Efficient Distributed Differentially Private Synthetic Data with High Utility via Secure Aggregation

Ratang Sedimo
University of Vermont
rsedimo@uvm.edu

Joseph P. Near
University of Vermont
jnear@uvm.edu

Abstract

We introduce a new approach to differentially private synthetic data generation in federated settings that achieves high utility without relying on a trusted curator. Our method centers on a simple top-k selection mechanism called Distributed Gaussian Top-k that enables clients to collaboratively identify the most informative queries using only secure aggregation and Gaussian noise. Unlike existing approaches such as FLAIM, which suffer from local selection bias, or FHE-based approaches like FedDPSyn, which incur prohibitive computational costs, our protocol strikes a balance between scalability and utility. We develop a distributed variant of the AIM algorithm, Distributed AIM; our empirical evaluation shows that it achieves utility much closer to centralized baselines than prior work, while scaling efficiently to thousands of clients. Our results suggest that lightweight secure aggregation combined with Gaussian noise provides a practical path to high-utility, differentially private synthetic data in federated environments.

1 Introduction

Differential privacy [7, 8] provides a rigorous framework for disclosing statistical information while limiting the risk of individual re-identification. *Differentially private synthetic data* is an important and active research area, and a number of different mechanisms have been developed for this task in the central model of differential privacy [**?**]. The AIM mechanism [16] is a leading example: it *selects* queries using the exponential mechanism, *measures* them privately, and *generates* a synthetic dataset to match the measurement results. AIM achieves strong accuracy but assumes that a centralized server can compute all queries over the raw data. Many other DP synthetic data mechanisms follow the same structure and adjust individual pieces [1, 12, 13, 17, 19].

Federated synthetic data generation requires data holders to collaboratively generate synthetic data without sharing the data itself. Extending approaches like AIM to the federated setting brings significant challenges, and existing solutions struggle to balance trust assumptions, accuracy, and scalability. The *measure* step of mechanisms like AIM can be accomplished using lightweight techniques like secure aggregation [2–4, 23]. However, the *select* step is more difficult, because it fundamentally requires non-linear operations. The FLAIM protocol [14] solves this problem by applying the exponential mechanism locally at each client, which leads to biased selections if the data is not uniformly distributed among the clients. Solutions based on fully-homomorphic encryption (FHE) [24] and secure multiparty computation (MPC) [9] produce unbiased selections, but take hours or days to run.

We propose **Distributed AIM**, a novel protocol leveraging only lightweight cryptography that addresses the biased selection problem. Our approach allows adapting any central-model synthetic data mechanism that follows the select-measure-generate paradigm (e.g. AIM) into a federated version with only a minor reduction in utility and negligible performance overhead. Compared to FLAIM, our approach reduces the utility gap to central-model AIM by 50%; unlike approaches based on heavyweight cryptography [9, 24], it runs in just a few seconds longer than central-model AIM. Our approach **applies to any synthetic data mechanism that follows the select-measure-generate framework**.

To solve the federated selection problem, we propose a simple but effective approach: *just use the Gaussian mechanism for top-k*—an approach that is not optimal from a utility perspective, but avoids bias and is easy to adapt to the federated setting. Our approach, called Distributed Gaussian Top-k, reveals noisy scores for all options via the Gaussian mechanism, and selects the top-k of them. Compared to the exponential mechanism, this approach has reduced utility, but can be easily adapted to the federated setting using lightweight secure aggregation and the results are unbiased, regardless of how data is distributed among the clients.

We evaluate the scalability and utility of Distributed AIM. Following the AIM work [16], we use the UCI Adult dataset to evaluate utility against central-model AIM and FLAIM; our results show that Distributed AIM approaches the accuracy of AIM and significantly outperforms FLAIM. The protocol scales linearly with the number of clients and incurs minimal communication cost; for fewer than 100 clients, it generally takes only a few seconds longer to run than central-model AIM, and it adds overhead of several minutes for fewer than 5000 clients. We will release our implementation as open-source upon publication.

Contributions. In summary, we make three contributions:

- Distributed Gaussian Top-k: A simple and scalable method for distributed unbiased private top-k selection using Gaussian noise and secure aggregation.
- Distributed AIM: A practical synthetic data generation protocol for federated settings that removes the need for a trusted central server.
- Extensive empirical evaluation: We show that our method approaches centralized AIM in utility, exceeds FLAIM under realistic settings, and scales efficiently to large deployments.

2 Background

This section reviews key concepts and prior work that form the foundation of our distributed differentially private synthetic data generation framework.

2.1 Differential Privacy

Differential privacy [7, 8] is a formal privacy definition that prevents observers from learning too much about an individual who participates in a data analysis. Formally:

Definition 1 (Differential privacy). A mechanism $\mathcal{M}$ satisfies (ϵ, δ)-differential privacy if for all neighboring databases $x, x' \in \mathcal{D}$, and for all possible sets of outcomes S:

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

Two databases are neighbors if they differ in one person's data. Here, we use a variant of differential privacy called *zero-concentrated differential privacy* (zCDP) [5]:

Definition 2 (Zero-concentrated differential privacy (zCDP) [5]). A randomized mechanism $\mathcal{M}$ satisfies ρ-zCDP if for all $\alpha \in (1, \infty)$ and all neighboring databases x and x',

$$D_\alpha\big(\, \mathcal{M}(x) \,\big\|\, \mathcal{M}(x') \,\big) \;\leq\; \rho\, \alpha,$$

where D_α is the order-α Rényi divergence.

The Gaussian mechanism for zCDP adds Gaussian noise calibrated to the privacy parameter ρ and the L_2 sensitivity Δ_2:

Definition 3 (Gaussian mechanism for zCDP). Let $q : \mathcal{D} \to \mathbb{R}^d$ have L_2-sensitivity Δ_2. For an input database x, the Gaussian mechanism releases:

$$q(x) + \mathcal{N}(0, \sigma^2) \text{ where } \sigma^2 = \frac{\Delta_2^2}{2\rho}$$

Sensitivity is the maximum change in a function's output when one record differs between two neighboring databases. Sensitivity for vector-valued functions can be defined in terms of L_1 or L_2 norms:

Definition 4 (L_2-sensitivity). For a (vector-valued) query $q : \mathcal{X}^n \to \mathbb{R}^d$, the (global) L_2-sensitivity is

$$\Delta_2(q) \;=\; \sup_{\text{neighbors } x \sim x'} \| \, q(x) - q(x') \, \|_2 \, .$$

If mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_k$ each satisfy ρ_i-zCDP, their composition satisfies $\left(\sum_{i=1}^k \rho_i\right)$-zCDP. This property allows for straightforward cumulative privacy accounting over multiple iterations. A ρ-zCDP mechanism also satisfies (ε, δ)-DP for all $\delta > 0$, where

$$\varepsilon = \rho + 2\sqrt{\rho \log(1/\delta)}.$$

This facilitates interpretation of zCDP bounds within the more familiar (ε, δ)-DP framework. If $\mathcal{M}$ satisfies ρ-zCDP and f is any function, then $f(\mathcal{M}(D))$ also satisfies ρ-zCDP.

2.2 Central-Model DP Synthetic Data: AIM

In AIM [16], the synthetic dataset is constructed iteratively by identifying and measuring the most informative queries under differential privacy, as shown in Algorithm 1. The procedure begins with an initialization phase (lines 1–3), where AIM constructs an empty measurement set, initializes the synthetic distribution $\hat{p}_0$ to be uniform over $\mathcal{X}$, and resets the accumulated privacy expenditure. In Step 1 (*Select*), shown in lines 5–6, AIM computes the utility of each query in the workload by evaluating the discrepancy between the true measurement and the current synthetic estimate, and then privately selects the next informative query q_t via the exponential mechanism. Step 2 (*Measure*), in lines 8–9, privately answers the selected query by adding Gaussian noise according to the annealing schedule and appends the resulting noisy measurement to the measurement set. Step 3 (*Generate*), shown in lines 11–16, updates the synthetic distribution by solving the Private-PGM inference objective, producing a new distribution $\hat{p}_t$ that best fits all previously collected noisy measurements. After completing T iterations of selection, measurement, and generation, line 18 samples a synthetic dataset from the final distribution $\hat{p}_T$, which is returned as the output. The full version adaptively adjusts the privacy budget per iteration using an annealing step.

The select-merge-generate paradigm embodied by AIM represents the state-of-the-art approach for generating synthetic tabular data with DP. Many other algorithms follow the same paradigm, and primarily make changes to the generate step [1, 12, 13, 17, 19].

Parameters:
- Parties: one server S and n clients $c_1, \ldots, c_n$.
- Each client c_i holds a vector $x_i \in \mathbb{F}_p^k$.

Functionality:
(1) Each client sends x_i to $\mathcal{F}_{\text{agg}}$.
(2) $\mathcal{F}_{\text{agg}}$ returns $\sum_{i=1}^n x_i$ to S.

Functionality 1: Secure aggregation functionality $\mathcal{F}_{\text{agg}}$.

Algorithm 1: Central-model AIM (Accurate Iterative Mechanism) [16]

Input : Dataset $\mathcal{D}$ over domain $\mathcal{X}$;
Workload of candidate queries $\mathcal{W}$;
Total privacy budget ρ (or (ε, δ));
Number of iterations T

Output : Synthetic dataset $\hat{\mathcal{D}}$ sampled from $\hat{p}_T$

1 Initialize measurement set $\mathcal{M} \leftarrow \emptyset$;
2 Initialize synthetic distribution $\hat{p}_0$ to uniform over $\mathcal{X}$;
3 **for** $t \leftarrow 1$ **to** T **do**
    `// 1. Select`
4   Choose $q_t \in \mathcal{W}$ using the exponential mechanism on u_t, where
    $$u_t(q) \leftarrow \left\| \mathcal{M}_q(\mathcal{D}) - \mathcal{M}_q(\hat{p}_{t-1}) \right\|_1$$
    `// 2. Measure`
5   $\tilde{y}_t \leftarrow \mathcal{M}_{q_t}(\mathcal{D}) + \mathcal{N}(0, \sigma_t^2 I)$;
6   Add measurement $(q_t, \tilde{y}_t, \sigma_t)$ to $\mathcal{M}$;
    `// 3. Generate`
7   $$\hat{p}_t \leftarrow \arg\min_{p \in \mathcal{P}} \sum_{(q, \tilde{y}, \sigma) \in \mathcal{M}} \frac{1}{2\sigma^2} \left\| \mathcal{M}_q(p) - \tilde{y} \right\|_2^2$$
8 Sample synthetic dataset $\hat{\mathcal{D}}$ from $\hat{p}_T$;
9 **return** $\hat{\mathcal{D}}$;

2.3 Secure Aggregation

Single-server *secure aggregation* is a protocol between a single server and many clients that enables an untrusted server to compute the sum of private client vectors without revealing any individual vector [4]. Modern single-server secure aggregation protocols

scale to millions of clients with minimal communication rounds and lightweight cryptography [2, 3, 23], forming the cornerstone of privacy-preserving federated learning and distributed DP algorithms. As formalized in Functionality 1, the goal of SA is to ensure that the server learns only the aggregate $\sum_{i=1}^{n} x_i$, while each client's vector x_i remains hidden from both the server and other clients.

Many different protocols have been proposed to realize Functionality 1, leveraging different approaches and targeting different threat models. Recent work in this area (e.g. [2, 3]) targets a strong threat model (malicious security, with corrupted server and fraction of the clients) and scales to thousands of clients in seconds. See Section 3 for a discussion of threat models.

2.4 Federated Synthetic Data

Several frameworks have sought to decentralize AIM's architecture. The measurement step can be decentralized using secure aggregation alone, since it relies only on the Gaussian mechanism. However, the use of the exponential mechanism in AIM is more challenging to decentralize: previous attempts to approximate or emulate centralized selection each introduce significant drawbacks. Table 1 summarizes the approaches used in prior work and contrasts them with our approach.

FLAIM. FLAIM [14] adapts AIM for federated settings by using secure aggregation for the measure step, and *local* evaluation of the exponential mechanism at each client for the select step. This approach avoids the complexity of distributed evaluation of the exponential mechanism, but it introduces *local-selection bias* due to data heterogeneity. Consequently, the aggregated selected queries may diverge from globally optimal ones, reducing accuracy. Empirical results suggest that the utility penalty is significant when data is not uniformly distributed among the clients.

FHE-based Approach. FedDPSyn [24] employs fully homomorphic encryption (FHE) to conduct both the select and measure steps on encrypted data. It achieves near-centralized accuracy but at substantial computational cost (up to 20 minutes even on small datasets), making it challenging to deploy in practice.

MPC-based Approach. Fu et al. [9] introduce a multi-party computation (MPC) protocol for vertically partitioned data. Their method runs significantly faster than the FHE-based approach, but is limited to a small set of semi-honest compute servers.

Our work. Our protocol addresses this challenge by combining secure aggregation with a distributed Gaussian Top-k mechanism (Section 4), combining utility and scalability. Notably, our approach (like FLAIM) is designed around *horizontally distributed* data; the FHE- and MPC-based approaches also extend to vertically distributed data, which is not feasible in our approach.

3 Threat Models

We consider two threat models: semi-honest, and a variant of malicious security without correctness associated with single-server secure aggregation protocols. Both settings consider a static set of parties *corrupted* by the adversary: $C \subset \{S\} \cup \{c_1, \ldots, c_n\}$. Our protocols are secure even when both the server S and a θ fraction of the clients are corrupted.

Semi-honest security. In the semi-honest (honest-but-curious) threat model, all parties follow the protocol, and do not change their inputs or outputs. The adversary cannot affect the correctness of the result in this setting, since all parties follow the protocol. However, the adversary may try to learn information about the honest parties' inputs by observing messages received by the corrupt parties. Single-server secure aggregation protocols generally provide semi-honest security when both the server and a fraction of clients are corrupted.

Malicious security. In the malicious (active) threat model, each corrupted party can deviate arbitrarily from the protocol, including by changing their inputs, their message contents, and their outputs. Single-server secure aggregation protocols [2–4] typically do not ensure correctness of their outputs in the presence of a corrupted server in the malicious model, since the server is the single source of the final result and is allowed to output an arbitrary value when corrupted. These protocols do provide confidentiality for the clients' inputs, even in the presence of a malicious adversary. This variant of malicious security is weaker than the traditional definition, which does include correctness of the output. However, this threat model is often considered to be a good match for practical deployments, in which the operator of the server is incentivized to produce the correct result, since their goal is to use the collected statistics for some purpose; the clients, in contrast, primarily want to ensure confidentiality of their inputs, and care less about the utility of the output.

Single- vs. multi-server aggregation. Highly-efficient secure aggregation protocols exist for the multi-server setting, where multiple non-colluding servers collaboratively compute the aggregate result. In practice, however, deployed applications of secure aggregation protocols typically involve collection of data by a single organization, and identifying additional organizations which verifiably do not collude with the first one is difficult. Our presentation and experiments focus on single-server setting, since it is more challenging and typically more applicable than the multi-server setting, but our approach extends in a trivial way to multi-server aggregation protocols.

Input validation. Malicious clients may have no incentive to provide correct inputs to the protocol, and may destroy the final output by providing garbage inputs. Our approach is compatible with existing work that uses a zero-knowledge (zk) proof framework to prove that all clients' inputs are within a reasonable range. Our malicious-secure protocol is compatible with several existing solutions for client input validation, including those proposed by ACORN [3] and EiFFeL [22]. Employing any of these methods ensures that we can guarantee correctness of client inputs, in case some clients behave maliciously.

4 Distributed Top-k Mechanism

Selecting the most informative queries while preserving privacy is a central challenge in differentially private synthetic data generation. In centralized settings, top-k selection is implemented using mechanisms such as the exponential mechanism [18] and its variants, including Report Noisy Max, which perturbs utility scores computed over the entire dataset and returns the k highest noisy values. These methods achieve high utility because the curator has full access to all data.

Approach	Model	Selection	Measurement
AIM [16]	Central	Exponential Mechanism	Gaussian Mechanism
FLAIM [14]	Distributed	Local Exponential Mechanism	Secure Aggregation + Gaussian Mechanism
FedDPSyn [24]	Distributed	FHE-based Exponential Mechanism	FHE-based Gaussian Mechanism
MPC-based [9]	Distributed	MPC-based Exponential Mechanism	MPC-based Gaussian Mechanism
Ours	Distributed	Secure Aggregation + Gaussian Top-k (Section 4)	Secure Aggregation + Gaussian Mechanism

Table 1: Comparison of major approaches for differentially private synthetic data generation. Our method combines global selection accuracy with federated scalability using secure aggregation and distributed Gaussian Top-k selection.

Parameter	Description
S	Server coordinating protocol execution
n	Number of participating clients
$c_1, \ldots, c_n$	Clients holding private datasets
θ	Upper bound on fraction of corrupted clients
γ	Scaling parameter for encoding
T	Total number of measurement rounds
$\mathcal{W}$	Workload (set of candidate queries)
$\lvert\mathcal{W}\rvert$	Total number of queries in the workload
k	Number of top queries selected in each round
d	Domain size or dimension of the histogram

Table 2: List of parameters and notation used in the paper.

Top-k selection is a non-linear function, and is therefore difficult to adapt to a distributed setting. Naïve local selection, where each client independently selects top-k queries from its own data, results in inconsistent and biased outcomes since query utilities vary across the population. This limitation is evident in distributed algorithms such as FLAIM [14].

We propose the **Distributed Gaussian Top-k (DGT-k)** mechanism, which performs global top-k selection in the distributed setting by simply releasing all of the utility scores with Gaussian noise. This approach results in reduced utility compared to central-model mechanisms, but improved utility compared to local selection. We first describe central-model top-k selection and the challenges of adapting it to the distributed setting, then present our approach and prove its security and privacy properties.

4.1 Background: Central-Model Top-k Selection

Let $\mathcal{D}$ be a dataset, $\mathcal{R}$ be a set of potential outputs, and $u(\mathcal{D}, r)$ be a utility function that produces a real-valued score for output $r \in \mathcal{R}$ on dataset $\mathcal{D}$. The exponential mechanism [18] is a generic tool for selecting high-utility outputs in a differentially private manner. It defines a randomized mechanism $\mathcal{M}$ that selects output $r \in \mathcal{R}$ with probability:

$$\Pr[\mathcal{M}(\mathcal{D}, u, \mathcal{R}) = r] \propto \exp\left(\frac{\epsilon \cdot u(\mathcal{D}, r)}{2\Delta u}\right)$$

where Δu is the global sensitivity of u, typically $\Delta u = 1$ if each individual can change a score by at most 1.

For top-k selection, we iteratively apply the exponential mechanism k times, removing the previously selected queries at each step. This yields a k-length ordered list of queries likely to have the highest true values while satisfying differential privacy, where:

$$\mathcal{M}_{\text{top-}k}(\mathcal{D}, u, \mathcal{R}) = \begin{cases} \text{Select } r_1 \text{ using } M(x, u, R), \\ \text{Select } r_2 \text{ using } M(x, u, R \setminus \{r_1\}), \\ \vdots \\ \text{Select } r_k \text{ using } M(x, u, R \setminus \{r_1, \ldots, r_{k-1}\}). \end{cases}$$

This approach (sometimes called *peeling*) applies the exponential mechanism k times with zCDP noise calibrated to parameter ρ'. By composition, the total mechanism satisfies $k\rho'$-zCDP. In the central model, more complicated alternatives can be simpler to implement (e.g. they run in one iteration) but produce roughly similar utility [6, 10, 21, 25].

4.2 Challenges of Distributed Top-k Selection

Implementing differentially private top-k query selection in a distributed setting presents a key technical challenge: the top-k operation is inherently *non-linear* and not directly compatible with the operations supported by secure aggregation protocols. These protocols are fundamentally designed for summing vectors, and they do not support comparisons or conditional logic over encrypted or secret-shared data. This leads to three possible design options for distributed top-k selection:

(1) **Local selection with the exponential mechanism** (as in FLAIM [14]): Each client privately selects a query using the exponential mechanism based on its local data. This method is lightweight and fast but introduces *local-selection bias* when data is heterogeneously distributed, which degrades utility.
(2) **Global selection via secure MPC [9] or FHE [24]**: Heavyweight cryptography can evaluate the exponential mechanism directly, but is computationally expensive and impractical for large-scale deployments.
(3) **Our approach: secure aggregation of noisy scores**: Our solution is to approximate the exponential mechanism using the Gaussian mechanism, which enables the use of lightweight secure aggregation protocols for top-k selection.

4.3 Central-Model Gaussian Top-k

To illustrate our approach, we describe a central-model version first (Algorithm 2); we implement the same mechanism in a distributed protocol in Section 4.6. The mechanism computes utility scores, adds Gaussian noise to the vector of scores, and then publishes the whole list of noisy scores. Then, the mechanism picks the top-k by post-processing the vector of noisy scores. The mechanism is a trivial extension of the Gaussian mechanism and satisfies ρ-zCDP.

Algorithm 2: Central-model Gaussian Top-k Mechanism

Input : Dataset $\mathcal{D}$, utility function u with sensitivity Δ, set of outputs $\mathcal{R}$, privacy parameter ρ, number of desired results k

Output : Indices I_k of the top-k queries

$\sigma^2 \leftarrow |\mathcal{R}|\Delta^2/(2\rho_i)$;
for $r_i \in \mathcal{R}$ **do**
  $s_i \leftarrow u(\mathcal{D}, r_i)$; // Utility of r_i
  $\tilde{s}_i \leftarrow s_i + \mathcal{N}(0, \sigma^2)$; // Add Gaussian noise
$S \leftarrow \langle \tilde{s}_1, \ldots, \tilde{s}_m \rangle$;
$I_k \leftarrow \mathrm{TopK}(S, k)$; // Indices of top-$k$ noisy scores
return I_k

This method differs from the exponential mechanism-based top-k selection discussed earlier in an important way: instead of selecting high-utility queries probabilistically in a privacy-preserving way, it privately releases *all* noisy scores and then selects the top-k deterministically. Because it releases all of the scores, the amount of noise (line 1) grows with $\sqrt{|\mathcal{R}|}$ (not k, as in top-k via the exponential mechanism). For large $\mathcal{R}$ and small k, this mechanism can be significantly worse than the traditional approach.

However, this approach has a significant advantage over the exponential mechanism: its results can be computed by linear operations on the scores, with selection as a post-processing step. For linear utility functions, this design immediately leads to a distributed version of the mechanism based on secure aggregation, as described in Section 4.6.

4.4 Encoding

To use a secure aggregation protocol, we need to transform the inputs into elements of a finite field ($\{0, \ldots, p-1\}$ for some prime p). We use a simple encoding algorithm that converts a client's real-valued measurement vector into a finite-field representation suitable for secure aggregation. Given a measurement vector $v \in \mathbb{R}^k$ produced by the HDMM (High-Dimensional Matrix Mechanism) measurement matrix, the encoder performs the following operations (Algorithm 3, lines 1–3): (i) scales and quantizes the entries (line 1), (ii) adds discrete Gaussian noise calibrated to the privacy parameter ρ (line 2), and (iii) maps the noisy result into $\mathbb{F}_p$ (line 3). The resulting encoded vector $\tilde{v} \in \mathbb{F}_p^k$ is then transmitted to the secure aggregation functionality $\mathcal{F}_{\text{agg}}$.

4.5 Decoding

The decoding algorithm inverts the encoding process described in Section 4.4, transforming the aggregated result—a vector of finite field elements—back into real-valued measurements. Given an aggregated vector $v_s \in \mathbb{F}_p^k$, the decoder first converts each element from its finite field representation to a signed integer, and then de-scales it to recover the original floating-point magnitude. Each field element $x \in \mathbb{F}_p$ is interpreted as a signed integer in the range $\left[-\frac{p-1}{2}, \frac{p-1}{2}\right]$ by applying the inverse of the integer-to-field mapping used in the encoding phase (Algorithm 4). The resulting integer vector $x' \in \mathbb{Z}^k$ is then divided by the public scaling factor γ to restore the floating-point representation.

Algorithm 3: Encoding procedure for a client's measurement vector.

Input : $v \in \mathbb{R}^k$, scaling factor γ, L_2 sensitivity Δ_2, privacy parameter ρ

Output : $\tilde{v} \in \mathbb{F}_p^k$

$v_s \leftarrow \lfloor \gamma v \rfloor$; // Scale and truncate
$\hat{v}_s \leftarrow v_s + \mathcal{N}_{\mathbb{Z}}\left(0, \frac{\gamma^2 \Delta_2^2}{2(1-\theta)n\rho} \mathbf{I}_k\right)$; // Add noise elementwise
$\tilde{v} \leftarrow \hat{v}_s \bmod p$; // Map into $\mathbb{F}_p^k$
return $\tilde{v}$

Algorithm 4: Decoding procedure

Input : Encoded vector $v_s \in \mathbb{F}_p^k$, scaling factor γ

Output : Decoded vector $v \in \mathbb{R}^k$

for $i \leftarrow 1$ **to** k **do**
  if $v_s[i] \leq \frac{p-1}{2}$ **then**
    $x \leftarrow v_s[i]$;
  else
    $x \leftarrow v_s[i] - p$;
  end
  $v[i] \leftarrow x/\gamma$; // De-scale to recover measurement
end
return v;

4.6 Distributed Gaussian Top-k

Protocol 1 outlines our Distributed Gaussian Top-k mechanism. The mechanism requires only a single use of a secure aggregation protocol. In Step 1, each client computes a local utility score vector using the utility function u. In Step 2, each client uses the encode procedure to encode their score vector and add discrete Gaussian noise. This noise does not guarantee local differential privacy (the scale of the noise is not sufficient), but when aggregated with the other clients' vectors, will be sufficient to ensure differential privacy. In Step 3, the clients and server run the secure aggregation functionality $\mathcal{F}_{\text{agg}}$ to sum their vectors. This functionality can be instantiated with any secure aggregation protocol with the desired threat model; our experiments evaluate the Bonawitz protocol [4] and the Bell protocol [2]. In step 4, the server decodes the result of the sum, which corresponds to the global score vector for all options. In Step 5, the server determines the top-k elements by post-processing.

4.7 Privacy Guarantee

The Distributed Gaussian Top-k protocol satisfies ρ-zCDP, where neighboring datasets differ in one row of one client's data. However, the noise is not added in a single step, as in Algorithm 2; instead, each client adds a small amount of noise, and the sum of the noisy vectors must satisfy zCDP. This approach is necessary because both the server and a fraction of the clients may be corrupted, so we cannot rely on any single party to sample the noise and keep it secret. By distributing noise generation among the clients, who are

Protocol: Secure Aggregation-Based Gaussian Top-k Selection

Parties:

- n clients: each holds a local dataset
- One central server

Inputs:

- A public set of possible outputs $\mathcal{R} = r_1, \ldots, r_m$
- A public utility function $u : \mathcal{D} \times \mathcal{R} \to \mathbb{R}$
- For each client i, a private dataset D_i
- Privacy parameter ρ (zCDP)
- Public scaling factor γ and finite field modulus p from the encoding scheme

Steps:

(1) **Local Score Computation:** Each client c_i computes their local score vector
$$s_i = \langle u(D_i, r_1), \ldots, u(D_i, r_m) \rangle.$$
(2) **Encoding and Local Noise Addition:** Apply the encoding procedure from Section 4.4:
$$\hat{s}_i \leftarrow \mathsf{encode}(v_s, \gamma, \Delta, \rho).$$
where Δ is the sensitivity of u.
(3) **Secure Aggregation:** All clients run $\mathcal{F}_{\text{agg}}$ to compute the elementwise sum in $\mathbb{F}_p^m$:
$$y \leftarrow \sum_{j=1}^{n} \hat{s}_j \bmod p$$
(4) **Decoding:** The server applies the decoding procedure from Section 4.5 to recover the aggregated noisy scores in $\mathbb{R}^m$:
$$\bar{s} \leftarrow \mathsf{decode}(y).$$
(5) **Top-k Selection (Post-processing):** The server computes
$$I_k = \mathrm{TopK}(\bar{s}, k),$$
where ties among identical noisy scores are resolved using a deterministic rule.

Output: The server learns only the global top-k query indices I_k, without observing any individual scores or local datasets.

Protocol 1: $\Pi_{\text{top-k}}$: Distributed Gaussian Top-k via Secure Aggregation, integrating the encoding and decoding procedures to maintain ρ-zCDP privacy while ensuring correct finite-field aggregation.

mostly honest, we ensure that the noise is sufficient for the privacy guarantee. Specifically, each client adds discrete Gaussian noise scaled to:
$$\frac{\gamma^2 \Delta_2^2}{2(1-\theta)n\rho}$$
Where there are n clients and at most θn clients are corrupted. If no clients are corrupted, and we used *continuous* Gaussian noise, then the sum of the noisy submissions would satisfy ρ-zCDP trivially by the Gaussian mechanism. Since we use *discrete* Gaussian noise (required due to the encoding of inputs as field elements), the analysis is not quite so simple.

The *distributed discrete Gaussian* mechanism [11] addresses precisely this issue. That result showed that the sum of distributed discrete Gaussian noise satisfies zCDP in a similar way to the continuous case, with a small added privacy cost.

Lemma 5 (Distributed discrete Gaussian [11]). *Let $\sigma \geq 1$ and $X_i \sim \mathcal{N}_{\mathbb{Z}}(0, \sigma^2)$ independently for each $i \in [n]$. Let $Z_n = \sum_{i=1}^{n} X_i$. An algorithm that adds Z_n to a sensitivity-Δ query satisfies ρ'-zero concentrated differential privacy, where:*
$$\rho' = \frac{\Delta^2}{2n\sigma^2} + 5 \sum_{k=1}^{n-1} e^{-4\pi^2\sigma^2 \frac{k}{k+1}}.$$

We use this result to prove that $\mathcal{F}_{\text{Top-}k}$ (Functionality 2), the ideal functionality corresponding to the protocol $\Pi_{\text{Top-}k}$, satisfies zCDP. We prove correspondence between the protocol and functionality in the next section.

Theorem 6 (Privacy of $\mathcal{F}_{\text{Top-}k}$). *The functionality $\mathcal{F}_{\text{Top-}k}$ (Functionality 2) satisfies ρ'-zCDP under row-level adjacency, where $\rho' = \rho + \eta$ and η is negligible in γ.*

Proof. We proceed in four steps.

(1) **Sensitivity.** $\mathcal{F}_{\text{Top-}k}$ computes a local score vector $s_i = g(D_i)$ with global ℓ_2-sensitivity Δ.
(2) **Noise addition.** During the encode phase, $\mathcal{F}_{\text{Top-}k}$ adds discrete Gaussian noise via encode with variance:
$$\sigma^2 = \frac{\gamma^2 \Delta_2^2}{2(1-\theta)n\rho}$$
(3) **Aggregation.** $\mathcal{F}_{\text{top-}k}$ sums the noisy score vectors to obtain $y = \sum_{j=1}^{n} \hat{s}_j \bmod p$. In the worst case, the server can subtract the noise of the $n\theta$ corrupted clients, so $n(1-\theta)$ noise samples remain. By Lemma 5, y satisfies ρ'-zCDP, where
$$\begin{aligned}\rho' =& \frac{\Delta^2}{2(1-\theta)n\sigma^2} + 5 \sum_{k=1}^{(1-\theta)n-1} e^{-4\pi^2\sigma^2 \frac{k}{k+1}} \\ =& \frac{\Delta^2}{2(1-\theta)n\sigma^2} + \eta \\ =& \frac{\gamma^2\Delta_2^2}{2(1-\theta)n\frac{\gamma^2\Delta_2^2}{2(1-\theta)n\rho}} + \eta \\ =& \rho + \eta\end{aligned}$$
and η is a small additional privacy cost that shrinks exponentially fast with σ^2.
(4) **Post-processing.** The subsequent top-k selection and reporting of indices I_k and noisy measurements $\tilde{y}$ are post-processing operations and do not degrade the privacy guarantee by the post-processing property of zCDP.

□

The η value in the definition of ρ' represents the extra cost of using discrete Gaussian noise rather than continuous noise. However, the extra cost shrinks exponentially with the noise scale σ^2; in our setting, σ^2 depends on the scaling parameter γ, so η can easily

Ideal Functionality: $\mathcal{F}_{\text{Top-}k}$: Gaussian Top-$k$
Parties:

- n clients: each holds a local dataset
- One central server
- $\mathcal{F}_{\text{Top-}k}$, the functionality

Inputs:

- A public set of possible outputs $\mathcal{R} = r_1, \ldots, r_m$
- A public utility function $u : \mathcal{D} \times \mathcal{R} \to \mathbb{R}$
- For each client i, a private dataset D_i
- Privacy parameter ρ (zCDP)
- Public scaling factor γ and finite field modulus p from the encoding scheme

Steps:

(1) **Score Computation:** Each client c_i submits its data D_i to $\mathcal{F}_{\text{Top-}k}$. $\mathcal{F}_{\text{Top-}k}$ computes the vector of scores for the datasets:
$$s_i = \langle u(D_i, r_1), \ldots, u(D_i, r_m) \rangle$$
for each i.
(2) **Encoding and Local Noise Addition:** $\mathcal{F}_{\text{Top-}k}$ applies the encoding procedure from Section 4.4:
$$\hat{s}_i \leftarrow \mathsf{encode}(s_i, \gamma, \Delta, \rho).$$
for each i, where Δ is the sensitivity of u.
(3) **Aggregation:** $\mathcal{F}_{\text{Top-}k}$ computes:
$$y \leftarrow \sum_{j=1}^{n} \hat{s}_j \bmod p$$
(4) $\mathcal{F}_{\text{Top-}k}$ sends y to the server
(5) **Decoding:** The server applies the decoding procedure from Section 4.5 to recover the aggregated noisy scores in $\mathbb{R}^m$:
$$\bar{s} \leftarrow \mathsf{decode}(y).$$
(6) **Top-k Selection (Post-processing):** The server computes
$$I_k = \mathrm{TopK}(\bar{s}, k),$$
where ties are resolved using a deterministic rule.

Output: The server learns only the global top-k query indices I_k, without observing any individual scores or local datasets.

Functionality 2: Top-k ideal functionality $\mathcal{F}_{\text{Top-}k}$.

be made negligible by setting γ large enough. For example, when $\rho = 0.1$ and $n = 5000$, setting $\gamma = 100$ results in $\eta = 9.39 \times 10^{-86}$. In our experiments, we set $\gamma = 1000$, which results in a η that is too small to calculate using 64-bit floating-point numbers.

4.8 Security Guarantee

We prove confidentiality of client inputs for our protocols using the standard ideal/real methodology. We first formalize the ideal functionality that captures the *intended* leakage, then show that our concrete protocols leak nothing beyond the ideal outputs via black-box simulators. The ideal functionality appears in Functionality 2; it has identical structure to the protocol, but it uses a trusted third party named $\mathcal{F}_{\text{Top-}k}$ to perform the computation.

Adversarial Model and Views. We consider a static adversary controlling the server S and an arbitrary subset $C \subseteq [n]$ of clients (with $|C| \le \theta n$). In the *semi-honest* model, corrupted parties follow the protocol but try to learn extra information. In the *malicious (no correctness)* model, corrupted parties may deviate arbitrarily; we only prove confidentiality of honest inputs. We denote a party's *view* by all its random coins and received messages. We prove security using a standard simulator argument.

Definition 7 (Real/Ideal Indistinguishability). A real protocol Π securely realizes an ideal functionality $\mathcal{F}$ in the presence of a static adversary class $\mathcal{A}$ if for every PPT real adversary $\mathcal{A}$ there exists a PPT simulator Sim such that for all inputs $(D_1, \ldots, D_n)$ and all environments $\mathcal{Z}$,
$$\left| \Pr[\mathsf{Real}_{\Pi,\mathcal{A},\mathcal{Z}} = 1] - \Pr[\mathsf{Ideal}_{\mathcal{F},\mathsf{Sim},\mathcal{Z}} = 1] \right| \le \mathrm{negl}(\lambda),$$
where λ is the security parameter.

Theorem 8 (Confidentiality of Distributed Gaussian Top-k). *$\Pi_{\text{Top-}k}$ (Protocol 1) securely realizes $\mathcal{F}_{\text{Top-}k}$ (Functionality 2) in the $\mathcal{F}_{agg}$-hybrid model against any static adversary corrupting S and up to a θ-fraction of clients, in the semi-honest model and in the malicious model without correctness.*

Proof. We construct a simulator Sim assuming a corrupted server and an arbitrary corrupted fraction θ of clients. Given the ideal output I_k (and public parameters), Sim proceeds in two steps.

(1) Simulating secure aggregation transcripts. By the security of the secure aggregation protocol used, there exists $\mathsf{Sim}_{\text{agg}}$ that, given only the aggregate sum, generates a view computationally indistinguishable from the real execution (including dropouts and recovery) for the secure aggregation protocol. Sim provides the value of y from the ideal functionality to $\mathsf{Sim}_{\text{agg}}$ to generate the transcript for the secure aggregation step.

*(2) Simulating post-processing (*TopK*).* The additional server-side computation (e.g. decoding and TopK) can be simulated as a function of the aggregate y. □

Our proofs establish *confidentiality* of honest inputs against a server and a θ-fraction of corrupted clients, both in the semi-honest model and in a malicious model *without correctness*. Because a malicious single server can output an arbitrary value, we do not claim end-to-end correctness in that model (Section 3). When desired, ACORN-style input validation [3] and audit trails can strengthen robustness against malformed client inputs; these are fully compatible with our approach. The approach could also be extended with verifiability for the output, by asking the server to construct a zero-knowledge proof demonstrating that the result is consistent with the protocol specification.

5 Distributed AIM

We now present Distributed AIM, a lightweight protocol that achieves utility similar to the central-model AIM algorithm without the need for a trusted curator. By integrating the Distributed Gaussian Top-k mechanism introduced in Section 4, clients collaboratively run the selection step of AIM using secure aggregation. This design

Parties: A server S and clients $C_1, \ldots, C_n$, where client C_i holds dataset D_i.
Input:

- Each client C_i holds a private dataset $D_i \in \mathcal{X}^m$.
- A fixed query workload $\mathcal{W}$ shared among all parties.
- A target privacy parameter ρ.
- Algorithm parameters: MAX-SIZE, number of iterations T, and decay rate α.

Functionality:

(1) **Initialization:** Initialize the synthetic distribution $\hat{p}_0$ to uniform.
(2) **For** $t \in 1 \ldots T$ **iterations do:**
  (a) **Query Selection via Top-k:** The clients and server invoke $\mathcal{F}_{\text{Top-}k}$ (Functionality 2) with privacy parameter $\rho/(2T)$ to select query r_t from workload $\mathcal{W}$. The utility score is:

$$q_r(D) = w_r \cdot \|\mathcal{M}_r(D) - \mathcal{M}_r(\hat{p}_{t-1})\|_1 - \sqrt{\tfrac{2}{\pi}} \cdot \sigma_t \cdot n_r.$$

  (b) **Private Measurement (with Encoding):** For the selected query r_t, the clients and server invoke $\mathcal{F}_{\text{agg}}$. Each client computes a noisy encoded input:

$$y_i = \text{encode}(r_t(D_i), \gamma, \Delta_2(r_t), \rho/(2T))$$

  $\mathcal{F}_{\text{agg}}$ outputs $y = \sum_i y_i$ to S.
  (c) **Decoding and Private PGM Update:** S computes $\tilde{y} = \text{decode}(y, \gamma)$, then updates the synthetic distribution via Private-PGM:

$$\hat{p}_t = \arg\min_{p \in S} \sum_{i=1}^{t} \frac{1}{2} \left(\mathcal{M}_{r_i}(p) - \tilde{y}^i \right)^2 / \sigma_i^2.$$

(3) **Output:** At termination, S outputs the final synthetic dataset $\hat{\mathcal{D}}$ sampled from $\hat{p}_T$.

Protocol 2: Π_{AIM}: Distributed AIM Protocol

Parties: A set of n clients $C_1, \ldots, C_n$ and a server S.
Input:

- Each client C_i holds a private dataset $D_i \in \mathcal{X}^m$.
- A fixed query workload $\mathcal{W}$ shared among all parties.
- A target privacy parameter ρ.
- Algorithm parameters: MAX-SIZE, number of iterations T, and decay rate α.

Functionality:

(1) **Initialization:** Initialize the synthetic distribution $\hat{p}_0$ to uniform.
(2) Each client C_i sends their data D_i to the functionality $\mathcal{F}_{\text{AIM}}$
(3) **For** $t \in 1 \ldots T$ **iterations do:**
  (a) **Query Selection via Top-k:** $\mathcal{F}_{\text{AIM}}$ invokes $\mathcal{F}_{\text{Top-}k}$ (Functionality 2) with privacy parameter $\rho/(2T)$, acting as both clients and server, and providing the D_is as the client inputs, to select query r_t from workload $\mathcal{W}$. The utility score is:

$$q_r(D) = w_r \cdot \|\mathcal{M}_r(D) - \mathcal{M}_r(\hat{p}_{t-1})\|_1 - \sqrt{\tfrac{2}{\pi}} \cdot \sigma_t \cdot n_r.$$

  (b) $\mathcal{F}_{\text{AIM}}$ sends r_t to S
  (c) **Private Measurement (with Encoding):** For the selected query r_t, $\mathcal{F}_{\text{AIM}}$ computes:

$$y = \sum_i \text{encode}(r_t(D_i), \gamma, \Delta_2(r_t), \rho/(2T))$$

  (d) $\mathcal{F}_{\text{AIM}}$ sends y to S
  (e) **Decoding and Private PGM Update:** S computes $\tilde{y} = \text{decode}(y, \gamma)$, then updates the synthetic distribution via Private-PGM:

$$\hat{p}_t = \arg\min_{p \in S} \sum_{i=1}^{t} \frac{1}{2} \left(\mathcal{M}_{r_i}(p) - \tilde{y}^i \right)^2 / \sigma_i^2.$$

(4) **Output:** At termination, S outputs the final synthetic dataset $\hat{\mathcal{D}}$ sampled from $\hat{p}_T$, and $\mathcal{F}_{\text{AIM}}$ outputs all messages sent to S.

Functionality 3: Ideal functionality $\mathcal{F}_{\text{AIM}}$ for Distributed AIM, mirroring the structure of Protocol 1. All steps are performed jointly by the functionality, which represents the idealized secure aggregation and noise addition of the clients.

ensures that query selection is unbiased regardless of how data is distributed among clients, and uses only lightweight cryptography.

5.1 Protocol Description

Protocol 2 describes the Distributed AIM protocol; the ideal functionality appears in Functionality 3. The protocol follows the standard select-measure-generate paradigm of the AIM mechanism, but uses our distributed top-k protocol for the select step and secure aggregation for the measure step.

Step 2a implements the select step of AIM, by calling $\mathcal{F}_{\text{Top-}k}$ to select the query with the highest error. Step 2b implements the measure step of AIM, by encoding the result of local measurement using the encode procedure from Section 4.4 and invoking $\mathcal{F}_{\text{agg}}$. Step 2c implements the generate step of AIM, by asking the server to decode the measurement results and update the model. After T iterations, the server outputs the synthetic data.

5.2 Privacy Guarantee

The Distributed AIM protocol satisfies ρ/T-zCDP for each iteration. Each iteration of the protocol spends privacy budget on the select step and on the measure step; the total cost of all iterations is ρ-zCDP by composition.

Theorem 9. *$\mathcal{F}_{AIM}$ satisfies ρ-zCDP.*

Proof. We consider each step of Functionality 3:

- In each iteration, the select step satisfies $\rho/(2T)$-zCDP by Theorem 6
- In each iteration, the measure step satisfies $\rho/(2T)$-zCDP by Lemma 5
- The generate step is post-processing and consumes no additional budget

After T iterations, the total privacy cost is ρ. □

5.3 Security Guarantee

We prove security of the Distributed AIM protocol under the standard ideal/real simulation paradigm. In particular, we show that Π_{AIM} (Protocol 2) securely realizes the ideal functionality $\mathcal{F}_{\text{AIM}}$ (Functionality 3), in the same way as our security proof for the Distributed Gaussian Top-k protocol in Section 4.8.

Theorem 10 (Security of Π_{AIM}). *The protocol Π_{AIM} securely realizes the functionality $\mathcal{F}_{AIM}$ (Functionality 3) in the $\mathcal{F}_{agg}$, $\mathcal{F}_{\text{Top-}k}$-hybrid model.*

Proof. We construct a simulator Sim assuming a corrupted server and an arbitrary corrupted fraction θ of clients. The ideal functionality outputs the final synthetic data $\hat{\mathcal{D}}$, and the intermediate measurements $y_1, \ldots, y_T$. Since the functionality outputs these intermediate (differentially private) results, we can essentially rely on the sequential composition of the sub-protocols. By their security properties, we assume simulators $\text{Sim}_{\text{Top-}k}$ and Sim_{agg} exist for the top-k and secure aggregation protocols. The simulator Sim simulates the transcript for each iteration i of the mechanism by passing y_i as the ideal output to Sim_{agg}, and passing the selected query r_t to $\text{Sim}_{\text{Top-}k}$. □

6 Evaluation

Our empirical evaluation sees to answer four evaluation questions:

- **Q1**: How does the utility of the Distributed Gaussian Top-k mechanism compare to the exponential mechanism?
- **Q2**: How does the utility of Distributed AIM compare to central-model AIM and prior distributed protocols?
- **Q3**: What is the performance overhead of Distributed AIM, and how well does it scale?
- **Q4**: How do network conditions affect the performance of Distributed AIM?

Implementation & experiment setup. We implement our mechanisms in Python, leveraging the Olympia simulator for large-scale experiments on distributed protocols [20]. We use the Private PGM implementation of AIM [15]. We evaluate our mechanisms on synthetic data and (following McKenna et al. [16]) the UCI Adult dataset.

6.1 Q1: Utility of Distributed Gaussian Top-k

Experimental Setup. We evaluate the utility of the Distributed Gaussian top-k mechanism using synthetic data and the Adult dataset. We vary $k \in \{1, \ldots, 10\}$ and compute average error, measuring how well each mechanism identifies the true top-k elements under differential privacy. We measure accuracy using the Jaccard similarity between the true top-k set T and the mechanism's output A:

$$U(T, A) = \frac{|T \cap A|}{|T \cup A|}.$$

Results. The results appear in Figure 1 (Adult dataset) and 2 (synthetic data). For small values of ϵ and low k, as expected, the Distributed Gaussian Top-k mechanism delivers worse utility than central-model top-k. As k increases, however, their utilities become similar. Our results suggest that when the domain of options is moderate (e.g. $\mathcal{R} < 1000$) and k is significantly greater than 1 (e.g. $k = 5$ or $k = 10$), the Distributed Gaussian Top-k mechanism can achieve utility similar to the central-model mechanism.

6.2 Q2: Utility of Distributed AIM

Experiment Setup. We evaluate Distributed AIM on the `Adult` dataset under two adversarial settings: low corruption ($\theta = 0.05$) and high corruption ($\theta = 0.33$). We compare against three baselines: the original central-model AIM mechanism, AugFLAIM (private) [14], and Naive FLAIM [14], a simpler version of FLAIM without heuristics to correct for the bias of local selection. All methods use the same total privacy budget. We vary the value of k used in Distributed AIM to select multiple queries at once.

Results. Figure 3 reports ℓ_2 error as a function of k. The utility is similar overall for both corruption settings, and improves significantly over the utility of FLAIM for all values of k. A utility gap to central-model AIM remains for all the values of k we tested, due to the lower utility of the Distributed Gaussian Top-k mechanism compared to the central-model exponential mechanism.

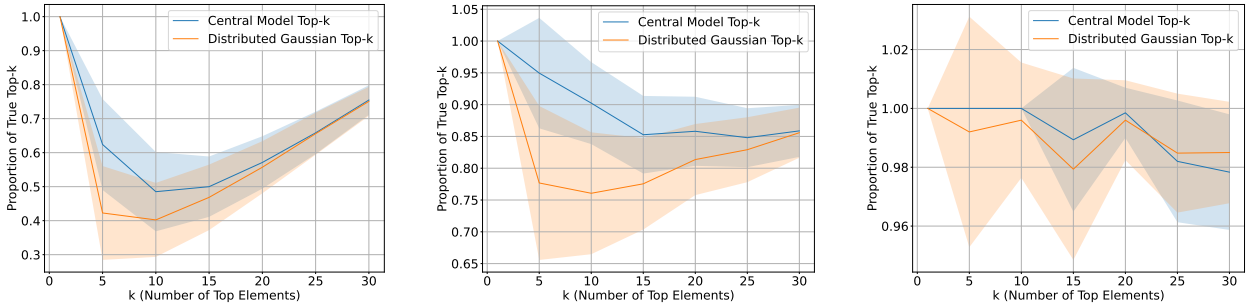
We distributed data uniformly among the clients and do not vary this setting in our experiments, since both Distributed AIM and central-model AIM are not affected by this setting. With greater heterogeneity in data distribution, however, FLAIM's utility degrades further [14].

6.3 Q3: Scalability of Distributed AIM

Experiment Setup. We evaluate the scalability of Distributed AIM under realistic federated environments where the number of clients may grow to the thousands. We consider two experiments: (1) varying both the number of selected queries k and the number of clients, and (2) varying only the number of clients (from 1,000 to 5,000) while holding k and bandwidth constant. Our goal is to measure how end-to-end runtime grows with client population, and whether the protocol remains efficient at federated scale.

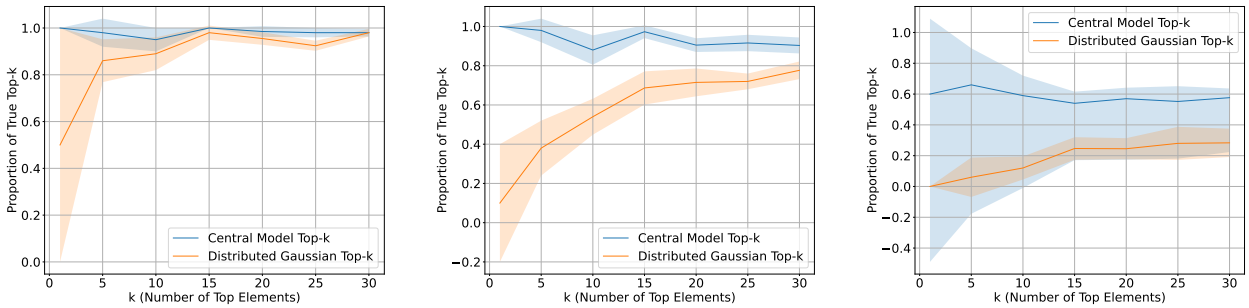
Results. Figure 5 reports runtime as the number of clients increases to 100, using the Bonawitz protocol [4] for secure aggregation. Figure 7 reports runtime for up to 5000 clients, using the Bell protocol [2] for secure aggregation (since it better supports thousands or millions of clients). For small values of k and fewer than 100 clients, Distributed AIM actually runs *faster* than central-model AIM, because it selects multiple queries in each iterations and therefore exhausts the privacy budget in fewer total iterations. The growth pattern matches the behavior of the underlying secure-aggregation primitives: using the Bonawitz protocol, runtime scales linearly in the number of clients; using the Bell protocol, runtime grows logarithmically due to its tree-based aggregation structure. In both cases, end-to-end overhead remains small. Even at 5,000 clients, Distributed AIM adds under five minutes of additional time relative to centralized AIM, demonstrating that our approach scales to a huge number of clients while retaining practical running time.

Figure 4 shows the relative contribution of the AIM algorithm (mostly the generate step) against the running time of the secure aggregation protocols used in our approach, to visualize the additional overhead of our distributed approach. The results show that the additional overhead is minimal, and the primary source of running time is the underlying AIM algorithm rather than the distributed components.



$\epsilon = 10.0$ $\epsilon = 1.0$ $\epsilon = 0.1$

Figure 1: Utility comparison of Distributed Gaussian Top-k against central-model top-k via the exponential mechanism, on the Adult dataset.



$N = 1000, \epsilon = 1.0$ $N = 500, \epsilon = 1.0$ $N = 100, \epsilon = 1.0$

Figure 2: Utility comparison of Distributed Gaussian Top-k against central-model top-k via the exponential mechanism, on synthetic data.

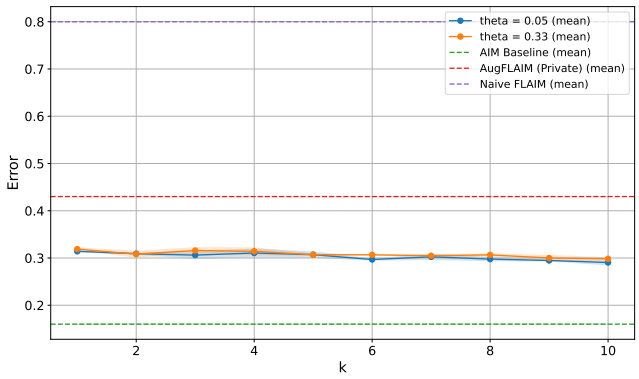


Figure 3: ℓ_2 error of Distributed AIM and baselines as a function of k under two adversarial settings. Distributed AIM substantially outperforms decentralized alternatives.

Table 3 compares average running time for the semi-honest and malicious-secure variants of our approach, using different underlying secure aggregation primitives to target the different threat models. Malicious security adds only minimal additional overhead.

Semi-honest	Malicious
$298s \pm 48s$	$301s \pm 54s$

Table 3: Comparison of total running time between semi-honest and malicious settings. Running time is similar for both protocols. Totals are averages over experiments with {20, 40, 80, 100} clients, 3 trials for each setting, and $k = 5$.

6.4 Q4: Impact of Network Constraints

Experiment Setup. We study how Distributed AIM behaves under varying network conditions. Because clients transmit only perturbed top-k utility vectors—rather than gradients or full local data, we expect client-side bandwidth requirements to be minimal. We vary server bandwidth from 10 Mbps (low bandwidth) to 1,000,000 Mbps (high bandwidth) and measure total runtime while keeping all other parameters fixed.

Results. Figure 6 summarizes the effect of network constraints. Client bandwidth has negligible impact on runtime: the communication footprint per client is extremely small and independent of dataset size. In contrast, server bandwidth is the dominant factor. When server bandwidth is constrained, the time required for secure aggregation increases noticeably, mirroring well-known scalability

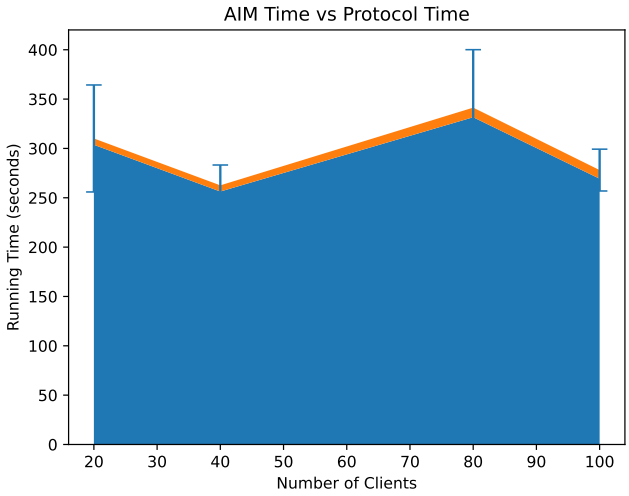


Figure 4: Runtime contribution breakdown (semi-honest setting). AIM computation dominates total runtime, while protocol overhead grows moderately with the number of clients.

characteristics of federated learning aggregation protocols. However, even at relatively low bandwidth (10 Mbps), the total runtime remains within practical bounds.

7 Conclusion

We have presented Distributed AIM, a novel approach for federated differentially private synthetic data. Our approach leverages a simple but effective technique for distributed top-k: we simply use the Gaussian mechanism, which ensure compatibility with lightweight cryptographic techniques like secure aggregation. Experimental results show that our approach significantly improves utility compared to prior work on efficient distributed synthetic data generation, and incurs minimal performance overhead compared to central-model AIM.

Generative AI Use

The authors used generative AI-based tools to correct typos, grammatical errors, and phrasing.

References

[1] Sergul Aydore, William Brown, Michael Kearns, Krishnaram Kenthapadi, Luca Melis, Aaron Roth, and Ankit A Siva. 2021. Differentially private query release through adaptive projection. In *International Conference on Machine Learning*. PMLR, 457–467.

[2] James Bell, Adrià Gascón, Peter Kairouz, and et al. 2020. Practical Secure Aggregation for Federated Learning on User-Held Data. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 1175–1191.

[3] James Bell, Adrià Gascón, Tancrède Lepoint, Baiyu Li, Sarah Meiklejohn, Mariana Raykova, and Cathie Yun. 2022. ACORN: Input Validation for Secure Aggregation. In *31st USENIX Security Symposium (USENIX Security 22)*. https://www.usenix.org/conference/usenixsecurity22/presentation/bell

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

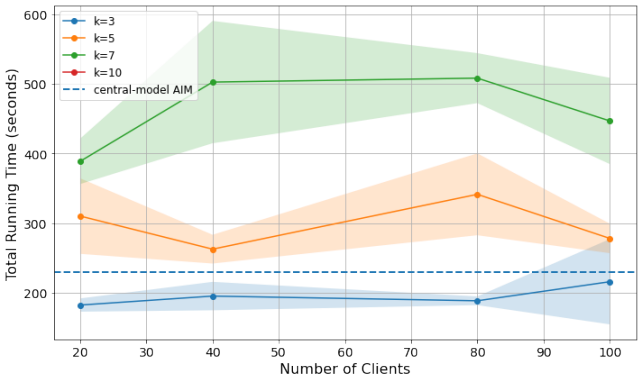


Figure 5: Number of clients and k on performance (Bonawitz protocol, semi-honest)

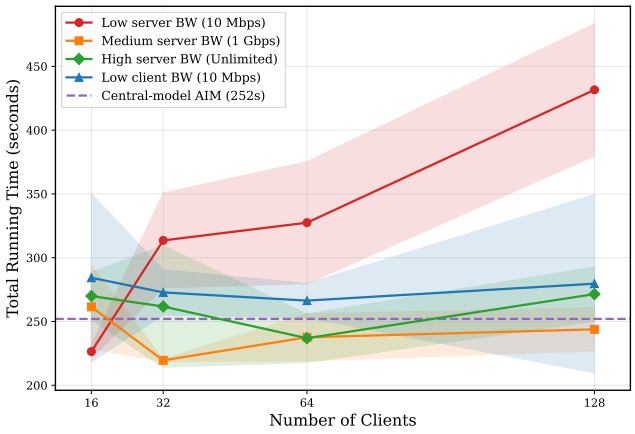


Figure 6: Impact of bandwidth limitations on running time

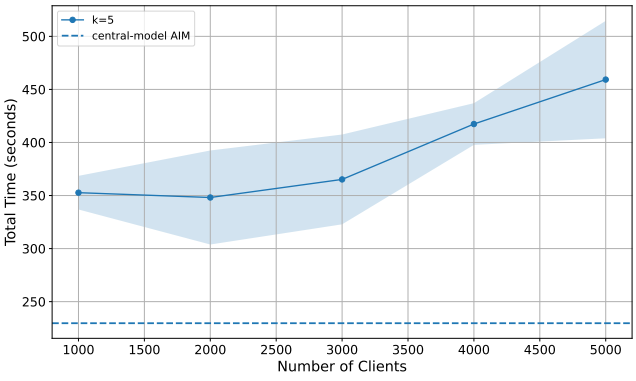


Figure 7: Running time for many clients (Bell protocol, semi-honest)

[5] Mark Bun and Thomas Steinke. 2016. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference (TCC)*. Springer, 635–658.

[6] David Durfee and Ryan M Rogers. 2019. Practical differentially private top-k selection with pay-what-you-get composition. *Advances in Neural Information*

Processing Systems 32 (2019).

[7] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*. Springer, 265–284.

[8] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[9] Yucheng Fu, Tiaoyao Gu, Elaine Shi, and Tianhao Wang. 2025. Towards Vertically Distributed Differentially Private Synthetic Data Generation. *Theory and Practice of Differential Privacy (TPDP)* (2025). Available at https://tpdp.journalprivacyconfidentiality.org/2025/pdf/fu.pdf.

[10] Zhengyang Hu, Jian Liu, Katrina Ligett, Aaron Roth, and Zhiwei Steven Wu. 2023. Optimal and Practical Algorithms for Differentially Private Top-k Selection. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*. PMLR, 16590–16614.

[11] Peter Kairouz, Ziyu Liu, and Thomas Steinke. 2021. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *International conference on machine learning*. PMLR, 5201–5212.

[12] Terrance Liu, Giuseppe Vietri, Thomas Steinke, Jonathan Ullman, and Steven Wu. 2021. Leveraging public data for practical private query release. In *International Conference on Machine Learning*. PMLR, 6968–6977.

[13] Terrance Liu, Giuseppe Vietri, and Steven Z Wu. 2021. Iterative methods for private synthetic data: Unifying framework and new methods. *Advances in Neural Information Processing Systems* 34 (2021), 690–702.

[14] Samuel Maddock, Graham Cormode, and Carsten Maple. 2024. FLAIM: AIM-based synthetic data generation in the federated setting. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2165–2176.

[15] Ryan McKenna. [n.d.]. Private PGM repository. https://github.com/ryan112358/mbi.

[16] Ryan McKenna, Brett Mullins, Daniel Sheldon, and Gerome Miklau. 2022. Aim: An adaptive and iterative mechanism for differentially private synthetic data. *arXiv preprint arXiv:2201.12677* (2022).

[17] Ryan McKenna, Daniel Sheldon, and Gerome Miklau. 2019. Graphical-model based estimation and inference for differential privacy. In *International Conference on Machine Learning*. PMLR, 4435–4444.

[18] Frank McSherry and Kunal Talwar. 2007. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. IEEE, 94–103.

[19] Brett Mullins, Miguel Fuentes, Yingtai Xiao, Daniel Kifer, Cameron Musco, and Daniel R Sheldon. 2024. Efficient and private marginal reconstruction with local non-negativity. *Advances in Neural Information Processing Systems* 37 (2024), 141356–141389.

[20] Ivoline C Ngong, Nicholas Gibson, and Joseph P Near. 2024. Olympia: A simulation framework for evaluating the concrete scalability of secure aggregation protocols. In *2024 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*. IEEE, 534–551.

[21] Zhikun Qiao, Katrina Ligett, Aaron Roth, and Zhiwei Steven Wu. 2021. The One-Shot Top-k Mechanism. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*. PMLR, 8654–8664.

[22] Amrita Roy Chowdhury, Chuan Guo, Somesh Jha, and Laurens Van der Maaten. 2022. Eiffel: Ensuring integrity for federated learning. In *Proceedings of the 2022 ACM SIGSAC conference on computer and communications security*. 2535–2549.

[23] Timothy Stevens, Joseph Near, and Christian Skalka. 2022. Secret Sharing Sharing For Highly Scalable Secure Aggregation. *arXiv preprint arXiv:2201.00864* (2022).

[24] Ge Zhang, Jinhui Xu, Ryan McKenna, Daniel Sheldon, and Michael Hay. 2025. FedDPSyn: Homomorphic Federated DP Synthetic Data Generation. In *Theory and Practice of Differential Privacy (TPDP)*. Available at https://cs.uwaterloo.ca/~s693zhan/papers/TPDP/2025/TPDP_2025.pdf.

[25] Ligeng Zhu and Yu-Xiang Wang. 2022. Instance-Optimal Private Top-k Selection. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 35. 3155–3167.

Ratang Sedimo, Joseph P. Near„