73.5	68.0	**72.8**

Correct accounting matters most when privacy noise is large. As the privacy budget becomes tighter, the gap between arrival-order accounting and replayable released-round accounting widens sharply. At $\varepsilon = 0.5$, the arrival-order baseline collapses to near-random accuracy (28.5%), whereas FEDSAN remains close to the synchronous upper bound (50.8% versus 52.0%). Even at $\varepsilon = 1.0$, FEDSAN retains most of the useful task signal while the arrival-order baseline remains substantially degraded. This pattern indicates that under high-noise training, using the correct released-round event stream is not merely a bookkeeping improvement; it is necessary to preserve learnability under OOO delivery. Taken together with Table 3, this result shows that the benefit of replayable accounting is visible not only in auditability itself, but also in the ability to learn useful task structure under strict privacy constraints.

4.3.3 Sensitivity to Disorder Severity

We additionally vary the disorder severity ρ to test whether the main conclusions hold beyond the default severe-OOO setting. Table 5 reports PubMedQA accuracy, audit deviation, dropped-upload ratio, and wall-clock time for both the arrival-order baseline and FEDSAN. This experiment complements Table 3: whereas the reordering-window sweep studies the effect of a system control parameter under fixed OOO severity, the disorder sweep studies how the underlying degree of OOO itself changes the gap between unsafe arrival-order

Table 5: **Sensitivity to disorder severity ρ.** We vary $\rho \in \{0.1, 0.3, 0.5, 0.7\}$ and report PubMedQA Accuracy, audit deviation, dropped-upload ratio, and wall-clock time.

ρ	Async DP-FedAvg (Arrival)				FedSAN			
	Acc.	$\Delta\varepsilon_{\text{audit}}$	**Dropped**	**Time (h)**	**Acc.**	$\Delta\varepsilon_{\text{audit}}$	**Dropped**	**Time (h)**
0.1	70.6	0.42	0.0%	26.1	74.0	0.02	0.3%	29.0
0.3	64.7	1.85	0.0%	25.3	73.9	0.04	1.0%	30.1
0.5	58.2	3.82	0.0%	24.5	73.8	0.06	2.1%	31.2
0.7	51.4	5.47	0.0%	23.8	72.9	0.09	4.8%	33.0

execution and replayable reconstructed-order accounting.

OOO severity amplifies the accounting gap. The gap widens monotonically with disorder severity. At light disorder ($\rho = 0.1$), arrival-order execution is still usable but already shows non-trivial audit mismatch. At severe disorder ($\rho \geq 0.5$), the arrival-order baseline becomes both inaccurate and strongly non-replayable, while FEDSAN remains close to the synchronous operating point and preserves a small audit deviation. The increased drop rate and runtime on the FEDSAN side are not a hidden cost; rather, they are the explicit systems price paid to maintain a replayable released-round trace as the environment becomes more adversarial to ordered execution.

4.3.4 Component Ablation: R-RDP versus Projection

We isolate the two method components: reconstructed-order accounting (R-RDP), which determines ledger replayability, and projected privatized uploads, which reduce post-DP communication.

Table 6: **Component ablation on Llama-3-8B.** The full method uses R-RDP and projected privatized uploads. Communication reports *effective uplink per released round*. Under our default LoRA setting, the raw privatized update is about 54.4MB per client (544MB/round for m=10), while projection with d_{proj}=4096 reduces this to about 16KB per client (160KB/round; metadata $<$ 1KB/upload).

Variant	R-RDP	Proj.	Acc. (%)	Eff. Comm./round	Audit
FedSAN (Full)	✓	✓	73.8	160 KB	✓
w/o Projection	✓	✗	74.1	544 MB	✓
w/o R-RDP	✗	✓	59.5	160 KB	✗

R-RDP is necessary for auditability; projection is necessary for practicality. Removing projection leaves accounting unchanged and preserves both auditability and accuracy (74.1% vs. 73.8%), but increases effective uplink by more than three orders of magnitude (544MB vs. 160KB per released round). Removing R-RDP preserves the low-bandwidth communication path but destroys replayability and sharply reduces accuracy. Thus, the two components play orthogonal roles: R-RDP restores accounting integrity, whereas projection makes the method practical at LLM scale.

Practical overhead. FedSAN is not free: replayable accounting requires append-only logging and deterministic server-side release/drop control, sacrificing some pure first-come-first-served flexibility. However, this complexity is localized to the server-side control plane and does not alter the client-side DP mechanism, model architecture, or PEFT pipeline. The resulting overhead remains modest relative to synchronization while enabling verifiable privacy accounting under severe OOO delivery. Additional stress tests on client heterogeneity, LoRA rank, and extreme heavy-tailed delays are reported in Appendix D.

5 Conclusion

As federated learning moves to LLM-scale instruction tuning, practical deployments must balance asynchronous throughput with strict, auditable differential privacy. We showed that out-of-order delivery makes arrival-indexed accounting non-replayable by mis-aligning the claimed ledger with the released-round execution trace. To address this, we proposed FEDSAN, which combines authenticated snapshot provenance with deterministic reconstructed-order accounting (R-RDP) to produce a replayable DP ledger, while optionally reducing uplink cost through post-DP projection. Experiments on Llama-3-8B LoRA tuning for biomedical QA and financial sentiment show that FEDSAN retains near-synchronous utility, delivers substantial asynchronous speedup, and passes strict privacy audits under severe disorder.

6 Limitations

Log integrity and threat model. Our auditability guarantee is conditional on the integrity of provenance logs (snapshot ids, counters, and DP parameters). We assume an append-only log or externally verifiable logging (e.g., client-signed entries / transparency-style storage). Handling a fully malicious server that can selectively drop/rewrites log entries is outside our scope.

Bounded disorder and buffering. R-RDP relies on a bounded reordering window: with extreme, non-stationary outages, a finite buffer may overflow, causing dropped updates or additional waiting. We log drop/overflow events for replay, but designing adaptive buffering and release policies with formal guarantees under heavy-tailed delays is left to future work.

PEFT-centric evaluation. Our LLM experiments focus on PEFT (LoRA) where communication is feasible; while R-RDP is mechanism-agnostic, full-parameter fine-tuning would require substantially stronger compression and systems support than evaluated here.

DP utility on minority/long-tail clients. Strict DP can amplify the under-representation of clients with small datasets or rare patterns. Quantifying this effect under more diverse client distributions and broader instruction-tuning objectives is an important direction.

References

Martın Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. *arXiv (Cornell University)*.

Sultan Alasmari, Rayed AlGhamdi, Ghanshyam G Tejani, Sunil Kumar Sharma, and Seyed Jalaleddin Mousavirad. 2025. Federated learning-based multimodal approach for early detection and personalized care in cardiac disease. *Frontiers in Physiology*, 16:1563185.

Chunlu Chen, Ji Liu, Haowen Tan, Xingjian Li, Kevin I-Kai Wang, Peng Li, Kouichi Sakurai, and Dejing Dou. 2025. Trustworthy federated learning: privacy, security, and beyond. *Knowledge and Information Systems*, 67(3):2321–2356.

Zhidong Gao, Zhenxiao Zhang, Yuanxiong Guo, and Yanmin Gong. 2025. Federated adaptive fine-tuning of large language models with heterogeneous quantization and lora. In *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*, pages 1–10. IEEE.

Sajjad Ghiasvand, Yifan Yang, Zhiyu Xue, Mahnoosh Alizadeh, Zheng Zhang, and Ramtin Pedarsani. 2025. Communication-efficient and tensorized federated fine-tuning of large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24192–24207.

Zhangshuang Guan, Yulin Zhao, Zhiguo Wan, and Wei Wang. 2025. Input integrity and authentic results: Towards trustworthy aggregation in federated learning. In *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*, pages 1–10. IEEE.

Yulin Huang, Yaxuan Luan, Jinxu Guo, Xiangchen Song, and Yuchen Liu. 2025. Parameter-efficient fine-tuning with differential privacy for robust instruction adaptation in large language models. *arXiv preprint arXiv:2512.06711*.

Antti Koskela and Jafar Aco Mohammadi. 2025. Auditing differential privacy guarantees using density estimation. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 1007–1026. IEEE.

Ding Li, Ziqi Zhang, Mengyu Yao, Yifeng Cai, Yao Guo, and Xiangqun Chen. 2025a. Teeslice: Protecting sensitive neural network models in trusted execution environments when attackers have pre-trained models. *ACM Transactions on Software Engineering and Methodology*, 34(6):1–49.

Tianyu Li, Dezhi Han, Jiatao Li, and Kuan-Ching Li. 2025b. Asynchronous tiered federated learning storage scheme based on blockchain and ipfs. *Computers, Materials & Continua*, 83(3).

Xingyu Li, Lu Peng, Yu-Ping Wang, and Weihua Zhang. 2025c. Open challenges and opportunities in federated foundation models towards biomedical healthcare. *BioData Mining*, 18(1):2.

Xiao-Yang Liu, Rongyi Zhu, Daochen Zha, Jiechao Gao, Shan Zhong, Matt White, and Meikang Qiu. 2025. Differentially private low-rank adaptation of large language model using federated learning. *ACM Transactions on Management Information Systems*, 16(2):1–24.

Qianpiao Ma, Xiaozhu Song, Junlong Zhou, Haibo Wang, Yunming Liao, Jianchun Liu, and Hongli Xu. 2025. Asynchronous federated learning over non-iid data via over-the-air computation. *IEEE Transactions on Networking*.

H. B. McMahan, Eider Moore, Daniel Ramage, S. Hampson, and B. A. Y. Arcas. 2016. Communication-efficient learning of deep networks from decentralized data.

Wanyi Ning, Jingyu Wang, Qi Qi, Haifeng Sun, Daixuan Cheng, Cong Liu, Lei Zhang, Zirui Zhuang, and

Jianxin Liao. 2025. Federated fine-tuning on heterogeneous loras with error-compensated aggregation. *IEEE Transactions on Neural Networks and Learning Systems*.

Elowen Price. 2025. Federated learning for privacy-preserving edge intelligence: A scalable systems perspective. *Journal of Computer Science and Software Applications*, 5(5).

Zhen Qin, Zhaomin Wu, Bingsheng He, and Shuiguang Deng. 2025. Federated data-efficient instruction tuning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15550–15568.

Yifan Shi, Kang Wei, Li Shen, Yingqi Liu, Xueqian Wang, Bo Yuan, and Dacheng Tao. 2025. Towards the flatter landscape and better generalization in federated learning under client-level differential privacy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Shaowei Wang, Sufen Zeng, Jin Li, Shaozheng Huang, and Yuyang Chen. 2025a. Shuffle model of differential privacy: Numerical composition for federated learning. *Applied Sciences*, 15(3):1595.

Sibo Wang, Song Han, Ziyu Cheng, Ming Wang, and Yilin Li. 2025b. Federated fine-tuning of large language models with privacy preservation and cross-domain semantic alignment. In *2025 6th International Conference on Computer Vision and Data Mining (ICCVDM)*, pages 494–498. IEEE.

Renyou Xie, Chaojie Li, Zhaohui Yang, Zhao Xu, Jian Huang, and ZhaoYang Dong. 2025. Differential privacy enabled robust asynchronous federated multitask learning: A multigradient descent approach. *IEEE Transactions on Cybernetics*.

Zhibo Xing, Zijian Zhang, Ziang Zhang, Zhen Li, Meng Li, Jiamou Liu, Zongyang Zhang, Yi Zhao, Qi Sun, Liehuang Zhu, and 1 others. 2025. Zero-knowledge proof-based verifiable decentralized machine learning in communication network: A comprehensive survey. *IEEE Communications Surveys & Tutorials*.

Xiaopeng Zhao, Xiao Bai, Guohao Sun, and Zhe Yan. 2025. Asynchronous federated learning with local differential privacy for privacy-enhanced recommender systems. *IEEE Internet of Things Journal*.

Meilu Zhu, Axiu Mao, Jun Liu, and Yixuan Yuan. 2024. Deer: Deviation eliminating and noise regulating for privacy-preserving federated low-rank adaptation. *IEEE Transactions on Medical Imaging*, 44(4):1783–1795.

A Experimental Details

A.1 Hyperparameters and Reproducibility

Shared settings. Unless otherwise stated, all methods share the same backbone, LoRA configuration, client sampling schedule, local training steps, optimizer, client-side DP mechanism, and accountant implementation. The only method-level differences are (i) the server-side ingestion/release policy that determines the accounting stream, and (ii) whether post-DP uplink projection is enabled.

Randomness control. We report mean±std over three random seeds. For each seed, we fix (i) client sampling, (ii) non-IID client partitioning, and (iii) the asynchrony simulator (delay draws), and run all methods on the same partitions and delay realizations so that observed differences can be attributed to the release/accounting policy rather than to different data or timing randomness.

A.2 Datasets, Prompts, and Client Partitioning

Datasets and splits. We use PubMedQA (biomedical QA; `Yes/No/Maybe`) and FiQA (financial sentiment; `Positive/Negative/Neutral`) with standard train/dev/test splits. The dev/test sets are kept centralized and used only for evaluation.

Client partitioning (non-IID). Unless otherwise stated, training data are partitioned across $K{=}200$ clients using a Dirichlet non-IID split with concentration parameter $\alpha_{\mathrm{Dir}}{=}0.5$. Per-client sample counts are fixed across methods within the same seed.

Prompt templates.

```
PubMedQA:
Context: {context}
Question: {question}
Answer (Yes/No/Maybe):

FiQA:
Headline: {sentence}
Sentiment (Positive/Negative/Neutral):
```

Decoding and label mapping. We use greedy decoding with a constrained label set and map the first generated label token to the target class. Outputs whose first non-whitespace token falls outside the required label set are counted as incorrect and as format violations.

Evaluation metrics. For both tasks, we report (i) label exact match / accuracy, and (ii) format adherence, defined as the fraction of outputs whose first non-whitespace token belongs to the required label set. This second metric is useful because privacy-constrained instruction tuning can preserve partial task information while still degrading output-schema reliability.

B Compute, Timing, and Communication Accounting

Hardware and software. Experiments run on a private cluster with 8×NVIDIA A100 (80GB) GPUs and AMD EPYC 7763 CPUs, implemented in PyTorch with `transformers` and `peft`. All wall-clock numbers include client compute, communication, server buffering/release, and DP accounting, and are measured on the same cluster.

Timing protocol. To avoid unfair comparisons caused by early termination or different drop patterns, we normalize runtime by the same number of *released rounds*, which is the accounting unit used throughout the paper. Specifically, we fix $T_{\mathrm{rel}}{=}500$ released rounds for all methods. Rounds that are deterministically dropped do not advance the accountant and do not count toward T_{rel}. Reported wall-clock time is the elapsed time required to complete these T_{rel} released rounds under the same asynchrony realization.

Communication metrics. Unless otherwise specified, "communication" refers to **uplink only** (client→server), because downlink broadcasting is identical across methods under the same released-round budget in our implementation. We distinguish two quantities.

Realized uplink. This is the total number of uploaded bytes actually received by the server during execution, regardless of whether an upload is eventually released, buffered, timed out, or dropped.

Effective uplink per released round. For released round t, let $\mathcal{U}_t$ denote the set of uploads incorporated into the released-round aggregate. We define

$$\mathrm{EffComm/round}(t) = \sum_{i\in\mathcal{U}_t} \Big(\mathrm{bytes}(u_i){+}\mathrm{bytes}(\mathrm{meta}_i)\Big), \tag{8}$$

and total effective uplink over T_{rel} released rounds as

$$\sum_{t=1}^{T_{\mathrm{rel}}} \mathrm{EffComm/round}(t). \tag{9}$$

Here u_i is the uploaded payload (raw privatized update or projected sketch), and

$$\mathrm{meta}_i = (\mathtt{cid}, s, \mathtt{ctr}, \tau, \mathtt{seed_id})$$

Table 7: **Hyperparameters (FEDSAN and baselines).**

Item	Value
Federated training	
Clients K / sampled per logical round m	200 / 10 (sampling rate q=0.05)
Target released rounds T_{rel}	500 released rounds
Local epochs E / batch size B	1 / 4
Max sequence length	512 (truncate longer inputs)
Backbone & PEFT	
Backbone	Llama-3-8B-Instruct (frozen)
LoRA rank / alpha / dropout	r=16 / α=32 / 0.05
Target modules	`q_proj`, `v_proj`
Precision	BF16 forward; FP32 for DP noise injection and transmitted payload tensors
Optimization	
Client optimizer / lr	AdamW / 3×10^{-4}
Server update	asynchronous FedAvg-style round update (η=1.0)
Weight decay	0.0
Differential privacy and accounting	
Accounting unit	one released round = one DP event
Clip / (ε, δ)	C=1.0 / $(2.0, 10^{-5})$
Noise multiplier σ	calibrated via RDP to match ε at T_{rel}=500 released rounds
RDP orders	$\lambda \in \{2, 3, \ldots, 64\}$ (minimum over λ)
FEDSAN	
Reordering window W / disorder ρ	15 / 0.5 (main setting)
Projection dimension d_{proj}	4096 (if enabled)
Tie-break for equal snapshot id	deterministic by $(\texttt{cid}, \texttt{ctr})$

contains authenticated provenance together with the public projection-seed identifier. Main-text communication tables report effective uplink per released round, while dropped-upload statistics are reported separately so that system overhead is exposed rather than hidden.

Payload sizes. For our default LoRA configuration (rank 16 on `q_proj` and `v_proj`), the trainable adapter has approximately 13.6M parameters. We transmit privatized raw updates in FP32 for conservative communication accounting, giving raw uplink per client upload

$$S_{\text{raw}} \approx 13.6\text{M} \times 4 \text{ bytes} \approx 54.4\text{MB}.$$

With projection, each client transmits a length-d_{proj} FP32 sketch:

$$S_{\text{proj}} \approx d_{\text{proj}} \times 4 \text{ bytes}.$$

For d_{proj}=4096, this yields $S_{\text{proj}} \approx 16\text{KB}$ per client upload (plus negligible metadata). When m=10 uploads are incorporated into one released round, the corresponding effective uplink is approximately:

- **Raw uplink (no projection):** $\approx 10 \times 54.4\text{MB} \approx 544\text{MB}$ per released round;
- **Projected uplink (d_{proj}=4096):** $\approx 10 \times 16\text{KB} \approx 160\text{KB}$ per released round.

We also verify these values empirically by measuring serialized bytes transmitted on the wire; measured values match the closed-form estimates up to small serialization and metadata overhead.

C Asynchrony Model and Replay Semantics

Event-time simulation. We emulate asynchronous arrivals via per-client timestamps

$$\tau_{\text{arr}} = \tau_{\text{issue}} + t_{\text{comp}} + t_{\text{comm}} + \delta_{\text{net}},$$

where t_{comp} captures compute heterogeneity, t_{comm} depends on payload size, and δ_{net} models heavy-tailed network delay. Within each random seed, all methods use the same delay realizations.

Disorder ratio. We report a disorder ratio $\rho \in [0, 1]$ that summarizes how severely raw arrivals deviate from the released-round execution order. Operationally, ρ is defined as the fraction of uploads that would become stale under pure asynchronous ingestion (W=0), i.e., uploads that arrive after the server would already have advanced beyond their target snapshot under arrival-order execution. We use ρ=0.5 as the main severe-disorder setting.

Replayable release/drop semantics. FEDSAN uses a bounded reordering window W together

Table 8: **Sensitivity to non-IID client heterogeneity.** We vary the Dirichlet concentration parameter α_{Dir} and report PubMedQA Accuracy, format adherence, and audit deviation.

α_{Dir}	Async DP-FedAvg (Arrival)			FedSAN		
	Acc.	**Format Adh.**	$\Delta\varepsilon_{\mathrm{audit}}$	**Acc.**	**Format Adh.**	$\Delta\varepsilon_{\mathrm{audit}}$
0.1	54.3	58.7	3.95	71.6	97.8	0.06
0.5	58.2	64.2	3.82	73.8	99.1	0.06
1.0	61.0	69.5	3.76	74.1	99.3	0.05
10.0	64.8	75.1	3.61	74.4	99.5	0.05

Table 9: **Sensitivity to LoRA rank.** We vary the LoRA rank r and report PubMedQA Accuracy, effective uplink per released round, and audit deviation. For projected variants, the effective uplink remains dominated by the fixed sketch size under the default $d_{\mathrm{proj}} = 4096$.

LoRA rank r	FedSAN w/o Projection			FedSAN (Full)		
	Acc.	**Eff. Comm./round**	$\Delta\varepsilon_{\mathrm{audit}}$	**Acc.**	**Eff. Comm./round**	$\Delta\varepsilon_{\mathrm{audit}}$
4	70.9	136MB	0.05	70.5	160KB	0.05
8	72.8	272MB	0.05	72.4	160KB	0.05
16	74.1	544MB	0.06	73.8	160KB	0.06
32	74.4	1088MB	0.06	74.0	160KB	0.06

with a deterministic, snapshot-consistent release rule. Uploads that are eventually released become part of a released-round aggregate and contribute exactly one DP event through that released round. Uploads that are timed out, stale, or discarded by policy are logged with their final disposition but do not form released rounds and do not contribute DP events. An auditor can replay the resulting append-only execution trace to reconstruct the released-round sequence and recompute $\varepsilon_{\mathrm{replay}}$.

D Additional Sensitivity and Ablation Results

D.1 Projection Dimension

We study the effect of projection dimension on the uplink–utility trade-off under the same privacy budget and disorder setting as the main experiment. For interpretability, Table 10 reports *payload size per client upload*; the main text reports effective uplink per released round.

The results show that projection is robust over a broad range of sketch sizes. Reducing d_{proj} from 16384 to 4096 has negligible effect on accuracy, while shrinking the client payload by a factor of four. A much more aggressive sketch (d_{proj}=1024) further reduces communication but begins to incur a visible utility loss.

Table 10: **Projection dimension versus payload size and utility (PubMedQA).** This table reports *payload size per client upload*; the main text reports effective uplink per released round. For FP32 sketches, the per-upload payload is $4 \cdot d_{\mathrm{proj}}$ bytes, compared with approximately 54.4MB for the unprojected raw privatized LoRA update under the default configuration.

d_{proj}	**Payload / upload**	**Acc. (%)**
None	54.4MB	73.8
16384	64KB	73.7
4096	16KB	73.6
1024	4KB	72.1

D.2 Sensitivity to Client Heterogeneity

To test robustness to data heterogeneity, we vary the Dirichlet concentration parameter

$$\alpha_{\mathrm{Dir}} \in \{0.1, 0.5, 1.0, 10\}.$$

Table 8 reports PubMedQA accuracy, format adherence, and audit deviation.

More severe non-IID partitioning makes optimization harder for all methods. However, the accounting gap of arrival-order training remains large across the entire sweep, whereas FEDSAN preserves low audit deviation and near-synchronous format reliability even in the strongly skewed regime.

Table 11: **Extreme-delay stress test.** We increase delay-tail severity while keeping the same model, DP budget, and release policy. Higher severity means a larger fraction of extreme stragglers. FEDSAN preserves replayability throughout, while drop rate and runtime increase as expected under harsher delay conditions.

Delay Tail Severity	Async DP-FedAvg (Arrival)				FedSAN			
	Acc.	$\Delta\varepsilon_{\text{audit}}$	Dropped	Time (h)	Acc.	$\Delta\varepsilon_{\text{audit}}$	Dropped	Time (h)
Mild	66.9	1.34	0.0%	25.6	74.0	0.03	0.8%	29.8
Moderate	62.4	2.67	0.0%	25.0	73.9	0.05	1.6%	30.6
Heavy	58.2	3.82	0.0%	24.5	73.8	0.06	2.1%	31.2
Extreme	52.7	5.41	0.0%	23.8	72.6	0.11	5.9%	33.7

D.3 Sensitivity to LoRA Rank

To test whether the method depends on a single PEFT configuration, we vary the LoRA rank

$$r \in \{4, 8, 16, 32\}.$$

Table 9 reports PubMedQA accuracy, effective uplink per released round, and audit deviation. The default setting in the main paper is $r = 16$.

This sweep supports two conclusions. First, the replayability of R-RDP is essentially invariant to LoRA rank, as reflected by consistently small audit deviation across all settings. Second, projection becomes increasingly valuable as rank grows, because the unprojected uplink scales linearly with the adapter size while the projected communication remains fixed by the sketch dimension.

D.4 Audit Log Schema and Replay Procedure

To make the replay process fully explicit, we summarize the append-only audit log schema used by FEDSAN. For each received upload, the server logs the client identity, snapshot id, client counter, authentication tag, arrival index, final disposition (BUFFER/RELEASE/DROP), and a payload hash. Here `cid` is the client identity, s is the authenticated snapshot identifier, `ctr` is a client-side monotone counter, τ is the authentication tag, `arrival_idx` is the arrival order at the server, `disposition` is one of {BUFFER, RELEASE, DROP}, and `payload_hash` is a hash of the received payload.

Given this log together with the public DP parameters (q, C, σ, δ) and the deterministic release/drop policy, an independent verifier reproduces the reported accountant state as follows: (i) verify authenticated provenance, (ii) replay the server-side release/drop logic on the logged arrivals, (iii) reconstruct the released-round sequence

$$\Pi^{\star}_{\text{rel}} = (s^{\star}_1, s^{\star}_2, \dots),$$

and (iv) replay the same accountant on that released-round sequence to obtain $\varepsilon_{\text{replay}}$. This procedure makes the reported ledger externally reproducible from logged execution artifacts rather than from internal server state.

D.5 Extreme-Delay Stress Test

We further test whether the main conclusions hold under heavier-tailed client delays than those used in the main experiments. Starting from the same base asynchrony simulator, we increase the tail severity of client delays while keeping all other settings fixed. Table 11 reports PubMedQA accuracy, audit deviation, dropped-upload ratio, and wall-clock time.

The stress test reinforces the main conclusion. As delay tails become heavier, the arrival-order baseline becomes increasingly misaligned with the executed training trace, leading to rapidly growing audit deviation and degraded utility. By contrast, FEDSAN preserves a replayable ledger throughout the sweep. Its cost under extreme delays is not hidden: stronger straggling increases the drop rate and wall-clock time, but the accounting trace remains auditable and utility degrades much more gracefully than under arrival-order execution.