FastLloyd: Federated, Accurate, Secure, and Tunable k-Means Clustering with Differential Privacy

Abdulrahman Diaa
University of Waterloo
abdulrahman.diaa@uwaterloo.ca

Thomas Humphries
University of Waterloo
thomas.humphries@uwaterloo.ca

Florian Kerschbaum
University of Waterloo
florian.kerschbaum@uwaterloo.ca

Abstract

We study the problem of privacy-preserving k-means clustering in the horizontally federated setting. Existing federated approaches using secure computation suffer from substantial overheads and do not offer output privacy. At the same time, differentially private (DP) k-means algorithms either assume a trusted central curator or significantly degrade utility by adding noise in the local DP model. Naively combining the secure and central DP solutions results in a protocol with impractical overhead. Instead, our work provides enhancements to both the DP and secure computation components, resulting in a design that is faster, more private, and more accurate than previous work. By utilizing the computational DP model, we design a lightweight, secure aggregation-based approach that achieves five orders of magnitude speed-up over state-of-the-art related work. Furthermore, we not only maintain the utility of the state-of-the-art in the central model of DP, but we improve the utility further by designing a new DP clustering mechanism.

1 Introduction

Unsupervised learning allows data analysts to extract meaningful patterns from raw data that may be difficult to label. The canonical example of unsupervised learning is the Euclidean k-means clustering problem, where data is grouped into clusters with similar features. Clustering has a plethora of important applications in recommendation systems, fraud detection, and healthcare analytics [30]. In such applications, the dataset often contains sensitive data, which necessitates the use of privacy-preserving techniques. Combining databases horizontally split across multiple parties in the federated scenario yields more robust insights about the global population at the cost of further exasperating the privacy risks.

To counteract the privacy risks, a number of works focus on solving the federated k-means problem using secure multi-party computation [10, 29, 34–38, 50, 56, 59, 61, 67, 70, 71]. Secure computation enables analysts to solve the k-means problem while keeping the sensitive data encrypted. However, all of these works still output the exact result of the clustering. This gives a false sense of privacy; although the input and intermediate computations all remain private, no effort is made to protect the privacy of the output. It is well known in the literature that publishing exact statistics can leak significant information about the input data and allow attacks such as dataset reconstruction [19]. This level of privacy does not justify the significant runtime overhead incurred by solely using secure computation.

A separate line of research considers using differential privacy (DP) to protect the data. Differential privacy guarantees that the output (the cluster centroids) will be approximately the same regardless of any individual user's participation. Differential privacy can be applied in the central, local, or shuffle models. In the central model, a trusted aggregator receives the input as plaintext (unprotected) and randomizes the cluster centroids to ensure the output satisfies DP. Several works [3,8,16,45,49,52,55,65,72] have considered the central setting. In the local model, instead of using a trusted aggregator, each party perturbs their data locally before sending it to the aggregator. While this removes the need for a trusted aggregator (similar to secure computation), the utility at high privacy levels is low. Specifically, the utility is asymptotically worse than the central model by a factor of $\sqrt{n}$, where n is the number of data points [11]. Several works [12, 54, 63] have considered the local model. The shuffle model is a hybrid between the local and central models, with two mutually distrustful parties: a shuffler and an aggregator. Despite improving over the local model, the shuffle model often has a worse utility than the central model and requires additional parties or computational overhead to implement an oblivious shuffle.

To summarize, no prior work offers a solution that protects both the input and output privacy of the federated k-means problem and provides a good utility vs. privacy trade-off. A straw man solution to our problem would be to combine the current state-of-the-art solutions in federated k-means using secure computation and k-means in the central DP model.

However, this solution would be unacceptable as the runtime of current federated k-means approaches using secure computation is prohibitively large. The state-of-the-art approach of Mohassel et al. [50] incurs runtime overheads on the order of tens of minutes. Naively adding differentially private noise to this computation will further increase this runtime.

Our work focuses on designing an efficient DP k-means in the federated setting (using secure computation) that protects the privacy of the input, output, and intermediate computations. Following related work in federated k-means [38, 50], we use Lloyd's algorithm as the foundation of our protocol. However, we design a new variant of DP-Lloyd [8, 65] that achieves a tighter bound on the sensitivity. Our algorithm significantly improves the clustering utility over the state-of-the-art DP algorithm of Su et al. [65], especially in higher dimensions and number of clusters.

Our algorithm is of independent interest in the central model of DP. First, we enforce a bound on the radius of each cluster in the assignment step of Lloyd's algorithm. We then modify Lloyd's algorithm to compute updates relative to the previous centroid. Our relative updates have a sensitivity proportional to the bounded cluster radius rather than the domain size used in previous work [8, 65]. We find that our radius-based sensitivity bound is best suited to the use of the analytic Gaussian mechanism [5, 15], which further improves utility. Finally, we utilize additional post-processing steps based on the radius and the domain to improve the algorithm further. Together, these components lead to an improvement of up to 88% in utility (reduction in clustering error).

To tailor our DP-Lloyd algorithm to secure computation, we design a protocol that publishes intermediate computations in each iteration. This allows operations that would be expensive in secure computation (such as assignment and division) to be conducted in plain text. However, unlike prior work [29, 38], we ensure that the intermediate computations are protected by DP guarantees. We prove that the protocol is secure in the computational DP definition [48], which allows DP-bounded leakage during the computation. Typically, utilizing the computational DP model to leak intermediate steps implies sacrificing privacy or utility for an efficient runtime. However, in our case, we specifically choose to leak intermediate values already accounted for in the central DP proof of DP-Lloyd [8, 65]. In other words, we get this speed-up with no additional cost to the privacy parameter ε. Furthermore, rather than decrease the utility, we improve the utility of the clustering over state-of-the-art approaches [65].

To perform the aggregation over multiple parties, we design a lightweight secure aggregation protocol that keeps the aggregator oblivious to the client's inputs and the global centroids that are output at each iteration of the protocol. This allows the aggregator to add DP noise equivalent to the central model of DP. The combination of our improved DP algorithm, leaking the DP centroids, and our lightweight, secure aggregation protocol allows us to reduce the computation time by up to five orders of magnitude compared to related work [50] (from minutes to milliseconds per iteration).

In summary, our contributions are four-fold:

- We design the first DP protocol for horizontally federated private k-means.
- We improve the clustering utility over state-of-the-art DP k-means solutions by developing a new DP algorithm with various improvements, such as enforcing a radius constraint on the centroids and using relative cluster updates.
- We design an efficient protocol using DP and a lightweight secure aggregation protocol to implement our protocol in the local trust model.
- We prove our protocol is secure and preserves the end-to-end privacy in the computational model of DP and reduces the runtime by five orders of magnitude over the state-of-the-art secure federated approaches.

The remainder of the paper is organized as follows. We begin with some problem-specific background information in Section 2 and formally define the problem in Section 3. In Section 4, we summarize the relevant literature. We then present our complete protocol for federated DP k-means (FastLloyd) and prove its privacy and utility in Section 5. Finally, in Section 6, we give an in-depth evaluation of our protocol in terms of utility, runtime, and communication size over various real-world and synthetic datasets.

2 Background

2.1 Notation

Throughout this paper, we will denote objects that we intend to further slice/index with boldface notation (e.g. $\boldsymbol{\mu}$), while keeping atomic objects (whether scalars or vectors) as normal face (e.g. ε). A summary of the notation used in this paper is provided in Table 1, and we will define the notation as it is used.

2.2 k-Means Problem

The k-means problem is a discrete optimization problem that aims to partition a set of N d-dimensional observations into k clusters, each represented by its mean or centroid. Given a dataset of observations $D = \{x_1, x_2, ..., x_N\}$, where each observation is a d-dimensional real vector, the k-means problem is to find an assignment of data points to clusters, and a set of cluster centroids, that minimizes the Within-cluster Sum of Squares (WCSS) objective:

$$\underset{\mathbf{O},\boldsymbol{\mu}}{\operatorname{argmin}} \sum_{j=1}^{k} \sum_{x_l \in O_j} ||x_l - \mu_j||_2^2 \tag{1}$$

Symbol	Description
$\boldsymbol{p}$	Set of M clients: $\{p_1, p_2, ..., p_M\}$
S	Service provider
D_i	Local dataset of client p_i: $\{x_1, x_2, ..., x_{N_i}\}$
d	Dimension of the data points
k	Number of clusters
$\mathbf{O}$	Set of k clusters: $\{O_1, O_2, ..., O_k\}$
$\boldsymbol{\mu}$	Centroids of clusters: $\{\mu_1, \mu_2, ..., \mu_k\}$
S_j	Sum of data points in cluster j
C_j	Count of data points in cluster j
R_j	Relative sum of data points in cluster j
T	Number of iterations in Lloyd's algorithm
$\boldsymbol{\gamma}$	Differentially-private noise
η	Maximum radius bound
(ε, δ)	Privacy budget
σ	Noise multiplier

Table 1: Notation used in the paper.

where $\mathbf{O} = \{O_1, O_2, ..., O_k\}$, $O_j \subseteq D$ are the clusters, and $\boldsymbol{\mu} = \{\mu_1, \mu_2, ..., \mu_k\}$ are the centroids of the clusters, defined to be the (arithmetic) mean of points in O_j.

2.2.1 Lloyd's Algorithm

The k-means problem is NP-hard in Euclidean space [2]. However, a simple heuristic known as Lloyd's algorithm is commonly applied for practical applications [44]. We detail the basic procedure of Lloyd's algorithm:

1. Initialization step: Randomly sample k centroids $\{\mu_1, \mu_2, ..., \mu_k\}$ (typically from the datapoints).
2. Repeat until convergence (the assignments no longer change) or a predetermined number of iterations has been reached:
   (a) Assignment step: Assign each observation x_l to the nearest centroid (using the Euclidean distance). This creates clusters O_j for $j = 1, 2, ..., k$. Formally, the assignment is:

$$O_j^{(t)} = \left\{x_l : ||x_l - \mu_j^{(t-1)}||_2^2 \leq ||x_l - \mu_c^{(t-1)}||_2^2 \; \forall c \in \{1, \ldots, k\}\right\}$$

   (b) Update step: Calculate the new centroids to be the mean of the observations in the cluster:

$$\mu_j^{(t)} = \frac{\sum_{x_l \in O_j^{(t)}} x_l}{\left|O_j^{(t)}\right|}$$

2.3 Differential Privacy

Differential privacy (DP) [17] is an increasingly popular notion to protect the privacy of individuals while allowing the computation of aggregate statistics. Differential privacy guarantees that an algorithm's output is approximately the same, regardless of the participation of any single user. More formally, differential privacy can be defined as follows.

Definition 2.1 (Differential Privacy)**.** *A randomized algorithm $M : \mathcal{D} \mapsto \mathbb{R}$ is (ε, δ)-DP, if for any pair of neighbouring datasets $D, D' \in \mathcal{D}$, and for any $S \subseteq \mathbb{R}$ we have*

$$\Pr[M(D) \in S] \leq e^{\varepsilon} \Pr[M(D') \in S] + \delta. \tag{2}$$

The privacy parameter ε defines how similar the outputs must be, and δ allows a small chance of failure in the definition. We use the unbounded neighbouring definition where datasets are neighbours if $|D \backslash D' \cup D' \backslash D| = 1$. That is, we allow for the addition or removal of a single data point. We note that arbitrary computations can be carried out on the output of a DP mechanism without affecting privacy (the post-processing lemma [18]). Finally, DP is composed naturally with multiple runs of a mechanism. If we apply a differentially private mechanism(s) sequentially, the privacy parameters are composed through summation or more advanced methods [18]. If a mechanism is applied multiple times over disjoint subsets of the dataset, then the total privacy leakage is the maximum privacy parameter over each subset (parallel composition [18]).

Definition 2.2 (Sensitivity)**.** *Let $f : \mathcal{D} \mapsto \mathbb{R}^k$. If $\mathbb{D}$ is a distance metric between elements of $\mathbb{R}^k$ then the $\mathbb{D}$-sensitivity of f is*

$$\Delta^{(f)} = \max_{(D,D')} \mathbb{D}(f(D), f(D')), \tag{3}$$

where (D, D') are pairs of neighbouring datasets.

We will focus on the ℓ_2 norm in this work as we use the Gaussian Mechanism. To analyze the Gaussian Mechanism, we will use Gaussian Differential Privacy (GDP) [15].

Definition 2.3 (GDP [15])**.** *A mechanism M is said to satisfy θ-Gaussian Differential Privacy (θ-GDP) if it is G_θ-DP. That is,*

$$\mathcal{T}\big(M(D), M(D')\big) \geq G_\theta$$

for all neighbouring datasets D and D', where $\mathcal{T}$ is a trade-off function measuring the difficulty for attackers in identifying presence of an individual data point and $G_\theta = \mathcal{T}\big(\mathcal{N}(0,1), \mathcal{N}(\theta,1)\big)$ (see Dong et al. [15] for specifics of the definition).

Naturally, the Gaussian Mechanism satisfies GDP.

Theorem 2.1. *(Gaussian Mechanism GDP [15]) Define the Gaussian mechanism that operates on a statistic f as $M(D) = f(D) + \gamma$, where $\gamma \sim \mathcal{N}(0, (\Delta^{(f)})^2/\theta^2)$. Then, M is θ-GDP.*

Similar to DP, GDP composes over multiple adaptive uses of a mechanism.

Theorem 2.2 (GDP Composition [15]). *The n-fold composition of θ_i-GDP mechanisms is $\sqrt{\theta_1^2 + \cdots + \theta_n^2}$-GDP.*

Finally, it is possible to convert a GDP guarantee to DP and vice versa:

Theorem 2.3 (GDP to DP [5, 15]). *A mechanism is θ-GDP if and only if it is $(\varepsilon, \delta(\varepsilon))$-DP for all $\varepsilon \geq 0$, where*

$$\delta(\varepsilon) = \Phi\Big(-\frac{\varepsilon}{\theta} + \frac{\theta}{2}\Big) - e^{\varepsilon}\Phi\Big(-\frac{\varepsilon}{\theta} - \frac{\theta}{2}\Big).$$

In practice, we use the algorithm derived by Balle and Wang to solve this function for θ [5, Algorithm 1].

3 Problem Statement

We consider M clients denoted by $\boldsymbol{p} = \{p_1, p_2, ..., p_M\}$, with each party p_i owning a private dataset $D_i = \{x_1, x_2, ..., x_{N_i}\}$, where $x_l \in \mathbb{R}^d$, d denotes the dimensionality of the dataset and N_i is the size of the dataset of party i. The total dataset that we compute on, can be very large, even if there are only a small number of clients. The objective is to compute a collaborative k-means clustering $(\mathbf{O}, \boldsymbol{\mu})$ over $\boldsymbol{D}$ that minimizes the WCSS objective (1); where $\boldsymbol{D} = \bigcup_{i=1}^{i=M} D_i$ is the union of the datasets held by the clients, $\mathbf{O} = \{O_1, O_2, ..., O_k\}$ are the clusters, and $\boldsymbol{\mu} = \{\mu_1, \mu_2, ..., \mu_k\}$ are the respective centroids of the clusters.

We aim to optimize this objective while formally proving the privacy of the output and intermediate computations. Specifically, the protocol should be secure in the computational model of differential privacy introduced by Mirnov et al. [48] and extended to the multi-party setting by Humphries et al. [33]. We further extend this model to the approximate DP setting by incorporating the failure probability δ.

Definition 3.1 (IND-CDP-MPC [33, 48]). *A multi-party protocol Π for computing function f satisfies $(\varepsilon(\lambda), \delta)$- indistinguishable computationally differential privacy (IND-CDP-MPC) if for every probabilistic polynomial time (in λ) adversary A with input dataset D_A, and for neighbouring datasets D, D' belonging to the honest parties (i.e. $D, D' = \cup_{i \setminus A} D_i$),*

$$\begin{aligned}&\Pr[\mathcal{A}(\text{VIEW}_A^{\Pi}(D_A, D)) = 1] \\ \leq &\exp(\varepsilon) \cdot \Pr[\mathcal{A}(\text{VIEW}_A^{\Pi}(D_A, D')) = 1] + negl(\lambda) + \delta.\end{aligned}$$

where $\mathcal{A}$ is the function representing the adversary's decision on whether the dataset was D or D' based on their VIEW_A^{Π}, which is the transcript of all messages observed by adversary A during execution of protocol Π. $negl(\lambda)$ is a negligible function decreasing faster than any inverse polynomial in λ. Likewise, the definition holds for every other party's view of neighbours (D, D').

Intuitively, a protocol that satisfies IND-CDP-MPC *securely* simulates a TTP executing a central DP mechanism. Specifically, even after observing the output and intermediate computations, any (computationally bounded by a polynomial in λ) party should not learn more about any other party's local dataset than what could be learned from the differentially private leakage of the central DP mechanism itself.

We make the following assumptions when designing our protocol:

- The existence of an honest-but-curious service provider S to assist with multiparty computations. This party should be oblivious to all inputs and results, i.e., the service provider learns nothing about the client's input or output during the execution of the protocol (a weaker assumption than the local and shuffle models). We also discuss alternatives to this assumption in Appendix C.

- Clients share a common secret used as a Pseudorandom Number Generator (PRNG) seed and do not collude with the server. This secret can be established through standard public-key schemes (e.g., Bonawitz et al. [9] uses Diffie-Hellman key exchange [14]) or TLS-secured channels.

3.1 Comparison to other models

In the **local DP model**, each client independently perturbs their data before sharing it. Since privacy is guaranteed at the source, there is no need to trust an aggregator or rely on any cryptographic protocols. However, it significantly reduces accuracy because noise is added individually to each data point rather than to aggregated statistics.

The **central DP model**, in contrast, involves a trusted third-party (TTP) aggregator that collects raw data from clients and applies noise only to aggregated results. This significantly improves accuracy over the local model since the sensitivity of aggregated statistics is lower than individual points. However, this model requires strong trust assumptions, as the aggregator has full access to the clients' raw data.

The **shuffle DP model** offers a compromise between local and central DP by introducing a non-colluding semi-trusted shuffler that permutes messages before aggregation. This shuffling enhances privacy, allowing clients to inject less noise compared to local DP while still not requiring trust in the aggregator. Although accuracy improves compared to local DP, the shuffle model remains inherently less accurate than the central DP model. Additionally, it necessitates either trusting the shuffler to remain non-colluding or employing an oblivious shuffle protocol.

On the other hand, the **IND-CDP-MPC model** "replaces" the trusted aggregator from central DP with a secure multi-party computation (MPC) protocol that simulates the central mechanism. Therefore, it retains the accuracy of the central DP model while operating under a local model of trust (only revealing noised data). This allows for a significantly better

privacy vs. utility trade-off than the local and shuffle DP models, and is therefore the focus of this work.

The **exact MPC model** allows multiple parties to jointly compute a function over their inputs while keeping those inputs private. The model assumes that no information beyond the final (non-private) output is revealed. However, without protecting the output with DP, the result is vulnerable to reconstruction attacks that can leak the entire dataset in the worst case. Thus, we focus on the IND-CDP-MPC model.

4 Related Work

4.1 Secure Exact k-means

We first discuss secure computation based approaches to k-means that do not preserve the privacy of the output. This literature shows a trade-off between privacy and efficiency. Approaches that prioritize efficiency often compromise privacy by allowing the leakage of intermediate computations such as sums, counts, or centroids, which can be exploited to infer sensitive data [29, 38]. For instance, Gheid et al. [29] rely on computing secure sums for aggregating sums and counts across clients. While this is very efficient in practice, it reveals aggregate intermediate values to all clients, which is not considered secure [31]. As we discuss in Appendix C, naively fixing this leakage would require a significant degradation in utility or performance. Another approach by Jiang et al. [38] employs homomorphic encryption and garbled circuits to conceal sums and counts. However, it still reveals the intermediate centroids to all clients in the clear. Many other works in this category are explained in the SoK of Hedge et al. [31].

Conversely, secure approaches that do not leak intermediate computations employ heavy cryptographic techniques, like fully homomorphic encryption, leading to significant performance degradation [10, 36, 40, 59]. Bunn and Ostrovsky [10] utilize arithmetic secret sharing alongside homomorphic encryption, with high-performance overheads. Homomorphic encryption has also been used by Rao et al. [59], Jäschke et al. [36], and Kim et al. [40], and consistently results in prohibitively high computation time due to the computation demands of the encryption. The work of Mohassel et al. [50] is a notable exception, offering a scheme that is significantly faster than previous state-of-the-art methods. By using optimized 2-party computation primitives, batched oblivious transfers, and garbled circuits, they achieve a speedup of five orders of magnitude over Jäschke et al. [36]. However, the runtime is still on the order of minutes; and the communication size is on the order of gigabytes. In contrast, our approach is as efficient as the protocols that leak intermediate computations, while protecting the privacy of the output and intermediate computations with DP.

4.2 Differentially Private k-means

DP k-means algorithms aim to address output privacy by introducing noise during clustering, hiding individual data contributions, and ensuring a formal privacy guarantee. Research in DP k-means is divided into central and local models.

4.2.1 Central DP

The central model operates with a trusted curator who collects data for analysis, hiding individuals' information in the output, but not from the curator itself. Among the central approaches, DP-Lloyd introduces noise during each centroid update, by adding Laplace noise to both the sum (numerator) and the counts (denominator) when computing the arithmetic mean of points within a cluster [8]. DP-Lloyd was later implemented in the PinQ framework [47] with a fixed number of iterations. Dwork [16] extended this framework to allow for an arbitrary number of iterations, allocating an exponentially decreasing privacy budget to each iteration, although it was noted that utility degrades beyond a certain number of iterations due to increasing noise levels. The state-of-the-art in DP-Lloyd was achieved by Su et al. [65], who included an error analysis to determine the optimal number of iterations, optimized the splitting of privacy budget between sum and count queries, and introduced a "sphere-packing" centroid initialization method that significantly improves clustering quality.

The Sample and Aggregate Framework (*SaF*) partitions the dataset into multiple subsets, upon which the non-private Lloyd's algorithm is executed independently [53]. The resulting centroids from each subset are then aggregated using a standard DP mechanism. SaF was later implemented within the GUPT system [49]. However, experiments conducted by Su et al. [65] suggest that DP-Lloyd consistently outperforms SaF across various synthetic and real datasets. Synopsis-based methods take a different approach by first generating a synopsis of the dataset using a differentially private algorithm, then applying the k-means clustering algorithm on this synopsis. Qardaji et al. [57] proposed a synopsis technique for 2D datasets, which was extended and optimized by Su et al. [65] to higher dimensions. However, this approach only outperforms DP-Lloyd in datasets with less than three dimensions [65]. Another, more theoretical, line of work focuses on minimizing approximation error bounds, without implementation or empirical evaluation [3, 20, 21, 27, 54, 64]. The work of Ghazi et al. [27] concludes this long line of work by achieving the same approximation factor as the best non-private algorithms. However, these methods suffer from superlinear running times, making them impractical for large datasets, where privacy is most important.

4.2.2 Local DP

In contrast to central DP, the local model operates under the premise that no trusted curator is available. This necessitates that individuals randomize their data locally before it is aggregated, which reduces utility by introducing a significantly higher level of noise (proportional to the number of data points). Several works have been proposed in this area [12, 54, 63], most of which focus on the theoretical error bounds rather than practical applications. The most efficient and accurate local DP protocol by Chang et al. [12] could be extended to our setting, by instantiating the aggregation oracles with our MSA protocol. This would significantly improve the accuracy of their protocol. However, this would introduce high communication complexity ($O(M\sum_{i=1}^{M} N_i)$), since the complete net-tree would need to be aggregated across all parties. In contrast, our protocol transmits only centroids and counts per iteration, leading to a complexity of $O(MkT)$. Another line of work studies private aggregation (a building block for federated DP k-means) in local and shuffle DP models [4, 28]. However, these works cannot overcome the utility gap stemming from the noise being added to each data point. Finally, Li et al. [43] attempt to bridge the gap between these central and local models in the vertically federated setting. Their approach relies on an untrusted curator aggregating noisy local centers and membership encodings from the clients to generate a private synopsis. However, this approach is specifically tailored for the vertical setting and cannot be applied to our configuration. Our work yields utility that is even better than state-of-the-art work in the central model [65] while obtaining the trust assumptions of the local model and operating in the federated setting.

5 Distributed DP k-means

First, we overview our modified DP algorithm using the maximum radius constraint and relative updates. Then, we describe FastLloyd, including the initialization, assignment, and update steps. Finally, we provide our security theorem, and discuss how we choose the various parameters.

5.1 Radius Constrained DP k-means

In this section, we describe how we improve the clustering utility over the work of Su et al. [65], a contribution that is independently applicable to the central model of DP. We observe that despite being common in the non-private literature [7, 42], neither federated exact k-means nor DP k-means algorithms in the literature apply any constraints to the k-means objective. In this work, we focus on a type of constraint not studied in the non-private literature: bounding the radius of each cluster. This constraint is particularly useful in the context of DP k-means, as it allows a tighter sensitivity bound when computing cluster updates relative to the previous centroid.

In the DP k-means literature [8, 16, 47, 65], a typical private clustering algorithm is to run Lloyd's algorithm, with DP noise added to the update step. The DP mean is computed by two DP queries: one for the sum and one for the count. The sum is then divided by the count as a post-processing step. The challenge in these protocols is that the sensitivity of the sum is bounded by the domain size. If the datapoints x_l are contained in $[-B, B]^d$, then the l_2 sensitivity of the sum is $B\sqrt{d}$. This is because, with no other constraints, there exists a worst-case data point that could add B to the sum in all dimensions, regardless of the cluster it is assigned to.

In this work, we modify the k-means objective such that in addition to minimizing the WCSS (Eqn 1), a constraint on the maximum radius of each cluster must also be satisfied. Specifically, no data point in a cluster can be more than η away from the cluster's centroid. Thus, the objective becomes to minimize (Eqn 1) s.t.:

$$||x_l - \mu_j||_2 \leq \eta \quad \forall x_l \in O_j, \forall j \tag{4}$$

To enforce this constraint, we modify the assignment step to not assign a data point to a cluster if it is more than η away from its nearest centroid. Any unassigned data points are discarded and, therefore, do not contribute to any sum or count query. Intuitively, bounding the maximum radius of each cluster limits how much a cluster can move by adding or removing a data point since a worst-case data point must be close to a cluster's centroid to be factored into the mean. However, simply bounding the radius of the cluster does not tighten the sensitivity of the sum query. This is because the sensitivity analysis must consider the worst-case cluster. If a cluster is within η of the domain boundary, then the ℓ_2 sensitivity of the sum query is still $B\sqrt{d}$.

To realize the reduction in sensitivity from bounding the radius, we must modify the update step as well as the assignment step. Instead of perturbing the sum of the data points themselves, we perturb the difference between the data points and their assigned cluster centroid. Specifically:

$$R_j^{(t)} = \sum_{x_l \in O_j^{(t)}} x_l - \tilde{\mu}_j^{(t-1)} \tag{5}$$

Essentially, we are computing the "updates" to the centroids, rather than the centroids themselves. This simple change has a significant impact on the sensitivity of the sum query. The sensitivity of the relative sum is now bounded by the maximum radius parameter η:

Theorem 5.1. *If the constraint (4) is satisfied, then:*

$$\Delta^R = \max_{D, D' \in \mathcal{D}} ||R_j^{(t)}(D) - R_j^{(t)}(D')||_2 \leq \eta \tag{6}$$

for all clusters $j \in [k]$, and iterations $t \in [T]$.

We prove this in Appendix A.1. The intuition is that a worst-case data point only contributes its distance to the centroid (which is bounded by η), rather than its distance to the origin.

Our maximum distance constraint naturally bounds the l_2 sensitivity as it bounds the Euclidean distance, and thus, Gaussian noise is a natural choice that also scales more efficiently to higher dimensions. It also allows for a simple and tight privacy analysis using Gaussian-DP [15]. We use the analysis of Balle and Wang [5, Algorithm 1] to obtain the noise multiplier σ. We divide the privacy budget between the relative sum and the count following our analysis in Section 5.5 as:

$$\sigma^R = \frac{\sigma\sqrt{1+\sqrt{4d}}}{\sqrt[4]{4d}} \quad \sigma^C = \sigma\sqrt{1+\sqrt{4d}} \tag{7}$$

We then add noise as follows:

$$\tilde{R}_j^{(t)} = R_j^t + \gamma^R \quad \text{where} \quad \gamma^R \sim \mathcal{N}(0,(\sigma^R)^2\eta^2 T) \tag{8}$$

$$\tilde{C}_j^{(t)} = C_j^t + \gamma^C \quad \text{where} \quad \gamma^C \sim \mathcal{N}(0,(\sigma^C)^2 T). \tag{9}$$

We prove in Section 5.4 that adding noise in this way satisfies DP. After adding noise, we compute the new centroid as:

$$\tilde{\mu}_j^{(t)} = \frac{\tilde{R_j}^{(t)}}{\tilde{C_j}^{(t)}} + \tilde{\mu}_j^{(t-1)}. \tag{10}$$

Without DP noise, this yields an equivalent cluster update.

$$\mu_j^{(t)} = \frac{S_j^{(t)} - C_j^{(t)}\mu_j^{(t-1)}}{C_j^{(t)}} + \mu_j^{(t-1)} = \frac{S_j^{(t)}}{C_j^{(t)}} \tag{11}$$

where $S_j^{(t)} = \sum_{x_l \in O_j^{(t)}} x_l$ is the sum of the data points. With the DP noise, we get an additional noise (error) term in the sum compared to noising the sum directly:

$$\tilde{S}_j^{(t)} = \tilde{R}_j^{(t)} + \tilde{C}_j^{(t)}\tilde{\mu}_j^{(t-1)} \tag{12}$$

$$= S_j^{(t)} - C_j^{(t)}\tilde{\mu}_j^{(t-1)} + \gamma^R + (C_j^{(t)} + \gamma^C)\tilde{\mu}_j^{(t-1)} \tag{13}$$

$$= S_j^{(t)} + \gamma^R + \gamma^C\tilde{\mu}_j^{(t-1)}. \tag{14}$$

However, as we show in Section 6.2, this additional error is compensated for by the increase in utility from reducing the sensitivity from $B\sqrt{d}$ to η. The maximum radius further decreases error by reducing the effect of outliers, as data points far away from any centroid will not (and, in some cases, should not) be assigned to any cluster. We also use the maximum radius constraint as an additional post-processing constraint, which we call *Radius Clipping*. Namely, if a noisy centroid is computed to be more than η away from the previous centroid, we truncate it to be η away.

5.2 Overview of FastLloyd

In this section, we describe our protocol for federated DP k-means, FastLloyd. The focus of FastLloyd is to create the most efficient and accurate protocol possible under the threat model defined in Section 3. Specifically, we refrain from adding additional noise or computations that would be needed to handle client failures or collusion between clients and the server. We leave it to future work to adapt our approach to use more resilient aggregation protocols [9, 22, 39, 62].

5.2.1 Protocol Intuition

A naive IND-CDP-MPC implementation of Lloyd's algorithm would use an end-to-end secure protocol (e.g., [50]) to compute the exact centroids in every iteration, then add DP noise using a secure computation. This entails gigabytes of communication and tens of minutes of runtime as we show in Section 6.

Instead, our key insight is that the tightest DP analysis for Lloyd's algorithm is an (adaptively)-compositional proof [8, 65] that assumes that the (perturbed) intermediate computations are published in each iteration. Therefore, revealing these intermediate DP updates does not violate securely simulating the TTP, as required by the IND-CDP-MPC model [1]. This allows us to compute divisions and assignments locally, rather than in a secure computation protocol.

We note that leaking the intermediate computations is not possible in the exact MPC model. Thus our approach is not applicable to the exact MPC model (where the output is not private). In this paper, we demonstrate that, by working within the computational differential privacy framework, the state-of-the-art DP version of Lloyd's algorithm can be computed significantly more efficiently (five orders of magnitude faster) in MPC than its non-private counterpart.

5.3 Algorithm Description

Algorithm 1 overviews the protocol from the perspective of a single client. Following the outline of Lloyd's algorithm, the following sections describe how we design each of its main steps: initialization, assignment, and update.

5.3.1 Initialization: Sphere Packing

We modify the initialization so that it can be carried out in a federated manner. We employ the sphere packing initialization approach of Su et al. [65] as it was shown to outperform random initialization. The sphere packing approach is data-independent and thus does not use any privacy budget. The process can be outlined as follows:

1. Initialize a radius parameter a.
2. For $j \in \{1,2,...,k\}$, generate a point μ_j such that it is at least of distance a away from the domain boundaries and

[1] This insight would not apply to any future DP analyses that achieve privacy amplification by specifically hiding intermediate computations. Our approach would still be applicable, but at would not benefit from this amplification.

Algorithm 1 FastLloyd from p_i's perspective

Inputs: Local Dataset D_i.
Output: Cluster Centres $\tilde{\boldsymbol{\mu}}^{(T)}$

1: $\tilde{\boldsymbol{\mu}}^{(0)} = \text{INITIALIZATION}(seed)$
2: **for** $t \in \{1:T\}$ **do**
3: // **Assignment Step**
4: **for** $x_l \in D_i$ **do**
5: $j' = \arg\min_{j' \in [k]} ||x_l - \tilde{\mu}_{j'}^{(t-1)}||_2$
6: **if** $||x_l - \tilde{\mu}_{j'}^{(t-1)}||_2 < \eta$ **then**
7: $O_{ij'}^{(t)} \leftarrow x_l$
8: // **Local Update**
9: **for** $j \in \{1,\ldots,k\}$ **do**
10: Compute $\bar{R}_{ij}^{(t)} = \sum_{x_l \in O_{ij}^{(t)}} x_l - \tilde{\mu}_j^{(t-1)}$
11: Compute $\bar{C}_{ij}^{(t)} = |O_{ij}^{(t)}|$
12: // **Global Update**
13: $\widehat{\boldsymbol{U}} = \text{GLOBALMSA}(\bar{R}_{ij}^{(t)}, \sigma^R, \text{seed}, t)$
14: $\widehat{\boldsymbol{C}} = \text{GLOBALMSA}(\bar{C}_{ij}^{(t)}, \sigma^C, \text{seed}, t)$
15: // **Post Process Result**
16: **for** $j \in \{1,\ldots,k\}$ **do**
17: $\widehat{\mu}_j = \frac{\widehat{\boldsymbol{U}}}{\widehat{\boldsymbol{C}}} + \tilde{\mu}_j^{(t-1)}$
18: $\tilde{\boldsymbol{\mu}}^{(t)} = \text{FOLD}(\widehat{\boldsymbol{\mu}})$

at least of distance $2a$ away from any previously chosen centroid. If a randomly generated point does not meet this condition, generate another one.

3. If, after 100 repeated attempts, it is not possible to find such a point, decrease the radius a and repeat the process.

The radius a is determined via a binary search to find the maximum a that allows for the generation of k centroids. In Algorithm 1, each client independently calls the `Initialization` function (Line 1) with the same random seed, which results in each client starting with the same centroids.

5.3.2 Assignment: Radius Constrained k-means

In Line 5, each client locally computes the closest cluster to each of their data points. If the data point is within η of the cluster's centroid, it is assigned to that cluster (Line 7). Any points further than η from the cluster's centroid are not assigned to any cluster. We discuss how we set η in Section 5.6.

5.3.3 Local Update

The client has already locally assigned each of their data points to a cluster, following the constraints in the previous step (Line 7). Next, they compute the relative sum and the count for each cluster using their local dataset. We call this the local update step. The output of the relative sum (Line 10) and the count (Line 11) are two matrices of dimensions $(k \times d)$ and $(k \times 1)$ respectively.

5.3.4 Global Update: Masked Secure Aggregation

We modify the update step to release a perturbed version of relative sums and the counts to each client under the CDP security model. By publishing the previous noisy centroids, the assignment and local update computation can be performed locally by each client instead of using a secure computation protocol. In the global update step (Lines 13 and 14), we privately aggregate (sum) the local updates of all clients, perturb the result, and reveal the noisy global relative sums and counts for the next round.

We describe our aggregation protocol in Figure 1. We call this protocol Masked Secure Aggregation (MSA). In MSA, similar to regular secure aggregation protocols [9], a service provider S securely aggregates values from M clients, all while being oblivious to every client's contributions. However, in MSA, the server is also oblivious to the result of the aggregation operation; the server only acts as an aggregator (who also adds DP noise) and is oblivious to both the input and output. Each client p_i possesses a private value (a matrix $v_i \in \mathbf{R}^{k \times d}$, where $\mathbf{R}$ could be $\mathbb{Z}_{2^{64}}$). These clients aim to collectively compute the element-wise sum of their private matrices: $\boldsymbol{v} = \sum_{i=1}^{M} \frac{v_i}{M}$. In addition to the client's values, we assume the noise scale σ and a random *seed* (that all clients know) are also provided.

Clients Send Data. The first step in the protocol is for each of the M clients to generate a random mask set. The random mask set is composed of M random matrices of the same dimensions as the client items: $\{r_1, r_2, \ldots r_M\}$ where each value is sampled uniformly from $\mathbf{R}$. Because all the clients have access to a shared seed and *PRNG* they can each compute the entire set locally. In Line 1, each client samples the entire set and sums it (in Line 2) to get the global mask matrix $\boldsymbol{r}$ (which will be used to decrypt later).

In Line 3, each private value is converted to a fixed-point format, $\tilde{v_i}$, by scaling it up with a power-of-2 scale factor, 2^q. Formally, $\tilde{v_i} = \lceil v_i \times 2^q \rfloor$. All fractional values are rounded to the nearest representable value in this fixed-point representation. The choice of the scale factor, 2^q, determines the precision of the representation[2]. In practice, we empirically choose $q = 16$.

Then, in Line 4, each client encrypts (masks) their input $\tilde{v_i}$ by adding their share of the mask r_i. Finally, each client sends their masked values to the server. This step masks the actual value v_i from the server because r_i acts as a one-time pad. Note that this masked value is never sent to other clients,

[2]A larger 2^q allows for greater precision but also reduces the number of bits available for the integer part of the number, which might cause overflows. A workaround is to increase the bit-width of the operations, which increases the computational load and the communication cost.

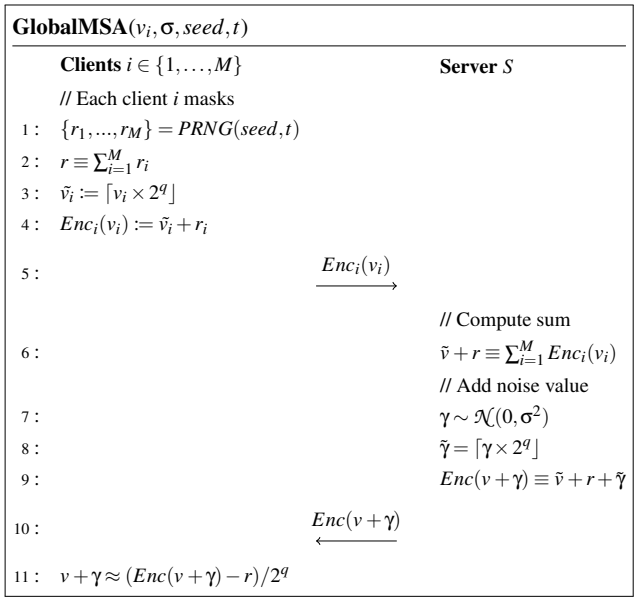


Figure 1: Global Masked Secure Aggregation Protocol with M Clients.

as they would be able to unmask it easily.

Server Aggregates Data. Upon receiving the masked matrices, the server first sums over each client's contribution in Line 6. This yields the masked global sum $\boldsymbol{v} + \boldsymbol{r}$. The server then samples from a zero mean Gaussian distribution with standard deviation equal to the supplied σ for each entry in the result matrix. This noise must also be converted to fixed-point by computing $\tilde{\gamma} = \lceil \gamma \times 2^q \rfloor$. We show why this preserves differential privacy in Appendix A.3. Finally, the server adds the noise $\tilde{\gamma}$ to the sum and broadcasts it to the clients.

Client Unmasks Data. Upon receiving the result from the server, each client must unmask the result. They do this by simply subtracting the mask r that they computed in Line 2. After unmasking, each client must scale down the result to retrieve the correct answer. This is done by dividing the unmasked sum by the scale factor, 2^q, to reverse the initial scaling operation.

5.3.5 Post-processing

In Algorithm 1, each client locally post-processes the results to obtain the centroids for the next iteration.

First, each party divides the relative sum by the count and shifts the result by the previous centroid to get the new centroid (Line 17) following Eqn 10. Then, if the new centroid is more than η away from the previous centroid, we truncate it to be η away.

Finally, we apply a post-processing step to the centroids to ensure they remain within the domain. A naive post-processing strategy is to simply truncate out-of-bounds centroids to the boundary. However, in practice, we find (in Appendix B) that *folding* [32] the value (reflecting it over the boundary) gives better utility. More formally, the operation folds a value x into the range $[-B, B]$ through modular arithmetic. We first compute $(x+B) \bmod (2B)$, then if this result exceeds B, we reflect it about B by subtracting it from $2B$. Finally, we return the value to the target range by subtracting B. This approach creates a periodic folding pattern that naturally reflects values across the boundaries while preserving distances from the nearest boundary point. We show in Appendix B that folding improves the algorithm's utility.

5.4 Privacy Analysis

FastLloyd allows for a central party to add noise, retaining the utility of the central model of DP. However, since the server only interacts with masked values, and the clients can only observe a differentially private view of the final (noised) centroids, the protocol provides a level of privacy akin to that of the local model of DP.

We state the end-to-end security Theorem of our algorithm and defer the proof to Appendix A.2.

Theorem 5.2. *Algorithm 1 ensures* $(\varepsilon(\lambda), \delta)$*-IND-CDP-MPC in the presence of a semi-honest, polynomial time adversary who controls at most a single party.*

5.5 Error Analysis

To analyze the error of our approach, we follow a similar approximate error analysis as Su et al. [65]. The purpose of the analysis is primarily to choose the ratio of the privacy parameters and the number of iterations. Thus, the analysis makes a series of approximations. Following Su et al., we consider a single iteration and a single cluster for this analysis. To simplify the notation, we omit the cluster index j and instead index the variables by the dimension h. We analyze the mean-squared error between the true centroid (μ) and the differentially-private centroid ($\tilde{\mu}$) for one iteration across all dimensions.

$$MSE(\tilde{\mu}^{(t)}) = \mathbb{E}\left[\sum_{h=1}^{d} (\mu_h^{(t)} - \tilde{\mu}_h^{(t)})^2\right] \tag{15}$$

We first expand the following term using the definition of our DP mechanism from Section 5.1:

$$\begin{aligned} \mu_h^{(t)} - \tilde{\mu}_h^{(t)} &= \frac{(C^{(t)} + \gamma^C)\mu_h^{(t)}}{C^{(t)} + \gamma^C} - \frac{S_h^{(t)} + \gamma_h^R + \gamma^C \tilde{\mu}_h^{(t-1)}}{C^{(t)} + \gamma^C} \\ &= \frac{\gamma^C(\mu_h^{(t)} - \tilde{\mu}_h^{(t-1)}) - \gamma_h^R}{C^{(t)} + \gamma^C}. \end{aligned}$$

Let $\nabla_h = \mu_j^{(t)} - \tilde{\mu}_j^{(t-1)}$. Then, after squaring, we get:

$$(\mu_h^{(t)} - \tilde{\mu}_h^{(t)})^2 = \frac{(\gamma^C)^2(\nabla_h)^2 - 2\gamma^C\gamma_h^R\nabla_h + (\gamma_h^R)^2}{(C^{(t)} + \gamma^C)^2}.$$

After taking the expectation over all clusters and assuming that $C+\gamma^C \approx N/k$, following Su et al. [65], we get:

$$\begin{aligned}\mathsf{MSE}\left(\tilde{\mu}_h^{(t)}\right) &\approx \frac{k^3}{N^2}\left(\mathbb{E}\left[(\gamma^C)^2\right]\mathbb{E}\left[\nabla_h\right]^2+\mathbb{E}\left[(\gamma_h^R)^2\right]\right)\\ &= \frac{k^3}{N^2}\left(\mathsf{Var}\left(\gamma^C\right)\mathbb{E}\left[\nabla_h\right]^2+\mathsf{Var}\left(\gamma_h^R\right)\right)\end{aligned}$$

where, in the first line, the middle term goes is zero as the noise terms are i.i.d. and zero mean. The second line holds because each γ is an independent variable with zero mean, and so

$$\mathsf{E}[(\gamma)^2] = \mathsf{Var}\left(\gamma\right) - (\mathsf{E}[\gamma])^2 = \mathsf{Var}\left(\gamma\right).$$

Finally, we approximate $\mathsf{E}[\sum_{h=1}^{d}\nabla_h] \approx \frac{\eta}{2}$. We argue that if the data were uniformly distributed in a hypersphere of radius η around the centroid, the expected distance from the centroid would be $\eta/2$. This gives the final error term over all dimensions:

$$\mathsf{MSE}\left(\tilde{\mu}^{(t)}\right) \approx \frac{k^3}{4N^2}\left(\mathsf{Var}\left(\gamma^C\right)\eta^2+4d\mathsf{Var}\left(\gamma^R\right)\right) \tag{16}$$

Privacy Budget Splitting We now use the approximate analysis in Eqn 16 to determine the optimal privacy budget split between the relative sum and the count. We substitute the variance of the noise terms from Eqn 7 into Eqn 16:

$$\mathsf{MSE}\left(\tilde{\mu}^{(t)}\right) \approx \frac{k^3\eta^2 T}{4N^2}\left((\sigma^C)^2+4d(\sigma^R)^2\right). \tag{17}$$

In our privacy analysis, we compute the noise multiplier σ such that we achieve $\frac{1}{\sigma}$-GDP. Thus, we need to split the noise multiplier between the relative sum and the count following Theorem 2.2:

$$\sqrt{\left(\frac{1}{\sigma^R}\right)^2+\left(\frac{1}{\sigma^C}\right)^2} = \frac{1}{\sigma} \tag{18}$$

We minimize the per iteration error, subject to this constraint, using Lagrange multipliers, which gives:

$$\sigma^C = \sqrt[4]{4d}\sigma^R. \tag{19}$$

Simply scaling each sigma by this ratio gives Eqn 7. Substituting Eqn 7 back into the error analysis gives:

$$\mathsf{MSE}\left(\tilde{\mu}^{(t)}\right) = \frac{k^3\eta^2 T\sigma^2(1+\sqrt{4d})^2}{4N^2} \tag{20}$$

Optimal Number of Iterations Using this analysis, we can determine a heuristic for the number of iterations. Following Su et al. [65], we assume that the error in each iteration is less than $0.004\times B$. Re-arranging for T gives:

$$T < \frac{4N^2(0.004)}{k^3\eta^2\sigma^2(1+\sqrt{4d})^2}. \tag{21}$$

However, as Su et al. [65] explain, in practice, there is no need to go beyond seven iterations and at least two iterations are needed to gain useful results. Therefore, we truncate this analysis such that the number of iterations is in the range $[2,7]$.

5.6 Setting the Maximum Radius Parameter

We derive a dimensionality-aware radius parameter η based on the geometric properties of the feature space. Consider a d-dimensional bounded feature space where each dimension is constrained to the interval $[-B,B]$. The domain diagonal β, representing the maximum possible distance between any two points in this space, is given by:

$$\beta = \sqrt{d}\times(2B) \tag{22}$$

When partitioning this space into k clusters, each cluster occupies a fraction of the total volume. The total volume of the feature space is $(2B)^d$. Assuming uniform partitioning, each cluster occupies a volume of $(2B)^d/k$. Consequently, the effective length of each cluster along any dimension can be expressed as:

$$L_c = \frac{2B}{k^{1/d}} \tag{23}$$

Assuming hypercube-shaped clusters, the maximum distance from a centroid to any point within the cluster (the cluster radius) is half the cluster's diagonal. This can be calculated as:

$$\text{Cluster Radius} = \frac{L_c}{2}\times\sqrt{d} = \frac{B}{k^{1/d}}\times\sqrt{d} = \frac{\beta}{2k^{1/d}} \tag{24}$$

Based on this analysis, we propose a heuristic η_d to constrain the cluster radius in a d-dimensional space as:

$$\eta_d = \frac{\alpha\beta}{2k^{1/d}} \tag{25}$$

where α is a scaling factor we set to 0.8 based on experiments using synthetic data in Appendix B.

In scenarios with low d and high k, η becomes increasingly restrictive as $k^{1/d}$ grows larger, providing tighter bounds on cluster radii. However, as dimensions increase, the curse of dimensionality necessitates larger cluster radii to accommodate the exponential growth of volume in the space, $(2B)^d$. Our approach takes this into account through $k^{1/d}$ approaching unity in high dimensions. This dimensional scaling aligns with the intuition that radius constraints are most meaningful in lower-dimensional spaces, where cluster boundaries remain well-defined, whereas, in high-dimensional spaces, the curse of dimensionality renders such constraints increasingly less effective as distance metrics lose their discriminative power.

We consider two distinct approaches to implement the radius constraint. In the "Constant" approach, the radius constraint is consistently enforced throughout the clustering process. Alternatively, the "Step" approach initializes with a

broader constraint of $\beta/2$ at iteration zero and then transitions to the computed radius η for the remaining iterations. This two-phase strategy allows for initial flexibility in centroid placement while gradually imposing stricter constraints for fine-tuning the centroids. We show in Appendix B that the "Step" approach is superior.

6 Evaluation

In this section, we provide an extensive evaluation of our work in terms of the following questions:

- Q1 How does the utility of FastLloyd compare with state-of-the-art in central model [65] for interactive DP k-means?
- Q2 How does the utility of FastLloyd scale with varying the number of dimensions and number of clusters?
- Q3 How does the runtime, communication, and number of rounds for FastLloyd compare with the state-of-the-art in federated k-means using secure computation [50]?
- Q4 How does the runtime and communication of FastLloyd scale with varying the number of dimensions, clusters, and data points?

6.1 Experimental Setup

6.1.1 Implementation

Our experimental evaluation was conducted on a Macbook Pro M2 Max (30-core CPU, 38-core GPU, 64GB RAM) using Open MPI [26] for multiparty communication. Following Mohassel et al. [50], we evaluate in the LAN setting with simulated network latency (0.25ms per send operation), noting that WAN runtimes can be derived using a linear cost model. Our experimental framework utilizes a two-client setup with balanced dataset partitioning. The MSA protocol (Figure 1) operates over the ring $\mathbf{R} = \mathcal{Z}_{2^{32}}$, employing fixed-point representation with precision factor $q = 16$. All experiments use randomly partitioned datasets, with results averaged over 100 runs and reported with 95% confidence intervals where applicable. For FastLloyd, we set the radius constraint $\alpha = 0.8$ using the "Step" strategy (see Appendix B for detailed analysis and justification of these parameter choices). We set $\delta = \frac{1}{N \log N}$ and report the total ε over all iterations for all experiments. Our implementation is publicly available at https://doi.org/10.5281/zenodo.15530617.

6.1.2 Baselines

We evaluate FastLloyd against several baselines. For evaluating utility, we implement three protocols. First, Lloyd [44], a non-private Lloyd's algorithm adapted to our federated framework with sphere packing initialization. Second, SuLloyd [65], which adapts Su et al.'s centralized differentially private k-means clustering algorithm to our federated setting using our masked secure aggregation (MSA). Third, GLloyd, a modification of SuLloyd replacing Laplace noise with Gaussian noise and using composition and privacy budget analysis similar to our work. In Appendix B.1, we provide the details of how the analysis in Section 5.5 differs for GLloyd. For computational and communication efficiency benchmarking, we compare against MohLloyd [50], established by Hedge et al. [31] as the most efficient secure k-means protocol.

6.1.3 Datasets

Our evaluation employs both real and synthetic datasets. We use established datasets from the clustering datasets repository [25], following Su et al. [65] and Mohassel et al. [50] for comparability. For Birch2 [73], we take $25,000$ random samples from the $100,000$ sample dataset.

For evaluating scalability, following [65], we generate synthetic datasets (*Synth*) using the `clusterGeneration` R package [58], which enables control over inter-cluster separation (in $[-1,1]$). The synthetic datasets contain $N = 10,000$ samples across k clusters, with cluster sizes following a $1:2:\ldots:k$ ratio. We incorporate a random number (in $[0,100]$) of randomly sampled outliers and set the cluster separation degrees in $[0.16, 0.26]$, spanning partially overlapping to separated clusters. While Su et al. [65] evaluate configurations up to $k = 10$ and $d = 10$, we extend the evaluation to $k = 32$ and $d = 512$ in powers of two, creating 45 parameter combinations. Each combination generates three datasets with different random seeds. To assess scalability at higher cluster counts, we further extend our evaluation with *Synth-K*, incorporating configurations up to $k = 128$ for $d = 2$.

For benchmarking performance, we create synthetic datasets (*TimeSynth*) with balanced cluster sizes ($C_{avg} = \frac{N}{k}$), varying N (10K-100K), d (2-5), and k (2-5), following Mohassel et al. [50]. Table 2 summarizes these datasets. Following standard practice in DP literature [8, 16, 47, 65], all datasets are normalized to $[-1,1]$.

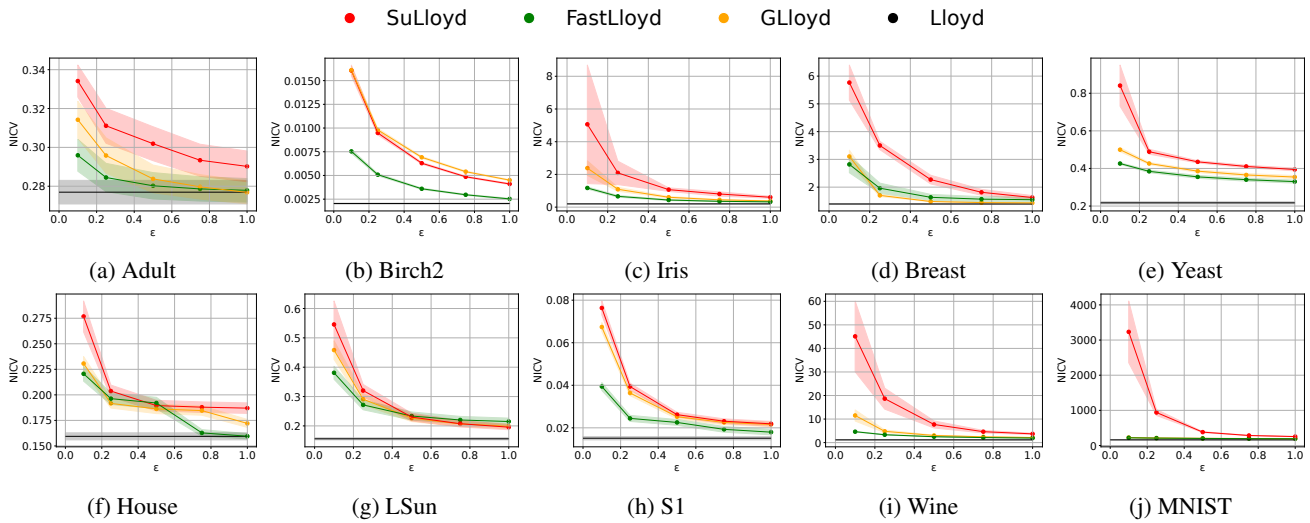
6.2 Utility Evaluation

6.2.1 Metrics

We use the Normalized Intra-cluster Variance (NICV) from prior work [65] as our primary utility metric. NICV normalizes the k-means objective function (Eqn 1) by dividing it by the dataset size:

$$NICV(O) = \frac{1}{N} \cdot \sum_{j=1}^{k} \sum_{x_l \in O_j} ||x_l - \mu_j||^2$$

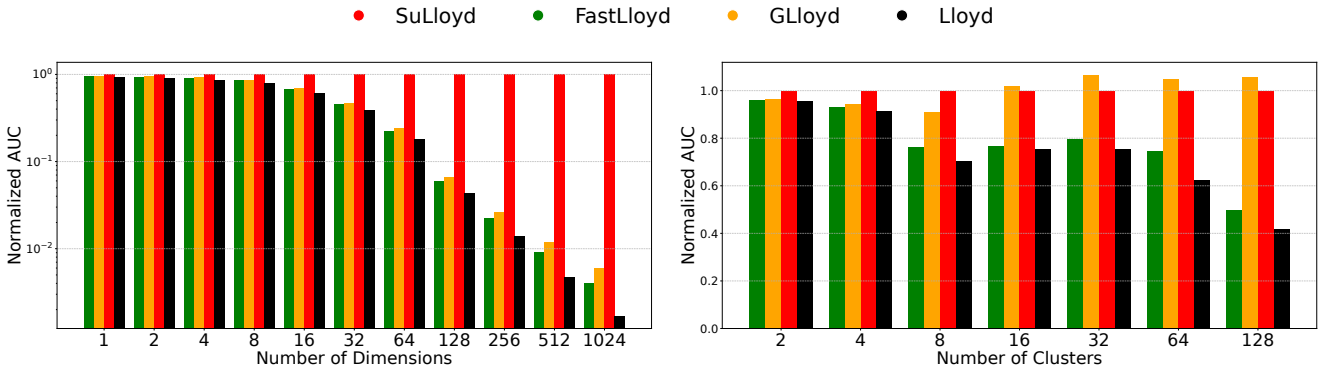
To facilitate systematic comparison across methods for the synthetic datasets, we summarize each method's performance



(a) Adult (b) Birch2 (c) Iris (d) Breast (e) Yeast

(f) House (g) LSun (h) S1 (i) Wine (j) MNIST

Figure 2: Normalized Intra-cluster Variance (NICV) vs ε for real datasets.



(a) Varying the number of dimensions using G2 ($k = 2$, log scale).

(b) Varying the number of clusters using *Synth-K* ($d = 2$).

Figure 3: Scaling of AUC over number of dimensions and clusters.

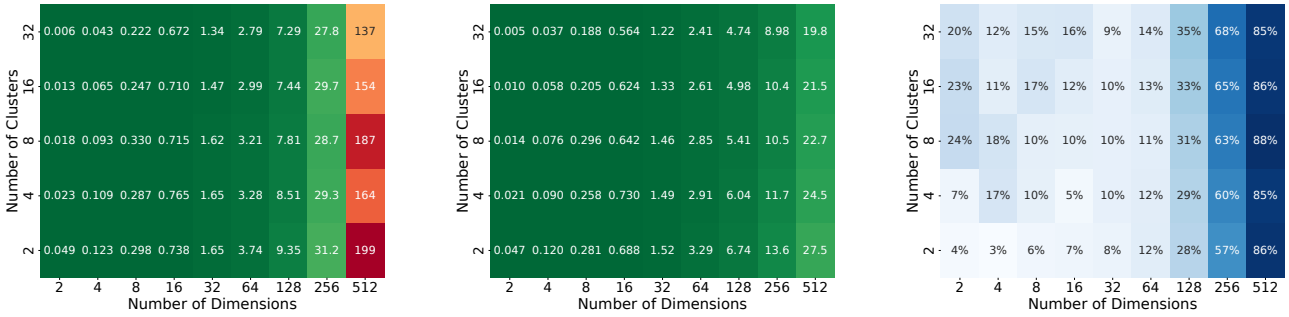
over different privacy budgets using the area under the curve (AUC) of NICV values against $\varepsilon \in [0.1, 0.25, 0.5, 0.75, 1.0]$, computed via the trapezoidal rule:

$$AUC = \sum_{i=1}^{n-1} \frac{NICV_i + NICV_{i+1}}{2} \cdot (\varepsilon_{i+1} - \varepsilon_i) \qquad (26)$$

We evaluate additional metrics in Appendix D.

6.2.2 Real World Datasets

Figure 2 shows the NICV against various privacy budgets on ten real-world datasets to answer Q1. The shading shows the 95% confidence interval of the mean over the 100 experiments. We observe that FastLloyd and GLloyd consistently outperform SuLloyd across all high-dimensional datasets (e.g., Wine, Breast, Yeast, MNIST). This aligns with our theoretical expectations as the Gaussian mechanism composes much more favourably in high-dimensional spaces than the Laplace mechanism. Additionally, FastLloyd outperforms GLloyd in all cases, especially in datasets with low dimensions and a high number of clusters (e.g., Birch2, S1), where GLloyd does not outperform SuLloyd. This highlights the effect of imposing a radius constraint on the protocol, which, as discussed in Section 5.1, is most restrictive in low-dimensional, high-cluster settings, further reducing the noise added to the centroids. While all differentially private methods eventually approach the non-private baseline as privacy budget increases, FastLloyd achieves this convergence at substantially lower privacy budgets, establishing it as the superior approach for DP k-means clustering across all datasets.



(a) SuLloyd (b) FastLloyd (c) Improvement

Figure 4: AUC comparison of FastLloyd vs. SuLloyd on the *Synth* datasets.

Table 2: Summary of Datasets Used in Evaluation

Dataset	N	d	k
Iris [23]	150	4	3
LSun [66]	400	2	3
S1 [24]	5000	2	15
House [25]	1837	3	3
Adult [6]	48842	6	3
Wine [1]	178	13	3
Breast [68]	699	9	2
Yeast [51]	1484	8	10
MNIST [41]	10000	784	10
Birch2 [73]	25000	2	100
G2 [46]	2048	2-1024	2
Synth	10K	2-512	2-32
Synth-K	10K	2	2-128
TimeSynth	10K, 100K	2,5	2,5

6.2.3 Synthetic Datasets

To answer Q2, we follow an approach similar to Su et al. [65] and evaluate FastLloyd and SuLloyd on our synthetic datasets varying both k and d. Figure 4 shows heatmaps of the AUC values for SuLloyd and FastLloyd on the synthetic datasets. To clarify the difference between the protocols, Figure 4c shows the percentage of reduction in AUC of FastLloyd over that of SuLloyd. From the heatmaps, we observe that FastLloyd outperforms SuLloyd across all datasets, with the improvement being more pronounced in higher dimensions and in higher k.

To investigate higher limits, we further extend the evaluation to the G2 [46] datasets with d up to 1024, and *Synth-K* datasets with k up to 128. Figures 3a and 3b present a comparative analysis of AUC performance across Lloyd, FastLloyd, GLloyd, and SuLloyd protocols, evaluated on G2 and *Synth-K* datasets. The AUC values are normalized relative to SuLloyd as the baseline. Figure 3a examines dimensionality scaling by varying d up to 1024 while maintaining $k = 2$, whereas Figure 3b shows cluster scaling by varying k up to 128 while fixing $d = 2$. In Figure 3a, we observe that for low dimensions, all protocols perform very closely to the baseline (Lloyd), but as the dimension increases, the task becomes much more challenging for the private protocols, with FastLloyd outperforming all others. In Figure 3b, we observe that FastLloyd outperforms all other protocols across all cluster counts, with the improvement being more pronounced at higher cluster counts. This is to be expected as the radius constraint becomes more restrictive with higher cluster counts, leading to a more significant reduction in noise added to the centroids.

Since Su et al. found SuLloyd outperformed all other approaches for $d > 3$ [65], and we outperform SuLloyd, we claim that FastLloyd is state-of-the-art in DP k-means for $d > 3$. FastLloyd also fixes the scalability issue that Su et al.'s work [65] observed in high k and enables us to scale to much larger dimensions.

6.3 Runtime Evaluation

To answer questions Q3 and Q4, we compare FastLloyd with the MohLloyd protocol of Mohassel et al. [50] on a variety of common datasets in terms of runtime and communication size (per iteration). Table 3 shows the summary of the evaluation with two clients (the setting considered in Mohassel et al.'s work [50]). Values for MohLloyd are taken as reported in the paper by Mohassel et al. [50], noting that the setup is similar to the one used in our evaluation, and the gap in performance is more than what could be accounted for by different setups. In terms of runtime, MohLloyd executes in the order of minutes, while FastLloyd executes in milliseconds, offering five orders of magnitude speedup. In terms of communication size, MohLloyd requires communicating gigabytes of data per iteration, while FastLloyd needs a fraction of a kilobyte, offering up to seven orders of magnitude reduction in size. In terms of communication rounds, MohLloyd requires $\Theta(\lceil \log k \rceil)$ communication rounds per iteration [31], while FastLloyd only requires one. Since MohLloyd was found to be the state-of-

Dataset	Parameters			Runtime (ms)			Comm (bytes)		
	N	k	d	FastLloyd	MohLloyd [50]	Speedup	FastLloyd	MohLloyd [50]	Reduction
TimeSynth	10K	2	2	3.03 ± 0.02	10500	3465×	192	2.557e8	$1.33e6\times$
		2	5	3.24 ± 0.01	-	-	384	-	-
		5	2	4.32 ± 0.02	34050	7882×	480	9.746e8	$2.03e6\times$
		5	5	4.95 ± 0.04	-	-	960	-	-
TimeSynth	100K	2	2	12.57 ± 0.02	105120	8363×	192	2.467e9	$1.28e7\times$
		2	5	13.17 ± 0.03	-	-	384	-	-
		5	2	22.05 ± 0.03	347250	15748×	480	9.535e9	$1.99e7\times$
		5	5	22.56 ± 0.02	-	-	960	-	-
LSun	400	3	2	2.6 ± 0.04	1481	570×	288	-	-
S1	5K	15	2	5.81 ± 0.08	49087	8449×	1440	-	-

Table 3: Overhead comparison per iteration against MohLloyd [50] for two clients (100 runs with mean and 95% confidence reported for FastLloyd)

the-art in secure federated k-means [31], we conclude that we advance the state-of-the-art while offering output privacy (which would only further slow down Mohassel et al. [50]).

7 Conclusion

In this work, we designed FastLloyd, a new private protocol for federated k-means. We have shown that FastLloyd is secure in the computational model of DP and analyzed its utility. Compared to state-of-the-art solutions in the central model of DP, FastLloyd results in higher utility across a wide range of real datasets and scales effectively to larger dimensions and number of clusters. FastLloyd also achieves five orders of magnitude faster runtime than the state-of-the-art in secure federated k-means across a variety of problem sizes. In summary, we provide an efficient, private, and accurate solution to the horizontally federated k-means problem.

Acknowledgments

We gratefully acknowledge the support of NSERC for grants RGPIN-2023-03244, IRC-537591, the Government of Ontario and the Royal Bank of Canada for funding this research.

Open Science

We make all source code and datasets used in our paper available here: https://doi.org/10.5281/zenodo.15530617. This repository includes all relevant code and supporting scripts necessary to reproduce our experiments and results.

Ethics Considerations

Our work develops a new protocol for privately clustering data. Our primary stakeholders are the data scientists who will deploy our protocol. The secondary stakeholders are the subjects of the analyses whose data is aggregated and analyzed in this protocol. We improve the accuracy, privacy, and runtime of existing protocols in the private clustering domain. Simultaneous improvements in all three categories have positive ethical implications for both stakeholders. Improved privacy protection (input and output privacy) reduces the risks for both stakeholders. Our accuracy improvements make the output more useful to the primary stakeholders. Finally, our runtime improvements reduce the computational cost and, thus, the environmental impact of conducting the analysis. All of these improvements incentivize the use of private protocols over non-private alternatives. While our improvements significantly reduce the risks compared to related work, care must be taken to appropriately communicate the inherent risks of deploying any protocol that satisfies a similar security model to both stakeholders.

References

[1] Stefan Aeberhard and M. Forina. Wine. UCI Machine Learning Repository, 1992. DOI: https://doi.org/10.24432/C5PC7J.

[2] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. NP-hardness of euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, January 2009.

[3] Maria-Florina Balcan, Travis Dick, Yingyu Liang, Wenlong Mou, and Hongyang Zhang. Differentially private clustering in high-dimensional Euclidean spaces. In

Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 322–331. PMLR, 06–11 Aug 2017.

[4] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. Private summation in the multi-message shuffle model. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 657–676, New York, NY, USA, 2020. Association for Computing Machinery.

[5] Borja Balle and Yu-Xiang Wang. Improving the Gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 394–403. PMLR, 10–15 Jul 2018.

[6] Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

[7] K.P. Bennett, P.S. Bradley, and A. Demiriz. Constrained k-means clustering. Technical Report MSR-TR-2000-65, Microsoft Research, May 2000.

[8] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the SuLQ framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '05, pages 128–138. Association for Computing Machinery, 2005.

[9] 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. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery.

[10] Paul Bunn and Rafail Ostrovsky. Secure two-party k-means clustering. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, page 486–497, New York, NY, USA, 2007. Association for Computing Machinery.

[11] T-H. Hubert Chan, Elaine Shi, and Dawn Song. Optimal Lower Bound for Differentially Private Multi-party Aggregation. In Leah Epstein and Paolo Ferragina, editors, *Algorithms – ESA 2012*, pages 277–288, Berlin, Heidelberg, 2012. Springer.

[12] Alisa Chang, Badih Ghazi, Ravi Kumar, and Pasin Manurangsi. Locally private k-means in one round. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1441–1451. PMLR, 18–24 Jul 2021.

[13] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979.

[14] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *IEEE Transactions On Information Theory*, 22(6), 1976.

[15] Jinshuo Dong, Aaron Roth, and Weijie J. Su. Gaussian differential privacy. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(1):3–37, 02 2022.

[16] Cynthia Dwork. A firm foundation for private data analysis. *Communications of the ACM*, 54(1):86–95, 2011.

[17] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[18] Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, August 2014.

[19] Cynthia Dwork, Adam Smith, Thomas Steinke, and Jonathan Ullman. Exposed! a survey of attacks on private data. *Annual Review of Statistics and Its Application*, 4(1):61–84, 2017.

[20] Dan Feldman, Amos Fiat, Haim Kaplan, and Kobbi Nissim. Private coresets. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, STOC '09, pages 361–370. Association for Computing Machinery, 2009.

[21] Dan Feldman, Chongyuan Xiang, Ruihao Zhu, and Daniela Rus. Coresets for differentially private k-means clustering and applications to privacy in mobile sensor networks. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 3–15. ACM, 2017.

[22] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, et al. Safelearn: Secure aggregation for private federated learning. In *2021 IEEE*

Security and Privacy Workshops (SPW), pages 56–62. IEEE, 2021.

[23] R. A. Fisher. Iris. UCI Machine Learning Repository, 1988. DOI: https://doi.org/10.24432/C56C76.

[24] P. Fränti and O. Virmajoki. Iterative shrinking method for clustering problems. *Pattern Recognition*, 39(5):761–765, 2006.

[25] Pasi Fränti and Sami Sieranoja. K-means properties on six clustering benchmark datasets, 2018.

[26] Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph H. Castain, David J. Daniel, Richard L. Graham, and Timothy S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation, September 2004.

[27] Badih Ghazi, Ravi Kumar, and Pasin Manurangsi. Differentially private clustering: Tight approximation ratios. In *Advances in Neural Information Processing Systems*, volume 33, pages 4040–4054. Curran Associates, Inc., 2020.

[28] Badih Ghazi, Pasin Manurangsi, Rasmus Pagh, and Ameya Velingker. Private aggregation from fewer anonymous messages. In *Advances in Cryptology – EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part II*, page 798–827, Berlin, Heidelberg, 2020. Springer-Verlag.

[29] Zakaria Gheid and Yacine Challal. Efficient and privacy-preserving k-means clustering for big data mining. pages 791–798, 08 2016.

[30] Attri Ghosal, Arunima Nandy, Amit Kumar Das, Saptarsi Goswami, and Mrityunjoy Panday. A short review on different clustering techniques and their applications. *Emerging Technology in Modelling and Graphics: Proceedings of IEM Graph 2018*, pages 69–83, 2020.

[31] Aditya Hegde, Helen Möllering, Thomas Schneider, and Hossein Yalame. SoK: Efficient privacy-preserving clustering. Publication info: Published elsewhere. PoPETs '21.

[32] Naoise Holohan, Stefano Braghin, Pól Mac Aonghusa, and Killian Levacher. Diffprivlib: the IBM differential privacy library. *ArXiv e-prints*, 1907.02444 [cs.CR], July 2019.

[33] Thomas Humphries, Rasoul Akhavan Mahdavi, Shannon Veitch, and Florian Kerschbaum. Selective MPC: Distributed computation of differentially private key-value statistics. In *Proceedings of sthe 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, pages 1459–1472, New York, NY, USA, 2022. Association for Computing Machinery.

[34] Geetha Jagannathan, Krishnan Pillaipakkamnatt, Rebecca Wright, and Daryl Umano. Communication-efficient privacy-preserving clustering. *Transactions on Data Privacy*, 3:1–25, 04 2010.

[35] Geetha Jagannathan and Rebecca N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, page 593–599, New York, NY, USA, 2005. Association for Computing Machinery.

[36] Angela Jäschke and Frederik Armknecht. Unsupervised machine learning on encrypted data. *IACR Cryptol. ePrint Arch.*, 2018:411, 2018.

[37] Somesh Jha, Luis Kruger, and Patrick McDaniel. Privacy preserving clustering. In *Proceedings of the 10th European Conference on Research in Computer Security*, ESORICS'05, page 397–417, Berlin, Heidelberg, 2005. Springer-Verlag.

[38] Zoe L. Jiang, Ning Guo, Yabin Jin, Jiazhuo Lv, Yulin Wu, Zechao Liu, Junbin Fang, S.M. Yiu, and Xuan Wang. Efficient two-party privacy-preserving collaborative k-means clustering protocol supporting both storage and computation outsourcing. *Information Sciences*, 518:168–180, 2020.

[39] Swanand Kadhe, Nived Rajaraman, O Ozan Koyluoglu, and Kannan Ramchandran. Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning. *arXiv preprint arXiv:2009.11248*, 2020.

[40] Hyeong-Jin Kim and Jae-Woo Chang. A privacy-preserving k-means clustering algorithm using secure comparison protocol and density-based center point selection. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 928–931, 2018.

[41] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[42] Josh Levy-Kramer. k-means-constrained, April 2018.

[43] Zitao Li, Tianhao Wang, and Ninghui Li. Differentially private vertical federated clustering. *Proc. VLDB Endow.*, 16(6):1277–1290, apr 2023.

[44] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982.

[45] Zhigang Lu and Hong Shen. Differentially private k-means clustering with guaranteed convergence. pages 1–1.

[46] P. Fränti R. Mariescu-Istodor and C. Zhong. Xnn graph. *Joint Int. Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, LNCS 10029:207–217, 2016.

[47] Frank D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, page 19–30, New York, NY, USA, 2009. Association for Computing Machinery.

[48] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. Computational differential privacy. In *Advances in Cryptology – CRYPTO 2009*, pages 126–142, 2009.

[49] Prashanth Mohan, Abhradeep Thakurta, Elaine Shi, Dawn Song, and David Culler. Gupt: privacy preserving data analysis made easy. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 349–360. ACM, 2012.

[50] Payman Mohassel, Mike Rosulek, and Ni Trieu. Practical privacy-preserving k-means clustering. *Proceedings on Privacy Enhancing Technologies*, 2020:414 – 433, 2020.

[51] Kenta Nakai. Yeast. UCI Machine Learning Repository, 1991. DOI: https://doi.org/10.24432/C5KG68.

[52] Tianjiao Ni, Minghao Qiao, Zhili Chen, Shun Zhang, and Hong Zhong. Utility-efficient differentially private k-means clustering based on cluster merging. *Neurocomputing*, 424:205–214, 2021.

[53] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, page 75–84, New York, NY, USA, 2007. Association for Computing Machinery.

[54] Kobbi Nissim and Uri Stemmer. Clustering algorithms for the centralized and local models. In *Proceedings of Algorithmic Learning Theory*, pages 619–653. PMLR, 2018. ISSN: 2640-3498.

[55] Mijung Park, James Foulds, Kamalika Choudhary, and Max Welling. Dp-em: Differentially private expectation maximization. In *Artificial Intelligence and Statistics*, pages 896–904, 2017.

[56] Sankita Patel, Sweta Garasia, and Devesh Jinwala. An efficient approach for privacy preserving distributed k-means clustering based on shamir's secret sharing scheme. In *IFIP Advances in Information and Communication Technology*, pages 129–141. Springer Berlin Heidelberg, 2012.

[57] Wahbeh H. Qardaji, Weining Yang, and Ninghui Li. Differentially private grids for geospatial data. *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 757–768, 2012.

[58] Weiliang Qiu and Harry Joe. *Random Cluster Generation (with Specified Degree of Separation)*, 2023. R package version 1.3.8.

[59] Fang-Yu Rao, Bharath K. Samanthula, Elisa Bertino, Xun Yi, and Dongxi Liu. Privacy-preserving and outsourced multi-user k-means clustering. In *2015 IEEE Conference on Collaboration and Internet Computing (CIC)*, pages 80–89, 2015.

[60] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[61] Arlei Silva and Gowtham Bellala. Privacy-preserving multi-party clustering: An empirical study. In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pages 326–333, 2017.

[62] Jinhyun So, Başak Güler, and A Salman Avestimehr. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. *IEEE Journal on Selected Areas in Information Theory*, 2(1):479–489, 2021.

[63] Uri Stemmer. Locally private k-means clustering. 22(1):176:7964–176:7993, 2021.

[64] Uri Stemmer and Haim Kaplan. Differentially private k-means with constant multiplicative error. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[65] Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, and Hongxia Jin. Differentially private k-means clustering. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, CODASPY '16, pages 26–37. Association for Computing Machinery, 2016.

[66] Alfred Ultsch. Clustering wih som: U* c. *Proc. Workshop on Self-Organizing Maps*, 01 2005.

[67] Jaideep Vaidya and Chris Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*,

KDD '03, page 206–215, New York, NY, USA, 2003. Association for Computing Machinery.

[68] WIlliam Wolberg. Breast Cancer Wisconsin (Original). UCI Machine Learning Repository, 1990. DOI: https://doi.org/10.24432/C5HP4Z.

[69] Genqiang Wu, Yeping He, Jingzheng Wu, and Xianyao Xia. Inherit differential privacy in distributed setting: Multiparty randomized function computation. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 921–928, 2016.

[70] Liu Xiaoyan, Zoe Jiang, Sm Yiu, Xuan Wang, Chuting Tan, Li Ye, Zechao Liu, Yabin Jin, and Junbin Fang. Outsourcing two-party privacy preserving k-means clustering protocol in wireless sensor networks. pages 124–133, 12 2015.

[71] Jiawei Yuan and Yifan Tian. Practical privacy-preserving mapreduce based k-means clustering over large-scale dataset. *IEEE Transactions on Cloud Computing*, 7(2):568–579, 2019.

[72] Jun Zhang, Xiaokui Xiao, Yin Yang, Zhenjie Zhang, and Marianne Winslett. Privgene: differentially private model fitting using genetic algorithms. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 665–676. ACM, 2013.

[73] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, 1997.

A Proofs

A.1 Proof of Theorem 5.1

Theorem 5.1. *If the constraint (4) is satisfied, then:*

$$\Delta^R = \max_{D,D' \in \mathcal{D}} ||R_j^{(t)}(D) - R_j^{(t)}(D')||_2 \leq \eta \tag{6}$$

for all clusters $j \in [k]$, and iterations $t \in [T]$.

Proof. w.l.o.g assume that the datasets differ by a single point x' so that $D' = D \cup \{x'\}$. If x' is not within η of $\tilde{\mu}_j^{(t-1)}$, it will not be assigned to O_j and thus $R_j(D) = R_j(D')$. Therefore, we only need to consider the case where x' is at most η from $\tilde{\mu}_j^{(t-1)}$. By definition, we have:

$$\begin{aligned}
\Delta^R &= \max_{D,D' \in \mathcal{D}} ||S_j^{(t)} - C_j^{(t)} \tilde{\mu}_j^{(t-1)} - (S_j'^{(t)} - C_{j'}{}^{(t)} \tilde{\mu}_j^{(t-1)})||_2 \\
&= \max_{D,D' \in \mathcal{D}} ||S_j^{(t)} - C_j^{(t)} \tilde{\mu}_j^{(t-1)} - S_j^{(t)} - x' + (C_j^{(t)} + 1)\tilde{\mu}_j^{(t-1)}||_2 \\
&= \max_{D,D' \in \mathcal{D}} ||\tilde{\mu}_j^{(t-1)} - x'||_2 \\
&\leq \eta
\end{aligned}$$

where the last line follows by (Eqn 4). □

A.2 Proof of Theorem 5.2

We first prove the following helpful lemma.

Lemma A.1. *Our noise mechanism (defined in Eqn 8) of $f(D) + \gamma$ where $\gamma \sim \mathcal{N}(0, \sigma^2 T (\Delta^{(f)})^2)$ applied over T adaptive iterations of f is $(\frac{1}{\sigma})$-GDP, where $\Delta^{(f)}$ is the sensitivity of f.*

Proof. After choosing σ, by Theorem 2.1 from Dong et al. [15, Theorem 1], we have that each application of a Gaussian mechanism with noise multiplier σ is $\frac{\Delta^{(f)}}{\sigma}$-GDP. We apply the Gaussian mechanism adaptively over T iterations. Which by Theorem 2.2 from Dong et al. [15, Corollary 2], gives us $\frac{\Delta^{(f)}\sqrt{T}}{\sigma}$-GDP. Thus, by multiplying σ by $\sqrt{T}\Delta^{(f)}$, we get $\frac{1}{\sigma}$-GDP. □

We now restate and prove Theorem 5.2.

Theorem 5.2. *Algorithm 1 ensures $(\varepsilon(\lambda), \delta)$-IND-CDP-MPC in the presence of a semi-honest, polynomial time adversary who controls at most a single party.*

Proof. To prove the algorithm satisfies Definition 3.1, we need to consider the view from each party. We begin with the view of the server. First, let us assume that each client samples the random mask r_i from a random oracle over $\mathbf{R}$. Under this assumption, the $Enc(\cdot)$ function satisfies information-theoretic security as it is a one-time pad using r_i as the pad. In practice, we implement the sampling of r_i using a PRNG and thus reduce from information-theoretic to computational security with a negligible term $negl(\lambda)$. Therefore, the view of the server satisfies $\varepsilon(\lambda)$-IND-CDP-MPC with $\varepsilon = 0$.

Regarding the clients, a client's view consists of their own dataset D_A, and the output from each iteration of the protocol. We note that the initialization is data-independent and thus is indistinguishable between neighbouring datasets. Each iteration consists of the assignment and updating of the cluster centroids. The assignment step uses the published centroids from the previous iteration (post-processing) to divide the dataset into clusters. We can then apply parallel composition over each of the clusters. Thus, we can focus on the privacy cost of a single cluster for the remainder of the proof.

First, we set σ using Theorem 2.3 from Dong et al. [15, Corollary 1]. We choose the minimum σ such that (ε, δ)-DP iff $\frac{1}{\sigma}$-GDP by Theorem 2.3. This is equivalent to finding the noise multiplier for a sensitivity one, single application, of the Gaussian mechanism with noise multiplier σ. To solve this minimization, we use Algorithm 1 from Balle and Wang [5].

The updating of the cluster centroid applies the Gaussian mechanism twice, once to the relative sums and once to the counts. We split the overall σ into σ^R and σ^C following Eqn 7. We begin with the analysis of the relative sum. In Theorem 5.1, we show that the sensitivity of the relative sum is η. Applying Lemma A.1 we get that the relative sum over T iterations is $\frac{1}{\sigma^R}$-GDP. The sensitivity of the count is 1, and thus the count

is $\frac{1}{\sigma^C}$-GDP by similar analysis. Finally, applying Theorem 2.2. We get that

$$\sqrt{\left(\frac{1}{\sigma^R}\right)^2 + \left(\frac{1}{\sigma^C}\right)^2} = \frac{1}{\sigma} \tag{27}$$

and thus the we get $\frac{1}{\sigma}$-GDP over the entire protocol which implies (ε, δ)-DP (because of how we chose σ). Applying parallel composition over all clusters, the client's view is (ε, δ)-IND-CDP-MPC, with no computational assumption (as we use the information-theoretic DP properties of differential privacy). Taking the worst case over the clients and the server, the result follows. □

A.3 Proof of Quantized DP Noise

We argue why quantizing the DP noise is acceptable in Figure 1. In essence, because the sensitive data is already quantized, adding quantized noise to it is equivalent to adding the noise first and then performing the quantization, which aligns with the post-processing lemma in differential privacy.

This can be intuitively understood by noting that:

- The data before noise addition is already quantized.
- The sum of a quantized value and a non-quantized value will only have information of the non-quantized variable in the less significant bits (i.e., those lost to quantization).
- Hence, quantizing the sum is equivalent to dropping this lower bit information, which is similar to quantizing the second variable prior to addition.

We show this formally in the following Theorem.

Theorem A.2 (Quantization and Differential Privacy)**.** *Quantizing Laplace noise at the same level of quantization as that used in the Masked Secure Aggregation (MSA) protocol does not violate the privacy guarantees offered by differential privacy.*

Proof. We begin by observing that any value, say v, can be split into two parts upon quantization: the integral part, $\bar{v}$, and the fractional part, $\hat{v}$, such that $\bar{v} \times 2^q$ is the (rounded) integral part of $v \times 2^q$ (i.e. it is $\lceil v \times 2^q \rfloor$), and $\hat{v}$ denotes the fractional part that gets lost due to quantization. This notation can similarly be applied to the noise variable, η.

In our context, the sensitive data we wish to protect is $\bar{v}$ since the entire protocol operates in a quantized environment, i.e., v is never transmitted. We aim to demonstrate the following equivalence:

$$\tilde{v} + \tilde{\eta} = \lceil v \times 2^q \rfloor + \lceil \eta \times 2^q \rfloor = \lceil (\bar{v} + \eta) \times 2^q \rfloor$$

The right-hand side represents quantization applied as post-processing to the sum of the sensitive data and the noise, which adheres to the rules of differential privacy.

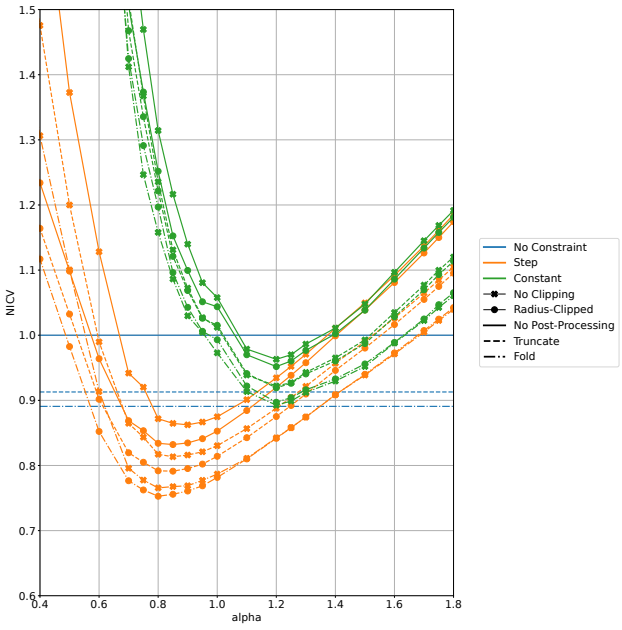


Figure 5: Ablation of α and post-processing strategy on NICV for $\varepsilon = 0.1$.

After expanding the right-hand side, we obtain:

$$\lceil (\bar{v} + \eta) \times 2^q \rfloor = \lceil \bar{v} \times 2^q + \bar{\eta} \times 2^q + \hat{\eta} \times 2^q \rfloor$$

We can take out $\bar{v} \times 2^q$ and $\bar{\eta} \times 2^q$ from the rounding operator because they are exact integers (indeed $\bar{v} \times 2^q$ is $\lceil v \times 2^q \rfloor$). We then obtain:

$$\lceil (\bar{v} + \eta) \times 2^q \rfloor = \lceil v \times 2^q \rfloor + \lceil \eta \times 2^q \rfloor + \lceil \hat{\eta} \times 2^q \rfloor$$

However, $\hat{\eta} \times 2^q$ will be lost due to quantization, hence:

$$\lceil (\bar{v} + \eta) \times 2^q \rfloor = \lceil v \times 2^q \rfloor + \lceil \eta \times 2^q \rfloor = \tilde{v} + \tilde{\eta}$$

□

B Ablation of Parameters

In this section, we empirically justify our choice of hyperparameter α and the post-processing strategy. We choose these parameters to minimize the Normalized Intra-Cluster Variance (NICV) metric over a set of synthetic datasets. The synthetic datasets were generated using the `clusterGeneration` package [58] varying the number of dimensions (d) and clusters (k) as $(k,d) \in \{2,4,8,16\}^2$, resulting in 16 variations. For each variation, we generate three different datasets with the degrees of separation $\{0.25, 0.5, 0.75\}$. The average cluster size is fixed to $C_{avg} = \frac{2048}{k}$ and the cluster sizes are randomly sampled in the range $[0.70 \cdot C_{avg}, 1.30 \cdot C_{avg}]$, leading to a total dataset size in the range $[1433, 2662]$.

All the ablation experiments were done on $\varepsilon = 0.1$. The results are shown in Figure 5, where we plot the average NICV over all datasets for different values of α and post-processing strategies. The average is then divided by the NICV of the naive baseline of no post-processing and no radius constraint. We observe that the "folding" post-processing strategy consistently outperforms the "truncation" and "none" strategies. We also observe that the "Step" strategy outperforms the "Constant" strategy and provides most performance improvement for $\alpha \approx 0.8$. We also note that "radius clipping" provides a consistent improvement. Based on these results, we choose $\alpha = 0.8$ and the "Step" constraint strategy with a post-processing strategy of "folding" after "radius clipping" is applied.

B.1 Error Analysis of Simple Gaussian Mechanism

We follow a similar analysis to Section 5.5 to derive the parameters used for GLloyd. Since we do not modify the algorithm (only change the noise distribution), we can use Su et al.'s MSE analysis unchanged:

$$\mathsf{MSE}\left(\tilde{\mu}^{(t)}\right) \approx \frac{k^3}{N^2}\left(\mathsf{Var}\left(\gamma^S\right) + 4\rho^2 \mathsf{Var}\left(\gamma^C\right)\right) \tag{28}$$

Substituting the variance following our noise mechanism (with the domain-based sensitivity) gives:

$$\mathsf{MSE}\left(\tilde{\mu}^{(t)}\right) \approx \frac{k^3}{N^2}\left(dT(\sigma^S)^2 + 4\rho^2 T(\sigma^C)^2\right). \tag{29}$$

Minimizing this using Lagrange multipliers such that:

$$\sqrt{\left(\frac{1}{\sigma^R}\right)^2 + \left(\frac{1}{\sigma^C}\right)^2} = \frac{1}{\sigma} \tag{30}$$

gives the following ratio of noise multipliers:

$$\sigma^C = \sqrt{\frac{\sqrt{d}}{2\rho}}\sigma^S \tag{31}$$

Substituting this back into the MSE equation gives:

$$\mathsf{MSE}\left(\tilde{\mu}^{(t)}\right) \approx \frac{k^3 dT\sigma^2(2\rho + \sqrt{d})^2}{N^2} \tag{32}$$

Setting the per iteration MSE to be less than 0.004 and rearranging for T gives:

$$T \leq \frac{N^2(0.004)}{k^3 dT\sigma^2(2\rho + \sqrt{d})^2} \tag{33}$$

which we also truncate to be in $[2, 7]$.

C Extensions

C.1 Serverless MSA

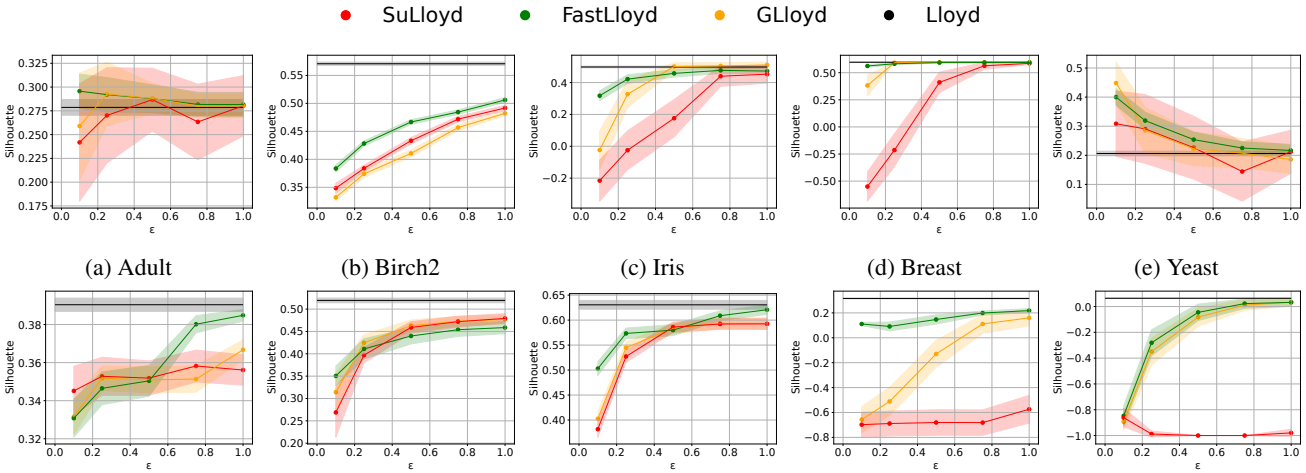
Masked Secure Aggregation (MSA) is at the core of our protocol and requires a semi-honest server to perform the aggregation. An alternative is for all or a subset of the clients to take the role of a computation node in an MPC protocol. In this case, the centroids would be secret shared and aggregated in MPC, then DP noise would need to be added before release every iteration. The noise addition can be implemented by each computation node sampling noise locally and adding it to their shares. Depending on how many nodes are assumed to be honest, this will necessarily add more noise than FastLloyd to ensure privacy in the presence of one or more colluding nodes. Alternatively, the computation nodes can jointly sample the noise using the protocol of Wu et al. [69]. The disadvantage of this approach is that it will add a large computational overhead. However, it would still be significantly faster than the strawman solution of applying an end-to-end MPC protocol such as Mohassel et al. [50] which would also need to sample noise this way (in addition to their already high computation cost).

C.2 Local and Shuffle DP

Another way to alleviate the need for a semi-honest server is to switch to a local or shuffle DP model. However, as discussed in Section 3.1, this is necessarily less accurate than the IND-CDP-MPC model that attains central DP accuracy (even when $N_i = 1$). We note that extending FastLloyd to the local or shuffle DP model is not advantageous, since the data is already noised in these models, secure computations are not required. Thus, improvements in the local and shuffle DP models are a tangential research direction and not the focus of this work.

D Additional Utility Evaluation

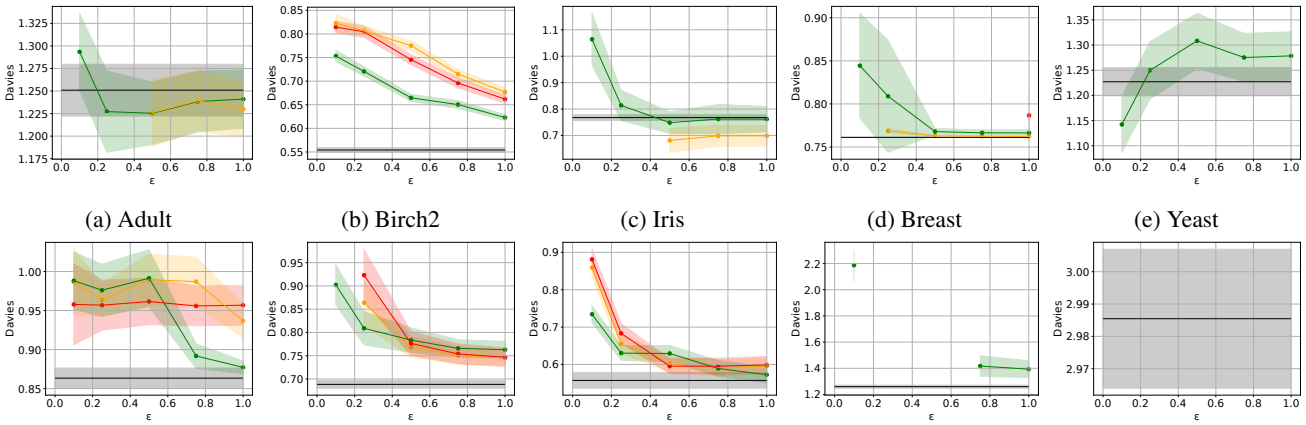
We evaluate our protocol with three metrics: Silhouette Score [60], Davies-Bouldin Index (DBI) [13], and Mean Squared Error (MSE). MSE is the average squared distance between output centroids and ground-truth centroids (from k-means++ matched via the Hungarian algorithm). Because MSE is sensitive to outliers, methods without post-processing (e.g. SuLloyd) can produce extreme MSE values, whereas FastLloyd remains close to the non-private baseline. For any single-cluster result, we set Silhouette Score to -1 and DBI to ∞; accordingly, infinite DBI values are omitted in their figure. Otherwise, all three metrics exhibit similar trends as reported in Section 6 of the main paper.



(a) Adult (b) Birch2 (c) Iris (d) Breast (e) Yeast

(f) House (g) LSun (h) S1 (i) Wine (j) MNIST

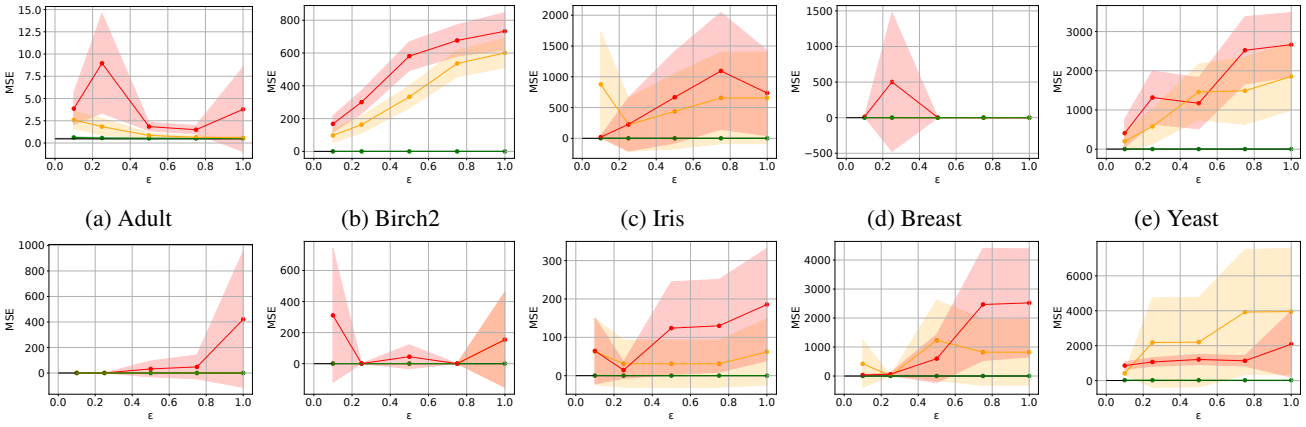
Figure 6: Silhouette Score vs ε for real datasets.



(a) Adult (b) Birch2 (c) Iris (d) Breast (e) Yeast

(f) House (g) LSun (h) S1 (i) Wine (j) MNIST

Figure 7: Davies-Bouldin Index vs ε for real datasets.



(a) Adult (b) Birch2 (c) Iris (d) Breast (e) Yeast

(f) House (g) LSun (h) S1 (i) Wine (j) MNIST

Figure 8: Mean Squared Error (Centroid Alignment) vs ε for real datasets.