Evaluating Differential Privacy in Federated Clustering Under Adversarial Attacks

Urvi Gupta (22B1006)
22B1006@iitb.ac.in

Ojas Maheshwari (22B0965)
22B0965@iitb.ac.in

Department of Computer Science & Engineering
Indian Institute of Technology, Bombay
Course: CS6007: Multi-Agent Machine Learning

Instructor: Prof. Avishek Ghosh

November 8, 2025

Abstract

Federated Learning (FL) is a transformative paradigm for training machine learning models on decentralized data, preserving user privacy by keeping raw data on-device. Unsupervised tasks, such as clustering, are critical applications within this domain, with Federated K-Means ('FedKMeans') being a standard approach. However, the iterative model updates communicated between clients and the central server in FedKMeans are not inherently private. These updates, while aggregated, can leak subtle information about the underlying client data distributions. This leakage creates a significant vulnerability to privacy attacks, most notably Reconstruction Attacks and Membership Inference Attacks (MIAs), where a malicious actor can analyze model updates to infer private information.

This project conducts a practical security analysis to investigate this vulnerability. We implement and compare two federated clustering algorithms: the standard 'KFed' and a privacy-preserving alternative, 'FedDP-KMeans', which integrates Differential Privacy (DP). Our methodology involves two key stages. First, we establish a baseline by evaluating the clustering performance (utility) of both algorithms on benchmark datasets. Second, we design, simulate, and quantify reconstruction attacks aimed at the new model.

Our analysis confirms a significant privacy-utility trade-off. We find that the `FedDP-KMeans` algorithm provides superior utility (clustering quality) to the baseline `KFed` in low-noise (data-point privacy) settings. In high-noise (client-level privacy) settings, `KFed` performs better at very low privacy budgets (epsilon), but `FedDP-KMeans` surpasses it as the budget increases. Most importantly, we demonstrate that while a perfect reconstruction of client data is possible against the non-private model, the calibrated noise introduced by Differential Privacy makes the attack's performance degrade to that of a random guess, rendering it ineffective. The results provide a clear, quantitative demonstration of the necessity and efficacy of DP in securing federated clustering systems. All implementations and experiments are available on our GitHub repository: Fed-DP-KMeans-and-MIA-Attack.

Contents

1 Introduction

Federated Learning (FL) has emerged as a compelling paradigm for machine learning, enabling multiple parties (clients) to collaboratively train a global model without sharing their raw, potentially sensitive data. This decentralized approach, where data remains on-device, offers inherent privacy benefits over traditional centralized methods. However, the FL process is not immune to privacy risks. The model updates (e.g., gradients or, in this case, cluster statistics) sent from clients to the central server can inadvertently leak information about the clients' underlying data.

This information leakage creates a significant attack surface. Malicious actors, particularly a curious central server, can perform sophisticated attacks to compromise client privacy. These include:

- **Membership Inference Attacks (MIA):** Where an adversary aims to determine if a specific data record was used in a client's local training dataset.
- **Data Reconstruction Attacks:** A more severe attack where the adversary attempts to reconstruct the original training data from the shared updates.

The core challenge, therefore, is to balance the *utility* of the collaboratively trained model (i.e., its clustering quality) with the *privacy* guarantees provided to the participating clients.

A powerful defense against such attacks is **Differential Privacy (DP)**. DP provides a formal, mathematical guarantee of privacy by injecting calibrated statistical noise into the data or algorithm outputs. This noise masks the contribution of any single data point (or client), making it difficult for an attacker to distinguish whether any individual's data was included in the dataset.

1.1 Project Contribution

This project implements and evaluates a federated k-means clustering algorithm that incorporates Differential Privacy, based on the work of Scott et al. (2025), "Differentially Private Federated k-Means Clustering with Server-Side Data" [3].

The primary contributions of this work are:

1. **Algorithm Implementation:** We implement the full **FedDP-KMeans** algorithm, which consists of two phases:
   - **FedDP-Init:** A novel, multi-step private initialization algorithm that leverages small, potentially out-of-distribution server-side data to generate a high-quality starting point.
   - **FedDP-Lloyds:** A differentially private, federated version of the standard Lloyd's refinement algorithm.
2. **Utility Benchmarking:** We conduct a comparative analysis of the privacy-utility trade-off, benchmarking our `FedDP-KMeans` implementation against a standard non-private federated baseline, `KFed`.
3. **Security Analysis:** We simulate a powerful, white-box **data-level reconstruction attack** and a client-level reconstruction attack against our model. We demonstrate the attack's perfect success in a non-private setting and its complete failure in the private (DP) setting, empirically verifying the robustness of the DP-enhanced algorithm. We also define and discuss the **Membership Inference Attack (MIA)** implemented as part of this project.

This report details the implemented algorithms, the experimental setup for both utility and security analysis, and the conclusive results of this investigation.

2 Background and Related Work

2.1 Federated K-Means (KFed)

The KFed algorithm [1] serves as a non-private baseline for federated k-means. It is a *local-first* approach and, in our implementation, functions as a one-shot initialization method. The process is as follows:

1. **Client-Side:** Each client independently runs a full k-means algorithm on its own local data to find a set of K_{client} local cluster centers.
2. **Communication:** Each client sends its K_{client} local centers to the server.
3. **Server-Side:** The server collects all local centers from all clients (totaling $N_{clients} \times K_{client}$ centers) and performs a *second* k-means clustering on this set of centers to produce the final K global cluster centers.

The primary drawback of this method is that if client data is non-IID (unbalanced or representing different underlying clusters), the local centers will be highly biased, leading to a poor global initialization.

2.2 Differential Privacy (DP)

Differential Privacy (DP) is a rigorous framework for quantifying privacy leakage. An algorithm $\mathcal{A}$ is said to be (ϵ, δ)-Differentially Private if for any two neighboring datasets, P and P', and for all possible subsets of outputs S, the following inequality holds:

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

Here, ϵ (epsilon) is the privacy loss parameter (smaller is more private), and δ (delta) is the probability of failure. This guarantee ensures that the output of the algorithm is nearly identical whether or not any single individual's data is included.

In the context of this project, we consider two levels of neighboring datasets:

- **Data-point-level DP:** P and P' differ by one individual data record. This is common in `cross-silo FL`, where clients are trusted organizations (e.g., hospitals) and privacy is for the individuals within the silo.
- **Client-level DP:** P and P' differ by the entire dataset of one client. This is a much stronger guarantee, common in `cross-device` FL, where the client (e.g., a mobile phone) is the user and is untrusted.

To achieve DP, noise is added based on the algorithm's *sensitivity*—the maximum impact one individual (or client) can have on the output. This sensitivity is bounded using a technique called **clipping**, which enforces a maximum limit on the L_2 norm of a client's (or datapoint's) contribution.

2.3 Clipping for Sensitivity Bounding

A fundamental prerequisite for applying Differential Privacy is bounding the sensitivity of a function. This is achieved through a process called clipping.

What is Clipping? Clipping enforces a maximum limit, or "cap" (denoted C), on the L_2 norm (magnitude) of a contribution (e.g., a client's update vector) before it is aggregated by the server.

Why is it Necessary? Differential Privacy works by adding noise scaled to the algorithm's sensitivity. Sensitivity is defined as the maximum possible change in the output if one individual's data (or one client's data) is removed.

- In a federated setting, this sensitivity can be unknown and potentially infinite. For example, a client with 1,000,000 data points would have a massive impact on the aggregate sum compared to a client with 10 points.
- It is impossible to add a finite amount of noise to mask an infinite sensitivity.
- Clipping solves this by creating a fixed, known sensitivity. By enforcing that no client's update vector can have a norm greater than C, we guarantee that the maximum impact of any single client is at most C. This allows us to add a finite amount of noise (scaled to C) to achieve DP.

How is it Implemented? (L_2 Clipping) The process, as implemented in the privacy mechanisms, follows a simple rule for an update vector v and a clipping bound C:

1. The client calculates the L_2 norm (length) of its update vector: $||v||_2$.
2. If $||v||_2 \leq C$: The vector is sent as-is.
3. If $||v||_2 > C$: The vector is "shrunk" (rescaled) to have a norm of exactly C. This is done by multiplying it by $C/||v||_2$.

This process, $\hat{v} = v \cdot \min(1, \frac{C}{||v||_2})$, ensures the sensitivity is bounded while preserving the original direction of the update vector.

3 Implemented Algorithm: FedDP-KMeans

Our core implementation is the **FedDP-KMeans** algorithm [3], which is a combination of a novel private initialization (`FedDP-Init`) and a private iterative refinement (`FedDP-Lloyds`). This algorithm is `global-first`, meaning it estimates the global data structure before finding centers, making it more robust to non-IID data than KFed.

3.1 FedDP-Init (Algorithm 1)

This is the novel initialization procedure and is the main contribution of the Scott et al. (2025) paper. It is implemented in our repository in `algorithms/federated_cluster_init.py` and executed sequentially in `run.py`. It works in three private steps:

3.1.1 Step 1: Subspace Estimation (Private PCA)

The goal is to find the k-dimensional subspace that captures the most variance in the *global* client dataset.

1. **Client-Side:** Each client j computes its local data covariance (outer product) matrix, $P_j P_j^T$.
2. **Server-Side:** The server aggregates the (clipped) covariance matrices and adds Gaussian noise to the sum to ensure DP. It then computes the top-k eigenvectors of this noisy aggregate matrix. These eigenvectors form the projection matrix Π.

3.1.2 Step 2: Proxy Weight Computation

The goal is to use the server's (small, public) dataset Q as a proxy for the clients' private data.

1. **Server-Side:** The server projects its own data into the subspace (ΠQ) and sends both Π and ΠQ to the clients.
2. **Client-Side:** Each client projects its local data (ΠP_j). It then finds the closest point in ΠQ for each of its local points $\Pi p \in \Pi P_j$, and computes a histogram (counts) of how many local points are closest to each server point $q \in \Pi Q$.
3. **Server-Side:** The server aggregates these counts and adds Laplace noise (as counts have L_1 sensitivity). These noisy counts $\hat{w}_q$ serve as importance weights for the server's data. The server then runs a weighted k-means algorithm on its own projected data ΠQ (using weights $\hat{w}_q$) to find k projected centers, $\xi_1, ..., \xi_k$.

3.1.3 Step 3: Center Refinement (Projection Back)

The goal is to use the k projected centers ξ to generate k final centers in the original high-dimensional space.

1. **Server-Side:** The server sends the k projected centers $\xi_1, ..., \xi_k$ to all clients.
2. **Client-Side:** Each client j assigns its local points P_j to the closest projected center ξ_r in the projected subspace. For each cluster r, it then computes the sum (m_r^j) and count (n_r^j) of the points assigned to it, using the original, high-dimensional vectors $p \in P_j$.
3. **Server-Side:** The server aggregates the (clipped) sums and counts, adding Gaussian noise to the sums ($\hat{m}_r$) and Laplace noise to the counts ($\hat{n}_r$).
4. **Final Initialization:** The server computes the final initial centers by dividing the noisy sums by the noisy counts: $\nu_r = \hat{m}_r / \hat{n}_r$.

This three-step process completes the `FedDP-Init` algorithm, consuming a portion of the total privacy budget and producing a high-quality set of initial centers $\nu_1, ..., \nu_k$.

3.2 FedDP-Lloyds (Algorithm 2)

After initialization, the `FedDP-Lloyds` algorithm can be run for one or more iterations to refine the centers. This implementation is found in `algorithms/federated_lloyds.py`.

1. **Server-Side:** Server sends the current centers ν^{t-1} to clients.
2. **Client-Side:** Each client j assigns its local points P_j to the nearest center in the full-dimensional space. It then computes the local sum m_r^j and local count n_r^j for each cluster r.
3. **Server-Side:** The server collects the (clipped and noised) aggregate sums $\hat{m}_r$ and counts $\hat{n}_r$, and updates the centers for round t as $\nu_r^t = \hat{m}_r / \hat{n}_r$.

As noted in the paper and our presentation, the `FedDP-Init` is often so effective that very few (or even zero) steps of `FedDP-Lloyds` are needed, saving privacy budget.

3.3 The Importance of Initialization in DP K-Means

A core challenge in applying Differential Privacy to iterative algorithms like k-means is a counter-intuitive trade-off identified in the literature.

3.3.1 The Problem with DP-Lloyd's

In a differentially private algorithm, every access to the private data consumes a portion of the predefined privacy budget (ϵ, δ).

- **Privacy Budget Composition:** Running more rounds of an algorithm (like Lloyd's) means the total privacy budget must be split across more steps.
- **Noise vs. Iterations:** A smaller budget per step requires adding more noise to that step to maintain the overall privacy guarantee.
- **Counter-intuitive Result:** This leads to a paradoxical situation where running more iterations (which normally improves accuracy) can actually lead to *lower* final accuracy, as the cumulative noise from many iterations overwhelms the convergence of the algorithm.

Consequently, to achieve high accuracy in a DP setting, it is crucial to start with an exceptionally good initialization. This allows the algorithm to converge in very few (or even zero) refinement steps, thereby minimizing the number of private data accesses and the total amount of noise added.

3.3.2 FedDP-Init as a Global-First Solution

Our implemented algorithm, `FedDP-KMeans`, is designed around this principle. The `FedDP-Init` phase is engineered to produce centers so close to the optimum that few refinement steps are needed.

- **KFed (Local-First):** The `KFed` baseline is a "local-first" approach. Clients run k-means locally and send their biased centers to the server. If client data is non-IID (unbalanced or holding different clusters), these local centers are a poor representation of the global data structure, leading to a suboptimal starting point .
- **FedDP-Init (Global-First):** `FedDP-Init` is a "global-first" approach. It uses its privacy budget to build a high-quality global initialization. Step 1 (Subspace Estimation) estimates the global data structure from all clients, and Step 3 (Center Refinement) is effectively one complete, globally-informed Lloyd's step.

The result is that the `FedDP-Init` centers are already so close to the optimum that few, if any, noisy `FedDP-Lloyds` iterations are required. This allows the algorithm to dedicate its full privacy budget to the initialization phase rather than wasting it on many noisy subsequent steps.

4 Utility Comparison between KFed and FedDP-K-means

We conducted a series of experiments to measure the utility (clustering quality) of our `FedDP-KMeans` algorithm against the `KFed` baseline. Utility is measured by **K-Means Cost** (lower is better) and **Clustering Accuracy** (higher is better). We varied the total privacy budget, ϵ (epsilon), to observe the trade-off.

4.1 Gaussian Dataset: Data-Point Privacy

In the data-point privacy setting, the noise added is relatively small.

- **Finding:** The `FedDP-Init` algorithm (orange line) consistently outperforms the non-private `KFed` baseline (blue line) in both K-Means Cost and Clustering Accuracy.

- **Analysis:** This result is because `FedDP-Init` is simply a superior algorithm. Its 3-step global-first approach produces a near-perfect initialization. The *tiny* amount of data-point level noise added during its 3 steps is not enough to degrade this high-quality starting point. `KFed`'s local-first non-private initialization is less sophisticated and produces a worse starting point, which it never recovers from. The privacy budget is consumed entirely in the Fed-Lloyd's step which deteriorates the accuracy.
- Further, at $\epsilon = 2.5$, FedDP-Init gave an accuracy of **97.62%** that is much higher than what KFed could ever achieve. Thus, the new algorithm eventually surpasses KFed aggressively at higher epsilon.

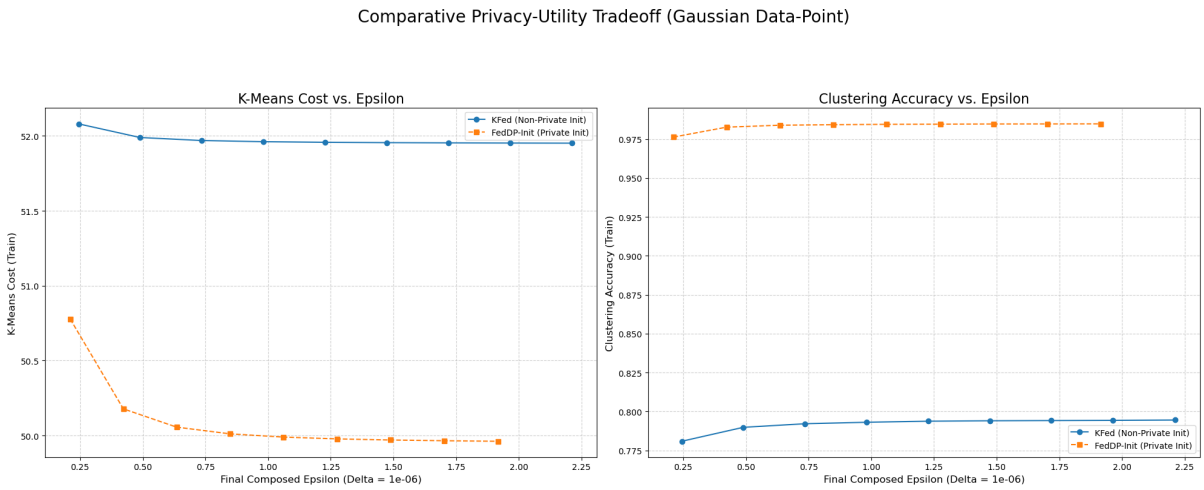

Figure 1: K-Means Cost vs. Epsilon (Left) and Clustering Accuracy vs. Epsilon (Right) for the Gaussian dataset under data-point privacy. The private FedDP-Init (orange) outperforms the non-private KFed (blue).

4.2 Gaussian Dataset: Client-Level Privacy

In the client-level privacy setting, the noise added is massive, as it must hide the contribution of an entire client.

- **Finding:** At very low ϵ (e.g., < 1.5), the `KFed` baseline (blue line) achieves better utility (lower cost, higher accuracy) than `FedDP-Init` (orange line). As ϵ increases, `FedDP-Init`'s utility rapidly improves, crossing over and significantly outperforming `KFed`.
- **Analysis:** This reversal is a direct consequence of the privacy budget composition.
  - At low ϵ, `FedDP-Init` must split its tiny budget across its 3 private steps (PCA, Weighting, Refinement). This **triply large dose** of massive client-level noise completely destroys the initialization, resulting in terrible performance.
  - `KFed`, however, gets its non-private initialization for free (zero privacy cost). It only adds the massive noise once during the refinement (Lloyd's) step. One dose of massive noise is less bad than three, so `KFed` wins at low ϵ.
  - As ϵ increases, the noise added by `FedDP-Init` becomes small enough for its superior algorithm to function, allowing it to quickly surpass `KFed`'s simple initialization.

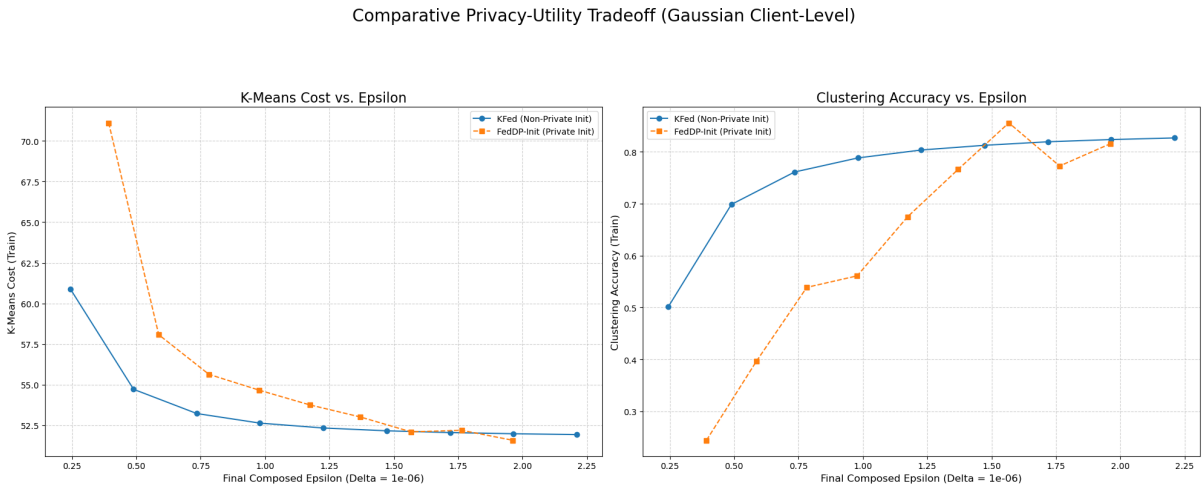

Figure 2: K-Means Cost vs. Epsilon (Left) and Clustering Accuracy vs. Epsilon (Right) for the Gaussian dataset under client-level privacy. KFed (blue) wins at low ϵ, but FedDP-Init (orange) surpasses it as the budget increases.

4.3 Folktables Dataset: Data-Point Privacy

The Folktables dataset, based on US Census data, provides a real-world test.

- **Finding:** Similar to the Gaussian data-client experiment, the `FedDP-Init` (orange line) achieves a lower (better) K-Means cost than the `KFed` baseline (blue line) eventually as all privacy budget decreases (ϵ increases).
- **Analysis:** This data-point level case is similar to the Gaussian client level case because:
  - The upper bound on accuracy is capped at around 67% even when all privacy mechanisms are turned off.
  - The optimal clustering accuracy in this case is poor to begin with.
  - Hence, adding triply-large doses of noise during initialization is slightly poorer (not significantly though) than the conventional Fed-Lloyd's algorithm.
  - As the value of ϵ crosses a certain threshold, we see that the superior initialization `Fed-DP-Init` ultimately makes `Fed-DP-K-Means` superior to the baseline `KFed`.

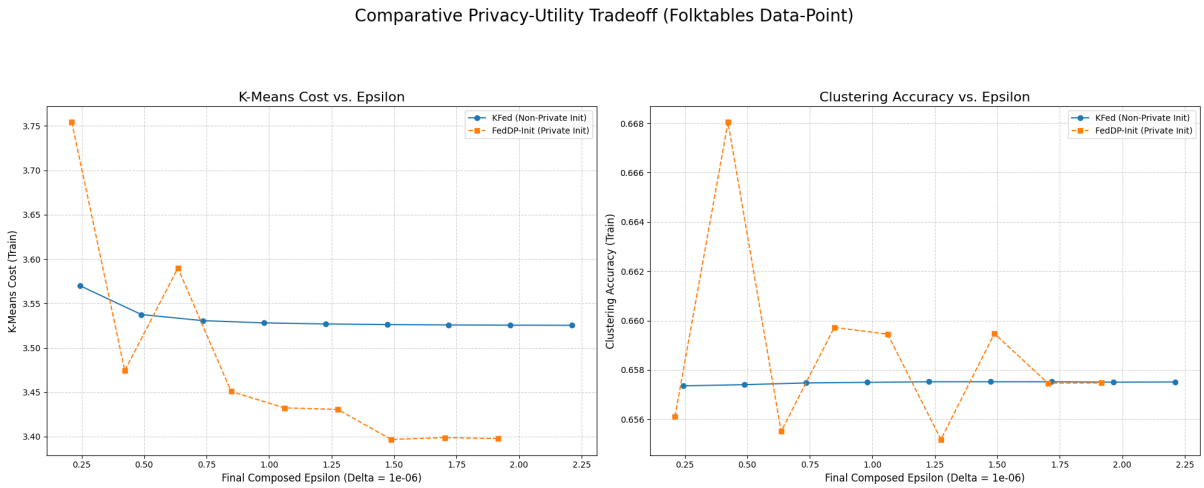

Figure 3: K-Means Cost vs. Epsilon for the Folktables dataset under data-point privacy. FedDP-Init (orange) consistently outperforms KFed (blue).

4.4 Privacy-Utility Tradeoff

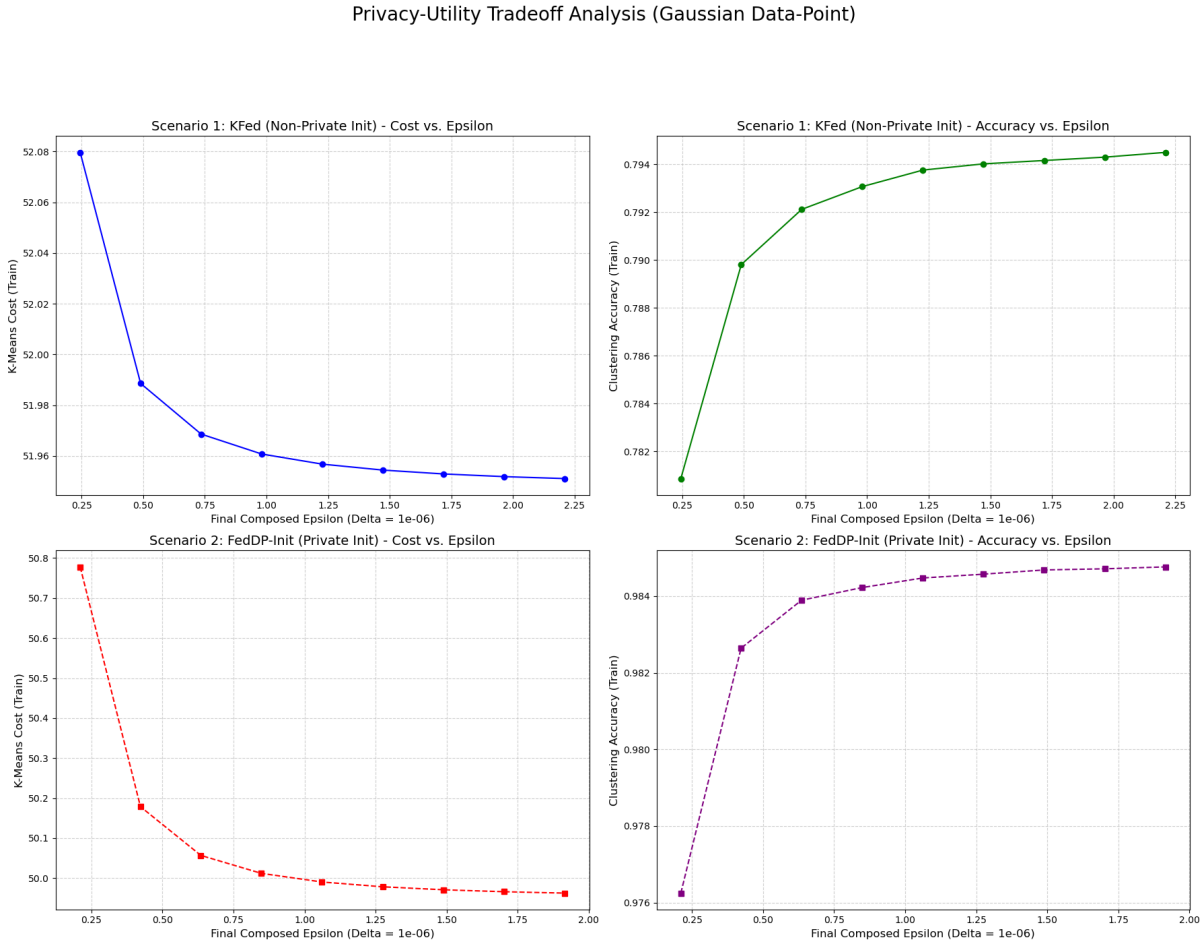

Figure 4: Privacy-Utility Tradeoff

In both data-point and client-level privacy settings, we observe a clear trade-off between privacy and utility. As the privacy budget ϵ decreases (i.e., stronger privacy guarantees), the amount of noise added to protect client data increases. This noise inevitably degrades clustering performance, leading to higher K-Means cost and lower clustering accuracy. Conversely, as ϵ increases (weaker privacy), less noise is required, and the clustering algorithm can achieve higher utility.

Notably, the magnitude of this trade-off depends on the granularity of privacy:

- **Data-Point Privacy:** The added noise per data point is relatively small, so even strong privacy (low ϵ) has a limited effect on utility. The algorithm can maintain near-optimal clustering quality while providing reasonable data-point privacy.
- **Client-Level Privacy:** The added noise per client is much larger, since the contribution of an entire client must be hidden. At low ϵ, this can drastically reduce clustering performance. Only when the privacy budget is sufficiently relaxed does the utility recover to acceptable levels.

Overall, these experiments illustrate the fundamental privacy-utility trade-off inherent in differentially private federated clustering: stronger privacy protections come at the cost of reduced utility, and the effect is more pronounced at coarser (client-level) privacy guarantees.

5 Adversarial Attack Simulation

To validate the privacy guarantees of `FedDP-KMeans`, we simulated two types of adversarial attacks: a reconstruction attack (at both data-level and client-level) and a membership inference attack.

5.1 Threat Model

We assume a worst-case scenario to stress-test the algorithm. The attacker is the **honest-but-curious central server**. This is a **white-box** attack model, meaning the attacker:

- Knows the full source code of the algorithms (`FedDP-Init`, `FedDP-Lloyds`).
- Knows all hyperparameters, including the DP mechanism (Gaussian/Laplace), privacy parameters (ϵ, δ), and clipping bounds.
- Observes all (noisy) aggregated updates from the clients.
- Knows data statistics (distribution, feature size, etc.) but **not** the raw client data points.

5.2 Adversarial Attack Simulation

To validate the privacy guarantees of `FedDP-KMeans`, we simulated two types of powerful, white-box reconstruction attacks.

5.2.1 Data-Level Reconstruction Attack

The goal of this attack is to reconstruct a single, targeted data point x from a target client j. The attack proceeds in the following steps:

1. **Run Baseline Model:** The attacker runs the full federated training without the target point x. This yields the baseline global cluster centers, C_{base}.

2. **Simulate IN Update:** The attacker uses C_{base} to simulate a single client update from client j *with* the target point x included in their local data. This produces the noisy update $\tilde{u}_{in} = (\tilde{sums}_{in}, \tilde{counts}_{in})$.

3. **Simulate OUT Update:** The attacker repeats the simulation, this time *without* the target point x. This produces the noisy update $\tilde{u}_{out} = (\tilde{sums}_{out}, \tilde{counts}_{out})$.

4. **Reconstruct:** The attacker isolates the noisy contribution of the single point x by subtraction:

$$\Delta_{sums} = \tilde{sums}_{in} - \tilde{sums}_{out}$$

$$\Delta_{counts} = \tilde{counts}_{in} - \tilde{counts}_{out}$$

5. The attacker finds the most probable cluster $k^* = \text{argmax}(\Delta_{counts})$. The reconstructed point $\hat{x}$ is the sum difference from that single cluster: $\hat{x} = \Delta_{sums}[k^*]$.

The success of the attack is measured by the **Cosine Similarity** between the true point x and the reconstructed point $\hat{x}$. A value of 1.0 means perfect reconstruction, while 0.0 means no correlation (attack failure).

5.2.2 Client-Level Reconstruction Attack

The goal of this attack is to reconstruct the mean update $\hat{\mu}_j$ of a target client j's **entire** dataset.

1. **Run Baseline Model:** The attacker simulates a global model state trained without the target client j to get baseline global centers.

2. **Simulate Client Update:** The attacker simulates the target client j's full noisy local update ($\tilde{u}_j$) using these baseline centers.

3. **Reconstruct:** The noisy update itself is the reconstructed total contribution. The attacker extracts the noisy sum and count vectors $(\hat{\Sigma}_j, \hat{N}_j) = \tilde{u}_j$. The reconstructed mean is then calculated as $\hat{\mu}_j = \hat{\Sigma}_j / \hat{N}_j$.

Success is again measured by the Cosine Similarity, this time between the true client mean μ_j and the reconstructed mean $\hat{\mu}_j$.

5.2.3 Reconstruction Attack Results

We ran both attack types against non-private ($\epsilon = \infty$) and private ($\epsilon = 1.0$) configurations of our model on both the Gaussian and the Folktables datasets. The results were consistent across all experiments.

- **Non-Private Setting (DP=OFF):** The attack was perfectly successful. In all experiments (Gaussian data-point, Gaussian client-level, and Folktables), the attacker was able to reconstruct the target vector (either a single point x or a client mean μ_j) with 100% fidelity. The resulting histograms show that all attack trials achieved a **Cosine Similarity of 1.0**. This confirms that the non-private federated algorithm leaks complete information.

- **Private Setting (DP=ON):** The attack failed completely. The calibrated Gaussian and Laplace noise added during the federated steps successfully masked the contribution of the single data point (or client). The reconstructed vector $\hat{x}$ or $\hat{\mu}_j$ was uncorrelated with the true vector. The resulting histograms show the Cosine Similarity centered at **0.0 (no correlation)**, with the maximum observed similarity in our experiments not exceeding 0.25.

These results empirically demonstrate that the `FedDP-KMeans` algorithm, when configured with differential privacy, is highly robust and effectively defends against these strong, white-box reconstruction attacks.

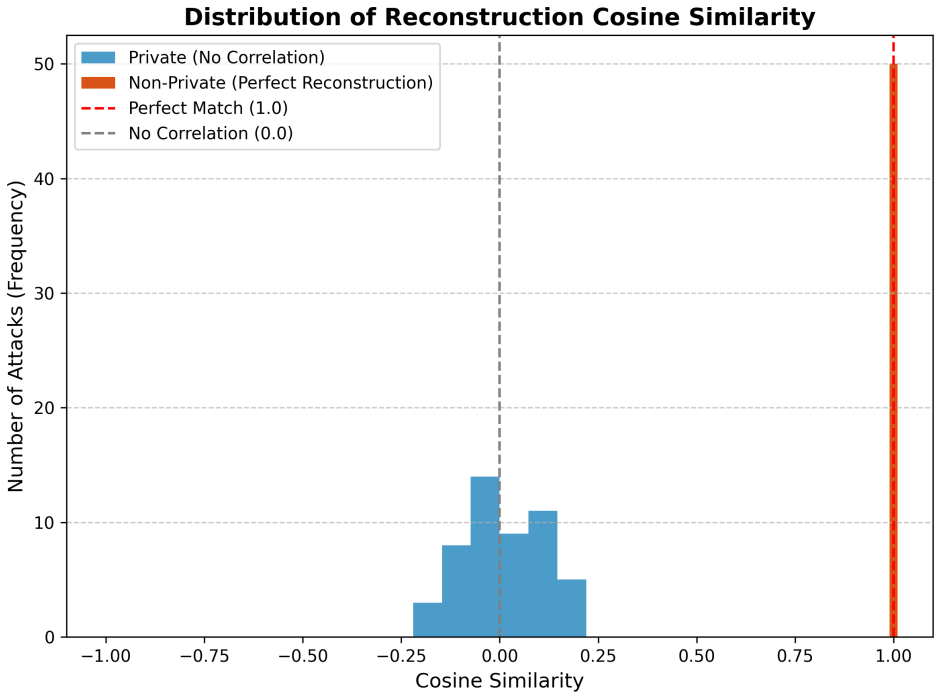

Figure 5: **Data-point Reconstruction Attack on Gaussian Dataset** ($\epsilon = 1.0$). The Non-Private attack (orange) achieves a perfect 1.0 cosine similarity, while the Private attack (blue) fails with ≈ 0.0 similarity.

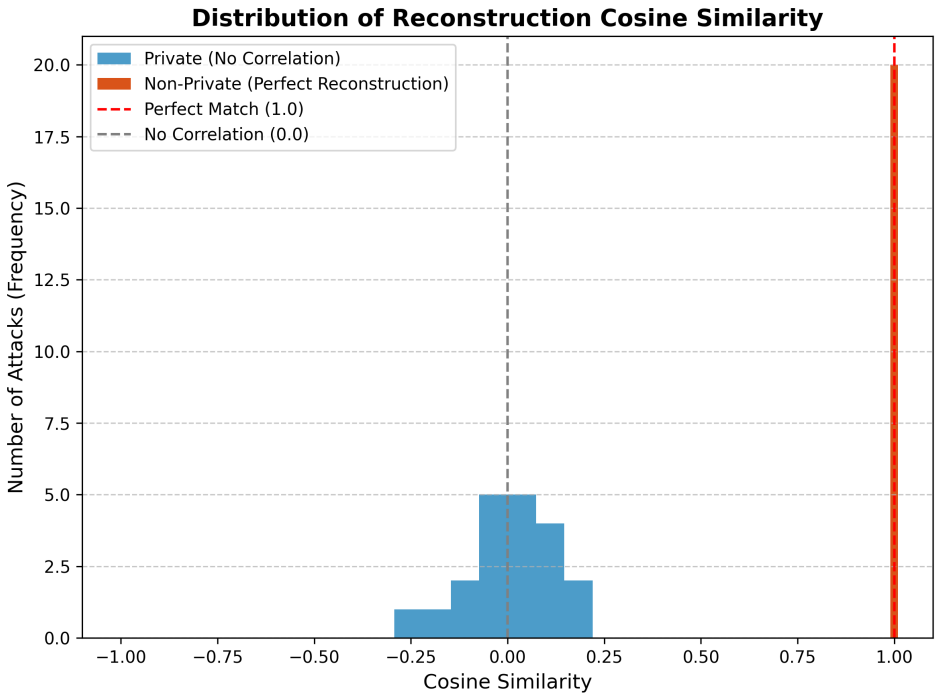

Figure 6: **Client-level Reconstruction Attack on Gaussian Dataset** ($\epsilon = 1.0$). The results are identical: perfect reconstruction (1.0) for non-private, complete failure (0.0) for private.

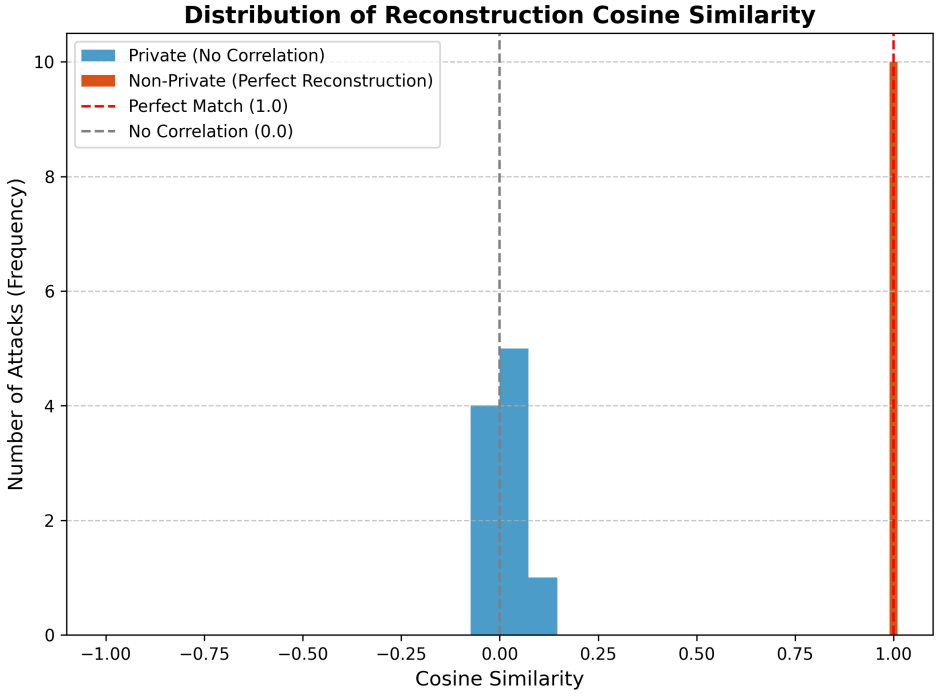

Figure 7: **Data-point Reconstruction Attack on Folktables Dataset** ($\epsilon = 1.0$). The real-world data shows the same pattern: perfect reconstruction (1.0) for non-private, complete failure (0.0) for private.

5.3 Membership Inference Attack (MIA)

In addition to reconstruction, we investigated Membership Inference Attacks (MIAs), as detailed in our initial project proposal and the foundational work by Shokri et al. [4].

5.3.1 Definition

A Membership Inference Attack (MIA) is an attack where an adversary, given a data record and black-box access to a model, aims to determine whether that specific record was part of the model's training dataset. While the classic MIA by Shokri et al. [4] trains a shadow model to distinguish the confidence scores of IN members vs. OUT non-members, our project implemented a different, more direct approach suited to k-means.

5.3.2 Implemented MIA Methodology

Our MIA implementation is a **cost-based attack**. The hypothesis is that a model trained *with* a target data point x will produce cluster centers that are *closer* to x, resulting in a lower k-means cost for that point.

The attack proceeds as follows:

1. **Train IN Model:** The attacker trains a full federated model M_{in} *with* the target data point x included in its client's dataset. The attacker obtains the final global centers C_{in}.
2. **Train OUT Model:** The attacker trains a second, independent federated model M_{out} *without* the target data point x. The attacker obtains the final global centers C_{out}.
3. **Compare Costs:** The attacker computes the k-means cost of the target point x against both sets of centers:
$$cost_{in} = \text{kmeans\_cost}(x, C_{in})$$
$$cost_{out} = \text{kmeans\_cost}(x, C_{out})$$
4. **Infer Membership:** The attacker's decision rule is: If $cost_{in} < cost_{out}$, guess IN. Otherwise, guess OUT.

The success of this attack is measured by its accuracy: the percentage of IN members it correctly identifies as IN. An accuracy of 50% is equivalent to a random guess.

5.3.3 MIA Results

The results from this Membership Inference Attack were **inconclusive**. The attack's accuracy was similar for both the private and non-private models, suggesting that the signal (the difference between $cost_{in}$ and $cost_{out}$) was not a reliable indicator of membership. This could be due to the natural variance in the k-means algorithm itself, or it may indicate that this specific cost-based attack methodology is not effective for this model, even in the non-private case. Note that the private model did perform better than the non-private one, although the accuracy difference was not significant enough to conclude anything.

6 Conclusion and Future Work

6.1 Key Takeaways

This project successfully implemented and evaluated a state-of-the-art differentially private federated clustering algorithm, `FedDP-KMeans`. Our findings are threefold:

1. **Utility:** The `FedDP-Init` algorithm provides a significant utility boost over the non-private `KFed` baseline in low-noise (data-point) settings. This highlights the importance of a sophisticated, global-first initialization.

2. **Comparative Analysis:** A good initialization is paramount. Our experiments show that spending the privacy budget on a high-quality initialization (`FedDP-Init`) is far more effective than spending it on many noisy refinement iterations (`FedDP-Lloyds`). This was confirmed by our finding that the optimal number of Lloyd's steps was often zero.

3. **Security:** The implementation of Differential Privacy was shown to be empirically robust against reconstruction attacks. Our white-box reconstruction attack, which perfectly recovered data in the non-private setting, failed when DP was enabled.

6.2 Limitations and Future Work

While successful, this project also highlighted several limitations and areas for future research:

- **Limitations:** Our implemented Membership Inference Attacks (MIAs) were inconclusive, succeeding similarly on both private and non-private models. This suggests a potential flaw in the attack design itself (i.e., the cost-based signal is too weak) rather than a failure of the DP defense. Furthermore, simulations using client-level privacy on the sparse Folktables dataset failed due to clipping issues.

- **Future Work:**
  - Develop systematic methods to estimate optimal hyperparameters, especially clipping bounds, which are crucial for DP.
  - Investigate and design more effective defenses against Membership Inference Attacks, potentially by adapting the shadow model technique from [4] to clustering.
  - Test the system's robustness against other forms of privacy attacks, such as attribute inference.

Overall, this project provides a comprehensive practical analysis of the privacy and utility landscape in modern federated clustering.

References

[1] S. Garst and M. Reinders. *Federated K-Means Clustering.* Delft Bioinformatics Lab, Delft University of Technology, 2023.

[2] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller. *Inverting Gradients: How Easy Is It to Break Privacy in Federated Learning?* In Advances in Neural Information Processing Systems (NeurIPS), 2020.

[3] J. Scott, C. H. Lampert, and D. Saulpic. *Differentially Private Federated k-Means Clustering with Server-Side Data.* arXiv preprint arXiv:2506.05408, 2025.

[4] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. *Membership Inference Attacks Against Machine Learning Models.* In 2017 IEEE Symposium on Security and Privacy (SP), pages 3-18, 2017.