POLLEN: SINGLE-SERVER SPARSE SECURE AGGREGATION WITH MINIMAL LEAKAGE

Anonymous authors
Paper under double-blind review

ABSTRACT

Secure aggregation protects the values of client updates in federated learning but assumes dense model updates. This assumption misaligned with modern language models and recommendation system models, whose index embedding and parameter-efficient fine-tuning layers produce highly sparse updates. While recent works such as Meridian (S&P 2026) and Clover (CCS 2025) have proposed sparse secure aggregation protocols, they focus on non-colluding multiple-server setting only, and the privacy effects of sparse updates comparing to dense are still not well understood.

In this paper, we identify a new leakage channel, termed *client index structure leakage*, arising from the co-occurrence structure of sparse updates. We show that under a positive distributional-gap assumption, any sparse aggregation protocol whose server view preserves per-client index co-occurrence information admits cross-round client re-identification attacks. Conversely, we show that revealing only the union of active input indices is sufficient, as this information is already derivable from the ideal functionality of dense (traditional) secure aggregation. Together, these results characterize a minimal leakage profile for sparse secure aggregation.

Guided by this characterization, we introduce a new primitive, *labeled multiparty private set union* (LMPSU) and present a lightweight single-server construction, Pollen, based on arithmetic invertible Bloom lookup tables and secure aggregation. Building on this primitive, we design a sparse secure aggregation protocol for sparse embedding updates. Experiments show that our approach eliminates re-identification attacks while achieving a $1.9 - 2803\times$ communication savings compared to dense secure aggregation.

1 INTRODUCTION

Introduced by SecAgg (Bonawitz et al., 2017), Secure Aggregation (SA) helps protect client individual weight updates from a single central server in Federated Learning. The line of work outputs to server the aggregated weight updates of client's local training while leaking no further information. Ideal functionality of SA is shown in Figure 2.

Despite being heavily optimizied by later works (Bell et al., 2020; Ma et al., 2023; Bell et al., 2023; Liu et al., 2025; Ma et al., 2025; Garg et al., 2026), SA cost stays dependent on the model size. This is because SA often assume client weight update to be sent fully: The client needs to send an encrypted/masked version of the full update vector to the server for the server to aggregate and decrypt/unmask.

In reality, client model update are often sparse: (1) As shown in recent work (Han et al., 2020), clients just need to send top k gradient value to the server rather than the full vector in order to achieve high accuracy and high communication efficient. (2) In addition to sparsified gradient, sparsity can also be found at embedding layer of large foundation language model or recommendation system model: only embeddings corresponding to input tokens or observed items receive gradient updates, while all other embedding parameters remain unchanged.

Related Work of Sparse SA. Acknowledging this phenomena, sparse SA has been studied extensively recently. Earlier works (Suimon et al., 2025; Ergun et al., 2021) has client to send masked

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

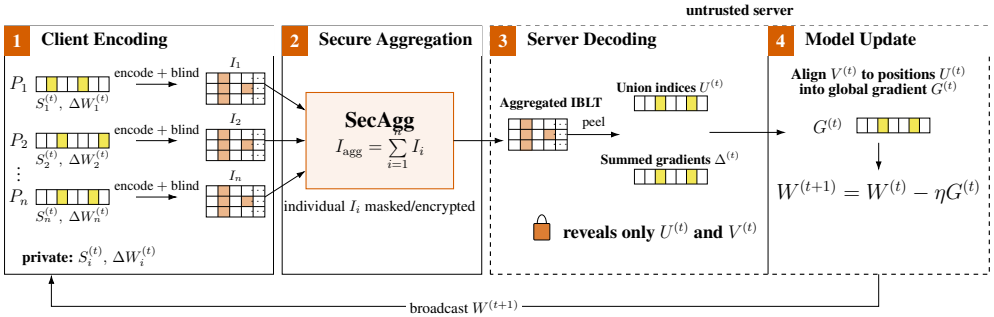

Figure 1: Overview of our Sparse SA protocol. Client encode and blind their sparse update, then perform normal SA on the local IBLT I. Finally, server perform decoding and aligning weight update using the union before sending back to the client. Dotted lines show computation being done at server.

top-k gradient value along with their indices in the full vector update for the server to align and aggregate. Later work (Us Sami and Güler, 2024) shows a gradient inversion attack taking advantages of the shared gradient index combined with aggregated gradient value. Hence, recent protocols (Tang et al., 2026; Us Sami and Güler, 2024) proposed that the client needs to hide the indices of the weight and that clients and server perform a heavy offline phase preparing a correlations proportional to full model size so that they can enjoy an efficient online phase where client only need to send a masked version of the top-k weight. In addition, Ding et al. (2026); Xu et al. (2025) takes a slightly different approach by proposing a setup of multiple servers instead of a single central one, getting rid of the heavy pre-processing phase.

Despite recent advances, two fundamental questions remain unresolved. First, does revealing gradient indices—even without the corresponding gradient values—compromise security? Second, can we achieve the best of all worlds: (A) eliminating the expensive offline phase, (B) supporting sparse SA in a single-server setting, and (C) reducing the client-side cost so that it is independent of the full model size? A comprehensive comparison of `Pollen` with prior work is presented in Table 1.

Table 1: Comparison of Sparse Federated Learning and SA Approaches. SA denotes SA. Structure Analysis indicates whether the approach provides a formal analysis of privacy leakage arising from sparse update structure.

Work	Sparse	SA	Single Server	Structure Analysis
FLASC (Kuo et al., 2024)	✓	✗	✓	✗
Top-k FL (Le et al., 2026)	✓	✗	✓	✗
SparseSA (Suimon et al., 2025)[1]	✓	✗	✓	✗
Tang et al. (Tang et al., 2026)	✓	✓	✓	✗
TinySA (Us Sami and Güler, 2024)	✓	✓	✓	✗
Meridian (Ding et al., 2026)	✓	✓	✗	✗
Clover (Xu et al., 2025)	✓	✓	✗	✗
Dense SA [a]	✗	✓	✓	✗
Ours	✓	✓	✓	✓

[a] SecAgg+ (Bell et al., 2020),Flamingo (Ma et al., 2023),Willow (Bell-Clark et al., 2024), Aion (Liu et al., 2025),LightHouse (Garg et al., 2026)

PARAMETERS: n clients $\{P_1, \ldots, P_n\}$, a server S.
FUNCTIONALITY:
- Waiting for input v_i from the $P_{i\in[n]}$
- Waiting for no input from the server S.
- Give each client $v = \sum_{i=1}^{n} v_i$

Figure 2: Secure Aggregation (SA) Ideal Functionality $\mathcal{F}_{SA}$

PARAMETERS: n parties $P_1, ..., P_n$ and the set size μ; a server $\mathcal{S}$ with no input.
FUNCTIONALITY:
1. Waiting for input $S_i = \{(\tau_{i,j}, \ell_{i,j}) \mid j \in [\mu]\}$ from P_i.
2. Output to the server the union $\{(\tau, \ell(\tau)) \mid \tau \in \bigcup_{i=1}^{n} T_i\}$, where $T_i = \{\tau_{i,j} \mid j \in [\mu]\}$ and $\ell(\tau)$ is the sum of all labels associated with τ across all parties holding τ.

Figure 3: Server-Aided Labled MPSU Ideal Functionality $\mathcal{F}_{LMPSU}$.

In this paper, we address both questions. First, we design an attack on the setting where the attacker use only the leaked set of gradient indices rather than any gradient value. Unlike the gradient + index attack shown in Us Sami and Güler (2024), our attack use only gradient indices as the attack vector. The attack shows that the gradient indices can uniquely identify the set of input embedding token along and their owner identity even in the case of anonymized communication. This attack works

even in case of multiple server setting where gradient update values are hidden. Second, we prove that the union of input embeddings can be obtained from dense SA, hinting that a private set union (PSU) solution can be used to obtain sparse SA. Finally, we propose a server-aided labeled multiparty private set union (LMPSU) protocol and its realization, `Pollen`, a sparse SA protocol based on LMPSU. Figure 1 illustrates the end-to-end workflow of `Pollen`, including client encoding, SA, server decoding, and model updating. The ideal functionality of LMPSU is shown in Figure 3.

Threat Model. We consider a standard federated learning threat model following prior SA systems (Bell et al., 2020; Ma et al., 2023; Bell-Clark et al., 2024; Garg et al., 2026). In a single training round, a central server coordinates the aggregation process among n clients. The server is *malicious* and may arbitrarily deviate from the protocol. Additionally, the server may collude with up to $n - T \geq 0$ corrupted clients, while the remaining clients are semi-honest who follow the protocol honestly but may try to extract information from the transcript they get during protocol execution. Here, the *corrupted clients* are clients under the adversarial control of the malicious server, which may arbitrarily control their inputs, outputs, and protocol behavior. The security goal is to protect the *input privacy* of honest clients from both the malicious server and the corrupted clients. Similar to prior works on SA (Bonawitz et al., 2017; Ma et al., 2023), we focus on *privacy rather than robustness*; ensuring the correctness of the aggregated output is left for future work and discussed in Section 6

We also provide further analysis on multi-round SA, inspired by Flamingo (Ma et al., 2023). At a high-level, the multi-round SA studies the effect of the accumulating information that server receive across training rounds and arguing whether a protocol provide input privacy for client when it participates in multiple rounds. The formal definition of multi-round SA can be found in Appendix E.

Our Contributions.

- We identify a new privacy vulnerability in sparse federated learning, termed *client index-structure leakage*. We show that any protocol revealing per-client active-index sets enables cross-round client re-identification. Through a formal analysis of sparse embedding updates, we prove that update co-occurrence patterns alone provide a stable client-specific signature, independent of gradient values.
- We characterize the minimal leakage profile relative to ideal dense SA, namely a profile that reveals no additional support-structure information beyond what is recoverable from aggregated sparse updates. Specifically, we show that hiding per-client co-occurrence information is necessary to prevent client re-identification, while revealing only the union of active items is sufficient and, in fact, can be derived from the aggregated output in dense aggregation.
- We introduce a new primitive, *Labeled Multiparty Private Set Union (LMPSU)*, which extends traditional MPSU by associating labels with set elements and aggregating labels across matching elements. We present a lightweight server-aided construction tailored to federated learning deployments and show that it achieves over $100\times$ speedup compared to state-of-the-art serverless MPSU protocols (Gao et al., 2025b; Dong et al., 2025b). We believe LMPSU is of independent interest and may find applications beyond sparse SA.
- We design and implement `Pollen`, an LMPSU-based sparse SA protocol and evaluate it on modern foundation models ranging from DistilBERT (67M) to Llama-3.3-70B. Compared to dense SA (Bell-Clark et al., 2024; Ma et al., 2023; Garg et al., 2026), our approach reduces client communication by up to $2803\times$ and runtime by up to $19.44\times$ while preserving the sparsity of model updates and eliminating client re-identification attacks.

2 Necessity of Hiding Co-occurrence Structure

In this section, we introduce a new attack, termed *client index-structure leakage*, which exploits the co-occurrence structure of sparse updates in federated learning. Unlike traditional attacks that rely on gradient values (Geiping et al., 2020; Zhao et al., 2023), this attack uses only the support structure of sparse updates. Consequently, the attack applies to any sparse aggregation protocol that reveals per-client support sets (Suimon et al., 2025; Ergun et al., 2021), regardless of whether gradient values

are protected by SA or differential privacy. In this section, we present our main theoretical results; the formal proofs are provided in Appendix F.

Let V be the embedding universe with size $|V|$. Each client P_i is associated with a fixed distribution p_i over the embedding universe set of all embedding items V, here $p_i(v) \geq 0$, $\sum_{v \in V} p_i(v) = 1$. In each round a participating client P_i produces an update by sampling s indices independently from p_i. For tractability we analyze ordered sampling with replacement, but later experiment in Section 5 shows the attack also works in practice for sampling without replacement.

Definition 1 (Collision Count). *Write $X = (X_1, \ldots, X_s)$ and $X' = (X'_1, \ldots, X'_s)$ for two independent ordered s-**draws** from the same client distribution p_i. For two distinct clients $P_i \neq P_j$ let $Y \leftarrow p_i^{\otimes s}$ and $Z \leftarrow p_j^{\otimes s}$ be independent draws [2]. Define the (ordered) collision count $C(Y, Z) = \sum_{a=1}^{s} \sum_{b=1}^{s} \mathbf{1}\{Y_a = Z_b\}$, so $0 \leq C \leq s^2$.*

Definition 2 (Key Distributional Quantities). *For a client i define the* **self-collision probability** *$q_i := \sum_{v \in V} p_i(v)^2$, and for two different clients i and j define the* **cross-collision probability** *$r_{ij} := \sum_{v \in V} p_i(v) p_j(v)$.*

Given such definition, the following lemma gives a formula for expectation of the collision count between random variables from similar and different distributions.

Lemma 1 (Expectations). *Let P_i and P_j be two distinct clients, and let X, X' be two independent s-draw from p_i, and Y be an s-draw from p_j. Under the sampling with replacement model, the expected collision count between X and Y (same distribution), and between X and Y (different distribution) are respectively:*

$$\mathbb{E}\big[C(X, X')\big] = s^2 q_i, \qquad \mathbb{E}\big[C(X, Y)\big] = s^2 r_{ij}.$$

Given the expectation formula, Theorem 1 derive overlap-based separability, which proposes a classifier that can tell if two random sampled sets are from same client. Then, Theorem 2 on multiple rounds effect of the classifier on re-identifying client across rounds.

Theorem 1 (Overlap-based separability; with replacement). *Fix distinct clients $P_i \neq P_j$. Consider the threshold test on two sampled sets of embedding items X, Y (each of size s) that declares "belong to the same client" whenever $\boxed{C(X, Y) \geq \Theta := \dfrac{s^2(q_i + r_{ij})}{2}}$. Under the with-replacement sampling model, the classification error (sum of false positive and false negative probabilities) is bounded by: $\boxed{Pr(\text{error}) \leq 2 \exp\left(-\dfrac{s(q_i - r_{ij})^2}{4}\right)}$*

Theorem 2 (Multi-round Re-identification from Overlap Separability). *Fix two distinct clients $P_i \neq P_j$. Suppose an adversary observes t independent sparse-update rounds from fixed client distributions and computes the cumulative overlap statistic: $C_t = \sum_{k=1}^{t} C(X^{(k)}, Y^{(k)})$ where $X^{(k)}, Y^{(k)}$ represents per round sampled sets of embedding items (size s).Then there exists a threshold-based linkage test that determines whether the t updates originate from the same client with a classification error bounded by: $\boxed{Pr(\text{error}) \leq 2 \exp\left(-\dfrac{ts(q_i - r_{ij})^2}{4}\right)}$*

Definition 3 (Per-update Co-occurence Preserving). *Let $\mathcal{V}_{\Pi}^{(k)}$ denote the server's view of a sparse aggregation protocol Π at training round k. We say Π is per-update co-occurrence preserving if there exists a deterministic polynomial-time extractor algorithm $\mathcal{E}$ such that $\mathcal{E}(\mathcal{V}_{\Pi}^{(k)})$ outputs an unordered collection of embedding item sets $\mathcal{U}^{(k)} = \{U_1^{(k)}, \ldots, U_n^{(k)}\}$ such that $U_a^{(k)} = S_{\pi_k(a)}^{(k)}$ for some unknown permutation π_k over $[n]$, where $S_i^{(k)}$ is the true active embedding item set submitted by client P_i.*

We make a specific assumption on our analysis termed "Positive Distributional Gap Assumption", which informally states that the underlying data distributions at different client are different. The formal definition of the assumption is shown in Definition 4.

[2] $p_i^{\otimes s}$ indicates s independent draw from p_i

Definition 4 (Positive Distributional Gap Assumption)**.** *Let Π be a sparse aggregation protocol that is per-update co-occurrence preserving, let n be the number of participating clients and $\Delta = \min_{i \neq j}(q_i - r_{ij})$ be the "minimum distributional gap" between any two clients. We call the condition $\Delta > 0$ the positive distributional-gap assumption.*

Implications Across Multiple Rounds. Theorem 3 is stated for two rounds for simplicity, but the attack naturally extends to an arbitrary number of rounds. By combining Theorem 3 with Theorem 2, the linkage error after observing t rounds is bounded by $\Pr(\text{Linkage Error}) \leq 2n \exp\left(-\frac{ts\Delta^2}{4}\right)$.

Theorem 3 (Co-occurrence Leakage Vulnerability)**.** *Under positive distributional-gap assumption, there exists a linkage adversary $\mathcal{A}$ that can link a targeted client's update from round 1 to their update in round 2 with a classification error bounded by $\mathcal{O}\left(n \cdot \exp\left(-\frac{s\Delta^2}{4}\right)\right)$.*

This bound shows that the adversary's distinguishing advantage improves exponentially as additional rounds are observed. Equivalently, to achieve a linkage error of at most ε, it suffices to observe $t \geq \frac{4}{s\Delta^2} \ln\left(\frac{2n}{\varepsilon}\right)$ rounds. Thus, the number of rounds required for reliable re-identification grows only logarithmically with the number of participating clients and the target error probability.

Necessity of Hiding Co-occurrence Structure. Theorem 3 shows that the privacy risk arises from the support sets themselves rather than from explicit client identities. As long as the server can observe each client's active embedding item set, it can compare overlap patterns across rounds and re-identify clients with high probability. Thus, defenses based solely on anonymization, shuffling, or metadata removal are insufficient. To prevent cross-round re-identification, a sparse aggregation protocol must hide the individual support sets and reveal only aggregate information. We therefore conclude the section with the following necessity result:

Necessity Result. Under the positive distributional-gap assumption, any sparse aggregation protocol that prevents cross-round client re-identification must hide per-update co-occurrence structure.

3 Minimal Leakage Characterization

Section 2 established that revealing per-client support sets is fundamentally incompatible with privacy. A natural question therefore arises: *what information can be safely revealed while preserving the efficiency benefits of sparse aggregation?* In this section, we show that union information is already recoverable from the dense SA output and therefore constitutes unavoidable baseline leakage in the sparse-update setting. Consequently, any sparse aggregation protocol that reveals only the union of active indices introduces no additional structural leakage beyond that already implied by the ideal functionality of dense SA. Throughout the remainder of the paper, we refer to this baseline-equivalent disclosure as the *minimal leakage* property.

3.1 Union Recoverability from Ideal SA

The intuition behind the recoverability is clear: once the server aggregate all sparse weight update, it can deduce the union of active embedding by looking at the support indices with non-zero aggregated gradient. The following theorem formalizes this observation and shows that the support of the aggregated update exactly recovers the union of active client indices.

Theorem 4 (Union Recoverability)**.** *Let V be the universe of sparse embedding indices, and let n be the number of participating clients in training round t. For each client $i \in [n]$, define $S_i^{(t)} \subseteq V$ as the private set of active embedding indices present in their local training batch. Let $\Delta W_i^{(t)}$ denote the sparse local model update for client i, defined such that for any embedding index $\tau \in V$:*

$$\Delta W_i^{(t)}[\tau] = \mathbf{0} \iff \tau \notin S_i^{(t)}$$

Let $G^{(t)} = \sum_{i=1}^{n} \Delta W_i^{(t)}$ be the global aggregated model update produced by an ideal dense SA protocol. Let δ_G be the probability that there is an exact cross-client gradient cancellation (i.e., $G^{(t)}[\tau] = \mathbf{0}$ for some $\tau \in \bigcup_{i=1}^{n} S_i^{(t)}$). Then, there exist a function $\mathcal{F}$ that maps $G^{(t)}$ (SA output) to $\bigcup_{i=1}^{n} S_i^{(t)}$ with probability $1 - \delta_G$. Formally: $P(\mathcal{F}(G^{(t)}) = \bigcup_{i=1}^{n} S_i^{(t)}) = 1 - \delta_G$

A probabilistic analysis of the cancellation event is provided in Appendix F.1. We argue that because of high dimension d, and continuous nature of the update, the probability δ_G is negligible, giving a near perfect probability of union recovery shown in Theorem 3.

3.2 Implications for Sparse SA Design

The results above imply that privacy-preserving sparse SA must hide individual support sets of active embedding indices. Defenses such as anonymization, shuffling, or metadata removal are insufficient if they preserve per-update co-occurrence structure, since the overlap-based linkage attack can still be applied.

Revealing only the union of active indices therefore emerges as a baseline-equivalent abstraction for privacy-preserving sparse aggregation.

LMPSU-based Defense. Recall that the attack in Section 2 relies on access to per-client support sets (or anonymized sets that preserve per-update co-occurrence structure). When LMPSU reduces the server's view to the per-round union $U^{(t)} = \bigcup_{i \in \mathcal{P}^{(t)}} S_i^{(t)}$, the adversary can no longer observe individual client support sets and therefore cannot compute overlap statistics between updates belonging to different clients. Instead, the server learns only aggregate union information, which eliminates the client-specific co-occurrence structure required by the attack. Consequently, the overlap-based linkage strategy of Theorem 1 no longer applies, motivating the following characterization of the minimal (baseline-equivalent) leakage profile of sparse SA.

Corollary 1 (**Minimal Leakage to Ideal SA**). *The minimal leakage profile relative to ideal dense SA (Figure 2) is precisely the functionality realized by LMPSU (Figure 3). Specifically, assuming no weight cancellation during aggregation process, in both ideal SA and LMPSU-based sparse SA, the server learns* $\left\{(\tau, \ell(\tau)) \ : \ \tau \in \bigcup_i S_i^{(t)}\right\}$, *namely the union of active embedding indices together with their aggregated labels, and nothing more about the individual client support sets or their co-occurrence structure.*

4 Labeled MPSU-based SA

We present a sparse SA protocol based on ideal LMPSU, followed by a lightweight IBLT-based LMPSU construction.

4.1 Secure Sparse Aggregation Protocol

Assuming the ideal LMPSU functionality $\mathcal{F}_{LMPSU}$ (Figure 3), we design a privacy-preserving sparse SA protocol, `Pollen`, shown in Figure 6 in Appendix B. At a high level, the protocol proceeds in two phases. (1) In the first phase, n clients participate in the training round, and the server learns the union $U^{(t)}$ of the embedding items used by the clients during the current update phase. In addition to learning $U^{(t)}$, the server also obtains the set $\Delta^{(t)}$ of aggregated local updates corresponding to the embedding indices in $U^{(t)}$. (2) In the second phase, the server aligns each aggregated gradient in $\Delta^{(t)}$ with its corresponding embedding index in $U^{(t)}$ to construct the aggregated update for the current training round, denoted by $G^{(t)}$. The server then updates the global model parameters according to $W^{(t+1)} = W^{(t)} - \eta \cdot G^{(t)}$, where η is the learning rate. Finally, the server distributes the updated model $W^{(t+1)}$ to all clients, enabling them to proceed to the next training round.

Security. Assuming the underlying LMPSU and the broadcast channel from server to client are secure against malicious server with corrupted clients, our sparse SA presented in Figure 6 realizes the sparse SA that provides input privacy against malicious server. We formalize this security guarantee in the following theorem:

Theorem 5 (Multi-Round Sparse SA Against Malicious Server, Corrupted Clients). *Let $\mathcal{F}_{MRSA}$ be the ideal multi-round sparse SA functionality defined in Figure 8. Assuming the underlying LMPSU protocol securely realizes $\mathcal{F}_{LMPSU}$ against a malicious adversary with security parameter λ, the Sparse SA protocol in Figure 6 securely realizes $\mathcal{F}_{MRSA}$ against a stateful, multi-round malicious server $\mathcal{A}$ colluding with a fixed set of up to $n - T$ corrupted clients over N training rounds, and for each round, the adversary can either: (1) aborts while revealing the SA-authorized abort and dropout*

leakage $\mathcal{L}$; or (2) selects an admissible survivor set $Q \in \Gamma$ and releases exactly the accepted sum $Y_Q = \sum_{i \in Q} I_i$, together with the survivor-set and delivery information authorized by $\mathcal{F}_{\mathsf{SA}}^{\Gamma,\mathcal{L}}$. with the advantage:

$$\boxed{Adv_{MRSA} \leq N \cdot (Adv_{LMPSU} + negl(\lambda))}$$

We deliver the proof of the theorem in Appendix F

Dropout Support. If the ideal $\mathcal{F}_{LMPSU}$ supports dropout, then our construction of sparse SA also supports dropout. It is because communication rounds requiring clients participation happens only within $\mathcal{F}_{LMPSU}$.

4.2 Our Labeled MPSU Protocol

Following the setup of FL, our protocol operates in a client-server communication-only manner: we have each client locally encode their set of items into a local IBLT, and interact with the central server, asking the server to aggregate the local IBLT in a way that can hide which items belong to multiple sets – which is the main reason why de-duplication is needed. We note that this leverage is well suited to the federated learning problem, where there exists a central server that has network capacity to connect to all clients in the training round. By removing such a need for pair-wise communication and de-duplication, our LMPSU design achieves scalable results even for a large number of clients up to $n = 1000$. We refer to Section D.2 for an empirical comparison to existing SOTA on MPSU.

Next, we discuss the challenges we face designing efficient IBLT-based LMPSU. Recall that in the arithmetic IBLT described in Section A, the `keysum`, `hashsum`, `count`, and `valuesum` fields are maintained as sums over the inserted elements, and a bucket is considered *pure* (and therefore peelable) if and only if its count equals one. A natural approach to constructing a union of multiple client sets is to aggregate their local IBLTs by summing the corresponding bucket contents. However, directly aggregating arithmetic IBLTs across clients introduces two challenges: ① **Double insertion:** When an item is inserted more than once, its corresponding count exceeds 1. Consequently, restricting peeling to buckets with count 1 prevents such items from being recovered; ② **Count leakage:** Because counts are aggregated across all local IBLTs, a peelable bucket reveals the number of parties holding the item. In our application, this reveals an important piece of information – the value N_τ for a embedding item τ. To ensure provable security, this count information must be hidden.

To address issue (1), we introduce a new notion of a "pure" bucket. To resolve issue (2), we mask the aggregated count with randomness while preserving bucket purity. Since randomness is introduced, we first apply the same randomness to other fields, such as the `keysum` field of the IBLT. We then exploit a duplication property: an item is deemed peelable if the modified key sum of the union is divisible by the modified count. This criterion enables us to define the notion of a "pure" bucket.

Randomized Arithmetic IBLT. Before describing the full LMPSU protocol, we isolate the randomized IBLT variant used by Pollen. When client P_i inserts item τ, it samples $r_{i,\tau} \leftarrow \mathrm{Unif}(\mathbb{Z}_p)$ and adds $(r_{i,\tau}\tau,\ r_{i,\tau}\mathtt{hash}(\tau),\ r_{i,\tau},\ \ell_i(\tau))$ to each bucket indexed by τ. If an aggregated bucket contains only one distinct item τ, possibly contributed by multiple clients, then $(K, H, C) = (R_\tau\tau,\ R_\tau\mathtt{hash}(\tau),\ R_\tau)$, where $R_\tau = \sum_{i:\tau \in S_i} r_{i,\tau}$. Whenever $R_\tau \neq 0$, the decoder recovers τ as K/C and verifies purity by checking $\mathtt{hash}(K/C) = H/C$. Thus, duplicate insertions of the same item remain peelable, while $C = R_\tau$ is uniformly distributed over $\mathbb{Z}_p$ and hides the number of clients holding τ. The decoding can fail only if $R_\tau = 0$, if standard IBLT peeling fails, or if a mixed bucket passes the algebraic purity check; these events are bounded in Theorem 6 in Appendix C.

Protocol Description. Our IBLT-based labeled MPSU protocol is defined in Figure 5. At a high level, the protocol consists of three phases: Local Encoding, Secure Aggregation, and Server Decoding. In the Local Encoding phase, each client P_i iterates over the items in their set and inserts each item into a local IBLT. The insertion follows the standard IBLT procedure with one key modification. For each item τ held by client P_i, the client samples a random integer $r_{i,\tau} \leftarrow \mathsf{Unif}(\mathbb{Z}_p)$. This value is used to blind the insertion count: the client multiplies $\mathtt{hash}(\tau)$ by $r_{i,\tau}$ before adding it to the `hashsum` field of each bucket indexed by $h_j(\tau)$ for $j \in [\kappa]$, multiplies τ by $r_{i,\tau}$ before adding it to the corresponding `keysum` field, and adds $r_{i,\tau}$ to the `count` field. The label payload $\ell_i(\tau)$ is added directly to the `valuesum` field $V_{h_j(\tau)}$ of each corresponding bucket. In the SA phase, the server obtains the element-wise sum of all local IBLTs without learning any information about the individual

Table 2: PR-AUC of the cross-round linkage attack under different observation models. S.O. refers to the Stackoverflow dataset, while Beauty, Sports, Games refer to 3 of the Amazon review dataset

Obs. Model	S.O.	Beauty	Sports	Games
Co-occurrence	0.875	0.952	0.961	0.968
MPSU	0.033	0.034	0.034	0.034
Random Guess	0.033	0.034	0.034	0.034

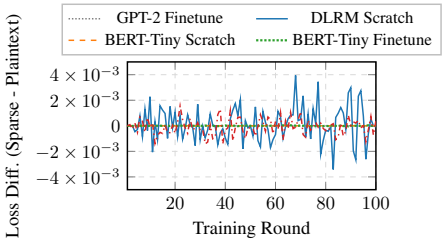

Figure 4: Loss difference between `Pollen` and the plaintext baseline across rounds.

Table 3: Comparison on Communication Cost of sparse SA protocols on Sparsified Gradient Aggregation Task across several value of K/L under ResNet20 model. TinySA refers to the work of Us Sami and Güler (2024).

Protocol	$K/L = 0.01$		$K/L = 0.05$	
	Offline	Online	Offline	Online
TinySA	24.40 GB	0.52 MiB	121.96 GB	0.57 MiB
Tang et al. (2026)	2.38 TiB	0.53 MiB	2.38 TiB	0.58 MiB
`Pollen`	–	2.69 MiB	–	8.93 MiB

Table 4: Measured run time comparison for the foundation models on LoRA $r = 16$ update. The model names are shorten due to space DistilBERT (D-BERT), RoBERTa-large (R-Large), Qwen2.5-1.5B(Q-1.5B), Llama-3.1-8B (Ll-8B), Llama3.3-70B (Ll-70B)

Protocol	D-BERT	R-Large	Q-1.5B	Ll-8B	Ll-70B
Flamingo	46.50	53.65	84.04	77.46	78.38
Willow	11.00	16.81	45.99	39.99	41.16
LightHouse	59.90	66.18	101.06	91.97	94.42
Ours, union only	11.22	11.26	11.63	12.28	13.45
Ours (10^{-2})	4.65	5.34	7.05	14.95	27.65
Ours (10^{-3})	**2.26**	**2.29**	**2.37**	**3.35**	**4.59**

local IBLTs. This is achieved by directly applying an existing SA protocol, such as SecAgg+ (Bell et al., 2020) and Willow (Bell-Clark et al., 2024), in which each client inputs their local IBLT. Finally, in the Server Decoding phase, the server locally iterates over the buckets and peels the "pure" ones. A bucket b is considered pure if and only if $\texttt{hash}(K_b \cdot C_b^{-1}) \equiv H_b \cdot C_b^{-1} \pmod{p}$. The peeled item is recovered as $\tau \equiv K_b \cdot C_b^{-1}$, and its corresponding label is obtained as $\ell(\tau) = V_b$.

Detailed discussion about Correctness and Security of our construction for LMPSU can be found in Appendix C

Efficiency. The communication complexity of the protocol is dominated by the size of the IBLT, which depends on the cardinality of the union $M = |U|$, rather than the embedding universe size V. Each client incurs communication cost $O(M)$, with updates securely aggregated via Willow (Bell-Clark et al., 2024).

On the computation side, the client-side cost is linear in the number of local updates, while the server-side cost is linear in the number of buckets m. Since sparse fine-tuning typically updates only a small fraction of the embedding universe ($M \ll V$), the additional overhead introduced by our protocol is negligible compared to the baseline cost of transmitting dense model updates.

Native Dropout Support. Our protocol natively handles dropout of clients during protocol execution if SA algorithm SA supports dropout (which is true for popular baseline protocol such as SecAgg (Bonawitz et al., 2017), SecAgg+ (Bell et al., 2020)). This is because the Labeled MPSU protocol only communicates to do SA, while the item encoding is done non-interactively at the client side and IBLT peeling is done non-interactively at the server side.

5 Experiment

We tested the protocol in four dimensions: attack validation, comparison against existing sparsified gradient works, comparison against existing dense SA works, and communication cost comparison on foundation model embedding layer. Details about dataset used and models used are discussed extensively in Appendix D.1, parameters used are detailed in Table 6 in Appendix C.

Attack Validation. We first validate the client index-structure leakage. We tested across different observation models and across 4 different datasets. The result is shown in Table 2, demonstrating that the attack identify client identity almost always for *Co-occurrence Preserving*– the server observes individual sparse update sets while client identities are removed. It also shows that MPSU solution brings down attack success rate to random guess level.

Table 5: Per-client communication cost (MiB) in one FL round for representative foundation models' embedding-layer updates. Plaintext denotes a full transmission of dense model updates without privacy protection. The number next to "**Ours**" represent the value of K/L in gradient sparsification, i.e. clients only communicate the top K gradients in the update vector; union-only refer to `Pollen` without gradient sparsification.

Update	Protocol	DistilBERT	RoBERTa-large	Qwen2.5-1.5B	Llama-3.1-8B	Llama-3.3-70B
Full Embedding	Plaintext	89.42	196.35	890.25	2004.00	4008.00
	Flamingo	178.85	392.71	1780.51	4008.01	8016.01
	LightHouse	89.43	196.35	890.26	2004.01	4008.01
	Willow	268.67	589.80	2671.15	6013.37	12025.70
	Ours, union only	78.24 (2.3×)	104.22 (3.8×)	156.17 (11.4×)	415.94 (9.6×)	831.56 (9.6×)
	Ours, (10^{-2})	3.08 (58.2×)	4.09 (96.0×)	6.12 (290.8×)	16.27 (246.4×)	33.26 (241.0×)
	Ours, (10^{-3})	0.32 (552.0×)	0.43 (918.0×)	0.64 (2803×)	1.65 (2425×)	3.34 (2401×)
LoRA ($r = 16$)	Plaintext	1.91	3.13	9.37	8.08	8.33
	Flamingo	3.83	6.27	18.74	16.17	16.67
	Lighthouse	1.92	3.14	9.37	8.08	8.33
	Willow	6.73	10.40	28.58	24.91	25.64
	Ours, union only	2.03 (1.9×)	2.06 (3.0×)	2.13 (8.8×)	2.44 (6.6×)	2.94 (5.7×)
	Ours, (10^{-2})	0.57 (6.7×)	0.74 (8.5×)	1.06 (17.6×)	2.69 (6.0×)	5.29 (3.1×)
	Ours, (10^{-3})	0.08 (47.9×)	0.09 (70.4×)	0.12 (151.0×)	0.29 (56.2×)	0.55 (30.3×)

Sparse SA comparison. Table 3 shows a comparison in communication cost between `Pollen` and state-of-the-art single-server sparsified gradient SA (Tang et al., 2026), TinySA (Us Sami and Güler, 2024) . The result shows that `Pollen` incur a small increase in online cost (several MiB) but helps the protocol avoid an expensive of gigabytes communication cost for offline phase.

Foundational Model Saving. As shown in previous experiment, current single server sparse SA requires expensive offline stage which requires clients sends gigabytes of data. We regard this as not practical. Thus, in experiment on foundational model, we only compare the computation and communication cost of `Pollen` against existing dense SA. Table 5 shows the communication saving and Table 4 shows the computation saving across several large foundation model embedding layers. We evaluate our protocol with 3 different configurations: union-only which uses the union of embedding indices for aggregation, and 2 configurations using gradient sparsification on top of embedding indices: we compute union on the top K gradient values among the L weights corresponding to the non-zero embedding indices. We set K/L to 0.01 and 0.001 following previous work of Shi et al. (2019); Bereyhi et al. (2025); Shi et al. (2020). The results show that `Pollen` helps saves communication cost by $1.9 - 2803\times$ and saves computation cost up to $19.44\times$. For running time, we only benchmark LoRA ($r = 16$) update rather than full model due to the large size of the full model. However, as shown in Table 5, the gap between `Pollen` and existing work is even larger in full embedding than the LoRA model that we use to test.

Utility Validation. We evaluate sparse SA across representative models and tasks: BERT-tiny for language understanding, GPT-2 (Radford et al., 2019) for next-token prediction, and DLRM (Naumov et al., 2019) for recommendation. We compare loss convergence when training with `Pollen` against plaintext training without SA. For BERT-tiny, we test both training from scratch and fine-tuning from a pretrained model; for GPT-2, we test fine-tuning only; and for DLRM, we test training from scratch only, matching common use cases for these models. All experiments use 100 clients, with 20 clients selected per round. Figure 4 shows negligible differences in loss convergence between plaintext training and sparse SA across all tasks, confirming that `Pollen` does not degrade model utility.

6 Conclusion

This paper studied sparse SA and a new leakage channel, in which adversaries infer honest-client information from the sparse structure of updates rather than from gradient values, as in traditional FL attacks. Our analysis shows that preventing this leakage requires hiding both client identities and item co-occurrence within each client's update set. This motivates LMPSU, a primitive that provides secure-aggregation-like privacy while preserving update sparsity and hiding co-occurrence information.

AI USE STATEMENT

In this work, we used generative AI tools for: formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, design or provide feedback on research methodology or experiments, implement methods and execution of software.

We have not used generative AI tools for generating synthetic datasets, help develop theoretical models or conceptual frameworks, propose or refine hypotheses, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results.

We have reviewed all AI-assisted work: We have checked the LLM-generated mathematical claims and corresponding proofs as well as their code implementation of the protocol and execution of existing works. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY

We provide an extensive Appendix that details the protocol pseudocode and set of parameters being used. The full codebase will be release upon the full release of the paper.

REFERENCES

James Bell, K. A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly)logarithmic overhead. Cryptology ePrint Archive, Paper 2020/704, 2020. URL https://eprint.iacr.org/2020/704.

James Bell, Adrià Gascón, Tancrède Lepoint, Baiyu Li, Sarah Meiklejohn, Mariana Raykova, and Cathie Yun. ACORN: Input validation for secure aggregation. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 4805–4822, 2023. ISBN 978-1-939133-37-3. URL https://www.usenix.org/conference/usenixsecurity23/presentation/bell.

James Bell-Clark, Adrià Gascón, Baiyu Li, Mariana Raykova, and Phillipp Schoppmann. Willow: Secure aggregation with one-shot clients. Cryptology ePrint Archive, Paper 2024/936, 2024. URL https://eprint.iacr.org/2024/936.

Ali Bereyhi, Ben Liang, Gary Boudreau, and Ali Afana. Regularized top-k: A bayesian framework for gradient sparsification. *IEEE Transactions on Signal Processing*, 73:4463–4478, 2025. ISSN 1941-0476. doi: 10.1109/tsp.2025.3624791. URL http://dx.doi.org/10.1109/TSP.2025.3624791.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy preserving machine learning. Cryptology ePrint Archive, Paper 2017/281, 2017. URL https://eprint.iacr.org/2017/281.

Youlong Ding, Peihua Mai, Jingqi Zhang, Sherman S. M. Chow, Minxin Du, and Yan Pang. Shared Spotlight Meridian: Distributed Sparse Pseudorandom Functions for Scalable Federated Learning . In *2026 IEEE Symposium on Security and Privacy (SP)*, 2026. doi: 10.1109/SP63933.2026.00065. URL https://doi.ieeecomputersociety.org/10.1109/SP63933.2026.00065.

Minglang Dong, Yu Chen, Cong Zhang, Yujie Bai, and Yang Cao. Multi-party private set operations from predicative zero-sharing. Cryptology ePrint Archive, Paper 2025/640, 2025a. URL https://eprint.iacr.org/2025/640.

Minglang Dong, Cong Zhang, Yujie Bai, and Yu Chen. Efficient multi-party private set union without non-collusion assumptions. In *Proceedings of the 34th USENIX Security Symposium (USENIX Security)*, 2025b.

Irem Ergun, Hasin Us Sami, and Basak Guler. Sparsified secure aggregation for privacy-preserving federated learning, 2021. URL https://arxiv.org/abs/2112.12872.

Jiahui Gao, Truong Son Nguyen, and Ni Trieu. Toward a practical multi-party private set union. In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*, 2024.

Jiahui Gao, Son Nguyen, Marina Blanton, and Ni Trieu. PULSE: Parallel private set union for large-scale entities. Cryptology ePrint Archive, Paper 2025/790, 2025a. URL https://eprint.iacr.org/2025/790.

Jiahui Gao, Truong Son Nguyen, Marina Blanton, and Ni Trieu. Pulse: Parallel private set union for large-scale entities. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2025b.

Sanjam Garg, Alireza Kavousi, Dimitris Kolonelos, Erkan Tairi, and Zhipeng Wang. Lighthouse: Single-server secure aggregation with o(1) server-committee communication at scale. Cryptology ePrint Archive, Paper 2026/346, 2026. URL https://eprint.iacr.org/2026/346.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients – how easy is it to break privacy in federated learning?, 2020. URL https://arxiv.org/abs/2003.14053.

Michael T. Goodrich and Michael Mitzenmacher. Invertible bloom lookup tables. In *49th Annual Allerton Conference on Communication, Control, and Computing*, 2011.

Pengchao Han, Shiqiang Wang, and Kin K. Leung. Adaptive gradient sparsification for efficient federated learning: An online learning approach. *CoRR*, abs/2001.04756, 2020. URL https://arxiv.org/abs/2001.04756.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding, 2020. URL https://arxiv.org/abs/1909.10351.

Kevin Kuo, Arian Raje, Kousik Rajesh, and Virginia Smith. Federated lora with sparse communication, 2024. URL https://arxiv.org/abs/2406.05233.

Duy-Dong Le, Duy-Thanh Huynh, and Tuong-Nguyen Huynh. Fedincsparse: A federated learning framework with top-k sparse delta transmission and incremental updates. In *Multi-disciplinary Trends in Artificial Intelligence*, 2026. ISBN 978-981-95-4957-3.

Yizhong Liu, Zixiao Jia, Xiao Chen, Song Bian, Runhua Xu, Dawei Li, and Yuan Lu. Aion: Robust and efficient Multi-Round Single-Mask secure aggregation against malicious participants. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 3025–3044, Seattle, WA, August 2025. USENIX Association. ISBN 978-1-939133-52-6. URL https://www.usenix.org/conference/usenixsecurity25/presentation/liu-yizhong.

Yiping Ma, Jess Woods, Sebastian Angel, Antigoni Polychroniadou, and Tal Rabin. Flamingo: Multi-round single-server secure aggregation with applications to private federated learning. Cryptology ePrint Archive, Paper 2023/486, 2023. URL https://eprint.iacr.org/2023/486.

Yiping Ma, Yue Guo, Harish Karthikeyan, and Antigoni Polychroniadou. Armadillo: Robust single-server secure aggregation for federated learning with input validation. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, CCS '25, page 2219–2233, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400715259. doi: 10.1145/3719027.3765216. URL https://doi.org/10.1145/3719027.3765216.

Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. Deep learning recommendation model for personalization and recommendation systems, 2019. URL https://arxiv.org/abs/1906.00091.

Lucas Piske and Ni Trieu. Is PSI really faster than PSU? achieving efficient PSU with invertible bloom filters. Cryptology ePrint Archive, Paper 2026/376, 2026. URL https://eprint.iacr.org/2026/376.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

Shaohuai Shi, Xiaowen Chu, Ka Chun Cheung, and Simon See. Understanding top-k sparsification in distributed deep learning, 2019. URL https://arxiv.org/abs/1911.08772.

Shaohuai Shi, Qiang Wang, Xiaowen Chu, Bo Li, Yang Qin, Ruihao Liu, and Xinxiao Zhao. Communication-efficient distributed deep learning with merged gradient sparsification on gpus. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 406–415, 2020. doi: 10.1109/INFOCOM41043.2020.9155269.

Takumi Suimon, Yuki Koizumi, Junji Takemasa, and Toru Hasegawa. Per-element secure aggregation against data reconstruction attacks in federated learning, 2025. URL https://arxiv.org/abs/2508.04285.

Hengxuan Tang, Jinbao Zhu, and Xiaohu Tang. Secure aggregation with top-k sparsification in decentralized federated learning, 2026. URL https://arxiv.org/abs/2606.10780.

Hasin Us Sami and Başak Güler. Secure gradient aggregation with sparsification for resource-limited federated learning. *IEEE Transactions on Communications*, 72(11):6883–6899, 2024. doi: 10.1109/TCOMM.2024.3403475.

Shuangqing Xu, Yifeng Zheng, and Zhongyun Hua. Harnessing sparsification in federated learning: A secure, efficient, and differentially private realization, 2025. URL https://arxiv.org/abs/2511.07123.

Joshua C. Zhao, Atul Sharma, Ahmed Roushdy Elkordy, Yahya H. Ezzeldin, Salman Avestimehr, and Saurabh Bagchi. Loki: Large-scale data reconstruction attack against federated learning through model manipulation, 2023. URL https://arxiv.org/abs/2303.12233.

A PRELIMINARIES

Labeled Multiparty Private Set Union (MPSU). MPSU allows multiple parties to compute the union of their private sets while revealing no additional information about the individual inputs. In this work, we introduce a *server-aided Labeled MPSU (LMPSU)* functionality. Unlike traditional MPSU protocols (Gao et al., 2025b; Dong et al., 2025b; Gao et al., 2024), LMPSU involves an untrusted server and a collection of clients whose set elements are associated with labels. The functionality reveals to the server the union of all client-held items. For each item appearing in the union, the server additionally learns the aggregate label obtained by summing the labels contributed by all clients holding that item. The ideal functionality is formally defined in Figure 3.

We use two equivalent notational conventions for labels. The value $\ell_{i,j}$ denotes the label associated with item $\tau_{i,j}$ held by client P_i. Alternatively, we write $\ell(\tau)$ to denote the aggregate label associated with item τ, defined as the sum of all labels contributed for τ across participating clients.

Invertible Bloom Lookup Table (IBLT). An arithmetic IBLT (Goodrich and Mitzenmacher, 2011) $\mathcal{I}$ has m buckets, κ hash functions $h_1, \ldots, h_\kappa$, and a fingerprint hash function $\texttt{hash}$. Each bucket b_j stores $(K_j, H_j, C_j, V_j) \in \mathbb{Z}_p^4$, initialized to zero, for the sums of keys, fingerprints, counts, and associated values. It supports two core operations:

- **Encode.** To insert a key-value pair $(x, \ell(x))$, for each $i \in [\kappa]$ with $j = h_i(x)$, the bucket b_j is updated as $C_j \leftarrow C_j + 1$, $K_j \leftarrow K_j + x$, $H_j \leftarrow H_j + \texttt{hash}(x)$, and $V_j \leftarrow V_j + \ell(x)$.
- **Peeling.** The peeling procedure iteratively identifies "pure" buckets with $C_j = 1$, recovers the corresponding key-value pair $(x, \ell(x))$, and removes it by resetting the bucket and subtracting its contribution from all buckets $h_i(x)$ for $i \in [\kappa]$.

IBLT based PSU. We use a modified IBLT to realize LMPSU. Piske and Trieu (2026) proposed the first IBLT-based PSU protocol but supports only two parties. Extending this approach to multiparty labeled unions is nontrivial and represents a key contribution of Pollen.

PARAMETERS: An item universe $\mathcal{T}$; a prime modulus $p > \kappa$; an injective encoding $\mathsf{enc} : \mathcal{T} \to \mathbb{Z}_p$; an IBLT size m; κ common encoding hash functions $h_1, \ldots, h_\kappa : \mathcal{T} \to [m]$; a fingerprint hash function $\mathtt{hash} : \mathcal{T} \to \mathbb{Z}_p$; a label dimension d; n clients $P_1, \ldots, P_n$; and a server. Each bucket $b \in [m]$ stores (K_b, H_b, C_b, V_b) where $K_b, H_b, C_b \in Z_p, V_b \in Z_p^d$.

INPUT: P_i inputs a set $S_i = \{(\tau, \ell_i(\tau)) : \tau \in \mathcal{T},\ \ell_i(\tau) \in \mathbb{Z}_p^d\}$

PROTOCOL: Each P_i does the following locally:
1. Initialize an empty IBLT I_i.
2. For every $(\tau, \ell_i(\tau)) \in S_i$:
   (a) Sample $r_{i,\tau} \leftarrow \mathsf{Uniform}(\mathbb{Z}_p)$.
   (b) Compute the set of distinct bucket locations $J(\tau) := \{h_a(\tau) : a \in [\kappa]\}$ and, for each $b \in J(\tau)$, its hash multiplicity $q_{\tau,b} := |\{a \in [\kappa] : h_a(\tau) = b\}|$.
   (c) For every $b \in J(\tau)$, update

$$K_b \mathrel{+}= q_{\tau,b}\, r_{i,\tau}\, \mathsf{enc}(\tau), \quad H_b \mathrel{+}= q_{\tau,b}\, r_{i,\tau}\, \mathtt{hash}(\tau),$$
$$C_b \mathrel{+}= q_{\tau,b}\, r_{i,\tau}, \qquad V_b \mathrel{+}= \ell_i(\tau).$$

The server:
1. Interacts with $P_1, \ldots, P_n$, securely aggregate the local IBLTs: $I_{\text{agg}} := \sum_{i=1}^n I_i \pmod p$.
2. Initializes the output set $\mathcal{U} \leftarrow \varnothing$.
3. While there exists a bucket b such that $C_b \neq 0$:
   (a) Compute $x := K_b C_b^{-1} \pmod p$.; Treat b as pure if $x \in \mathsf{enc}(\mathcal{T})$, and for $\tau := \mathsf{enc}^{-1}(x)$, $q_{\tau,b} > 0, \quad \mathtt{hash}(\tau) = H_b C_b^{-1} \pmod p$
   (b) Recover $\ell(\tau) := V_b, \quad R_\tau := C_b q_{\tau,b}^{-1} \pmod p$, and add $(\tau, \ell(\tau))$ to $\mathcal{U}$.
   (c) For every distinct bucket $u \in J(\tau)$, peel τ by updating

$$K_u \mathrel{-}= q_{\tau,u} R_\tau \mathsf{enc}(\tau), \quad H_u \mathrel{-}= q_{\tau,u} R_\tau \mathtt{hash}(\tau),$$
$$C_u \mathrel{-}= q_{\tau,u} R_\tau, \qquad V_u \mathrel{-}= \ell(\tau).$$

4. If every bucket is zero, output $\mathcal{U}$; otherwise output $\perp$.

Figure 5: Our server-aided LMPSU protocol.

B PROTOCOL FIGURES

In this section we present detail protocol figure for `Pollen` and $\mathcal{F}_{LMPSU}$; see Figure 6 and Figure 5.

C CORRECTNESS AND SECURITY OF OUR LMPSU CONSTRUCTION

C.1 CORRECTNESS.

We separate correctness into two parts: correctness of decoding and removing a pure bucket, and the probability that the IBLT peeling procedure recovers the entire union. For simplicity, use τ for both the item $\tau \in \mathcal{T}$ and its injective encoding $\mathrm{enc}(\tau)$ in $\mathbb{Z}_p$.

Decoding a pure bucket. For an item τ, define its set of distinct bucket locations and its multiplicity at bucket j as

$$J(\tau) := \{h_a(\tau) : a \in [\kappa]\}, \qquad q_{\tau,j} := |\{a \in [\kappa] : h_a(\tau) = j\}|.$$

The value field is updated once for every distinct bucket in $J(\tau)$, whereas the key, fingerprint, and count fields preserve the hash multiplicity $q_{\tau,j}$.

Suppose bucket j is pure, meaning that all its nonzero contributions arise from a single distinct item τ. Let $\mathcal{P}_\tau$ denote the clients holding τ, and define

$$R_\tau := \sum_{i \in \mathcal{P}_\tau} r_{i,\tau} \pmod p, \qquad \ell(\tau) := \sum_{i \in \mathcal{P}_\tau} \ell_i(\tau) \pmod p.$$

After secure aggregation, the bucket fields are

$$K_j = q_{\tau,j} R_\tau \tau, \quad H_j = q_{\tau,j} R_\tau \mathtt{hash}(\tau),$$
$$C_j = q_{\tau,j} R_\tau, \qquad V_j = \ell(\tau).$$

PARAMETERS: n clients $P_1, \ldots, P_n$; an untrusted server S; an ideal labeled MPSU functionality $\mathcal{F}_{LMPSU}$; learning rate η.

INPUT: All parties hold the current global model parameters $W^{(t)}$. Each participating client P_i computes a sparse local update comprising a private set of active embedding indices $S_i^{(t)}$ and corresponding gradient values $\Delta W_i^{(t)}$.

PROTOCOL:

Phase 1: Labeled MPSU Execution

1. **Client Input:** Each client P_i inputs their active embedding item-gradient pairs $\{(\tau, \Delta W_i^{(t)}[\tau]) \mid \tau \in S_i^{(t)}\}$ into $\mathcal{F}_{LMPSU}$.
2. **Server Output:** The n clients and the server jointly execute $\mathcal{F}_{LMPSU}$. As a result, the server learns exactly two pieces of information:
   - The exact union of active embedding indices: $U^{(t)} = \bigcup_{i=1}^{n} S_i^{(t)}$
   - The aggregated updates aligned to the union: $\Delta^{(t)} = \left\{ \sum_{i \in P_\tau} \Delta W_i^{(t)}[\tau] \;\middle|\; \tau \in U^{(t)} \right\}$

Phase 2: Server Update

1. **Gradient Alignment:** The server initializes a global aggregated gradient vector $G^{(t)}$ (sharing the exact dimensionality of $W^{(t)}$) to zero. For each active embedding index $\tau \in U^{(t)}$, the server assigns the corresponding aggregated gradient from $\Delta^{(t)}$ into $G^{(t)}[\tau]$.
2. **Global Model Update:** The server applies the dense aggregated gradient to update the global model parameters according to:
$$W^{(t+1)} = W^{(t)} - \eta \cdot G^{(t)}$$
3. **Broadcast:** The server distributes the newly updated model weights $W^{(t+1)}$ to all clients, enabling them to proceed to training round $t + 1$.

Figure 6: `Pollen` Construction using Ideal $\mathcal{F}_{LMPSU}$.

Because $1 \le q_{\tau,j} \le \kappa < p$, the multiplicity $q_{\tau,j}$ is invertible in $\mathbb{Z}_p$. Provided that $R_\tau \neq 0$, we also have $C_j \neq 0$, and the server recovers

$$K_j C_j^{-1} = q_{\tau,j} R_\tau \tau \left(q_{\tau,j} R_\tau\right)^{-1} = \tau \pmod{p}.$$

The server verifies purity by checking

$$\texttt{hash}(K_j C_j^{-1}) = H_j C_j^{-1} \pmod{p}.$$

Moreover, because the value field is inserted once per distinct bucket, $V_j = \ell(\tau)$ directly yields the aggregate label.

To remove τ from the IBLT, the server first recovers its aggregate mask as

$$R_\tau = C_j q_{\tau,j}^{-1} \pmod{p}.$$

For each distinct bucket $u \in J(\tau)$, it then subtracts

$$\left(q_{\tau,u} R_\tau \tau,\ q_{\tau,u} R_\tau \texttt{hash}(\tau),\ q_{\tau,u} R_\tau,\ \ell(\tau)\right).$$

This exactly removes τ from every bucket, including when two or more encoding hash functions map τ to the same bucket.

Peeling Entire Union. The decoder can fail for three reasons:

1. an aggregate mask satisfies $R_\tau = 0$ for some $\tau \in U$;
2. a bucket containing multiple distinct items passes the algebraic purity test; or
3. the peeling procedure terminates with a nonempty residual.

Let $P_{\text{peel}}^{\text{Pollen}}(M, m, \kappa)$ denote the probability of the third event for `Pollen`'s exact shared-table construction. This probability is taken over the κ hash locations sampled with replacement, including repeated locations for the same item, with the value-field deduplication and multiplicity-aware peeling procedure described above.

Theorem 6 (IBLT decoding-failure decomposition)**.** *Let U be a union of M distinct items, let m be the number of IBLT buckets, and let $p > \kappa$ be the prime modulus. Assume that the encoding hashes and fingerprint hash are independent and uniform over their respective ranges. Then*

$$\Pr[\mathsf{Fail}_{\mathrm{dec}}] \le P_{\mathrm{peel}}^{\mathrm{Pollen}}(M, m, \kappa) + \frac{\kappa^2 M^2}{mp} + \frac{M}{p}.$$

The three terms respectively account for ordinary peeling failure, false acceptance of a mixed bucket as pure, and cancellation of an aggregate mask.

Unlike the last two terms, we do not use an asymptotic expression for $P_{\mathrm{peel}}^{\mathrm{Pollen}}$. Tight analytical bounds are not known for our concrete parameter regime. Following the empirical parameter-selection methodology of Piske and Trieu Piske and Trieu (2026), we directly evaluate the peeling-failure probability of `Pollen`'s shared-table construction. We do not reuse their numerical estimates, because their experiments use a different, hash-partitioned table layout.

For every experimental workload, we fix a public union-capacity parameter M_{cap} before evaluation. We execute the peeling procedure over independently sampled IBLTs and report the observed failure rate

$$\widehat{P}_{\mathrm{peel}}^{\mathrm{Pollen}} := \frac{\text{number of peeling failures}}{\text{number of trials}}.$$

Peelability Experiment Setup. In our experiment, we fix a public union-capacity parameter $M_{\mathrm{cap}}^{(w)}$ before running the experiment and allocate $m = \lceil \alpha M_{\mathrm{cap}} \rceil$ IBLT buckets in every round with α being inverse load factor and would be a parameter to be tuned along with number of hash functions κ. The capacity is determined by the model, client participation, and update-sparsity configuration and is not adapted to the realized union of any individual round. We evaluate peeling failure conservatively at $M = M_{\mathrm{cap}}$. We swept $\kappa \in \{2, \ldots, 8\}$ at the smallest-table boundary for each capacity; $\kappa = 4$ was the only competitive choice. Each final row uses an independent seed. Hash locations are sampled with replacement; repeated locations of one item are deduplicated for structural degree and removed with their algebraic multiplicity. A reported zero means no observed peeling failures, not a proof that the probability is zero; the exact one-sided 95% binomial upper bound is 2.857×10^{-6}. The final column is the point estimate $\widehat{P}_{\mathrm{peel}} + \kappa^2 M_{\mathrm{cap}}^2/(mp) + M_{\mathrm{cap}}/p$. For the 100-client workloads, $M_{\mathrm{cap}} = 100 \cdot 100 = 10{,}000$ follows the public 100-active-items-per-client configuration. For ResNet20, $M_{\mathrm{cap}} = \min(L, nK)$.

The resulting concrete parameters and empirical failure rates are reported in Table 6.

Effect of Multiple Rounds. For configurations that may vary across rounds, a union bound gives

$$\Pr\left[\bigcup_{t=1}^{N} \mathsf{Fail}_{\mathrm{dec}}^{(t)}\right] \le \sum_{t=1}^{N} \left(P_{\mathrm{peel}}^{\mathrm{Pollen}}(M_t, m_t, \kappa_t) + \frac{\kappa_t^2 M_t^2}{m_t p} + \frac{M_t}{p} \right).$$

If the same parameters are used in every round, this reduces to N times the corresponding single-round expression. This is a correctness and availability statement only; decoding failure does not alter the input-privacy guarantee.

C.2 Security Analysis.

We present three theorems that establish the security of our protocol. Theorem 7 shows that the protocol completely hides embedding item frequency information, while Theorem 8 proves that the protocol achieves statistical security. Finally, Theorem 10 shows that, assuming at least T honest clients and a server that can corrupt up to $n - T$ clients and can actively alters messages sending to clients, the inputs of the remaining honest clients remain protected, provided that the underlying secure aggregation protocol (e.g., SecAgg+ Bell et al. (2020), Willow Bell-Clark et al. (2024)) is secure. For consistency, we collect all theorem proofs in one place. Accordingly, the detailed proofs of these theorems are deferred to Appendix F. At the core, the privacy guarantees stem from the blinding property of the random weights $r_{i,\tau}$, whose information-theoretically hide individual client contributions.

Theorem 7 (Frequency Hiding)**.** *For any embedding item τ held by a nonempty subset of clients $\mathcal{P}_\tau$, the aggregated count field C_{agg} of a pure bucket corresponding to τ is uniformly distributed over $\mathbb{Z}_p$.*

Table 6: Concrete IBLT parameters and empirical peeling-failure rates for the workloads evaluated in Section 5. M_{cap} denotes the public per-round union-capacity parameter. Prime modulus p is fixed at $2^{61}-1$

Model	K/L	M_{cap}	α	κ	m	Trials	$\widehat{P}_{\mathrm{peel}}$	$\widehat{\Pr}[\mathrm{Fail}_{\mathrm{dec}}]$
Union only								
All Table 5 models	none	10,000	1.33	4	13,300	2^{20}	0	5.65×10^{-14}
Full embedding								
DistilBERT	0.001	7,700	1.34	4	10,300	2^{20}	0	4.33×10^{-14}
DistilBERT	0.01	76,800	1.31	4	100,450	2^{20}	0	4.41×10^{-13}
RoBERTa-large	0.001	10,300	1.33	4	13,700	2^{20}	0	5.82×10^{-14}
RoBERTa-large	0.01	102,400	1.31	4	133,750	2^{20}	0	5.88×10^{-13}
Qwen2.5-1.5B	0.001	15,400	1.33	4	20,500	2^{20}	0	8.70×10^{-14}
Qwen2.5-1.5B	0.01	153,600	1.30	4	200,300	2^{20}	0	8.84×10^{-13}
Llama-3.1-8B	0.001	41,000	1.31	4	53,850	2^{20}	0	2.34×10^{-13}
Llama-3.1-8B	0.01	409,600	1.30	4	532,700	2^{20}	0	2.36×10^{-12}
Llama-3.3-70B	0.001	82,000	1.33	4	109,100	2^{20}	0	4.63×10^{-13}
Llama-3.3-70B	0.01	819,200	1.33	4	1,089,550	2^{20}	0	4.63×10^{-12}
Embedding LoRA $r=16$								
DistilBERT	0.001	1,400	1.64	4	2,300	2^{20}	0	6.52×10^{-15}
DistilBERT	0.01	13,900	1.33	4	18,450	2^{20}	0	7.87×10^{-14}
RoBERTa-large	0.001	1,800	1.44	4	2,600	2^{20}	0	9.43×10^{-15}
RoBERTa-large	0.01	18,000	1.32	4	23,800	2^{20}	0	1.02×10^{-13}
Qwen2.5-1.5B	0.001	2,700	1.39	4	3,750	2^{20}	0	1.47×10^{-14}
Qwen2.5-1.5B	0.01	26,200	1.32	4	34,500	2^{20}	0	1.49×10^{-13}
Llama-3.1-8B	0.001	6,800	1.34	4	9,100	2^{20}	0	3.82×10^{-14}
Llama-3.1-8B	0.01	67,200	1.31	4	87,950	2^{20}	0	3.85×10^{-13}
Llama-3.3-70B	0.001	13,300	1.33	4	17,700	2^{20}	0	7.51×10^{-14}
Llama-3.3-70B	0.01	132,700	1.30	4	173,150	2^{20}	0	7.63×10^{-13}

Theorem 8 (Statistical Security of Blinded IBLT-based MPSU)**.** *Let $U=(\tau,\ell(\tau)):\tau\in\bigcup_i S_i$ be the labeled union of embedding items, where $\ell(\tau)$ is the aggregate label associated with τ with $|U|=M$. Let m be the number of IBLT buckets, k the number of hash functions, and p a prime modulus, and let I_{agg} be the aggregated IBLT produced by the protocol. Assume that each client samples per-token masks $r_{i,\tau}\leftarrow\mathrm{Unif}(\mathbb{Z}_p)$ independently.*

There exists a simulator $\mathcal{S}$ such that the server's view in the real protocol execution is statistically indistinguishable from the view generated by $\mathcal{S}$ given only the ideal MPSU output, i.e. $\mathcal{S}$ can generate $I'_{\mathsf{agg}}\leftarrow\mathcal{S}(U)$ such that the statistical difference $\Delta\left(I_{\mathsf{agg}},I'_{\mathsf{agg}}\right)=0$.

Native Adversary Model Inheritance Our protocol also naturally adopts the security threat model of the underlying secure aggregation protocol: As the malicious server only communicates with client through the secure aggregation round to aggregate clients' local IBLT, if a secure aggregation protocol can guarantee input privacy of honest clients against malicious server, it provides input privacy for local IBLT of honest clients. Theorem 10 formally states this inherited adversary model.

Theorem 9 (Dense-SA-equivalent Aggregate View Under Collusion)**.** *Let $\mathcal{C}$ be the set of clients colluding with the server, let Q be the accepted survivor set, and let $\mathcal{H}=Q\setminus\mathcal{C}$.*

Consider the following two worlds:

- *In the* `Pollen` world*, the server receives $I_{\mathsf{agg}}=\sum_{i\in\mathcal{H}}I_i+\sum_{i\in\mathcal{C}\cap Q}I_i$.*
- *In the* ideal dense-SA world*, the server receives the aggregate update $G_Q=\sum_{i\in Q}\Delta W_i$ together with the leakage authorized by the underlying dense secure-aggregation functionality.*

Assume that the honest residual aggregate $G_{\mathcal{H}}:=G_Q-\sum_{i\in\mathcal{C}\cap Q}\Delta W_i$ satisfies $\mathrm{supp}(G_{\mathcal{H}})=\bigcup_{i\in\mathcal{H}}S_i$., i.e. there are no non-zero weight aggregate to zero, then there exists a simulator that, given the ideal dense-SA view and the colluding parties' states, produces an aggregate IBLT identically distributed to I_{agg}.

If the support equality fails with probability at most δ_G, the statistical distance between the simulated and real aggregate views is at most δ_G, as defined in Theorem 4.

Remark. Theorem 9 shows the security of `Pollen` under collusion is similar to the inherent leakage of dense SA.

Theorem 10 (Privacy against A Malicious Server, Corrupted Clients)**.** *Let I_i be the IBLT vector submitted by client P_i, where honest clients construct the local IBLT I_i according to Figure 5 and corrupted clients may submit arbitrary vectors. Suppose the instantiated secure-aggregation protocol actively realizes a sum functionality $\mathcal{F}_{\mathsf{SA}}^{\Gamma,\mathcal{L}}$ against a malicious server colluding with corrupted clients. For each session, this functionality can perform active malicious actions defined in Theorem 5. Then `Pollen` provides privacy against the same adversary and corruption pattern: for every PPT adversary $\mathcal{A}$, there exists a PPT simulator whose input consists only of the information exposed by $\mathcal{F}_{\mathsf{SA}}^{\Gamma,\mathcal{L}}$ and the corrupted parties' states, such that*

$$\mathrm{Adv}^{\mathsf{priv}}_{\mathsf{Pollen},\mathcal{A}} \leq \mathrm{Adv}^{\mathsf{act}}_{\mathsf{SA},\mathcal{B}} + \mathrm{negl}(\lambda).$$

In particular, `Pollen` introduces no leakage beyond that of the instantiated SA protocol. It provides no correctness, robustness, fairness, or guaranteed-delivery guarantee against corrupted clients or a malicious server.

D More Experiment Results

D.1 Union-recovery Validation

Table 7 shows perfect precision, recall, and F1-score across all workloads and federated scales. More importantly, exact recovery is obtained in every experimental trial (45 out of 45 trials total), including the largest setting with 1000 clients and a recovered union containing more than 1.6×10^4 active embedding indices. These results demonstrate that item-union information is directly recoverable from aggregated sparse updates generated by realistic optimization procedures, providing strong empirical support for the sufficiency result of Theorem 4.

Table 7: Union-recovery validation using sparse embedding gradients generated from real training objectives.

Model	Clients	Dim	Union Size	F1
BERT-tiny Jiao et al. (2020)	10	128	1034	1.0
	100	128	3824	1.0
	1000	128	9308	1.0
GPT2-med Radford et al. (2019)	10	1024	1347	1.0
	100	1024	6113	1.0
	1000	1024	16794	1.0
DLRM Naumov et al. (2019)	10	16	31	1.0
	100	16	248	1.0
	1000	16	2002	1.0

D.2 LMPSU Performance Comparison

This section compares our LMPSU protocol in `Pollen` with state-of-the-art server-less MPSU protocols Gao et al. (2025a); Dong et al. (2025a). Table 8 shows complexity comparison between our IBLT-based LMPSU and existing SOTA MPSU work. For empirical comparison, we also tested the protocols varying the number of clients $n \in \{10, 500, 1000\}$. Our experiment shows that our LMPSU is the only protocol that runs at $n = 500$ and $n = 1000$, while other protocols suffers due to their high peer-to-peer communication requirements. We estimate their run time and compare the results in Figure 7 . Comparing to SOTA, our protocol shows a significant improvement on run time at $\approx 100\times$.

E Multiple-Round Secure Aggregation

Because federated learning is inherently multi-round, the server accumulates protocol transcripts over time. We therefore consider an adversary with access to the complete transcript history across rounds.

Table 8: Complexity comparison between the proposed server-aided LMPSU and prior MPSU constructions. Here, s denotes the number of active items per client, n denotes the number of clients, M denotes size of the union ($M \ll ns$), d denotes dimension of the label. Client Channel refers to number of communication channels that clients need to interact.

Prot.	Labels	Clients			Server	
		Comm.	Channel	Comp.	Comp.	Comm.
Gao et al. (2025a)	✗	$O(ns)$	$O(n)$	$O(ns)$	–	–
Dong et al. (2025a)	✗	$O(ns)$	$O(n)$	$O(ns)$	–	–
Ours	✓	$O(Md)$	$O(1)$	$O(Md)$	$O(nMd)$	$O(nMd)$

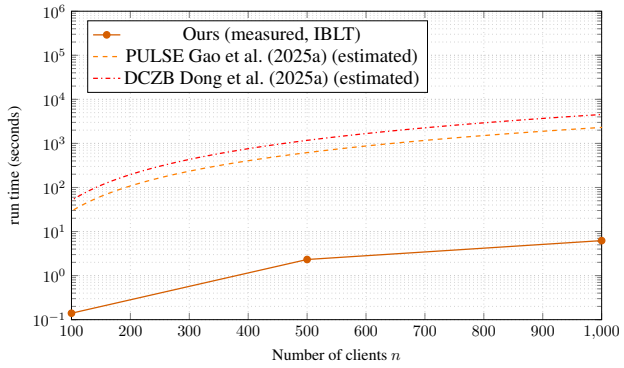

Figure 7: Run time (LAN) comparison between our lightweight server-aided MPSU protocol and PULSE Gao et al. (2025a) and DCZB Dong et al. (2025a). Serverless baselines (PULSE, DCZB) are extrapolated from their reported small-n measurements using fitted scaling models and are shown for reference.

This transcript is notably important in the sparse setting, where update values are protected by secure aggregation, but the structure of sparse updates may reveal information that can be correlated across rounds. Additionally, we assume a *persistent corruption model*: once a client is corrupted in any training round, it remains corrupted for the remainder of the protocol. This assumption captures the realistic setting in which an adversary that compromises a client can continue to access information from that client in subsequent rounds.

Therefore, in addition to the standard single-round adversary, we consider a *multi-round adversary* that: ① observes protocol transcripts across multiple training rounds, ② maintains a fixed set of corrupted clients across rounds, and ③ performs arbitrary cross-round analysis on all information learned during protocol execution.

Ideal Functionality $\mathcal{F}_{MRSA}$. We introduce a multi-round ideal secure aggregation functionality, $\mathcal{F}_{MRSA}$, that captures the threat model described in this section. The functionality is presented in Figure 8. It relies on an underlying secure aggregation phase, modeled by $\mathcal{F}_{\mathsf{SA}}$, together with a consistent broadcast round. Assuming the broadcast phase is secure, as discussed below, we later prove in Theorem 5 that $\mathcal{F}_{MRSA}$ can be reduced to the security of the underlying $\mathcal{F}_{\mathsf{SA}}$ functionality.

Broadcast Phase Security. In multi-round federated learning, single-server architectures are uniquely susceptible to *Model Inconsistency Attacks* (MIA), or split-view attacks. In such attacks, a malicious server distributes divergent global model weights to different subsets of clients in round t, artificially isolating their local states to compromise the anonymity set in round $t+1$. Therefore, we include a deterministic broadcast phase in the ideal functionality whose goal is to distribute the same model update to all clients.

In practice, the secure broadcast functionality can be instantiated using a transparency-log-based commitment mechanism. Prior to distributing the updated model, the server commits to a single model hash in an append-only transparency log. Clients verify consistency against the published commitment before proceeding to the next training round. We treat this mechanism as infrastructure and do not include its cost in our performance evaluation, as its computational and communication overhead is negligible relative to secure aggregation.

PARAMETERS: Clients $\mathcal{C} = \{P_1, \ldots, P_n\}$, server S, corrupted clients $\mathcal{B} \subseteq \mathcal{C}$, total rounds N, admissible survivor sets $\Gamma^{(t)}$, and Secure Aggregation functionality $\mathcal{F}_{\mathsf{SA}}$ as described in Figure 2 with leakage $\mathcal{L}_{\mathsf{SA}}^{(t)}$. The adversary $\mathcal{A}$ controls S and $\mathcal{B}$.

FUNCTIONALITY: For each round $t \in \{1, \ldots, N\}$ with unique session identifier sid_t:

Sparse Aggregation. Receive inputs $v_i^{(t)}$ from participating honest clients; corrupted clients may submit arbitrary inputs or no input. Let $\mathcal{R}^{(t)}$ be the clients whose inputs were accepted.
The adversary selects $Q^{(t)} \subseteq \mathcal{R}^{(t)}$ as permitted by the instantiated $\mathcal{F}_{\mathsf{SA}}$, or requests an abort. If it aborts or $Q^{(t)} \notin \Gamma^{(t)}$, reveal $(\perp, Q^{(t)}, \mathcal{L}_{\mathsf{SA}}^{(t)})$ and terminate.
Otherwise, output only to S

$$\left(Q^{(t)}, \sum_{P_i \in Q^{(t)}} v_i^{(t)}, \mathcal{L}_{\mathsf{SA}}^{(t)} \right).$$

Release at most one aggregate for sid_t.

Consistent Broadcast. After receiving the aggregate, $\mathcal{A}$ either submits $(\mathsf{deliver}, w^{(t+1)})$ or aborts. In the former case, output the same $w^{(t+1)}$ to every honest client in $Q^{(t)}$; otherwise, output $\perp$ and terminate.

Figure 8: Ideal Functionality for Multi-Round Sparse Secure Aggregation ($\mathcal{F}_{MRSA}$). This takes into account in-between round server message modification ability. The functionality guarantees privacy and broadcast consistency only; corrupted inputs may be arbitrary, and correctness, robustness, fairness, and guaranteed delivery are not provided.

F PROOFS FOR OUR PROTOCOL

Proof of Theorem 1 (Overlap-based separability; with replacement)

Proof. The statistic $C(X, Y)$ is a function of $2s$ independent random variables (the s draws in X and the s draws in Y). We cannot treat the s^2 pairwise indicators $\mathbb{I}\{X_a = Y_b\}$ as completely independent. Instead, we use bounded differences. Changing any single draw X_a to X'_a can change the total collision count C by at most s, because X_a can match at most s elements in Y. The same bounded difference of s applies to changing any Y_b. By McDiarmid's inequality, for any $\epsilon > 0$:

$$Pr(|C - \mathbb{E}[C]| \geq \epsilon) \leq 2 \exp\left(-\frac{2\epsilon^2}{2s \cdot s^2}\right) = 2 \exp\left(-\frac{\epsilon^2}{s^3}\right)$$

Setting $\epsilon = \frac{s^2(q_i - r_{ij})}{2}$ centers our threshold exactly between the same-client expectation $\mathbb{E}[C(X, X')] = s^2 q_i$ and the cross-client expectation $\mathbb{E}[C(X, Y)] = s^2 r_{ij}$. Substituting this ϵ into the bound yields the stated exponent. □

Proof of Theorem 2: (Multi-round Re-identification from Overlap Separability)

Proof. The multi-round cumulative overlap statistic is defined as $C_t = \sum_{k=1}^{t} C(X^{(k)}, Y^{(k)})$. Because the t training rounds are independent, C_t is a deterministic function of $2ts$ independent random variables (representing the s sampled embedding items per client, per round, across t rounds).

By linearity of expectation, the expected cumulative overlap is:

$$\mathbb{E}[C_t] = \begin{cases} ts^2 q_i & \text{if transcripts belong to same } P_i \\ ts^2 r_{ij} & \text{if transcripts belong to } P_i \& P_j (i \neq j) \end{cases}$$

We design the linkage test to classify the transcripts as "same-client" if $C_t \geq \Theta_t$, where the threshold is centered exactly halfway between the two expectations: $\Theta_t = \frac{ts^2(q_i + r_{ij})}{2}$.

To bound the classification error, we apply McDiarmid's inequality (Bounded Differences). Modifying any single embedding item draw in any of the t rounds alters the cumulative statistic C_t by at most s, because one embedding item can collide with at most s items in the opposing set for that specific round.

Therefore, setting the distance from the threshold to the expectation as $\epsilon = \frac{ts^2(q_i - r_{ij})}{2}$, the probability that the cumulative statistic deviates past the threshold is bounded by: $Pr(|C_t - \mathbb{E}[C_t]| \geq \epsilon) \leq 2\exp\left(-\frac{2\epsilon^2}{\sum_{k=1}^{2ts} s^2}\right)$. Substituting ϵ and evaluating the sum in the denominator ($2ts \cdot s^2 = 2ts^3$), we get:

$$Pr(\text{error}) \leq 2\exp\left(-\frac{2\left[ts^2(q_i - r_{ij})/2\right]^2}{2ts^3}\right)$$
$$= 2\exp\left(-\frac{ts(q_i - r_{ij})^2}{4}\right)$$

Since this bound applies symmetrically to both false positives (cross-client exceeding Θ_t) and false negatives (same-client falling below Θ_t), the overall classification error decays exponentially with respect to t. □

Proof of Theorem 3 (Co-occurrence Leakage Vulnerability)

Proof. Because Π is co-occurrence preserving, the adversary $\mathcal{A}$ runs the extractor $\mathcal{E}$ to obtain the unordered collections $\mathcal{U}^{(1)}$ and $\mathcal{U}^{(2)}$ from rounds 1 and 2, respectively. The adversary's goal is to find the mapping between the anonymized sets across the two rounds.

Let $U_a^{(1)} \in \mathcal{U}^{(1)}$ be a targeted update belonging to some unknown client P_i. To find P_i's corresponding update in round 2, $\mathcal{A}$ computes the overlap statistic $C(U_a^{(1)}, U_b^{(2)})$ for all candidate sets $b \in [n]$ in $\mathcal{U}^{(2)}$. The adversary links $U_a^{(1)}$ to the candidate $U_{b^*}^{(2)}$ that maximizes the overlap.

By Theorem 1, the probability that the true match (the set truly generated by P_i in round 2) fails the overlap threshold test is bounded by $2\exp\left(-s(q_i - r_{ij})^2/4\right)$. Conversely, the probability that any specific incorrect candidate (generated by some client $P_j \neq P_i$) falsely exceeds the threshold is bounded by the same exponent.

Since there are $n-1$ incorrect candidates in round 2, we apply a union bound. The probability that $\mathcal{A}$ incorrectly links the targeted update is bounded by the probability that the true match fails *or* any of the $n-1$ false matches succeed: $Pr(\text{Linkage Error}) \leq 2n\exp\left(-\frac{s\Delta^2}{4}\right)$ where $\Delta = \min_{i \neq j}(q_i - r_{ij})$. As the sparsity parameter s increases, this error decays exponentially, allowing $\mathcal{A}$ to deterministically link the anonymized updates. □

Proof of Theorem 4 (Union Recoverability)

Proof. We show that the support function $\text{supp}(G^{(t)})$ realizes $\mathcal{F}$.

By definition of the support function, an index $\tau \in \text{supp}(G^{(t)})$ if and only if the aggregated gradient $G^{(t)}[\tau] \neq 0$. By the linearity of the secure aggregation functionality, the global coordinate is the sum of local coordinates: $G^{(t)}[\tau] = \sum_{i=1}^{n} \Delta W_i^{(t)}[\tau]$.

Assume $\tau \notin \bigcup_{i=1}^{n} S_i^{(t)}$. This implies that $\tau \notin S_i^{(t)}$ for all $i \in [n]$. By definition of the sparse update, $\Delta W_i^{(t)}[\tau] = 0$ for every client i. Consequently, $G^{(t)}[\tau] = \sum_{i=1}^{n} 0 = 0$, which means $\tau \notin \text{supp}(G^{(t)})$.

Conversely, assume $\tau \in \bigcup_{i=1}^{n} S_i^{(t)}$. This implies there exists at least one client $k \in [n]$ such that $\tau \in S_k^{(t)}$, meaning $\Delta W_k^{(t)}[\tau] \neq \mathbf{0}$. At probability $1 - \delta_G$, there is no exact cross-client gradient cancellation, then the aggregated value $G^{(t)}[\tau] \neq \mathbf{0}$. Therefore, $\tau \in \text{supp}(G^{(t)})$.

Since containment holds in both directions, we conclude that $P(\text{supp}(G^{(t)}) = \bigcup_{i=1}^{n} S_i^{(t)}) = 1 - \delta_G$. □

Proof of Theorem 5 (Multi-Round Sparse Secure Aggregation Against Malicious Server, Corrupted Clients)

Proof. We prove this in the standard Real/Ideal paradigm using a hybrid simulation argument. Because we assume the underlying LMPSU protocol securely realizes $\mathcal{F}_{LMPSU}$ against a malicious adversary, there exists an active-secure simulator, $\mathcal{S}_{LMPSU}$, capable of extracting corrupted inputs and simulating the adversary's view for a single round of LMPSU encoding.

We construct a multi-round stateful simulator $\mathcal{S}_{MRSA}$ that interacts with the ideal functionality $\mathcal{F}_{MRSA}$ and simulates the real-world view of the malicious adversary $\mathcal{A}$ across all N rounds.

Simulator Construction ($\mathcal{S}_{MRSA}$): $\mathcal{S}_{MRSA}$ initializes the simulated state and runs the following procedure for each training round $t \in \{1, \ldots, N\}$:

1. **Phase 1 Extraction:** When the protocol enters the LMPSU phase of round t, $\mathcal{S}_{MRSA}$ invokes the single-round simulator $\mathcal{S}_{LMPSU}$. The adversary $\mathcal{A}$ interacts with $\mathcal{S}_{LMPSU}$, which extracts the effective active index sets $S_c^{(t)}$ and sparse gradients $\Delta W_c^{(t)}$ submitted by the actively corrupted clients.

2. **Translation to Ideal Functionality:** $\mathcal{S}_{MRSA}$ submits these extracted inputs to the ideal functionality $\mathcal{F}_{MRSA}$ on behalf of the corrupted clients.

3. **Receiving Ideal Output:** $\mathcal{F}_{MRSA}$ computes the exact global output $(U^{(t)}, \Delta^{(t)})$ based on the honest clients' true inputs and the corrupted clients' extracted inputs. $\mathcal{S}_{MRSA}$ receives this output.

4. **Simulating the Server's View:** $\mathcal{S}_{MRSA}$ forwards $(U^{(t)}, \Delta^{(t)})$ back to $\mathcal{S}_{LMPSU}$, which utilizes it to generate a computationally indistinguishable simulated transcript (the blinded aggregated IBLT) for $\mathcal{A}$ for round t.

   We do not need to simulate $\Delta^{(t)}$ as an additional hidden value. The value $\Delta^{(t)}$ is the public aggregate update determined by the aggregation and decoding procedure and used as a round-transition value. If the server is corrupted, it may attempt to replace $\Delta^{(t)}$ with an arbitrary value, or send different values $\Delta_i^{(t)}$ to different honest users. Such behavior does not reveal additional information about the honest users' round-t private inputs, since $\Delta^{(t)}$ is already a public output of the aggregation phase. Moreover, by the Phase 2 consistency requirement, all honest users either receive the same transition value or detect inconsistency and abort. Therefore, modifying $\Delta^{(t)}$ may affect correctness by causing an incorrect model update, state desynchronization, or abort, but it does not invalidate the privacy simulation of the round-t aggregation transcript.

5. **Phase 2 Broadcast Extraction:** After $\mathcal{A}$ locally computes the global model update, it must broadcast the new weights to initiate round $t + 1$. $\mathcal{S}_{MRSA}$ extracts the single, globally consistent payload $W_{server}^{(t+1)}$ that $\mathcal{A}$ commits to for the broadcast phase.

6. **Ideal Output Delivery:** $\mathcal{S}_{MRSA}$ forwards $W_{server}^{(t+1)}$ to $\mathcal{F}_{MRSA}$, instructing the ideal functionality to deterministically deliver this updated model to all surviving honest clients, formally fulfilling the consistency requirement.

7. **Handling Aborts and Dropouts:** If $\mathcal{A}$ deviates in a way that causes $\mathcal{S}_{LMPSU}$ to abort, if $\mathcal{A}$ fails to provide a broadcast model, or if the number of valid extracted client inputs drops below the threshold T, $\mathcal{S}_{MRSA}$ explicitly instructs $\mathcal{F}_{MRSA}$ to output $\perp$ and halt, perfectly matching real-world failure conditions.

Consistency Between Phase 1 and Phase 2. The Phase 2 consistency requirement links the aggregation output of round t to the common state used in round $t + 1$. Its role in the proof is to prevent the malicious server from using inconsistent transition values to create different honest-client views across rounds. Under consistent broadcast, all honest clients either accept the same transition value or abort. Thus, the real execution matches the ideal functionality's behavior: a single public transition value is delivered to all honest clients, or the protocol halts.

Indistinguishability and Sequential Composition. Because Pollen's LMPSU encoding is fundamentally stateless—with clients sampling fresh, independent blinding factors $r_{i,\tau} \leftarrow \text{Unif}(\mathbb{Z}_p)$ in every round—there is no shared ephemeral state or persistent cryptographic masking across rounds.

Thus, the simulated view of round t depends strictly on the ideal output of round t, and standard sequential composition applies natively.

For Phase 1 of any given round, by our premise, the view generated by $\mathcal{S}_{LMPSU}$ is computationally indistinguishable from the real execution. For Phase 2, the ideal deterministic broadcast of $\mathcal{F}_{MRSA}$ mathematically strips the server of the ability to equivocate. Because the real protocol assumes a consistent broadcast channel (preventing Model Inconsistency Attacks), the real-world view exactly matches the ideal-world guarantee where all honest clients update to an identical global state $W_{server}^{(t+1)}$.

By the triangle inequality over the N independent rounds, the total distinguishing advantage of the adversary against the multi-round protocol is bounded by the sum of the single-round advantages: $\text{Adv}_{MRSA} \leq N \cdot (\text{Adv}_{LMPSU} + \text{negl}(\lambda))$ □

Assuming the underlying LMPSU is realized by IBLT-based solution, substituting the single-round bounds for our specific IBLT-based instantiation of LMPSU (from Theorem 10), we obtain:

$$\text{Adv}_{MRSA} \leq N \cdot (\text{Adv}_{SA} + \text{negl}(\lambda))$$

Proof of Theorem 7 (Frequency Hiding)

Proof Sketch. The aggregated count is given by $C_{\text{agg}} = \sum_{i \in \mathcal{P}_\tau} r_{i,\tau} \pmod p$. Since each blinding factor $r_{i,\tau}$ is sampled independently and uniformly from $\mathbb{Z}_p$, their sum modulo p is itself uniformly distributed over $\mathbb{Z}_p$. Thus, C_{agg} is uniformly distributed over $\mathbb{Z}_p$, independently of $|P_\tau|$. With probability $1/p$, $C_{\text{agg}} = 0$, which causes a correctness failure but no additional privacy leakage. This property effectively reduces the Global Multiset Union problem to a Global Set Union problem, thereby preventing frequency-based inference attacks.

We now prove that the proposed blinded arithmetic IBLT protocol securely implements the server-aided labeled MPSU functionality (Figure 3). The proof follows a standard *real/ideal simulation* paradigm and explicitly separates correctness failures from privacy leakage.

View of the Server. In the real execution, the server observes the aggregated IBLT I_{agg}, consisting of bucket tuples (K_j, H_j, C_j, V_j). In the ideal execution, the server receives only the union set U and the aggregated labels $\{\ell(\tau)\}_{\tau \in U}$.

Proof of Theorem 8 (Statistical Security of Blinded IBLT-based MPSU)

Proof. **Simulator Construction.** Given the ideal functionality output $U = \{(\tau, \ell(\tau)) : \tau \in \cup_i S_i\})$, the simulator $\mathcal{S}$ proceeds as follows:

1. For each token $\tau \in \cup_i S_i$, sample an independent aggregate mask $R'_\tau \leftarrow \text{Unif}(\mathbb{Z}_p)$.
2. For each bucket j, let $T_j \subseteq \cup_i S_i$ be the set of tokens hashing to bucket j. The simulator computes

$$C'_j = \sum_{\tau \in T_j} R'_\tau, \quad K'_j = \sum_{\tau \in T_j} R'_\tau \cdot \tau,$$
$$H'_j = \sum_{\tau \in T_j} R'_\tau \cdot \texttt{hash}(\tau), \quad V'_j = \sum_{\tau \in T_j} \ell(\tau)$$

3. Output the simulated aggregated IBLT $I'_{\mathsf{agg}} = \{(K'_j, H'_j, C'_j, V'_j)\}_j$.

Indistinguishability Argument. For every token τ there exists at least one honest client contributing a uniformly random mask $r_{i,\tau} \in \mathbb{Z}_p$. Therefore, the aggregate mask $R_\tau = \sum_i r_{i,\tau}$ is itself uniformly distributed over $\mathbb{Z}_p$. Thus, R_τ and R'_τ are statistically identical. Since the bucket fields (K_j, H_j, C_j, V_j) are linear functions of the aggregate masks $\{R_\tau\}$, the joint distribution of the real aggregated IBLT is identical to that produced by $\mathcal{S}$. Consequently, I_{agg} and I'_{agg} are identically distributed. Thus, the real and ideal views are perfectly indistinguishable. □

Proof of Theorem 9 (Dense-SA-equivalent Aggregate View Under Collusion)

Proof. We consider two cases.

No corrupted clients. If $\mathcal{C} = \emptyset$, then $G_{\mathcal{H}} = G_Q$. Under the no-cancellation assumption, the simulator recovers the labeled union as

$$\mathcal{U}_{\mathcal{H}} = \left\{ \left(\tau, G_{\mathcal{H}}[\tau]\right) : \tau \in \mathrm{supp}(G_{\mathcal{H}}) \right\}.$$

It then invokes the simulator from Theorem 8 to generate an aggregate IBLT distributed identically to $\sum_{i\in\mathcal{H}} I_i$.

Colluding clients. If $\mathcal{C} \neq \emptyset$, the colluding parties' updates and accepted IBLT vectors are contained in their states. Hence, the simulator computes

$$G_{\mathcal{H}} = G_Q - \sum_{i\in\mathcal{C}\cap Q} \Delta W_i$$

and recovers $\mathcal{U}_{\mathcal{H}}$ as above. By Theorem 8, it simulates

$$I'_{\mathcal{H}} \equiv \sum_{i\in\mathcal{H}} I_i$$

and outputs

$$I'_{\mathsf{agg}} = I'_{\mathcal{H}} + \sum_{i\in\mathcal{C}\cap Q} I_i.$$

Because the second term is known from the colluding parties' states, I'_{agg} and I_{agg} are identically distributed.

The simulation is exact conditioned on support recovery. Therefore, if this condition fails with probability at most δ_G, the statistical distance is at most δ_G. □

Proof of Theorem 10 (Privacy against A Malicious Server, Corrupted Clients)

Proof. Fix a PPT adversary $\mathcal{A}$ controlling the server and a set $C \subseteq [n]$ of corrupted clients. Let $H = [n] \setminus C$. Corrupted clients may deviate arbitrarily and submit any vectors accepted by the underlying SA protocol.

For an honest client P_i, let $I_i = \mathsf{Encode}_{\mathsf{pp},\mathsf{sid}}(S_i; \{r_{i,\tau}\}_{\tau\in S_i})$ denote its local IBLT, where each $r_{i,\tau}$ is sampled independently and uniformly from $\mathbb{Z}_p$. A corrupted client's accepted input I_c may be an arbitrary vector of the appropriate dimension.

We construct an ideal-world simulator $\mathcal{S}_{\mathsf{Pollen}}$ for $\mathcal{A}$.

Hybrid H_0: Real execution

This is the real execution of Pollen. Honest clients locally construct their IBLTs and invoke Π_{SA}. If the underlying protocol succeeds for survivor set Q, the server receives $I_{\mathsf{agg},Q} = \sum_{i\in Q} I_i$. It then performs the public IBLT decoding algorithm. If the secure-aggregation execution aborts, Pollen aborts.

Hybrid H_1: Ideal secure aggregation

Replace the execution of Π_{SA} with its ideal functionality $\mathcal{F}_{\mathsf{SA}}^{\Gamma,\mathcal{L}}$ and its active-security simulator $\mathcal{S}_{\mathsf{SA}}$.

The simulator $\mathcal{S}_{\mathsf{Pollen}}$ runs $\mathcal{A}$ internally and forwards all of $\mathcal{A}$'s secure-aggregation actions to $\mathcal{S}_{\mathsf{SA}}$. In particular, $\mathcal{S}_{\mathsf{SA}}$ extracts the effective vectors I_c accepted from corrupted clients.

The ideal SA functionality then behaves according to the instantiated protocol:

- It may produce an abort together with the authorized abort and dropout leakage $\mathcal{L}_{\mathsf{SA}}$; or
- it selects an admissible survivor set $Q \in \Gamma_{\mathsf{SA}}$ and releases $I_{\mathsf{agg},Q} = \sum_{i\in Q} I_i$.

The choice and timing of Q, as well as any abort or output-delivery information, are exactly those permitted by $\mathcal{F}_{\mathsf{SA}}^{\Gamma,\mathcal{L}}$.

By the assumed active security of Π_{SA}, $H_0 \approx_c H_1$, and any distinguisher between these hybrids gives a distinguisher against the underlying secure-aggregation protocol. Therefore, $|\Pr[\mathcal{D}(H_0) = 1] - \Pr[\mathcal{D}(H_1) = 1]| \leq \mathrm{Adv}^{\mathsf{act}}_{\mathsf{SA},\mathcal{B}} + \mathrm{negl}(\lambda)$.

This replacement holds even when the malicious server and corrupted clients choose their messages, inputs, and dropout behavior adaptively within the capabilities represented by the SA functionality.

Simulation after SA. If $\mathcal{F}^{\Gamma,\mathcal{L}}_{\mathsf{SA}}$ aborts, $\mathcal{S}_{\mathsf{Pollen}}$ uses $\mathcal{S}_{\mathsf{SA}}$ to generate the corresponding transcript and exposes exactly the same survivor-set, abort, and delivery information. Since honest IBLT construction is local, there are no additional honest-client messages to simulate.

If the functionality returns $I_{\mathsf{agg},Q}$, the simulator gives this value to the internally executed adversary $\mathcal{A}$. It then lets $\mathcal{A}$ perform the public decoding algorithm and any subsequent local computation.

Everything following secure aggregation is deterministic public post-processing of $I_{\mathsf{agg},Q}$. In particular, the server may:

- decode the IBLT correctly or incorrectly;
- output an arbitrary value;
- replace the decoded aggregate update $\Delta^{(t)}$ with an arbitrary transition value, or attempt to send inconsistent values to different honest clients;
- stop before decoding;
- abort after learning the aggregate;
- withhold any subsequent model broadcast.

None of these actions requires further information about honest clients' inputs. Consequently, the distribution of H_1 is identical to the ideal privacy-only Pollen execution.

It follows that $\mathrm{Adv}^{\mathsf{priv}}_{\mathsf{Pollen},\mathcal{A}} \leq \mathrm{Adv}^{\mathsf{act}}_{\mathsf{SA},\mathcal{B}} + \mathrm{negl}(\lambda)$. Thus, Pollen reveals no information beyond the aggregate, survivor set, abort behavior, output-delivery information, and other leakage already authorized by the instantiated secure-aggregation functionality.

Relation to the LMPSU leakage. By Theorem 8, the aggregate IBLT of the honest survivors $H_Q = H \cap Q$ can be simulated from their labeled union and aggregated labels. The simulator adds the accepted corrupted-client vectors $\sum_{c \in C \cap Q} I_c$, extracted by the SA simulator, to obtain a simulated $I_{\mathsf{agg},Q}$ identically distributed to the real aggregate. Under the union-recoverability result, this honest-survivor LMPSU information is already derivable from the corresponding dense-SA output and corrupted clients' contributions. Therefore, Pollen introduces no additional leakage.

Selective omission merely changes the authorized survivor set Q. If an instantiated SA protocol permits multiple subset outputs or differencing attacks, the corresponding information is part of its authorized leakage and is inherited by Pollen. This theorem makes no robustness or correctness claim. □

For N sequential rounds with distinct session identifiers and fresh masks, the same argument applies by a hybrid over rounds: $\mathrm{Adv}^{N\text{-round}}_{\mathsf{Pollen}} \leq \sum_{t=1}^{N} \left(\mathrm{Adv}^{(t)}_{\mathsf{SA}} + \mathrm{negl}(\lambda) \right)$. This composition does not require the adversary or the rounds to be independent.

F.1 Embedding Cancellation Probability

We bound the probability of cross-client cancellation, the key assumption in Theorem 4.

Theorem 10 (Negligible Exact Cancellation for Quantized Gaussian Embedding Updates)**.** *Let $G^{(t)}[\tau] \in \mathbb{R}^d$ denote the aggregated embedding update vector at index τ. Assume that before quantization, each coordinate $G^{(t)}[\tau]_j$ is an independent Gaussian random variable $G^{(t)}[\tau]_j \sim \mathcal{N}(\mu_j, \sigma_j^2)$, and the implementation stores updates after uniform scalar quantization with step size $\Delta > 0$.*

Let $\widehat{G}^{(t)}[\tau]$ denote the quantized vector, and define the cancellation event $\mathcal{E}_\tau := \{\widehat{G}^{(t)}[\tau] = \mathbf{0}\}$. Then: $\Pr[\mathcal{E}_\tau] \le \left(\frac{\Delta}{\sigma_{\min}\sqrt{2\pi}}\right)^d$, *where* $\sigma_{\min} = \min_j \sigma_j$.

Proof. By definition,

$$G^{(t)}[\tau] = \mathbf{0} \quad \Longleftrightarrow \quad G^{(t)}[\tau]_j = 0 \text{ for all } j \in [d].$$

Since the coordinates are independent and each is distributed in quantized Gaussian distribution over $\mathbb{Z}_p$, for each coordinate j,

$$\Pr\Big[\widehat{G}^{(t)}[\tau]_j = 0\Big] = \Pr\Big[G^{(t)}[\tau]_j \in [-\Delta/2, \Delta/2]\Big] \le \frac{\Delta}{\sigma_j\sqrt{2\pi}}.$$

Consequently, $\Pr[\mathcal{E}_\tau] \le \prod_{j=1}^d \frac{\Delta}{\sigma_j\sqrt{2\pi}} \le \left(\frac{\Delta}{\sigma_{\min}\sqrt{2\pi}}\right)^d$, □