MACHINE UNLEARNING OF FEDERATED CLUSTERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated clustering is an unsupervised learning problem that arises in a number of practical applications, including personalized recommender and healthcare systems. With the adoption of recent laws ensuring the "right to be forgotten", the problem of machine unlearning for federated clustering methods has become of significant importance. This work proposes the first known unlearning mechanism for federated clustering with privacy criteria that support simple, provable, and efficient data removal at the client and server level. The gist of our approach is to combine special initialization procedures with quantization methods that allow for secure aggregation of estimated local cluster counts at the server unit. As part of our platform, we introduce *secure compressed multiset aggregation (SCMA)*, which is of independent interest for secure sparse model aggregation. In order to simultaneously facilitate low communication complexity and secret sharing protocols, we integrate Reed-Solomon encoding with special evaluation points into the new SCMA pipeline and derive bounds on the time and communication complexity of different components of the scheme. Compared to completely retraining K-means++ locally and globally for each removal request, we obtain an average speed-up of roughly 84x across seven datasets, two of which contain biological and medical information that is subject to frequent unlearning requests.

1 INTRODUCTION

The availability of large volumes of user training data has contributed to the success of modern machine learning models. For example, most state-of-the-art computer vision models are trained on large-scale image datasets including Flickr (Thomee et al., 2016) and ImageNet (Deng et al., 2009). Organizations and repositories that collect and store user data must comply with privacy regulations, such as the recent European Union General Data Protection Regulation (GDPR), the California Consumer Privacy Act (CCPA), and the Canadian Consumer Privacy Protection Act (CPPA), all of which guarantee the right of users to remove their data from the datasets (*Right to be Forgotten*). Data removal requests frequently arise in practice, especially for sensitive datasets pertaining to medical records (numerous machine learning models in computational biology are trained using UK Biobank (Sudlow et al., 2015) which hosts a collection of genetic and medical records of roughly half a million patients (Ginart et al., 2019)). Removing user data from a dataset is insufficient to ensure sufficient privacy, since training data can often be reconstructed from trained models (Fredrikson et al., 2015; Veale et al., 2018). This motivates the study of *machine unlearning* (Cao & Yang, 2015) which aims to efficiently eliminate the influence of certain data points on a model. Naively, one can retrain the model from scratch to ensure complete removal, yet retraining comes at a high computational cost and is thus not practical when accommodating frequent removal requests. To avoid complete retraining, specialized approaches have to be developed for each unlearning application (Ginart et al., 2019; Guo et al., 2020; Bourtoule et al., 2021; Sekhari et al., 2021).

At the same time, federated learning (FL) has emerged as a promising approach to enable distributed training over a large number of users while protecting their privacy (McMahan et al., 2017; Chen et al., 2020; Kairouz et al., 2021; Wang et al., 2021; Bonawitz et al., 2021). The key idea of FL is to keep user data on their devices and train global models by aggregating local models in a communication-efficient and secure manner. Due to model inversion attacks (Zhu et al., 2019; Geiping et al., 2020), secure local model aggregation at the server is a critical consideration in FL, as it guarantees that the server cannot get specific information about client data based on their local models (Bonawitz et al., 2017; Bell et al., 2020; So et al., 2022; Chen et al., 2022). Since data privacy is the main goal in FL, it should be natural for a FL framework to allow for frequent data removal of a subset of

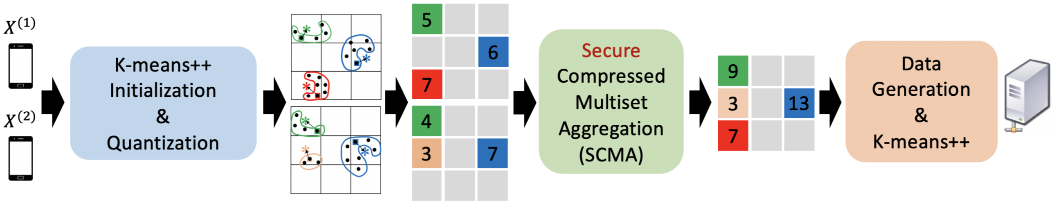

$X^{(1)}$
$X^{(2)}$
K-means++ Initialization & Quantization
5
6
7
4
3
7
Secure Compressed Multiset Aggregation (SCMA)
9
3
13
7
Data Generation & K-means++

Figure 1: Overview of our proposed federated clustering scheme. K-means++ initialization and quantization are performed at each client in parallel. The SCMA procedure ensures that only the server knows the aggregated statistics of clients, without revealing who contributed the points in each individual cluster. The server generates points from the quantization bins with prescribed weights and performs full K-means++ clustering to infer the global model.

client data in a cross-silo setting (e.g., when several patients request their data to be removed in the hospital database), or the entire local dataset for clients in a cross-device setting (e.g., when users request apps not to track their data on their phones). This leads to the largely unstudied problem termed *federated unlearning* (Liu et al., 2021; Wu et al., 2022; Wang et al., 2022). However, existing federated unlearning methods do not guarantee exact or approximate unlearning (Ginart et al., 2019; Guo et al., 2020), they do not come with theoretical performance guarantees after model updates, and often, they are vulnerable to adversarial attacks.

Our contributions. We describe the first unlearning framework for a newly developed federated clustering approach (Fig. 1) which addresses the following problems:

- Unlearning of clusters at the client level via an intuitive application of K-means++ initialization without subsequent Lloyd's iterations. The advantage of using K-means++ is that the models do not need to be updated unless removed points were selected as initial centroids. At the server level, full K-means++ can be performed to improve the overall federated clustering performance.
- Enabling secure aggregation of sparse vectors via a novel approach termed *secure compressed multiset aggregation* (SCMA). The gist of SCMA is to encode quantized local data into information pertaining to the identifiers of quantization bins and their occupancies using specialized Reed-Solomon code evaluations, and followed by secure model aggregations.
- Ensuring privacy-accuracy-efficiency trade-offs through the introduction of a privacy criterion suitable for federated clustering and unlearning with simple communication protocols. In this context, we derive theoretical performance guarantees for the computational and communication complexity of model training and unlearning procedures.
- Compiling a collection of datasets for benchmarking unlearning of federated clusters, including two new datasets which contain anonymized cancer genomics and microbiome information that is subject to frequent unlearning requests. Experimental results reveal that our one-shot algorithm offers an average speed-up of roughly 84x compared to complete retraining across seven datasets.

2 RELATED WORKS

Machine unlearning. For *centralized* machine unlearning problems, two types of unlearning requirements were proposed in previous works: exact unlearning and approximate unlearning. For exact unlearning, the unlearned model is required to perform identically as a completely retrained model. To achieve this, Cao & Yang (2015) introduced distributed learners, Bourtoule et al. (2021) proposed sharding-based methods, Ginart et al. (2019) used quantization to eliminate the effect of removed data in clustering problems, and Chen et al. (2021) applied sharding-based methods to Graph Neural Networks. For approximate unlearning, the "differences" in behavior between the unlearned model and the completely retrained model should be appropriately bounded, similarly to what is done in the context of differential privacy. Following this latter direction, Guo et al. (2020) introduced the inverse Newton update for linear models, Sekhari et al. (2021) studied the generalization performance of approximately unlearned models, while Chien et al. (2022) extended the analysis to linearized Graph Neural Networks. A limited number of recent works also investigated data removal in the FL settings: Liu et al. (2021) proposed to use fewer iterations during retraining for federated unlearning, Wu et al. (2022) introduced Knowledge Distillation into the unlearning procedure to eliminate the effect of data points requested for removal, and Wang et al. (2022)

considered removing all data from one particular class via inspection of the internal influence of each channel in Convolutional Neural Networks. These federated unlearning methods are (mostly) empirical and do not come with theoretical guarantees for model performance after removal and/or for the unlearning efficiency. In contrast, our proposed federated clustering framework not only enables efficient data removal in practice, but also provides theoretical guarantees for the unlearned model performance and for the expected time complexity of the unlearning procedure.

Federated clustering. The idea of federated clustering is to perform clustering using data that resides at different edge devices. It is closely related to clustered FL (Sattler et al., 2020), whose goal is to learn several global models simultaneously, based on the cluster structure of the dataset, as well as personalization according to the cluster assignments of client data in FL (Mansour et al., 2020). One difference between federated clustering and distributed clustering (Guha et al., 2003; Ailon et al., 2009) is the assumption of data heterogeneity across different clients. Recent works (Ghosh et al., 2020; Dennis et al., 2021; Chung et al., 2022) exploit the non-i.i.d nature of client data to improve the performance of some learners. Another difference pertains to data privacy. Most previous methods were centered around the idea of sending exact (Dennis et al., 2021) or quantized client (local) centroids (Ginart et al., 2019) to the server, which may not be considered private as it leaks the data statistics or cluster information of all the clients. To avoid sending exact centroids, Li et al. (2022) proposes sending distances between data points and centroids to the server without revealing the membership of data points to any of the parties involved. Note that there is currently no formal definition of computational or information-theoretic secrecy/privacy for federated clustering problems, making it hard to compare methods addressing different aspects of FL. Our method introduces a simple-to-unlearn clustering process and new privacy mechanism that is intuitively appealing as it involves communicating obfuscated point counts of the clients to the server.

3 PRELIMINARIES

We start with a formal definition of the centralized K-means problem. Given a set of n points $\mathcal{X}$ arranged into a matrix $X \in \mathbb{R}^{n\times d}$, and the number of clusters K, the K-means problem asks for finding a set of points $\mathbf{C} = \{c_1, ..., c_K\}, c_k \in \mathbb{R}^d, \forall k \in [K]$ that minimizes the objective

$$\phi_c(X; \mathbf{C}) = \|X - C\|_F^2, \tag{1}$$

where $||\cdot||_F$ denotes the Frobenius norm of a matrix, $\|\cdot\|$ denotes the ℓ_2 norm of a vector, and $C \in \mathbb{R}^{n\times d}$ records the closest centroid in $\mathbf{C}$ to each data point x_i (i.e., $c_i = \arg\min_{c_j\in\mathbf{C}} \|x_i - c_j\|$). Without loss of generality, we make the assumption that the optimal solution is unique in order to facilitate simpler analysis and discussion, and denote the optimum by $\mathbf{C}^* = \{c_1^*, ..., c_K^*\}$. The set of centroids $\mathbf{C}^*$ induces an optimal partition $\bigcup_{k=1}^K \mathcal{C}_k^*$ over $\mathcal{X}$, where $\forall k \in [K], \mathcal{C}_k^* = \{x_i : ||x_i - c_k^*|| \le ||x_i - c_j^*|| \ \forall i \in [n], j \in [K]\}$. We use $\phi_c^*(X)$ to denote the optimal value of the objective function for the centralized K-means problem. With a slight abuse of notation, we also use $\phi_c^*(\mathcal{C}_k^*)$ to denote the objective value computed over the optimal cluster $\mathcal{C}_k^*$. A detailed description of a commonly used approach for solving the K-means problem, K-means++, is available in Appendix A.

In federated clustering, the dataset $\mathcal{X}$ is no longer available at the centralized server. Instead, data is stored on L edge devices (clients) and the goal of federated clustering is to learn a global set of K centroids $\mathbf{C}_s$ over the union of client data. For simplicity, we assume that there exists no identical data points across clients, and that the overall dataset $\mathcal{X}$ is the union of the datasets $\mathcal{X}^{(l)}$ arranged as $X^{(l)} \in \mathbb{R}^{n^{(l)}\times d}$ on devices $l, \forall l \in [L]$. To make sure the global model can be learned securely, we allow all clients to communicate with each other as well as the server only to perform secure model aggregation (Bonawitz et al., 2017; Bell et al., 2020; So et al., 2022). The server will receive the aggregated cluster statistics of all clients, while each client will only know the cluster assignment of its own data $\mathcal{X}^{(l)}$ and not that of the global server clusters. In this case, the federated K-means problem asks for finding K global centroids $\mathbf{C}_s$ that minimize the objective

$$\phi_f(X; \mathbf{C}_s) = \sum_{l=1}^{L} \|X^{(l)} - C_s^{(l)}\|_F^2, \tag{2}$$

where $C_s^{(l)} \in \mathbb{R}^{n^{(l)}\times d}$ records the centroids of the global clusters that data points $\{x_i^{(l)}\}_{i=1}^{n^{(l)}}$ on client l belong to. A formal definition of global clusters is given in Definition 5.2. Here, ϕ_f stands for the objective of federated clustering, not to be confused with the centralized ϕ_c from Eq. (1).

Exact unlearning. For clustering problems, the *exact unlearning* criterion may be formulated as follows. Let $\mathcal{X}$ be a given dataset and $\mathcal{A}$ a (randomized) clustering algorithm that trains on $\mathcal{X}$ and outputs a set of centroids $\mathbf{C} \in \mathcal{M}$, where $\mathcal{M}$ is the chosen space of models. Let $\mathcal{U}$ be an unlearning algorithm that is applied to $\mathcal{A}(\mathcal{X})$ to remove the effects of one data point $x \in \mathcal{X}$. Then $\mathcal{U}$ is an exact unlearning algorithm if $\forall \mathbf{C} \in \mathcal{M}, x \in \mathcal{X}, \mathbb{P}(\mathcal{U}(\mathcal{A}(\mathcal{X}), \mathcal{X}, x) = \mathbf{C}) = \mathbb{P}(\mathcal{A}(\mathcal{X} \backslash x) = \mathbf{C})$. To avoid confusion, in certain cases, this criterion is referred to as *probabilistic (model) equivalence.*

Privacy-accuracy-efficiency trilemma. How to trade-off data privacy, model performance, communication and computational efficiency is a long-standing problem in distributed learning (Acharya & Sun, 2019; Chen et al., 2020; Gandikota et al., 2021) that also carries over to FL and federated clustering. Solutions that simultaneously address all these challenges in the latter context are still lacking. For example, Dennis et al. (2021) proposed a one-shot algorithm that takes model performance and communication efficiency into consideration by sending the *exact* centroids of each client to the server in a *nonanonymous* fashion. This approach may not be desirable under stringent privacy constraints as the server can gain information about individual client data. On the other hand, privacy considerations were addressed in Li et al. (2022) by performing K-means Lloyd's iterations anonymously via distribution of computations across different clients. Since the method relies on obfuscating pairwise distances for each client, it incurs computational overheads to hide the identity of contributing clients at the server and communication overheads due to interactive computations. None of the above methods is suitable for unlearning applications. To simultaneously enable unlearning and address the trilemma in the unlearning context, our privacy criterion involves transmitting *the number of client data points within local client clusters* in such a manner that the server cannot learn the data statistics of any specific client, but only the overall statistics of the union of client datasets. In this case, computations are limited and the clients on their end can perform efficient unlearning, unlike the case when presented with data point/centroid distances.

Random and adversarial removal. Most unlearning literature focuses on the case when all data points are equally likely to be removed, a setting known as *random removal*. However, adversarial data removal requests may arise when users are malicious in unlearning certain points that are critical for model training (i.e., boundary points in optimal clusters). We refer to such a removal request as *adversarial removal*. In Section 4, we provide theoretical analysis for both types of removal.

4 MACHINE UNLEARNING VIA SPECIALIZED SEEDING

We exhibit next a useful connection between exact unlearning and the K-means++ initialization procedure. Throughout this section we consider the problem in the centralized setting so as to set the stage for the discussion of the federated clustering setting in Section 5. The first step is to introduce the unlearning mechanism based on the initial centroid set, as described in Alg. 1.

4.1 PERFORMANCE ANALYSIS

Since the centroids chosen through K-means++ initialization are true data points, the results returned by Alg. 1 are guaranteed to contain no information about the data points that have been removed. To verify that Alg. 1 is an exact unlearning method, we also need to check that $\mathbf{C}'$ is probabilistically equivalent to the models generated by rerunning the K-means++ initialization process on $\mathcal{X}'$, the set of point remaining after removal. This observation is best explained through an example, while a formal proof is provided in Appendix C.

Suppose that one would like to unlearn the data point x_i from the clustering model represented by an initial centroid set $\mathbf{C}$. For the first chosen centroid c_1 in $\mathbf{C}$, there are only two possibilities: $c_1 = x_i$ and $c_1 \neq x_i$. In the first case, we will have to re-run the initialization over $\mathcal{X}'$, which is retraining the model. In the second case, since we know $c_1 \neq x_i$, the probability of choosing c_1 from $\mathcal{X}$ as the first centroid becomes the conditional probability $\mathbb{P}(\text{choose } c_1 \text{ from } \mathcal{X} \text{ as the first centroid} | c_1 \neq x_i) = \frac{1}{n-1} = \mathbb{P}(\text{choose } c_1 \text{ from } \mathcal{X}' \text{ as the first centroid})$, where n is the number of data points in $\mathcal{X}$. Therefore, the first centroid c_1' returned by Alg. 1 can always be viewed as the first centroid obtained by re-running the initialization over $\mathcal{X}'$. This can be generalized for $c_j \ \forall j \in [K]$ and arbitrary $\mathcal{X}_R$. Since Alg. 1 is an exact unlearning method, we can have the performance guarantees in Lemma 4.1 based on Theorem 1.1 of Vassilvitskii & Arthur (2006).

Lemma 4.1. *For any set of data points $\mathcal{X}$ and removal set $\mathcal{X}_R$, assuming that the remaining dataset is $\mathcal{X}'$ and the centroid set returned by Alg. 1 is $\mathbf{C}'$, we have $\mathbb{E}(\phi_c(X'; \mathbf{C}')) \leq 8(\ln K + 2)\phi_c^*(X')$.*

4.2 COMPLEXITY ANALYSIS

Algorithm 1 Unlearning via K-means++ Init.

```
input: Dataset $\mathcal{X}$, centroid set $\mathbf{C}$ obtained by $K$-means++ initialization on $\mathcal{X}$, removal request set $\mathcal{X}_R = \{x_{r_1}, \ldots, x_{r_R}\}$.
if $c_j \notin \mathcal{X}_R \; \forall c_j \in \mathbf{C}$ then
    $\mathbf{C}' \leftarrow \mathbf{C}$
else
    $i \leftarrow (\arg\min_j c_j \in \mathcal{X}_R) - 1$
    if $i = 0$ then
        $\mathbf{C}' \leftarrow \varnothing, \mathcal{X}' \leftarrow \mathcal{X} \backslash \mathcal{X}_R$.
    else
        $\mathbf{C}' \leftarrow \{c_1, \ldots, c_i\}, \mathcal{X}' \leftarrow \mathcal{X} \backslash \mathcal{X}_R$.
    end if
    for $j = i+1, \ldots, K$ do
        Sample $x$ from $\mathcal{X}'$ with prob $\frac{d^2(x, \mathbf{C}')}{\phi_c(X'; \mathbf{C}')}$.
        $\mathbf{C}' \leftarrow \mathbf{C}' \cup \{x\}$.
    end for
end if
return $\mathbf{C}'$
```

We show next that Alg. 1 supports removing R random points in expected time $O(RK^2 d)$ and R adversarial points in expected time $O(RK^3 \epsilon_1 \epsilon_2 d)$. In contrast, the corresponding time complexity of retraining is $O(nKRd)$. To analyze the removal time complexity of Alg. 1, we first state two assumptions pertaining to optimal cluster sizes and outliers.

Assumption 4.2. *Let $\epsilon_1 = \frac{n}{K s_{\min}}$ be a constant denoting what we refer to as* cluster size imbalance, *where $s_{\min}$ equals the size of the smallest cluster in the optimal clustering; when $\epsilon_1 = 1$, all clusters are of the same size $\frac{n}{K}$.*

Assumption 4.3. *Assume that $\epsilon_2 \geq 1$ is a fixed constant. An outlier x_i in $\mathcal{X}$ satisfies*

$$\|x_i - c_j^*\| \leq \|x_i - c_k^*\|, \forall k \in [K]$$

$$\text{and} \quad \|x_i - c_j^*\| > \sqrt{\frac{\epsilon_2 \phi_c^*(\mathcal{C}_j^*)}{|\mathcal{C}_j^*|}}. \tag{3}$$

The assumption is not overly restrictive, as outliers are commonly defined as points that are well-separated from all "regular" clusters: the distance between an outlier and its closest centroid is larger than a scaled proxy for the empirical standard deviation of the average distance between the cluster points and that centroid (Chien et al., 2018; Dennis et al., 2021). There also exist many different approaches for removing outliers for clustering purposes (Chawla & Gionis, 2013; Gan & Ng, 2017), which effectively reduce the probability of outliers to negligible values.

Assumptions 4.2 and 4.3 can be used to derive an expression for the probability of $x_i \in \mathbf{C}$ when x_i needs to be unlearned, which equals the probability of retraining with $\mathcal{X}'$.

Lemma 4.4. *Assume that the number of data points in $\mathcal{X}$ is n and the probability of the dataset containing at least one outlier is upper bounded by $O(1/n)$. Let $\mathbf{C}$ be the centroid set obtained by running K-means++ on $\mathcal{X}$. For unlearning a data point $x_i \in \mathcal{X}$ we have that for random removal: $\mathbb{P}(x_i \in \mathbf{C}) < O(K/n)$; for adversarial removal: $\mathbb{P}(x_i \in \mathbf{C}) < O\left(K^2 \epsilon_1 \epsilon_2 / n\right)$.*

Based on Alg. 1, we need to reinitialize the centroids only when the point that requests removal lie in the original centroid set. We need $O(K)$ time to check if this is the case, and we need extra $O(nd)$ time to sample every new centroid due to the distance computation. Therefore, we arrive at an estimate of the expected removal time as shown in Theorem 4.5, which is independent of the problem size n.

Theorem 4.5. *Assume that the number of data points in $\mathcal{X}$ is n and the probability of the dataset containing at least one outlier is upper bounded by $O(1/n)$. Algorithm 1 supports removing R points with time $O(RK^2 d)$ in expectation for random removals, and with time $O(RK^3 \epsilon_1 \epsilon_2 d)$ in expectation for adversarial removals. The time complexity of complete retraining equals $O(nKRd)$.*

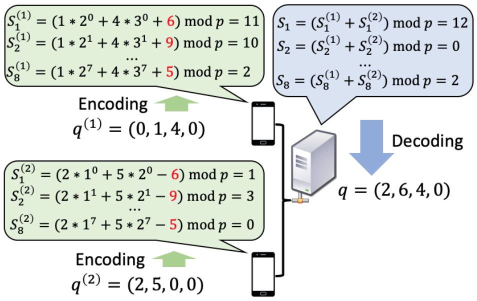

$S_1^{(1)} = (1 * 2^0 + 4 * 3^0 + 6) \bmod p = 11$
$S_2^{(1)} = (1 * 2^1 + 4 * 3^1 + 9) \bmod p = 10$
…
$S_8^{(1)} = (1 * 2^7 + 4 * 3^7 + 5) \bmod p = 2$
$S_1 = (S_1^{(1)} + S_1^{(2)}) \bmod p = 12$
$S_2 = (S_2^{(1)} + S_2^{(2)}) \bmod p = 0$
…
$S_8 = (S_8^{(1)} + S_8^{(2)}) \bmod p = 2$
Encoding
$q^{(1)} = (0, 1, 4, 0)$
Decoding
$q = (2, 6, 4, 0)$
$S_1^{(2)} = (2 * 1^0 + 5 * 2^0 - 6) \bmod p = 1$
$S_2^{(2)} = (2 * 1^1 + 5 * 2^1 - 9) \bmod p = 3$
…
$S_8^{(2)} = (2 * 1^7 + 5 * 2^7 - 5) \bmod p = 0$
Encoding
$q^{(2)} = (2, 5, 0, 0)$

Figure 2: Example of the SCMA procedure for $K = 2, L = 2, B^d = 4, n = 12, p = 13$.

5 UNLEARNING FEDERATED CLUSTERS

We introduce next our federated clustering framework (Alg. 2) based on the discussion from Section 4. The removal procedure is described in Alg. 4. We introduce an oversampling coefficient α for clients, following a common strategy for K-means clustering (Bahmani et al., 2012) to improve the final federated clustering performance. In practice, a constant value α suffices, and for simplicity, we set $\alpha = 1$ in our subsequent analysis. For Alg. 2 to satisfy the privacy criterion stated in Section 3, we need to design an efficient protocol for aggregating information at the server pertaining to the overall dataset without leaking individual client data statistics. This approach is discussed in Section 5.1. A simplified version of Alg. 2 is discussed in Appendix J, for applications where the privacy criterion is not an imperative concern.

Algorithm 2 Secure Federated Clustering

```
input: Dataset $\mathcal{X}$ distributed on $L$ clients ($\mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(L)}$), client oversampling coefficient $\alpha$.
Run $(\alpha K)$-means++ initialization on each client $l$ in parallel, obtain the initial centroid sets $\mathbf{C}^{(l)}$, and record the corresponding cluster sizes $(|\mathcal{C}_1^{(l)}|, \ldots, |\mathcal{C}_{\alpha K}^{(l)}|),\ \forall l \in [L]$.
Perform uniform quantization of $\mathbf{C}^{(l)}$ on each dimension, and flatten the quantization bins into a vector $q^{(l)}, \forall l \in [L]$.
Set $q_j^{(l)} = \left|\mathcal{C}_k^{(l)}\right|$ with $j$ being the index of the quantization bin where $c_k^{(l)}$ lies in for $\forall k \in [\alpha K]$. Set $q_j^{(l)} = 0$ for all other indices.
Securely sum up $q^{(l)}$ at server by Algorithm 3, with the aggregated vector denoted as $q$.
For index $j \in \{t : q_t \neq 0\}$, sample $q_j$ points based on pre-defined distribution and denote their union as new dataset $\mathcal{X}_s$ at server.
Run full $K$-means++ clustering at server with $\mathcal{X}_s$ to obtain the centroid set $\mathbf{C}_s$ at server.
return Each client retains its own centroid set $\mathbf{C}^{(l)}$, server retains $\mathcal{X}_s, q$ and $\mathbf{C}_s$.
```

Note that Step 6 of Alg. 2 involves generating q_j points for the j-th quantization bin. The simplest idea is to choose the center of the quantization bin as the representative point and assign weight q_j to it. Then, in Step 7, we can use the weighted K-means++ algorithm at the server to further reduce the computational complexity at the server side, by reducing the problem size at the server from n, which is the total number of data points, to KL. However, in practice we find that when the server computational power is not the bottleneck in the FL system, generating data points uniformly at random for the quantization bins with non-zero weights can often lead to better clustering performance. Thus, this is the default choice in our subsequent simulations.

5.1 SCMA AT THE SERVER

To utilize off-the-shelf secure aggregation methods developed for FL (Bonawitz et al., 2017; Bell et al., 2020; So et al., 2022), we first need to quantize the centroids of client data and convert the information into vectors. For simplicity, we use uniform quantization with step size γ for each dimension in Step 3 of Alg. 2. A detailed description of the quantization process is available in Appendix I. Once the vector representations $q^{(l)}$ of length B^d $(B = 1/\gamma)$, $l \in [L]$ of client data are generated, we can use standard secure model aggregation methods (Bonawitz et al., 2017; Bell et al., 2020; So et al., 2022) to sum up all $q^{(l)}$ securely and obtain the aggregated results q at the server. However, since the length of each $q^{(l)}$ is B^d, securely aggregating the whole vector would lead to very high communication complexity for each client if γ is small (i.e., if we need finer quantization) or d is large (i.e., if we have to cluster high dimensional data). Moreover, each $q^{(l)}$ is a sparse vector since the number of client centroids is much smaller than the number of quantization bins (i.e., $K \ll B^d$). It is inefficient and unnecessary to sum up $q^{(l)}$ for all dimensions. This motivates us to first compress the vectors and then perform the secure aggregation, and we refer to this process as SCMA.

By observing that there can be at most K nonzero entries in $q^{(l)}, \forall l \in [L]$ and at most KL nonzero entries in q, we invoke the Reed-Solomon code construction (Reed & Solomon, 1960) for designing SCMA. Let $\mathbb{F}_p = \{0, 1, \ldots, p-1\}$ be a finite field of prime order $p \geq \max\{n, B^d\}$. We treat the indices of the quantization bins as distinct elements from the underlying finite field, and use them as evaluation points of the encoder polynomial. In addition, we treat a nonzero entry $q_j^{(l)}$ in vector $q^{(l)}$ as a substitution error at the j-th entry in a codeword. Then, we use our SCMA scheme shown in Alg. 3, where the messages that the clients send to server can be treated as syndromes in Reed-Solomon decoding. The discussion about the differences between SCMA and a related problem termed private set union is available in Appendix M. Note that in our scheme, the server does not know $q^{(l)}, l \in [L]$ beyond the fact that $\sum_{l \in [L]} q^{(l)} = q$, which fits into our privacy criterion. This follows because $z_i^{(l)}$ is uniformly distributed over $\mathbb{F}_p$ and independently chosen for different $l \in [L]$ and $i \in [2KL]$. For details, please refer to Appendix K. One example of SCMA procedure is illustrated in Fig. 2.

5.2 PERFORMANCE ANALYSIS

Algorithm 3 SCMA

```
input: $L$ different vectors $q^{(l)}$ of length $B^d$ to be securely aggregated, a finite field $\mathbb{F}_p$.
Each client $l \in [L], i \in [2KL]$ communicates $\left(S_1^{(l)}, \ldots, S_{2KL}^{(l)}\right)$ to the server, where $S_i^{(l)} = (\sum_{j:q_j^{(l)} \neq 0} q_j^{(l)} \cdot j^{i-1} + z_i^{(l)}) \bmod p$, and $z_i^{(l)}$ is a random key uniformly distributed over $\mathbb{F}_p$ and hidden from the server. The keys $\{z_i^{(l)}\}_{l \in [L], i \in [2KL]}$ are generated offline using standard secure model aggregation so that $\sum_l z_i^{(l)} = 0$.
The server first computes the sum $S_i = \sum_{l \in [L]} S_i^{(l)}, \forall i \in [2KL]$. Given $S_i$, the server computes the coefficients of the polynomial $g(x) = \prod_{j:q_j \neq 0}(1 - j \cdot x)$ using the Berlekamp-Massey algorithm (Berlekamp, 1968; Massey, 1969). Then, the server factorizes $g(x)$ over the field $\mathbb{F}_p$ to determine the roots $j^{-1}$, $q_j \neq 0$, using the polynomial factorizing algorithm (Kedlaya & Umans, 2011). Finally, the server solves a set of $2KL$ linear equations $S_i = \sum_{l \in [L]} S_i^{(l)} = \sum_{j:q_j \neq 0} q_j \cdot j^{i-1}$ for $i \in [2KL]$, by considering $q_j$ as unknowns and $j^{i-1}$ as known coefficients for $q_j \neq 0$.
return $q$ reconstructed at the server.
```

We describe next the performance guarantees of Alg. 2 w.r.t. the objective defined in Eq. (2), using the definition of global clusters from Definition 5.2 originally proposed in Dennis et al. (2021). Note that only the global clustering information can be sent to clients by the server, if requested (i.e., global centroids will not be interactively communicated between clients and server). We first consider the case where no quantization is used (see Alg. 5). The proof is relegated to Appendix G.

Lemma 5.1. *Suppose that the entire dataset across clients is denoted by $\mathcal{X}$, and the set of server centroids returned by Alg. 5 is $\mathbf{C}_s$. Then $\mathbb{E}\left(\phi_f(X; \mathbf{C}_s)\right) < O(\log^2 K) \cdot \phi_c^*(X)$.*

The result of Lemma 5.1 holds for any distribution of data across clients, and Alg. 5 may perform even better than for the case of homogeneous assignments. This is best explained through the following example. In the extreme case of data heterogeneity, each client stores a different cluster w.r.t. the global optimal clustering ($L = K$). Then Alg. 5 can be viewed as seeding each optimal cluster by a data point uniformly at random chosen from that cluster, through 1-means clustering at each client. This leads to a performance guarantee of $\mathbb{E}\left(\phi_f(X; \mathbf{C}_s)\right) = 2\phi_c^*(X)$, where the approximation factor is independent of K. A more detailed discussion how data heterogeneity can benefit our unlearning method is available in Appendix G.

Definition 5.2. *Suppose that the local clusters at client l are denoted by $\mathcal{C}_k^{(l)}, \forall k \in [K], l \in [L]$, and that the clusters at the server are denoted by $\mathcal{C}_k^s, \forall k \in [K]$. The global clustering equals $\mathcal{P}_k = \left\{x_i^{(l)} | x_i^{(l)} \in \mathcal{C}_j^{(l)}, c_j^{(l)} \in \mathcal{C}_k^s, \forall j \in [K], l \in [L]\right\}$, where $c_j^{(l)}$ is the centroid of $\mathcal{C}_j^{(l)}$. Note that $(\mathcal{P}_1, \ldots, \mathcal{P}_K)$ forms a partition of the entire dataset $\mathcal{X}$, and the representative centroid for $\mathcal{P}_k$ is defined as $c_{s,k}$ in $\mathbf{C}_s$ returned by Alg. 2.*

Next, we consider the distortion introduced by quantization. The performance guarantee is given in Theorem 5.3. Note that SCMA does not contribute to the distortion as it always returns the exact sum. Furthermore, following the same reasoning as in Lemma 4.1, one can show that the result of Theorem 5.3 also holds for the remaining dataset $\mathcal{X}'$ after running the unlearning procedure in Alg. 4.

Theorem 5.3. *Suppose that we performed uniform quantization with step size γ in Algorithm 2. Then we have $\mathbb{E}\left(\phi_f(X; \mathbf{C}_s)\right) < O(\log^2 K) \cdot \phi_c^*(X) + O(nd\gamma^2 \log K)$.*

Remark. *To make the second term in the upper bound a constant w.r.t. n, we can choose $\gamma = \Theta(1/\sqrt{n})$, which is a good choice in practice offering good performance across different datasets.*

5.3 COMPLEXITY ANALYSIS

We derived a cohort of in-depth analysis pertaining to the computational complexity of both types of data removals, SCMA at the client and server side, as well as the communication complexity at the clients. Due to space limitations, these results are summarized in Appendix B.

6 EXPERIMENTAL RESULTS

Algorithm 4 Unlearning of Federated Clusters

```
input: Dataset $\mathcal{X}$ distributed on $L$ clients $(\mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(L)})$, $(\mathbf{C}^{(l)}, \mathcal{X}_s, q, \mathbf{C}_s)$ obtained by Algorithm 2 on $\mathcal{X}$, removal request set $\mathcal{X}_R^{(l)}$ for single-client removal or $\mathcal{L}_R$ for multi-client removal.
if single-client removal then
    Run Algorithm 1 on client $l$ and update $q^{(l)}$ if client $l$ has to perform retraining.
else
    $q^{(l)} \leftarrow \mathbf{0}$ on client $l, \forall l \in \mathcal{L}_R$.
end if
Securely sum up $q^{(l)}$ at server by Algorithm 3, with the aggregated vector denoted as $q'$.
if $q' = q$ then
    $\mathbf{C}'_s \leftarrow \mathbf{C}_s$.
else
    Generate $\mathcal{X}'_s$ with $q'$.
    Run full $K$-means++ at the server with $\mathcal{X}'_s$ to obtain $\mathbf{C}'_s$.
end if
return Client centroid sets $\mathbf{C}^{(l)\prime}$, server data $\mathcal{X}'_s, q'$ and centroids $\mathbf{C}'_s$.
```

To empirically characterize the trade-off between the efficiency of data removal and performance of our newly proposed federated clustering method, we compare it with baseline methods on both synthetic and real datasets. Due to space limitations, more in-depth experiments and discussions are delegated to Appendix N.

Datasets and baselines. We use one synthetic dataset generated by a Gaussian Mixture Model (Gaussian) and six real datasets (Celltype, Covtype, FEMNIST, Postures, TMI, TCGA) in our experiments. We preprocess the datasets such that the data distribution is non-i.i.d. across different clients. The symbol K' in Fig. 3 represents the maximum number of (true) clusters among clients, while K represents the number of true clusters in the global dataset. A detailed description of the data statistics and the preprocessing procedure is available in Appendix N. Since there is currently no off-the-shelf algorithm designed for unlearning federated clusters, we adapt DC-Kmeans (DC-KM) from Ginart et al. (2019) to apply to our problem setting, and use complete retraining as the baseline comparison method. To evaluate federated clustering performance on the complete dataset (before data removals), we also include the K-FED algorithm from Dennis et al. (2021) as the baseline method. In all plots, our Alg. 2 is referred to as MUFC. Note that in FL, clients are usually trained in parallel so that the estimated time complexity equals the sum of the longest processing time of a client and the processing time of the server.

Clustering performance. The clustering performance of all methods on the complete dataset is shown in the first row of Tab. 1. The loss ratio is defined as $\phi_f(X; \mathbf{C}_s)/\phi_c^*(X)^1$, which is the metric used to evaluate the quality of the obtained clusters. For the seven datasets, MUFC offered the best performance on TMI and Celltype, datasets for which the numbers of data points in different clusters are highly imbalanced. This can be explained by pointing out an important difference between MUFC and K-FED/DC-KM: the quantized centroids sent by the clients may have non-unit weights, and MUFC is essentially performing weighted K-means++ at the server. In contrast, both K-FED and DC-KM assign equal unit weights to the client's centroids. Note that assigning weights to the client's centroids based on local clusterings not only enables a simple analysis of the scheme but also improves the empirical performance, especially for datasets with highly imbalanced cluster distributions. For all other datasets except Gaussian, MUFC obtained competitive clustering performance compared to K-FED/DC-KM. The main reason why DC-KM outperforms MUFC on Gaussian data is that all clusters are of the same size in this case. Also note that DC-KM runs full K-means++ clustering for each client while MUFC only performs initialization. Although running full K-means++ clustering at the client side can improve the empirical performance on certain datasets, it also greatly increases the computational complexity during training and the retraining probability during unlearning, which is shown in Fig. 3. Nevertheless, we also compare the performance of MUFC with K-FED/DC-KM when running full K-means++ clustering on clients for MUFC in Appendix N.

We also investigated the influence of K' and γ on the clustering performance. Fig. 3(a) shows that MUFC can obtain a lower loss ratio when $K' < K$, indicating that data is non-i.i.d. distributed across clients. Fig. 3(b) shows that the choice of γ does not seem to have a strong influence on the clustering performance of Gaussian datasets, due to the fact that we use uniform sampling in Step 6 of Alg. 2 to generate the server dataset. Meanwhile, Fig. 3(c) shows that γ can have a significant influence on the clustering performance of real-world datasets, which agrees with our analysis in Theorem 5.3.

---

[1] $\phi_c^*(X)$ is approximated by running K-means++ multiple times and selecting the smallest objective value.

Table 1: Clustering performance of different federated clustering algorithms compared to centralized K-means++ clustering.

		TMI	Celltype	Gaussian	TCGA	Postures	FEMNIST	Covtype
Loss ratio	MUFC	1.24 ± 0.10	1.14 ± 0.03	1.25 ± 0.02	1.18 ± 0.05	1.10 ± 0.01	1.20 ± 0.00	1.03 ± 0.02
	K-FED	1.84 ± 0.07	1.72 ± 0.24	1.25 ± 0.01	1.56 ± 0.11	1.13 ± 0.01	1.21 ± 0.00	1.60 ± 0.01
	DC-KM	1.54 ± 0.13	1.46 ± 0.01	1.02 ± 0.00	1.15 ± 0.02	1.03 ± 0.00	1.18 ± 0.00	1.03 ± 0.02
Speed-up of MUFC (if no retraining is performed)		151x	1535x	2074x	483x	613x	53x	267x

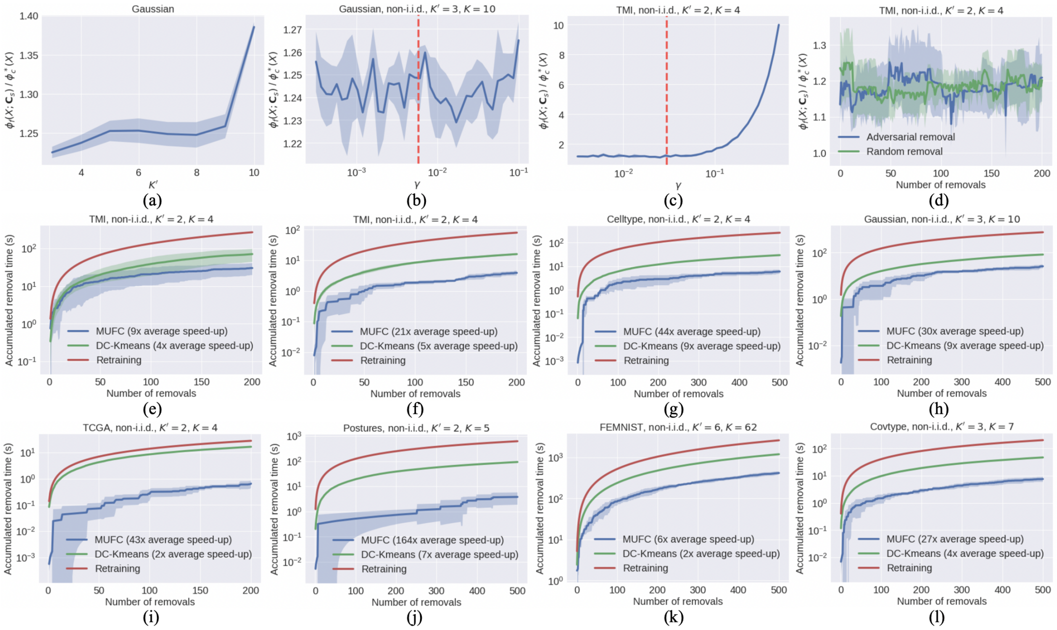


Figure 3: The shaded areas represent the standard deviation of results from different trails. (a) Influence of data heterogeneity on the clustering performance of MUFC: K' represents the maximum number of (global) clusters covered by the data at the clients, while $K' = 10$ indicates that the data points are i.i.d. distributed across clients. (b)(c) Influence of the quantization step size γ on the clustering performance of MUFC. The red vertical line indicates the default choice of $\gamma = 1/\sqrt{n}$, where n is the total number of data points across clients. (d) The change in the loss ratio after each round of unlearning. (e) The accumulated removal time for adversarial removals. (f)-(l) The accumulated removal time for random removals.

The red vertical line in both figures indicates the default choice of $\gamma = 1/\sqrt{n}$, where n stands for the number of total data points across clients.

Unlearning performance. Since K-FED does not support data removal, has high computational complexity, and its empirical clustering performance is worse than DC-KM (see Tab. 1), we only compare the unlearning performance of MUFC with that of DC-KM. For simplicity, we consider removing one data point from a uniformly at random chosen client l at each round of unlearning. The second row of Tab. 1 records the speed-up ratio w.r.t. complete retraining for one round of MUFC unlearning (Alg. 4) when the removed point does not lie in the centroid set selected at client l. Fig. 3(e) shows the accumulated removal time on the TMI dataset for adversarial removals, which are simulated by removing the data points with the highest contribution to the current value of the objective function at each round, while Fig. 3(f)-(l) shows the accumulated removal time on different datasets for random removals. The results show that MUFC maintains high unlearning efficiency compared to all other baseline approaches, and offers an average speed-up ratio of 84x when compared to complete retraining for random removals across seven datasets. We also report the change in the loss ratio of MUFC during unlearning in Fig. 3(d). The loss ratio remains nearly constant after each removal, indicating that our unlearning approach does not significantly degrade clustering performance. Similar conclusions hold for other tested datasets, as shown in Appendix N.

7 ETHICS STATEMENT

The seven datasets used in our simulations are all publicly available. Among these datasets, TCGA and TMI contain potentially sensitive biological data and are downloaded after logging into the database. We adhered to all regulations when handling this anonymized data and will only release the data processing pipeline and data that is unrestricted at TCGA and TMI. Datasets that do not contain sensitive information can be downloaded directly from their open-source repositories.

8 REPRODUCIBILITY STATEMENT

We upload our source code along with test datasets as supplementary materials to reproduce our results shown in Section 6. Detailed instructions are included as well in the source code.

REFERENCES

Jayadev Acharya and Ziteng Sun. Communication complexity in locally private distribution estimation and heavy hitters. In *International Conference on Machine Learning*, pp. 51–60. PMLR, 2019.

Nir Ailon, Ragesh Jaiswal, and Claire Monteleoni. Streaming k-means approximation. *Advances in neural information processing systems*, 22, 2009.

Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. *arXiv preprint arXiv:1203.6402*, 2012.

Constance Beguier, Mathieu Andreux, and Eric W Tramel. Efficient sparse secure aggregation for federated learning. *arXiv preprint arXiv:2007.14861*, 2020.

James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1253–1269, 2020.

Elwyn R. Berlekamp. Algebraic coding theory. In *McGraw-Hill series in systems science*, 1968.

Jock A Blackard and Denis J Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151, 1999.

Kallista Bonawitz, Peter Kairouz, Brendan McMahan, and Daniel Ramage. Federated learning and privacy: Building privacy-preserving systems for machine learning and data science on decentralized data. *Queue*, 19(5):87–114, 2021.

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*, pp. 1175–1191, 2017.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 141–159. IEEE, 2021.

Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A Benchmark for Federated Settings. *arXiv preprint arXiv:1812.01097*, 2018.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pp. 463–480. IEEE, 2015.

Sanjay Chawla and Aristides Gionis. k-means–: A unified approach to clustering and outlier detection. In *Proceedings of the 2013 SIAM international conference on data mining*, pp. 189–197. SIAM, 2013.

Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. Graph unlearning. *arXiv preprint arXiv:2103.14991*, 2021.

Wei-Ning Chen, Peter Kairouz, and Ayfer Ozgur. Breaking the communication-privacy-accuracy trilemma. *Advances in Neural Information Processing Systems*, 33:3312–3324, 2020.

Wei-Ning Chen, Christopher A Choquette Choo, Peter Kairouz, and Ananda Theertha Suresh. The fundamental price of secure aggregation in differentially private federated learning. In *International Conference on Machine Learning*, pp. 3056–3089. PMLR, 2022.

Eli Chien, Chao Pan, and Olgica Milenkovic. Certified graph unlearning. *arXiv preprint arXiv:2206.09140*, 2022.

I Chien, Chao Pan, and Olgica Milenkovic. Query k-means clustering and the double dixie cup problem. *Advances in Neural Information Processing Systems*, 31, 2018.

Jichan Chung, Kangwook Lee, and Kannan Ramchandran. Federated unsupervised clustering with generative models. In *AAAI 2022 International Workshop on Trustable, Verifiable and Auditable Federated Learning*, 2022.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. EMNIST: Extending MNIST to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pp. 2921–2926. IEEE, 2017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Don Kurian Dennis, Tian Li, and Virginia Smith. Heterogeneity for the win: One-shot federated clustering. In *International Conference on Machine Learning*, pp. 2611–2620. PMLR, 2021.

Alfredo Eisinberg and Giuseppe Fedele. On the inversion of the vandermonde matrix. *Applied mathematics and computation*, 174(2):1384–1397, 2006.

Irem Ergun, Hasin Us Sami, and Basak Guler. Sparsified secure aggregation for privacy-preserving federated learning. *arXiv preprint arXiv:2112.12872*, 2021.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1322–1333, 2015.

Keith Frikken. Privacy-preserving set union. In *International Conference on Applied Cryptography and Network Security*, pp. 237–252. Springer, 2007.

Guojun Gan and Michael Kwok-Po Ng. K-means clustering with outlier removal. *Pattern Recognition Letters*, 90:8–14, 2017.

Venkata Gandikota, Daniel Kane, Raj Kumar Maity, and Arya Mazumdar. vqsgd: Vector quantized stochastic gradient descent. In *International Conference on Artificial Intelligence and Statistics*, pp. 2197–2205. PMLR, 2021.

Andrew Gardner, Christian A Duncan, Jinko Kanno, and Rastko Selmic. 3D hand posture recognition from small unlabeled point sets. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 164–169. IEEE, 2014a.

Andrew Gardner, Jinko Kanno, Christian A Duncan, and Rastko Selmic. Measuring distance between unordered sets of different sizes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 137–143, 2014b.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33:19586–19597, 2020.

Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making AI forget you: Data deletion in machine learning. *Advances in neural information processing systems*, 32, 2019.

Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: Theory and practice. *IEEE transactions on knowledge and data engineering*, 15(3): 515–528, 2003.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3832–3842. PMLR, 13–18 Jul 2020.

Xiaoping Han, Renying Wang, Yincong Zhou, Lijiang Fei, Huiyu Sun, Shujing Lai, Assieh Saadatpour, Ziming Zhou, Haide Chen, Fang Ye, et al. Mapping the mouse cell atlas by microwell-seq. *Cell*, 172(5):1091–1107, 2018.

John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

Carolyn Hutter and Jean Claude Zenklusen. The Cancer Genome Atlas: Creating Lasting Value beyond Its Data. *Cell*, 173(2):283–285, 2018.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

Kiran S Kedlaya and Christopher Umans. Fast polynomial factorization and modular composition. *SIAM Journal on Computing*, 40(6):1767–1802, 2011.

Lea Kissner and Dawn Song. Privacy-preserving set operations. In *Annual International Cryptology Conference*, pp. 241–257. Springer, 2005.

Euiwoong Lee, Melanie Schmidt, and John Wright. Improved and simplified inapproximability for k-means. *Information Processing Letters*, 120:40–43, 2017.

Songze Li, Sizai Hou, Baturalp Buyukates, and Salman Avestimehr. Secure federated clustering. *arXiv preprint arXiv:2205.15564*, 2022.

Gaoyang Liu, Xiaoqiang Ma, Yang Yang, Chen Wang, and Jiangchuan Liu. Federaser: Enabling efficient client-level data removal from federated learning models. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, pp. 1–10. IEEE, 2021.

Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2): 129–137, 1982.

Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. *Theoretical Computer Science*, 442:13–21, 2012.

Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

James Massey. Shift-register synthesis and bch decoding. *IEEE transactions on Information Theory*, 15(1):122–127, 1969.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Jane Peterson, Susan Garges, Maria Giovanni, Pamela McInnes, Lu Wang, Jeffery A Schloss, Vivien Bonazzi, Jean E McEwen, Kris A Wetterstrand, Carolyn Deal, et al. The NIH Human Microbiome Project. *Genome research*, 19(12):2317–2323, 2009.

Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 32(8):3710–3722, 2020.

Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34:18075–18086, 2021.

Jae Hong Seo, Jung Hee Cheon, and Jonathan Katz. Constant-round multi-party private set union using reversed laurent series. In *International Workshop on Public Key Cryptography*, pp. 398–412. Springer, 2012.

Jinhyun So, Corey J Nolet, Chien-Sheng Yang, Songze Li, Qian Yu, Ramy E Ali, Basak Guler, and Salman Avestimehr. Lightsecagg: a lightweight and versatile design for secure aggregation in federated learning. *Proceedings of Machine Learning and Systems*, 4:694–720, 2022.

Cathie Sudlow, John Gallacher, Naomi Allen, Valerie Beral, Paul Burton, John Danesh, Paul Downey, Paul Elliott, Jane Green, Martin Landray, et al. Uk biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age. *PLoS medicine*, 12(3): e1001779, 2015.

Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

Sergei Vassilvitskii and David Arthur. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035, 2006.

Michael Veale, Reuben Binns, and Lilian Edwards. Algorithms that remember: model inversion attacks and data protection law. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2133):20180083, 2018.

Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021.

Junxiao Wang, Song Guo, Xin Xie, and Heng Qi. Federated unlearning via class-discriminative pruning. In *Proceedings of the ACM Web Conference 2022*, pp. 622–632, 2022.

Chen Wu, Sencun Zhu, and Prasenjit Mitra. Federated unlearning with knowledge distillation. *arXiv preprint arXiv:2201.09441*, 2022.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

A K-MEANS++ INITIALIZATION

The K-means problem is NP-hard even for $K = 2$, and when the points lie in a two-dimensional Euclidean space (Mahajan et al., 2012). Heuristic algorithms for solving the problem, including Lloyd's (Lloyd, 1982) and Hartigan's method (Hartigan & Wong, 1979), are not guaranteed to obtain the global optimal solution unless further assumptions are made on the point and cluster structures (Lee et al., 2017). Although obtaining the exact optimal solution for the K-means problem is difficult, there are many methods that can obtain quality approximations for the optimal centroids. For example, a randomized initialization algorithm (K-means++) was introduced in Vassilvitskii & Arthur (2006) and the expected objective value after initialization is a $(\log K)$-approximation to the optimal objective ($\mathbb{E}(\phi) \leq (8 \ln K + 16)\phi^*$). K-means++ initialization works as follows: initially, the centroid set $\mathbf{C}$ is assumed to be empty. Then, a point is sampled uniformly at random from X for the first centroid and added to $\mathbf{C}$. For the following $K - 1$ rounds, a point x from X is sampled with probability $d^2(x, \mathbf{C})/\phi_c(X; \mathbf{C})$ for the new centroid and added to $\mathbf{C}$. Here, $d(x, \mathbf{C})$ denotes the minimum ℓ_2 distance between x and the centroids in $\mathbf{C}$ chosen so far. After the initialization step, we arrive at K initial centroids in $\mathbf{C}$ used for running Lloyd's algorithm.

B COMPLEXITY ANALYSIS OF ALGORITHM 3 AND 4

Removal time complexity. We consider two types of removal requests in the federated clustering setting: removing R points from one client l (cross-silo, single-client removal), and removing all data points from R clients $l_1, \ldots, l_R$ (cross-device, multi-client removal). For the case where multiple clients want to unlearn only a part of their data, the approach is similar to that of single-client removal and is based on a simple union bound.

For the first case, we know from Theorem 4.5 that the expected removal time for client l is $O(RK^2d)$ for random requests and $O(RK^3\epsilon_1\epsilon_2 d)$ for adversarial requests. Other clients do not require additional computations, since their centroids will not be affected by the removal requests. Thus the expected removal time for the server is $O\left(RK/n^{(l)}\right) \cdot O(K^2LTd) = O\left(RK^3LTd/n^{(l)}\right)$ for random removal requests and $O\left(RK^4LT\epsilon_1\epsilon_2 d/n^{(l)}\right)$ for adversarial removal requests, where T is the maximum number of iterations of Lloyd's algorithm at the server before convergence, and $n^{(l)}$ is the number of data points on client l with $n^{(l)} \geq K$.

For the second case when we want to unlearn all data from R clients, no client needs to perform additional computations. Since in this case we always re-run the full K-means++ clustering at the server, the expected removal time complexity at the server equals $O((L - R)K^2Td)$.

Communication complexity of SCMA at the client end. Since each $S_i^{(l)}$ can be represented by $\lceil \log p \rceil$ bits, the information $\{S_i^{(l)}\}_{i \in [2KL]}$ sent by each client l can be represented by $2KL\lceil \log p \rceil \leq \max\{2KL \log n, 2KLd \log B\} + 1$ bits. Note that there are at most $\sum_{k \in [KL]} \binom{B^d}{k}\binom{n}{k-1}$ q-ary vectors of length B^d. Hence, the cost for communicating $q^{(l)}$ from the client to server l is at least $\log\left(\sum_{k \in [KL]} \binom{B^d}{k}\binom{n}{k-1}\right) = \max\{O(KL \log n), O(KLd \log B)\}$ bits, which implies that our scheme is order-optimal w.r.t. the communication cost. Note that following standard practice in the area, we do not take into account the complexity of noise generation in secure model aggregation, as it can be done offline and independently of the Reed-Solomon encoding procedure.

Computational complexity of SCMA at the client end. The computation of $S_i^{(l)}$ on client l requires at most $O(K \log i)$ multiplications over $\mathbb{F}_p$, $i \in [2KL]$. The total computational complexity equals $O(K^2L \log(KL))$ multiplication and addition operations over $\mathbb{F}_p$.

Computational complexity of SCMA at the server end. The computational complexity at the server is dominated by the complexity of the Berlekamp-Massey decoding algorithm (Berlekamp, 1968; Massey, 1969), factorizing the polynomial $g(x)$ over $\mathbb{F}_p$ (Kedlaya & Umans, 2011), and solving the linear equations $S_i = \sum_{l \in [L]} S_i^{(l)} = \sum_{j: q_j \neq 0} q_j \cdot j^{i-1}$ with known j, $q_j \neq 0$. The complexity of Berlekamp-Massey decoding over $\mathbb{F}_p$ is $O(K^2L^2)$. The complexity of factorizing a polynomial $g(x)$ over $\mathbb{F}_p$ using the algorithm in Kedlaya & Umans (2011) is $O((KL)^{1.5} \log p + KL \log^2 p)$ operations over $\mathbb{F}_p$. The complexity of solving for $S_i = \sum_{l \in [L]} S_i^{(l)}$ equals that of finding the inverse of a Vandermonde matrix, which takes $O(K^2L^2)$ operations over $\mathbb{F}_p$ (Eisinberg & Fedele, 2006). Hence, the total computational complexity at the server side is $\max\{O(K^2L^2), O((KL)^{1.5}\log p + KL\log^2 p)\}$ operations over $\mathbb{F}_p$.

C PROOF OF LEMMA 4.1

Proof. Assume that the number of data points in $\mathcal{X}$ is n, the size of $\mathcal{X}_R$ is R, and the initial centroid set for $\mathcal{X}$ is $\mathbf{C}$. We use induction to prove that $\mathbf{C}'$ returned by Alg. 1 is probabilistically equivalent to re-running the K-means++ initialization on $\mathcal{X}' = \mathcal{X}\backslash\mathcal{X}_R$.

The base case of induction amounts to investigating the removal process for c_1, the first point selected by K-means++. There are two possible scenarios: $c_1 \in \mathcal{X}_R$ and $c_1 \notin \mathcal{X}_R$. In the first case, we will re-run the initialization process over $\mathcal{X}'$, which is equivalent to retraining the model. In the second case, since we know $c_1 \notin \mathcal{X}_R$, the probability of choosing c_1 from $\mathcal{X}$ as the first centroid equals the conditional probability

$$\begin{aligned}\frac{1}{n-R} &= \mathbb{P}(\text{choose } c_1 \text{ from } \mathcal{X} \text{ as the first centroid}|c_1 \notin \mathcal{X}_R)\\ &= \mathbb{P}(\text{choose } c_1 \text{ from } \mathcal{X}' \text{ as the first centroid}).\end{aligned}$$

Next suppose that $K > 1, i = (\arg\min_j c_j \in \mathcal{X}_R) - 1$. The centroids $\mathbf{C}'_{i-1} = \{c'_1 = c_1, \dots, c'_{i-1} = c_{i-1}\}$ returned by Alg. 1 can be viewed probabilistically equivalent to the model obtained from re-running the initialization process over $\mathcal{X}'$ for the first $i-1$ rounds. Then we have

$$\begin{aligned}\mathbb{P}(\text{choose } c_i \text{ from } \mathcal{X} \text{ as } i\text{-th centroid}|c_i \notin \mathcal{X}_R) &= \frac{\mathbb{P}(\text{choose } c_i \text{ from } \mathcal{X} \text{ as } i\text{-th centroid} \cap c_i \notin \mathcal{X}_R)}{\mathbb{P}(c_i \notin \mathcal{X}_R)}\\ &\overset{(a)}{=} \frac{\mathbb{P}(\text{choose } c_i \text{ from } \mathcal{X} \text{ as } i\text{-th centroid})}{\mathbb{P}(c_i \notin \mathcal{X}_R)}\\ &= \frac{d^2(c_i, \mathbf{C}'_{i-1})/\phi_c(X; \mathbf{C}'_{i-1})}{1 - \sum_{x\in\mathcal{X}_R} d^2(x, \mathbf{C}'_{i-1})/\phi_c(X; \mathbf{C}'_{i-1})}\\ &= \frac{d^2(c_i, \mathbf{C}'_{i-1})/\phi_c(X; \mathbf{C}'_{i-1})}{\phi_c(X'; \mathbf{C}'_{i-1})/\phi_c(X; \mathbf{C}'_{i-1})}\\ &= \frac{d^2(c_i, \mathbf{C}'_{i-1})}{\phi_c(X'; \mathbf{C}'_{i-1})}\\ &= \mathbb{P}(\text{choose } c_i \text{ from } \mathcal{X}' \text{ as } i\text{-th centroid}),\end{aligned}$$

where (a) holds based on the definition of i, indicating that the i-th centroid is not in $\mathcal{X}_R$. Therefore, the centroid $c'_i = c_i$ returned by Alg. 1 can be seen as if obtained from rerunning the initialization process over $\mathcal{X}'$ in the i-th round. Again based on the definition of i, it is clear that for $j > i$, c'_j are the centroids chosen by the K-means++ procedure over $\mathcal{X}'$. This proves our claim that $\mathbf{C}'$ returned by Alg. 1 is probabilistic equivalent to the result obtained by rerunning the K-means++ initialization on $\mathcal{X}'$.

Theorem 1.1 of Vassilvitskii & Arthur (2006) then establishes that

$$\mathbb{E}(\phi_c(X'; \mathbf{C}')) \leq 8(\ln K + 2)\phi_c^*(X'), \tag{4}$$

which completes the proof. ☐

D PROOF OF LEMMA 4.4

Proof. Since outliers can be arbitrarily far from all true cluster points according to Eq. (3), during initialization they may be sampled as centroids with very high probability. For simplicity of analysis, we thus assume that outliers are sampled as centroids with probability 1 if they exist in the dataset. Note that for the adversarial removal scenario, we also assume that outliers will request unlearning with probability 1, meaning that we will always need to re-run the K-means++ initialization when outliers exist in the complete dataset before any removals.

For random removals, where the point requested for unlearning, x_i, is drawn uniformly at random from $\mathcal{X}$, it is clear that $\mathbb{P}(x_i \in \mathbf{C}) = \frac{K}{n}$, since $\mathbf{C}$ contains K distinct data points in $\mathcal{X}$.

For adversarial removals, we need to analyze the probability of choosing x_i as the $(k+1)$-th centroid, given that the first k centroids have been determined and $x_i \notin \mathbf{C}_k = \{c_1, \dots, c_k\}$. For simplicity we first assume that there is no outlier in $\mathcal{X}$. Then we have

$$\mathbb{P}(\text{choose } x_i \text{ from } \mathcal{X} \text{ as the } (k+1)\text{-th centroid}|\mathbf{C}_k) = \frac{d^2(x_i, \mathbf{C}_k)}{\sum_{y \neq x_i} d^2(y, \mathbf{C}_k) + d^2(x_i, \mathbf{C}_k)} \tag{5}$$

For the denominator $\sum_{y \neq x_i} d^2(y, \mathbf{C}_k) + d^2(x_i, \mathbf{C}_k)$, the following three observations are in place

$$\begin{aligned}
&\sum_{y \neq x_i} d^2(y, \mathbf{C}_k) + d^2(x_i, \mathbf{C}_k) \geq \phi_c^*(X) \geq \phi_c^*(\mathcal{C}_i^*), x_i \in \mathcal{C}_i^* \\
&\sum_{y \neq x_i} d^2(y, \mathbf{C}_k) + d^2(x_i, \mathbf{C}_k) \geq \sum_{y \neq x_i} d^2(y, \mathbf{C}^*) \\
&\sum_{y \neq x_i} d^2(y, \mathbf{C}_k) + d^2(x_i, \mathbf{C}_k) \geq \sum_{y \neq x_i} d^2(y, \mathbf{C}_k).
\end{aligned}$$

Therefore,

$$\begin{aligned}
\sum_{y \neq x_i} d^2(y, \mathbf{C}_k) + d^2(x_i, \mathbf{C}_k) &\geq \frac{\phi_c^*(\mathcal{C}_i^*)}{5} + \frac{2}{5}\left(\sum_{y \neq x_i} d^2(y, \mathbf{C}^*) + d^2(y, \mathbf{C}_k)\right) \\
&\overset{(a)}{\geq} \frac{1}{5}\left(\phi_c^*(\mathcal{C}_i^*) + \sum_{y \neq x_i} \|c_y - c_y^*\|^2\right),
\end{aligned} \tag{6}$$

where c_y, c_y^* are the closest centroid in $\mathbf{C}_k$ and $\mathbf{C}^*$ to y, respectively. Here, (a) is a consequence of the fact that $\|a-b\|^2 = \|a-c+c-b\|^2 \leq 2(\|a-c\|^2 + \|b-c\|^2)$. Since x_i is not an outlier for $\mathcal{C}_i^*$ based on our assumption, we have

$$\phi_c^*(\mathcal{C}_i^*) \geq \frac{|\mathcal{C}_i^*|}{\epsilon_2}\|x_i - c_i^*\|^2 \geq \frac{n}{K\epsilon_1\epsilon_2}\|x_i - c_i^*\|^2.$$

Consequently,

$$\begin{aligned}
\phi_c^*(\mathcal{C}_i^*) + \sum_{y \neq x_i} \|c_y - c_y^*\|^2 &\geq \frac{|\mathcal{C}_i^*|}{\epsilon_2}\|x_i - c_i^*\|^2 + \sum_{y \in \mathcal{C}_i^*} \|c_y - c_y^*\|^2 \\
&= \frac{|\mathcal{C}_i^*|}{\epsilon_2}\|x_i - c_i^*\|^2 + \sum_{y \in \mathcal{C}_i^*} \|c_y - c_i^*\|^2.
\end{aligned} \tag{7}$$

For $\forall y \in \mathcal{C}_i^*$, it hold $\|x_i - c_i^*\|^2 + \|c_y - c_i^*\|^2 \geq \frac{1}{2}\|x_i - c_y\|^2 \geq \frac{1}{2}d^2(x_i, \mathbf{C}_k)$. Thus, (7) can be lower bounded by

$$\frac{|\mathcal{C}_i^*|}{\epsilon_2}\|x_i - c_i^*\|^2 + \sum_{y \in \mathcal{C}_i^*} \|c_y - c_i^*\|^2 \geq \frac{|\mathcal{C}_i^*|}{2\epsilon_2}d^2(x_i, \mathbf{C}_k) \geq \frac{n}{2K\epsilon_1\epsilon_2}d^2(x_i, \mathbf{C}_k). \tag{8}$$

Combining (8) and (6) we obtain

$$\sum_{y \neq x_i} d^2(y, \mathbf{C}_k) + d^2(x_i, \mathbf{C}_k) \geq \frac{n}{10K\epsilon_1\epsilon_2}d^2(x_i, \mathbf{C}_k).$$

Using this expression in (5) results in

$$\mathbb{P}(\text{choose } x_i \text{ from } \mathcal{X} \text{ as the } (k+1)\text{-th centroid}|\mathbf{C}_k) \leq \frac{10K\epsilon_1\epsilon_2}{n}, \tag{9}$$

which holds for $\forall k \in [K]$. Thus, the probability $\mathbb{P}(x_i \in \mathbf{C})$ can be computed as

$$\begin{aligned}\mathbb{P}(x_i \in \mathbf{C}) &= \sum_{k=0}^{K-1} \mathbb{P}(\text{choose } x_i \text{ from } \mathcal{X} \text{ as the } (k+1)\text{-th centroid}|\mathbf{C}_k)\mathbb{P}(\mathbf{C}_k) \\ &\leq \sum_{k=0}^{K-1} \mathbb{P}(\text{choose } x_i \text{ from } \mathcal{X} \text{ as the } (k+1)\text{-th centroid}|\mathbf{C}_k) \\ &\leq \frac{1}{n} + \frac{10K(K-1)\epsilon_1\epsilon_2}{n} < O\left(\frac{K^2\epsilon_1\epsilon_2}{n}\right). \end{aligned} \tag{10}$$

Here, we assumed that $\mathbf{C}_0 = \varnothing$.

For the case where outliers are present in the dataset, we have

$$\begin{aligned}\mathbb{P}(x_i \in \mathbf{C}) &= \mathbb{P}(x_i \in \mathbf{C}|x_i \text{ is outlier})\mathbb{P}(x_i \text{ is outlier}) + \mathbb{P}(x_i \in \mathbf{C}|x_i \text{ is not outlier})\mathbb{P}(x_i \text{ is not outlier}) \\ &\leq 1 \cdot O\left(\frac{1}{n}\right) + O\left(\frac{K^2\epsilon_1\epsilon_2}{n}\right) \cdot 1 < O\left(\frac{K^2\epsilon_1\epsilon_2}{n}\right),\end{aligned}$$

which completes the proof for the adversarial removal scenario. □

E PROOF OF THEOREM 4.5

Proof. Assume that $\mathcal{X}_R$ denotes the removal request set over $\mathcal{X}$. From the union bound we have

$$\begin{aligned}&\text{Random removals: } \mathbb{P}(\exists j : c_j \in \mathcal{X}_R) < O\left(\frac{RK}{n}\right), \\ &\text{Adversarial removals: } \mathbb{P}(\exists j : c_j \in \mathcal{X}_R) < O\left(\frac{RK^2\epsilon_1\epsilon_2}{n}\right).\end{aligned}$$

Therefore, the expected removal time for random removals can be upper bounded by

$$\begin{aligned}\mathbb{E}(\text{Removal time}) &= \mathbb{E}(\text{Removal time}|\text{new initialization needed})\mathbb{P}(\text{new initialization needed}) + \\ &\quad \mathbb{E}(\text{Removal time}|\text{new initialization not needed})\mathbb{P}(\text{new initialization not needed}) \\ &\leq O(nKd + RK) \cdot O\left(\frac{RK}{n}\right) + O(RK) \cdot 1 \\ &< O(RK^2 d).\end{aligned}$$

Following a similar argument, we can also show that the expected removal time for adversarial removals can be upper bounded by $O(RK^3\epsilon_1\epsilon_2 d)$. This completes the proof. □

F COMPARISON BETWEEN ALGORITHM 1 AND QUANTIZED K-MEANS

In Ginart et al. (2019), quantized K-means were proposed to solve a similar problem of machine unlearning in the centralized setting. However, that approach substantially differs from Alg. 1. First, the intuition behind quantized K-means is that the centroids are computed by taking an average, and the effect of a small number of points is negligible when there are enough terms left in the clusters after removal. Therefore, if we quantize all centroids after each Lloyd's iteration, the quantized centroids will not change with high probability when we remove a small number of points from the dataset. Meanwhile, the intuition behind Alg. 1 is as described in Lemma 4.4. Second, the expected removal time complexity for quantized K-means equals $O\left(R^2K^3T^2d^{2.5}/\epsilon\right)$, which is high since one needs to check if all quantized centroids remain unchanged after removal at each iteration, where T denotes the maximum number of Lloyd's iteration before convergence and ϵ is some intrinsic parameter. In contrast, Alg. 1 only needs $O(RK^3\epsilon_1\epsilon_2 d)$ even for adversarial removals. Also note that the described quantized K-means algorithm does not come with performance guarantees on removal time complexity unless it is randomly initialized.

G PROOF OF LEMMA 5.1

Proof. Let $\mathbf{C}^*$ denote the optimal set of centroids that minimize the objective (1) for the entire dataset $X \in \mathbb{R}^{n\times d}$, let $C^* \in \mathbb{R}^{n\times d}$ be the matrix that records the closest centroid in $\mathbf{C}^*$ to each data point, $\mathbf{C}_s$ the set of centroids returned by Alg. 2, and $C_s \in \mathbb{R}^{n\times d}$ the matrix that records the corresponding centroid in $\mathbf{C}_s$ for each data point based on the global clustering defined in Definition 5.2. Since we perform K-means++ initialization on each client dataset, for client l it holds

$$\begin{aligned}\mathbb{E}\left(\|X^{(l)} - C^{(l)}\|_F^2\right) &\leq (8\ln K + 16)\|X^{(l)} - C_*^{(l)}\|_F^2, \quad \forall l \in [L]\\ &\leq (8\ln K + 16)\|X^{(l)} - C^{*,(l)}\|_F^2 \end{aligned} \tag{11}$$

where $C^{(l)} \in \mathbb{R}^{n^{(l)}\times d}$ records the closest centroid in $\mathbf{C}^{(l)}$ to each data point x_i in $\mathcal{X}^{(l)}$, $C_*^{(l)}$ is the optimal solution that can minimize the local K-means objective for client l, and $C^{*,(l)}$ denotes the row in C^* that corresponds to client l. Summing up (11) over all clients gives

$$\mathbb{E}\left(\sum_{l=1}^{L}\|X^{(l)} - C^{(l)}\|_F^2\right) \leq (8\ln K + 16)\sum_{l=1}^{L}\|X^{(l)} - C^{*,(l)}\|_F^2. \tag{12}$$

At the server side the client centroids are reorganized into a matrix $X_s \in \mathbb{R}^{n\times d}$. The weights of the client centroids are converted to replicates of rows in X_s. Since we perform full K-means++ clustering at the server, it follows that

$$\begin{aligned}\mathbb{E}\left(\|X_s - C_s\|_F^2\right) &= \mathbb{E}\left(\sum_{l=1}^{L}\|C^{(l)} - C_s^{(l)}\|_F^2\right)\\ &\overset{(a)}{\leq} (8\ln K + 16)\sum_{l=1}^{L}\mathbb{E}\left(\|C^{(l)} - C_{s,*}^{(l)}\|_F^2\right)\\ &\leq (8\ln K + 16)\sum_{l=1}^{L}\mathbb{E}\left(\|C^{(l)} - C^{*,(l)}\|_F^2\right), \end{aligned} \tag{13}$$

where $C_{s,*} \in \mathbb{R}^{n\times d}$ is the optimal solution that minimizes the K-means objective at the server. It is worth pointing out that $C_{s,*}$ is different from C^*, as they are optimal solutions for different optimization objectives. Note that we still keep the expectation on RHS for (a). The randomness comes from the fact that $C^{(l)}$ is obtained by K-means++ initialization, which is a probabilistic procedure.

Combining (12) and (13) results in

$$\begin{aligned}\mathbb{E}\left(\phi_f(X;\mathbf{C}_s)\right) &= \mathbb{E}\left(\sum_{l=1}^{L}\|X^{(l)} - C_s^{(l)}\|_F^2\right)\\ &\leq 2\cdot\mathbb{E}\left[\sum_{l=1}^{L}\left(\|X^{(l)} - C^{(l)}\|_F^2 + \|C^{(l)} - C_s^{(l)}\|_F^2\right)\right]\\ &\leq (16\ln K + 32)\sum_{l=1}^{L}\left[\|X^{(l)} - C^{*,(l)}\|_F^2 + \mathbb{E}\left(\|C^{(l)} - C^{*,(l)}\|_F^2\right)\right]. \end{aligned} \tag{14}$$

For $\mathbb{E}\left(\|C^{(l)} - C^{*,(l)}\|_F^2\right)$, we have

$$\begin{aligned}\mathbb{E}\left(\|C^{(l)} - C^{*,(l)}\|_F^2\right) &\leq 2\cdot\mathbb{E}\left(\|C^{(l)} - X^{(l)}\|_F^2 + \|X^{(l)} - C^{*,(l)}\|_F^2\right)\\ &= 2\cdot\|X^{(l)} - C^{*,(l)}\|_F^2 + 2\cdot\mathbb{E}\left(\|C^{(l)} - X^{(l)}\|_F^2\right). \end{aligned} \tag{15}$$

Replacing (15) into (14) shows that $\mathbb{E}\left(\phi_f(X;\mathbf{C}_s)\right) < O(\log^2 K)\cdot\phi_c^*(X)$, which completes the proof.

If we are only concerned with the performance of non-outlier points over the entire dataset, we can upper bound the term $\mathbb{E}\left(\sum_{l=1}^{L}\|C^{(l)}-C^{*,(l)}\|_F^2\right)$ by

$$\mathbb{E}\left(\sum_{l=1}^{L}\|C^{(l)}-C^{*,(l)}\|_F^2\right) \leq \epsilon_2 \phi_c^*(X). \tag{16}$$

Here, we used the fact that rows of $C^{(l)}$ are all real data points sampled by the K-means++ initialization procedure. For each data point x_i, it holds that $\|x_i - c_i^*\|^2|\mathcal{C}_i^*| \leq \epsilon_2 \phi_c^*(\mathcal{C}_i^*)$, where $x_i \in \mathcal{C}_i^*$. In this case, we arrive at $\mathbb{E}\left(\phi_f(X_t;\mathbf{C}_s)\right) < O(\epsilon_2 \log K)\cdot \phi_c^*(X_t)$, where X_t corresponds to all non-outlier points. $\square$

Remark. *In Theorem 4 of Guha et al. (2003) the authors show that for the distributed K-median problem, if we use a $O(b)$-approximation algorithm (i.e., $\phi \leq O(b)\cdot\phi^*$) for the K-median problem with subdatasets on distributed machines, and use a $O(c)$-approximation algorithm for the K-median problem on the centralized machine, the overall distributed algorithm achieves effectively a $O(bc)$-approximation of the optimal solution to the centralized K-median problem. This is consistent with our observation that Alg. 5 can offer in expectation a $O(\log^2 K)$-approximation to the optimal solution of the centralized K-means problem, since K-means++ initialization achieves a $O(\log K)$-approximation on both the client and server side.*

We also point out that in Dennis et al. (2021) the authors assume that the exact number of clusters from the global optimal clustering on client l is known and equal to $K^{(l)}$, and propose the K-FED algorithm which performs well when $K' = \max_{l\in[L]} K^{(l)} \leq \sqrt{K}$. The difference between K' and K represents the data heterogeneity across different clients. With a slight modifications of the proof, we can also obtain $\mathbb{E}\left(\phi_f(X;\mathbf{C}_s)\right) < O(\log K\cdot\log K')\cdot\phi_c^*(X)$, when $K^{(l)}$ is known for each client beforehand, and perform $K^{(l)}$-means++ on client l instead of K-means++ in Alg. 2. For the extreme setting where each client safeguards data of one entire cluster (w.r.t. the global optimal clustering ($L = K, K' = 1$)), the performance guarantee for Alg. 2 becomes $\mathbb{E}\left(\phi_f(X;\mathbf{C}_s)\right) < O(1)\cdot\phi_c^*(X)$, which is the same as seeding each optimal cluster by a data point sampled uniformly at random from that cluster. From Lemma 3.1 of Vassilvitskii & Arthur (2006) we see that we can indeed have $\mathbb{E}\left(\phi_f(X;\mathbf{C}_s)\right) = 2\phi_c^*(X)$, where the approximation factor does not depend on K. This shows that data heterogeneity across different clients can benefit the entire federated clustering framework introduced.

H PROOF OF THEOREM 5.3

Proof. Following the same idea as the one used in the proof of Lemma 5.1, we arrive at

$$\mathbb{E}\left(\phi_f(X;\mathbf{C}_s)\right) \leq 3\cdot\mathbb{E}\left[\sum_{l=1}^{L}\left(\|X^{(l)}-C^{(l)}\|_F^2 + \|C^{(l)}-\widehat{C}^{(l)}\|_F^2 + \|\widehat{C}^{(l)}-C_s^{(l)}\|_F^2\right)\right], \tag{17}$$

where $\widehat{C}^{(l)}$ is the quantized version of $C^{(l)}$. The first term can be upper bounded in the same way as in Lemma 5.1. For the second term, the distortion introduced by quantizing one point is bounded by $\frac{\sqrt{d}\gamma}{2}$, if we choose the center of the quantization bin as the reconstruction point. Therefore,

$$\mathbb{E}\left(\sum_{l=1}^{L}\|C^{(l)}-\widehat{C}^{(l)}\|_F^2\right) \leq n\left(\frac{\sqrt{d}\gamma}{2}\right)^2 = \frac{nd\gamma^2}{4}. \tag{18}$$

The third term can be bounded as

$$\mathbb{E}\left(\sum_{l=1}^{L}\|\widehat{C}^{(l)}-C_s^{(l)}\|_F^2\right) \leq (8\ln K + 16)\sum_{l=1}^{L}\mathbb{E}\left(\|\widehat{C}^{(l)}-C^{*,(l)}\|_F^2\right)$$

$$\mathbb{E}\left(\|\widehat{C}^{(l)}-C^{*,(l)}\|_F^2\right) \leq 3\cdot\mathbb{E}\left(\|\widehat{C}^{(l)}-C^{(l)}\|_F^2 + \|C^{(l)}-X^{(l)}\|_F^2 + \|X^{(l)}-C^{*,(l)}\|_F^2\right). \tag{19}$$

Replacing (18) and (19) into (17) leads to

$$\mathbb{E}\left(\phi_f(X;\mathbf{C}_s)\right) < O(\log^2 K)\cdot\phi_c^*(X) + O(nd\gamma^2\log K),$$

which completes the proof.

Similar as in Lemma 5.1, we can have that for non-outlier points X_t, $\mathbb{E}\left(\phi_f(X_t;\mathbf{C}_s)\right) < O(\epsilon_2 \log K)\cdot \phi_c^*(X_t) + O(nd\gamma^2 \log K)$. $\square$

I QUANTIZATION

For uniform quantization, we set $\hat{y} = \gamma \cdot a(y)$, where $a(y) = \arg\min_{j\in\mathbb{Z}} |y - \gamma j|, y \in \mathbb{R}$[2]. The parameter $\gamma > 0$ determines the number of quantization bins in each dimension. Suppose all client data lie in the unit hypercube centered at the origin, and that if needed, pre-processing is performed to meet this requirement. Then the number of quantization bins in each dimension equals $B = \gamma^{-1}$, while the total number of quantization bins for d dimensions is $B^d = \gamma^{-d}$.

J SIMPLIFIED FEDERATED K-MEANS CLUSTERING

When privacy criteria like the one stated in Section 3 are not enforced, and as done in the framework of Dennis et al. (2021), one can skip Steps 3-6 in Alg. 2 and send the centroid set $\mathbf{C}^{(l)}$ obtained by client l along with the cluster sizes $(|\mathcal{C}_1^{(l)}|,\ldots,|\mathcal{C}_{\alpha K}^{(l)}|)$ directly to the server. Then, one can run the weighted K-means++ algorithm at the server on the aggregated centroid set to obtain $\mathbf{C}_s$. The pseudocode for this simplified case is shown in Alg. 5. It follows a similar idea as the divide-and-conquer schemes of (Guha et al., 2003; Ailon et al., 2009), developed for distributed clustering.

Algorithm 5 Simplified Federated K-means Clustering

```
1: input: Dataset $\mathcal{X}$ distributed on $L$ clients $(\mathcal{X}^{(1)},\ldots,\mathcal{X}^{(L)})$, client oversampling coefficient $\alpha$.
2: Run $(\alpha K)$-means++ initialization on each client $l$ in parallel, obtain the initial centroid sets $\mathbf{C}^{(l)}$, and record the corresponding cluster sizes $\left(|\mathcal{C}_1^{(l)}|,\ldots,|\mathcal{C}_{\alpha K}^{(l)}|\right)$, $\forall l \in [L]$.
3: Send $\left(c_1^{(l)},\ldots,c_{\alpha K}^{(l)}\right)$ along with the corresponding cluster sizes $\left(|\mathcal{C}_1^{(l)}|,\ldots,|\mathcal{C}_{\alpha K}^{(l)}|\right)$ to the server, $\forall l \in [L]$.
4: Concatenate $\left(c_1^{(l)},\ldots,c_{\alpha K}^{(l)}\right)$ as rows of $X_s$ and set $\left(|\mathcal{C}_1^{(l)}|,\ldots,|\mathcal{C}_{\alpha K}^{(l)}|\right)$ as the weights for the corresponding rows, $\forall l \in [L]$.
5: Run full weighted $K$-means++ clustering at server with $X_s$ to obtain the centroid set at server $\mathbf{C}_s$.
6: return Each client retains their own centroid set $\mathbf{C}^{(l)}$ while the server retains $X_s$ and $\mathbf{C}_s$.
```

In Step 5 of Alg. 5, weighted K-means++ would assign weights to data points when computing the sampling probability during the initialization procedure and when computing the average of clusters during the Lloyd's iterations. Since the weights we are considering here are always positive integers, a weighted data point can also be viewed as there exist identical data points in the dataset with multiplicity equals to the weight.

K THE UNIQUENESS OF THE VECTOR q GIVEN $\{S_i\}_{i\in[2KL]}$

To demonstrate that the messages generated by Alg. 3 can be uniquely decoded, we prove that there exists a unique q that produces the aggregated values $\{S_i\}_{i\in[2KL]}$ at the server. The proof is by contradiction. Assume that there exist two different vectors q and q' that result in the same $\{S_i\}_{i\in[2KL]}$. In this case, we have the following set of linear equations $\sum_{j:q_j\neq 0} q_j \cdot j^{i-1} - \sum_{j:q'_j\neq 0} q'_j \cdot j^{i-1} = 0$, $i \in [2KL]$. Given that $\{q_j : q_j \neq 0\}$ and $\{q'_j : q'_j \neq 0\}$ represent at most $2KL$ unknowns and j^{i-1} coefficients, the linear equations can be described using a square Vandermonde matrix for the coefficients, with the columns of the generated by the indices of the nonzero entries in q. This leads to a contradiction since a square Vandermonde matrix with different column generators is

---

[2] We can also add random shifts during quantization as proposed in Ginart et al. (2019) to make the data appear more uniformly distributed within the quantization bins.

invertible, which we show below. Hence, the aggregated values $\{S_i\}$ must be different for different q. Similarly, the sums $\sum_{j:q_j^{(l)}\neq 0} q_j^{(l)} \cdot j^{i-1}$ are distinct for different choices of vectors $q^{(l)}$, $i \in [2KL]$, $l \in [L]$.

If two vectors q and q' result in the same $\{S_i\}_{i\in[2KL]}$, then $\sum_{j:q_j\neq 0} q_j \cdot j^{i-1} - \sum_{j:q'_j\neq 0} q'_j \cdot j^{i-1} = 0$, for all $i \in [2KL]$. Let $\{i_1, \ldots, i_u\} = (\{j : q_j \neq 0\} \cup \{j : q'_j = 0\})$ be the set of integers such that at least one of q_{i_m} and q'_{i_m} is nonzero for $m \in [u]$. Note that $u \leq 2KL$. Rewrite this equation as

$$\begin{bmatrix} 1 & \cdots & 1 \\ i_1 & \cdots & i_u \\ \vdots & \vdots & \vdots \\ i_1^{2KL-1} & \cdots & i_u^{2KL-1} \end{bmatrix} \begin{bmatrix} q_{i_1} - q'_{i_1} \\ \vdots \\ q_{i_u} - q'_{i_u} \end{bmatrix} = \mathbf{0}. \tag{20}$$

Since $u \leq 2KL$, we take the first u equations in (20) and rewrite them as

$$Bv = \mathbf{0},$$

where

$$B = \begin{bmatrix} 1 & \cdots & 1 \\ i_1 & \cdots & i_u \\ \vdots & \vdots & \vdots \\ i_1^{2KL-1} & \cdots & i_u^{2KL-1} \end{bmatrix}$$

is a square Vandermonde matrix and

$$v = \begin{bmatrix} q_{i_1} - q'_{i_1} \\ \vdots \\ q_{i_u} - q'_{i_u} \end{bmatrix}$$

is a nonzero vector since $q \neq q'$. It is known that the determinant of a square Vandermonde matrix B is given by $\prod_{m_1<m_2, m_1,m_2\in[u]}(i_{m_2} - i_{m_1})$, which in our case is nonzero since all the $i_1, \ldots, i_u$ are different. Therefore, B is invertible and does not admit a non-zero solution, which contradicts the equation $Bv = \mathbf{0}$.

L A DETERMINISTIC LOW-COMPLEXITY ALGORITHM FOR SCMA AT THE SERVER

In the SCMA scheme we described in Alg. 2, the goal of the server is to reconstruct the vector q, given values $S_i = \sum_{j:q_j\neq 0} q_j \cdot j^{i-1} \bmod p$ for $i \in [2KL]$. To this end, we first use the Berlekamp-Massey algorithm to compute the polynomial $g(x) = \prod_{j:q_j\neq 0}(1 - j \cdot x)$. Then, we factorize $g(x)$ over the finite field $\mathbb{F}_p$ using the algorithm described in Kedlaya & Umans (2011). The complexity $O((KL)^{1.5}\log p + KL\log^2 p)$ referred to in Section 5.3 corresponds to the average complexity (finding a deterministic algorithm that factorizes a polynomial over finite fields with $poly(\log p)$ worst-case complexity is an open problem). The complexity $\max\{O(K^2L^2), O((KL)^{1.5}\log p + KL\log^2 p)\}$ referred to in Appendix B for the SCMA scheme represents an average complexity.

We show next that the SCMA scheme has small worst-case complexity under a deterministic decoding algorithm at the server as well. To this end, we replace the integer p in Alg. 3 with a large number $p' \geq \max\{KLB^{2dKL}, n\} + 1$ such that p' is larger than the largest possible S_i and there is no overflow when applying the modulo p' operation on S_i. It is known (Bertrand's postulate) that there exists a prime number between any integer $n > 3$ and $2n - 2$, and hence there must be a prime number lower-bounded by $\max\{KLB^{2dKL}, n\} + 1$ and twice the lower bound $2(\max\{KLB^{2dKL}, n\} + 1)$. However, since searching for a prime number of this size can be computationally intractable, we remove the requirement that p' is prime. Correspondingly, $\mathbb{F}_{p'}$ is not necessarily a finite field. Then, instead of sending $S_i^{(l)} = (\sum_{j:q_j^{(l)}\neq 0} q_j^{(l)} \cdot j^{i-1} + z_i^{(l)}) \bmod p$, client l, $l \in [L]$, will send $S_i^{(l)} = (\sum_{j:q_j^{(l)}\neq 0} q_j^{(l)} \cdot j^{i-1} + z_i^{(l)}) \bmod p'$ to the server, $i \in [2KL]$, where random keys $z_i^{(l)}$ are independently and uniformly distributed over $\{0, \ldots, p'-1\}$ and hidden from the server. After obtaining S_i, $i \in [2KL]$, the server can continue performing operations over the field of reals since there is no overflow in computing $S_i \mod p'$. We note that though p' is exponentially large, the computation of $S_i^{(l)}$ and S_i, $l \in [L]$ and $i \in [2KL]$ is still manageable, and achieved by computing and storing $S_i^{(l)}$ and S_i using $O(KL)$ floating point numbers, instead of computing and storing $S_i^{(l)}$ in a single floating point number. Note that j^i can be computed using $O(i)$ floating point numbers with complexity almost linear in i (i.e., $O(i \log^c i)$ for some constant c).

We now present a low complexity secure aggregation algorithm at the server. After reconstructing S_i, we have $S_i = \sum_{j:q_j \neq 0} q_j \cdot j^{i-1}$. The server switches to computations over the real field. First, it uses the Berlekamp-Massey algorithm to find the polynomial $g(x) = \prod_{j:q_j \neq 0}(1 - j \cdot x)$ (the algorithm was originally proposed for decoding of BCH codes over finite fields, but it applies to arbitrary fields). Let m be the degree of $g(x)$. Then $h(x) = x^m g(1/x) = \prod_{j:q_j \neq 0}(x - j)$. The goal is to factorize $h(x)$ over the field of reals, where the roots are known to be integers in $[B^d]$ and the multiplicity of each root is one.

If the degree of $h(x)$ is odd, then $h(0) < 0$ and $h(B^d) > 0$. Then we can use bisection search to find a root of $h(x)$, which requires $O(\log B^d)$ polynomial evaluations of $h(x)$, and thus $O(MK \log B^d)$ multiplication and addition operations of integers of size at most $\log p'$. After finding one root j, we can divide $h(x)$ by $x - j$ and start the next root-finding iteration.

If the degree of $h(x)$ is even, then the degree of $h'(x)$ is odd, and the roots of $h'(x)$ are different and confined to $[B^d]$. We use bisection search to find a root j' of $h'(x)$. If $h(j') < 0$, then we use bisection search on $[0, j'] = \{0, 1, \ldots, j'\}$ to find a root of $h(x)$ and start a new iteration as described above when the degree of $h(x)$ is odd. If $h(j') > 0$, then $h'(j'-1) > 0$ and $h'(0) < 0$. We use bisection search to find another root of $h'(x)$ in $[j'-1]$. Note that for every two roots j'_1 and j'_2 ($j'_1 < j'_2$) of $h'(x)$ satisfying $h(j'_1) > 0$ and $h(j'_2) > 0$ we can always find another root j'_3 of $h'(x)$ in $[j'_1+1, j'_2-1]$. We keep iterating the search for every two such roots j'_1, j'_2 until we find a list of roots $r_1, \ldots, r_{2R+1}$ of $h'(x)$ such that $h(r_i) < 0$ for odd i in $[2R+1]$ and $h(r_i) > 0$ for even $i \in [2R+1]$. Then we can run bisection search on the sets $[0, r_1], [r_1, r_2], \ldots, [r_{2R}, r_{2R+1}], [r_{2R+1}, B^d]$, to find $2R+2$ roots of $h(x)$. Note that during the iteration we need $2R+1$ bisection search iterations to find the roots $r_1, \ldots, r_{2R+1}$ for $h'(x)$ and $2R+2$ bisection search iterations to find $2R+2$ roots for $h(x)$.

The total computations complexity is hence at most $O(MK \log B^d)$ evaluations of polynomials with degree at most $O(MK)$ and at most $O(MK)$ polynomial divisions, which requires at most $O((MK)^2 \log B^d)$ multiplications and additions for integers of size at most $\log p'$. This results in an overall complexity of $O((MK)^3 d^2 \log^c(MK) \log B)$, for some constant $c < 2$.

M COMPARISON AMONG SCMA, PRIVATE SET UNION AND SPARSE SECURE MODEL AGGREGATION

Our SCMA scheme is different from a related problem of private set union (Kissner & Song, 2005; Frikken, 2007; Seo et al., 2012), in which multiple parties communicate with each other to securely compute the union of their sets. In SCMA we aggregate multisets, which include the frequency of each element that is not revealed in the private set union problem. In addition, our scheme includes only one round of communication from clients to the server, while there is no server in the private set union problem and multi-round client to client communication is needed. Note that SCMA may also be adapted to solve problems related with sparse secure model aggregation (Beguier et al., 2020; Ergun et al., 2021), which is another interesting but orthogonal research direction to this work.

N EXPERIMENTAL SETUP AND ADDITIONAL RESULTS

N.1 DATASETS

In what follows, we describe the datasets used in our numerical experiments. Note that we preprocessed all datasets such that the absolute value of each element in the data matrix is smaller than 1. Each dataset has an intrinsic parameter K for the number of optimal clusters, and these are used in the centralized K-means++ algorithm to compute the approximation of the optimal objective value. We use $\phi_c^*(X)$ in subsequent derivation to denote the objective value returned by the K-means++ algorithm. Besides K, we set an additional parameter $K' \sim \sqrt{K}$ for each client data so that the number of true clusters at the client level is not larger than K'. This non-i.i.d. data distribution across clients is discussed in Dennis et al. (2021). For small datasets (e.g., TCGA, TMI), we consider the number of clients L as 10, and set $L = 100$ for all other datasets.

Celltype $[n = 12009, d = 10, K = 4]$ (Han et al., 2018; Gardner et al., 2014b) comprises single cell RNA sequences belonging to a mixture of four cell types: fibroblasts, microglial cells, endothelial cells and mesenchymal stem cells. The data, retrieved from the Mouse Cell Atlas, consists of 12009 data points and each sample has 10 feature dimensions, reduced from an original dimension of $23,433$ using Principal Component Analysis (PCA). The sizes of the four clusters[3] are $6506, 2328, 2201, 974$.

Postures $[n = 74975, d = 15, K = 5]$ (Gardner et al., 2014b;a) comprises images obtained via a motion capture system and a glove for 12 different users performing five hand postures – fist, pointing with one finger, pointing with two fingers, stop (hand flat), and grab (fingers curled). For establishing a rotation and translation invariant local coordinate system, a rigid unlabeled pattern on the back of the glove was utilized. There are a total of 74975 samples in the dataset and the data dimension is 15. The sizes of the given clusters are $19772, 17340, 15141, 12225, 10497$.

Covtype $[n = 15120, d = 52, K = 7]$ (Blackard & Dean, 1999) comprises digital spatial data for seven forest cover types obtained from the US Forest Service (USFS) and the US Geological Survey (USGS). There are 52 cartographic variables including slope, elevation, and aspect. The dataset has 15120 samples. The sizes of the seven clusters are $3742, 3105, 2873, 2307, 1482, 886, 725$.

Gaussian $[n = 30000, d = 10, K = 10]$ comprises ten clusters, each generated from a 10-variate Gaussian distribution centered at uniformly at random chosen locations in the unit hypercube. From each cluster, 3000 samples are taken, for a total of 30000 samples. Each Gaussian cluster is spherical with variance 0.5.

FEMNIST $[n = 36725, d = 784, K = 62]$ (Caldas et al., 2018) is a popular FL benchmark dataset comprising images of digits (0-9) and letters from the English alphabet (both upper and lower cases) from over 3500 users. It dataset is essentially built from the Extended MNIST repository (Cohen et al., 2017) by partitioning it on the basis of the writer of the digit/character. We extract data corresponding to 100 different clients, each of which contributed at least 350 data points. Each image has dimension 784. The size of the largest cluster is 1234, and that of the smallest cluster is 282.

TCGA $[n = 1904, d = 57, K = 4]$ methylation consists of methylation microarray data for 1904 samples from The Cancer Genome Atlas (TCGA) (Hutter & Zenklusen, 2018) corresponding to four different cancer types: Low Grade Glioma (LGG), Lung Adenocarcinoma (LUAD), Lung Squamous Cell Carcinoma (LUSC) and Stomach Adenocarcinoma (STAD). The observed features correspond to a subset of β values, representing the coverage of the methylated sites, at 57 locations on the promoters of 11 different genes (ATM, BRCA1, CASP8, CDH1, IGF2, KRAS, MGMT, MLH1, PTEN, SFRP5 and TP53). This subset of genes was chosen for its relevance in carcinogenesis. The sizes of the four clusters are $735, 503, 390, 276$.

TMI $[n = 1126, d = 984, K = 4]$ contains samples from human gut microbiomes. We retrieved 1126 human gut microbiome samples from the NIH Human Gut Microbiome (Peterson et al., 2009). Each data point is of dimension 983, capturing the frequency (concentration) of identified bacterial species or genera in the sample. The dataset can be roughly divided into four classes based on gender and age. The sizes of the four clusters are $934, 125, 46, 21$.

N.2 BASELINE SETUPS.

We use the publicly available implementation of K-FED and DC-KM as our baseline methods. For DC-KM, we set the height of the computation tree to 2, and observe that the leaves represent the clients. Since K-FED does not originally support data removal, has high computational complexity, and its clustering performance is not comparable with that of DC-KM (see Tab. 1), we thus only compare the unlearning performance of MUFC with DC-KM. During training, the clustering parameter K is

---

[3] The clusters are obtained by running centralized K-means++ clustering multiple times and selecting the one inducing the lowest objective value.

set to be the same in both clients and server for all methods, no matter how the data was distributed across the clients. Experiments on all datasets except FEMNIST were repeated 5 times to obtain the mean and standard deviations, and experiments on FEMNIST were repeated 3 times due to the high complexity of training. Note that we used the same number of repeated experiments as in Ginart et al. (2019).

N.3 ENABLING COMPLETE CLIENT TRAINING FOR MUFC

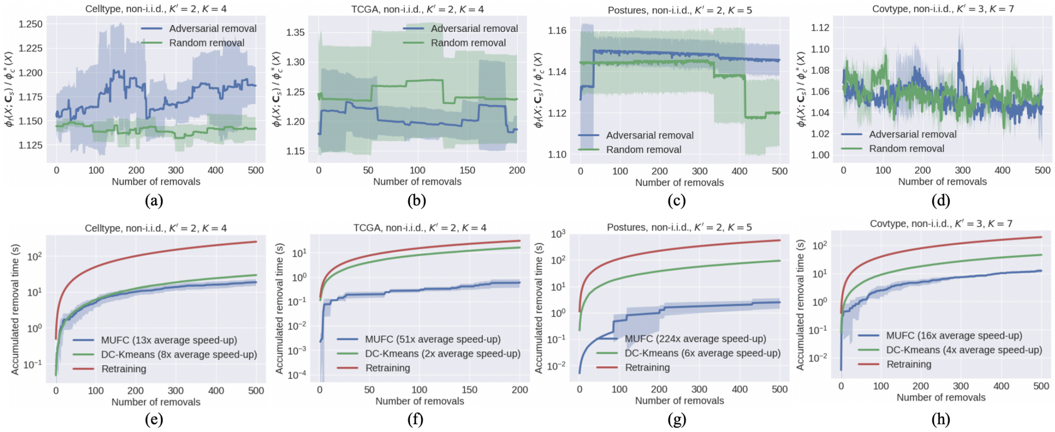
Note that both K-FED and DC-KM allow clients to perform full K-means++ clustering to improve the clustering performance at the server. Thus it is reasonable to enable complete client training for MUFC as well to compare the clustering performance on the full datasets. Although in this case we need to retrain affected clients and the server for MUFC upon each removal request, leading to a similar unlearning complexity as DC-KM, the clustering performance of MUFC is consistently better than that of the other two baseline approaches (see Tab. 2). This is due to the fact that we utilize information about the aggregated weights of client centroids.

Table 2: Clustering performance of different federated clustering algorithms compared to centralized K-means++ clustering.

		TMI	Celltype	Gaussian	TCGA	Postures	FEMNIST	Covtype
Loss ratio	MUFC	$\mathbf{1.05 \pm 0.01}$	$\mathbf{1.03 \pm 0.00}$	$\mathbf{1.02 \pm 0.00}$	$\mathbf{1.02 \pm 0.01}$	$\mathbf{1.02 \pm 0.00}$	$\mathbf{1.12 \pm 0.00}$	$\mathbf{1.02 \pm 0.00}$
	K-FED	1.84 ± 0.07	1.72 ± 0.24	1.25 ± 0.01	1.56 ± 0.11	1.13 ± 0.01	1.21 ± 0.00	1.60 ± 0.01
	DC-KM	1.54 ± 0.13	1.46 ± 0.01	$\mathbf{1.02 \pm 0.00}$	1.15 ± 0.02	1.03 ± 0.00	1.18 ± 0.00	1.03 ± 0.02

N.4 LOSS RATIO AND UNLEARNING EFFICIENCY

In Fig. 4 we plot results pertaining to the change of loss ratio after each removal request and the accumulated removal time when the removal requests are adversarial. The conclusion is consistent with the results in Section 6.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 4: The shaded areas represent the standard deviation of results from different trails for all subplots. (a)-(d) The change of loss ratio $\phi_f(X;\mathbf{C}_s)/\phi_c^*(X)$ after each round of unlearning procedure. (e)-(h) The accumulated removal time for adversarial removals.