FriendlyCore: Practical Differentially Private Aggregation

Eliad Tsfadia [1 2] **Edith Cohen** [1 2] **Haim Kaplan** [2 1] **Yishay Mansour** [2 1] **Uri Stemmer** [2 1]

Abstract

Differentially private algorithms for common metric aggregation tasks, such as clustering or averaging, often have limited practicality due to their complexity or to the large number of data points that is required for accurate results. We propose a simple and practical tool, FriendlyCore, that takes a set of points $\mathcal{D}$ from an unrestricted (pseudo) metric space as input. When $\mathcal{D}$ has effective diameter r, FriendlyCore returns a "stable" subset $\mathcal{C} \subseteq \mathcal{D}$ that includes all points, except possibly a few outliers, and is *guaranteed* to have diameter r. FriendlyCore can be used to preprocess the input before privately aggregating it, potentially simplifying the aggregation or boosting its accuracy. Surprisingly, FriendlyCore is light-weight with no dependence on the dimension. We empirically demonstrate its advantages in boosting the accuracy of mean estimation and clustering tasks such as k-means and k-GMM, outperforming tailored methods.

1. Introduction

Metric aggregation tasks are at the heart of data analysis. Common tasks include averaging, k-clustering, and learning a mixture of distributions. When the data points are sensitive information, corresponding for example to records or activities of particular users, we would like the aggregation to be private. The most widely accepted solution to individual privacy is differential privacy (DP) (Dwork et al., 2006b) that limits the effect that each data point can have on the outcome of the computation.

Differentially private algorithms, however, tend to be less accurate and practical than their non-private counterparts. This degradation in accuracy can be attributed, to a large extent, to the fact that the requirement of differential privacy is a *worst-case* kind of a requirement. To illustrate this point, consider the task of privately learning mixture of Gaussians. In this task, the learner gets as input a sample $\mathcal{D} \subseteq \mathbb{R}^d$, and, assuming that $\mathcal{D}$ was correctly sampled from some appropriate underlying distribution, then the learner needs to output a good hypothesis. That is, the learner is only required to perform well on *typical* inputs. In contrast, the definition of differential privacy is *worst-case* in the sense that the privacy requirement must hold for *any* two neighboring datasets, no matter how they were constructed, even if they are not sampled from any distribution. This means that in the privacy analysis one has to account for any *potential* input point, including "unlikely points" that have significant impact on the aggregation. The traditional way for coping with this issue is to bound the worst-case effect that a single data point can have on the aggregation (this quantity is often called the *sensitivity* of the aggregation), and then to add noise proportional to this worst-case bound. That is, even if all of the given data points are "friendly" in the sense that each of them has only a very small effect on the aggregation, then still, the traditional way for ensuring DP often requires adding much larger noise in order to account for a neighboring dataset that contains one additional "unfriendly" point whose effect on the aggregation is large.

In this paper we present a general framework for preprocessing the data (before privately aggregating it), with the goal of producing a *guarantee* that the data is "friendly" (or well-behaved). Given that the data is guaranteed to be "friendly", the private aggregation step can then be executed without accounting for "unfriendly" points that might have a large effect on the aggregation. Hence, our guarantee potentially allows for much less noise to be added in the aggregation step, as it is no longer forced to operate in the original "worst-case" setting.

1.1. Our Framework

Let us first make the notion of "friendliness" more precise.

Definition 1.1 (f-friendly and f-complete datasets)**.** *Let $\mathcal{D}$ be a dataset over a domain $\mathcal{X}$, and let $f\colon \mathcal{X}^2 \to \{0,1\}$ be a reflexive predicate. We say that $\mathcal{D}$ is f*-friendly *if for every $x, y \in \mathcal{D}$, there exists $z \in \mathcal{X}$ (not necessarily in $\mathcal{D}$) such that $f(x,z) = f(y,z) = 1$. As a special case, we call $\mathcal{D}$*

[1]Google Research [2]Blavatnik School of Computer Science, Tel Aviv University. Correspondence to: Eliad Tsfadia <eliadtsfadia@gmail.com>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

f-complete, *if* $f(x,y) = 1$ *for all* $x, y \in \mathcal{D}$.[1]

Example 1.2 (Points in a metric space)**.** *Let* $\mathcal{D}$ *be points in a metric space and* $f_r(x,y) := \mathbb{1}_{\{d(x,y)\le r\}}$*. Then if* $\mathcal{D}$ *is* f_r*-friendly, it is* f_{2r}*-complete (by the triangle inequality).*

We define a relaxation of differential privacy, where the privacy requirement must only hold for neighboring datasets which are both friendly. Formally,

Definition 1.3 (f-friendly DP algorithm)**.** *An algorithm* A *is called* f-friendly (ε,δ)-DP*, if for every neighboring databases* $\mathcal{D}, \mathcal{D}'$ *such that* $\mathcal{D}\cup\mathcal{D}'$ *is* f*-friendly, it holds that* $\mathsf{A}(\mathcal{D})$ *and* $\mathsf{A}(\mathcal{D}')$ *are* (ε,δ)*-indistinguishable.*

Note that nothing is guaranteed for neighboring datasets that are not f-friendly. Intuitively, this allows us to focus the privacy requirement only on well-behaved inputs, potentially requiring significantly less noise to be added.

We present a preprocessing tool, called FriendlyCore, that takes as input a dataset $\mathcal{D}$ and a predicate f, and outputs a subset $\mathcal{C} \subseteq \mathcal{D}$. If $\mathcal{D}$ is f-complete, then $\mathcal{C} = \mathcal{D}$ (i.e., no elements are removed from the core). In addition, for any neighboring databases $\mathcal{D}$ and $\mathcal{D}' = \mathcal{D}\cup\{z\}$, we show that FriendlyCore satisfies the following two key properties with respect to the outputs $\mathcal{C} = \mathsf{FriendlyCore}(\mathcal{D})$ and $\mathcal{C}' = \mathsf{FriendlyCore}(\mathcal{D}')$:

1. Friendliness: $\mathcal{C}\cup\mathcal{C}'$ is guaranteed to be f-friendly.
2. Stability: $\mathcal{C}$ is distributed "almost" as $\mathcal{C}' \setminus \{z\}$.

At the high level, FriendlyCore on input $\mathcal{D}$ acts as follows: For every element $x \in \mathcal{D}$, it counts $c = \sum_{y\in\mathcal{D}} f(x,y)$ (i.e., the number of x's "friends"), and puts x inside the core with probability $q(c)$, where q is a low-sensitivity monotonic function with $q(n/2) = 0$, $q(n) = 1$ and smoothness in the range $[n/2, n]$, i.e. $q(c) \approx q(c+1)$. The utility follows since if $\mathcal{D}$ is f-complete then all the counts are n. The friendliness is guaranteed since for every $x, y \in \mathcal{C}\cup\mathcal{C}'$, the set of x's friends and set of y's friends are both larger than $n/2$ and therefore must intersect. The stability follows by the smoothness of q. See Section 4 for more details.

Using this preprocessing tool, we prove the following theorem that converts a friendly DP algorithm into a standard (end-to-end) DP one using FriendlyCore.

Theorem 1.4 (Paradigm for DP, informal)**.** *If* A *is* f*-friendly* (ε,δ)-DP*, then* $\mathsf{A}(\mathsf{FriendlyCore}(\cdot))$ *is* $\approx (2\varepsilon, 2e^{3\varepsilon}\delta)$-DP.

In this work we also present a version of FriendlyCore for the δ-approximate ρ-zero-Concentrated Differential Privacy model of (Bun & Steinke, 2016) (in short, (ρ,δ)-zCDP). This version has similar utility guarantee (i.e., when $\mathcal{D}$ is f-complete, then $\mathcal{C} = \mathcal{D}$). In addition, this version gets additional privacy parameters ρ, δ, and satisfies the following privacy guarantee.

Theorem 1.5 (Paradigm for zCDP, informal)**.** *If* A *is* f*-friendly* (ρ,δ)-zCDP*, then* $\mathsf{A}(\mathsf{FriendlyCore}_{\rho',\delta'}(\cdot))$ *is* $(\rho+\rho', \delta+\delta')$-zCDP.

1.2. Example Applications

1.2.1. Private Averaging

Computing the average (center of mass) of points in $\mathbb{R}^d$ is perhaps the most fundamental metric aggregation task. The traditional way for computing averages with DP is to first bound the diameter Λ of the input space, say using the ball $B(0,\Lambda/2)$ with radius $\Lambda/2$ around the origin, clip all points to be inside this ball, and then add Gaussian noise per-dimension that scales with Λ. Now consider a case where the input dataset $\mathcal{D}$ contains n points from some small set with diameter $r \ll \Lambda$, that is located *somewhere* inside our input domain $B(0,\Lambda/2)$. Suppose even that we know the diameter r of that small set, but we do not know *where* it is located inside $B(0,\Lambda/2)$. Ideally, we would like to average this dataset while adding noise proportional to the effective diameter r instead of to the worst-case bound on the diameter Λ. This is easily achieved using our framework. Indeed, such a dataset is dist_r-complete for the predicate $\mathsf{dist}_r(\boldsymbol{x},\boldsymbol{y}) := \mathbb{1}_{\{\|\boldsymbol{x}-\boldsymbol{y}\|_2\le r\}}$, that is, two points are friends if their distance is at most r. Therefore, using our framework, it suffices to design an dist_r-friendly DP algorithm for averaging. Now, the bottom line is that when designing a dist_r-friendly DP algorithm for this task, we *do not* need to add noise proportionally to Λ, and a noise proportionally to r suffices. The reason is that we only need to account for neighboring datasets that are dist_r-friendly, and the difference between the averages of any two such neighboring datasets (i.e., the sensitivity) is proportional to r. See Figure 1 (Left) for an illustration.

We note that existing tailored methods for this averaging problem, for example (Nissim et al., 2016) and (Karwa & Vadhan, 2018) (applied coordinated wise after a random rotation), also provide sample complexity that is *asymptotically* optimal in that it matches that of dist_r-friendly DP averaging. These methods, however, have large constant factors in the sample complexity. The advantage of FriendlyCore is in its simplicity and dimension-independent sample complexity that allows for small overhead over what is necessary for friendly DP averaging.

In Section 6.1 we report empirical results of the averaging application. We observe that the zCDP version of our FriendlyCore framework provides significant practical bene-

[1] In an f-friendly dataset, every two elements have a common friend whereas in an f-complete dataset, all pairs are friends.

fits, outperforming the practice-oriented CoinPress (Biswas et al., 2020) for high d or Λ. This application is described in Section 5.1.

1.2.2. Private Clustering of Well-Separated Instances

Consider the problem of k-clustering of a set of points that is easily clusterable. For example, when the clusters are well separated or sampled from k well separated Gaussians. If data is not this nice, we should still be private, but we do not need the clusters. A classic approach (Nissim et al., 2007) is to split the data randomly into pieces, run some non-private off-the-shelf clustering algorithm on each piece, obtaining a set of k centers (which we call a k-tuple) from each piece, and privately aggregating the result. If the clusters are well separated, then the centers that we compute for different pieces should be similar.[2] Recently, Cohen et al. (2021) formulated the *private k-tuple clustering problem* as the aggregation step. That is, for an input set of such k-tuples (which are similar to each other), the task is to privately compute a new k-tuple that is similar to them. The k-tuple clustering problem is an easier private clustering task where all clusters are of the same size and utility is desired only when the clusters are separated. The application of FriendlyCore provides a simple solution: A tuple $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ is a "friend" of a tuple $Y = (\boldsymbol{y}_1, ..., \boldsymbol{y}_k)$, if for every $\boldsymbol{x}_i$ there is a unique $\boldsymbol{y}_j$ that is substantially closer to $\boldsymbol{x}_i$ than to any other $\boldsymbol{x}_\ell$, $\ell \neq i$. Formally, given a parameter $\gamma \leq 1$, we define the predicate $\mathsf{match}_\gamma(X, Y)$ to be 1, if there exists a permutation π over $[k]$ such that for every i it holds that $\left\|\boldsymbol{x}_i - \boldsymbol{y}_{\pi(i)}\right\| < \gamma \cdot \min_{j \neq i}\left\|\boldsymbol{x}_i - \boldsymbol{y}_{\pi(j)}\right\|$. Now given a database $\mathcal{D}$ of k-tuples as input, we can compute $\mathcal{C} = \mathsf{FriendlyCore}(\mathcal{D})$ with respect to the predicate match_γ for guaranteeing the friendliness of the core $\mathcal{C}$. In particular, if there are a few tuples that are not similar to the others (i.e., "outliers"), then they will be removed by FriendlyCore (see Figure 1 (Right) for an illustration). It follows that for small enough constant γ (as shown in Appendix D.2, $\gamma = 1/7$ suffices), the tuples are guaranteed to be separated enough for making the clustering problem almost trivial: We can use any tuple $Z = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k)$ in $\mathcal{C}$ to partition the tuple points to k parts (the partition is guaranteed to be the same no matter what tuple Z we choose). We can then take a private average of each part (with an appropriate noise) to get a tuple of DP centers. In this application, the use of FriendlyCore both simplifies the solution and lowers the sample complexity of private k-tuple clustering. This translates to using fewer parts in the clustering application and allowing for private clustering of much smaller datasets. We remark that the private averaging of each part can be done again by applying FriendlyCore on each part (as described in Section 1.2.1). It even turns out that the flexibility of FriendlyCore allows to do all k averaging using a single call to FriendlyCore using a special specification of a predicate for *ordered* tuples (see details in Appendix D.2.3).

[2] Starting from the work of (Ostrovsky et al., 2012), such separation conditions have been the subject of many interesting papers. See, e.g., (Shechner, 2021) for a survey of such separation conditions in the context of differential privacy.

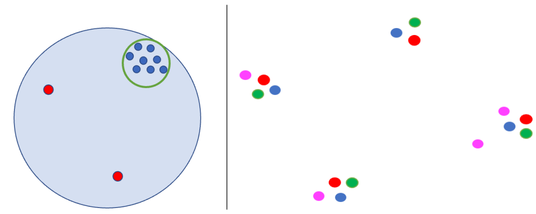

Figure 1: Left: Private averaging example. When we apply FriendlyCore with dist_r, the output is guaranteed to be dist_r-friendly (and dist_{2r} complete). When r is the diameter of the blue points then $\mathcal{C}$ includes all blue points and no red points. Right: k-tuple clustering. The predicate $\mathsf{match}_\gamma(X, Y)$ holds for $\gamma = 1/7$ for any pair of the red, blue, and green 4-tuples but does not hold for pairs that include the pink tuple.

In Section 6.2 we report empirical results of the clustering application, implemented in the zCDP model. We observe that in several different clustering tasks, it outperforms a recent practice-oriented implementation of Chang & Kamath (2021) that is based on local-sensitivity hashing (LSH). The clustering algorithm is described in Section 5.2.

1.2.3. Private Learning a Covariance Matrix

Recently, three independent and concurrent works of Kamath et al. (2021); Ashtiani & Liaw (2021); Kothari et al. (2021) gave a polynomial-time algorithm for privately learning the parameters of unrestricted Gaussians (all the three works were published after the first version of our work that did not include the covariance matrix application). The core of Ashtiani & Liaw (2021)'s construction consists of a framework in the DP model for privately learning average-based aggregation tasks, that has the same flavor of FriendlyCore. Their framework is then applied on private averaging and private learning an unrestricted covariance matrix.

For emphasizing the flexibility of FriendlyCore, in Appendix F we show how to apply FriendlyCore (based on the tools of (Ashtiani & Liaw, 2021)) for learning an unrestricted covariance matrix.

1.3. Related work

Our framework has similar goals to the smooth-sensitivity framework (Nissim et al., 2007) and to the propose-test-release framework (Dwork & Lei, 2009). Like our framework, these two frameworks aim to avoid worst-case restrictions and to perform well on well-behaved inputs. More formally, for a function f mapping datasets to the reals, and a dataset $\mathcal{D}$, define the *local sensitivity* of f on $\mathcal{D}$ as follows: $\mathrm{LS}_f(\mathcal{D}) = \max_{\mathcal{D}' \sim \mathcal{D}} \|f(\mathcal{D}) - f(\mathcal{D}')\|$, where $\mathcal{D}' \sim \mathcal{D}$ denotes that $\mathcal{D}'$ and $\mathcal{D}$ are neighboring datasets. That is, unlike the standard definition of (global) sensitivity which is the maximum difference in the value of f over *every pair* of neighboring datasets, with local sensitivity we consider only neighboring datasets w.r.t. the given input dataset. As a result, there are many cases where the local sensitivity can be significantly lower than the global sensitivity. One such classical example is the median, where on a dataset for which the median is very stable it might be that the local sensitivity is zero even though the global sensitivity can be arbitrarily large. Both the smooth-sensitivity framework and the propose-test-release framework aim to privately release the value of a function while only adding noise proportionally to its local sensitivity rather than its global sensitivity (when possible).

Our framework is very different in that it does not aim for local sensitivity, and is not limited by it. Specifically, in the application of private averaging, the local sensitivity is still very large even when the dataset is friendly. This is because even if all of the input points reside in a tiny ball, to bound the local sensitivity we still need to account for a neighboring dataset in which one point moves "to the end of the world" and hence causes a large change to the average of the points.

1.4. Paper Organization

Sections 2 to 6 in the main body are short versions of Appendices A to E (respectively). In Appendix F we present the covariance matrix application. Appendix G is about the running time of FriendlyCore. Other missing proofs (in particular, utility guarantees) appear in Appendix H.

2. Preliminaries

2.1. Notation

Throughout this work, a database $\mathcal{D}$ is an (ordered) vector over a domain $\mathcal{X}$. Given $\mathcal{D} = (x_1, \ldots, x_n) \in \mathcal{X}^n$, for $\mathcal{I} \subseteq [n]$ let $\mathcal{D}_{\mathcal{I}} := (x_i)_{i \in \mathcal{I}}$, let $\mathcal{D}_{-\mathcal{I}} := \mathcal{D}_{[n] \setminus \mathcal{I}}$, and for $i \in [n]$ let $\mathcal{D}_i := x_i$ and $\mathcal{D}_{-i} := \mathcal{D}_{-\{i\}}$ (i.e., $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$). For $\mathcal{D} = (x_1, \ldots, x_n)$ and $\mathcal{D}' = (x'_1, \ldots, x'_m)$ let $\mathcal{D} \cup \mathcal{D}' = (x_1, \ldots, x_n, x'_1, \ldots, x'_m)$.

For $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$, we let $\|\boldsymbol{x}\| := \sqrt{\sum_{i=1}^d x_i^2}$ (i.e., the ℓ_2 norm of $\boldsymbol{x}$). For $\mathcal{D} \in (\mathbb{R}^d)^*$ we denote $\mathrm{Avg}(\mathcal{D}) := \frac{1}{|\mathcal{D}|} \cdot \sum_{\boldsymbol{x} \in \mathcal{D}} \boldsymbol{x}$. For $r \geq 0$ and $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ we denote $\mathsf{dist}_r(\boldsymbol{x}, \boldsymbol{y}) := \mathbb{1}_{\{\|\boldsymbol{x} - \boldsymbol{y}\| \leq r\}}$ (i.e., 1 if $\|\boldsymbol{x} - \boldsymbol{y}\| \leq r$ and 0 otherwise).

Throughout this paper, we define neighboring databases with respect to the insertion/deletion model, where one database is obtain by adding or removing an element from the other database. Formally,

Definition 2.1 (Neighboring databases). *Let $\mathcal{D}$ and $\mathcal{D}'$ be two databases over a domain $\mathcal{X}$. We say that $\mathcal{D}$ and $\mathcal{D}'$ are* neighboring*, if either there exists $j \in [|\mathcal{D}|]$ such that $\mathcal{D}_{-j} = \mathcal{D}'$, or there exists $j \in [|\mathcal{D}'|]$ such that $\mathcal{D} = \mathcal{D}'_{-j}$.*

2.2. Zero-Concentrated Differential Privacy (zCDP)

Definition 2.2 (Rényi Divergence ((Rényi, 1961))). *Let X and X' be random variables over $\mathcal{X}$. For $\alpha \in (1, \infty)$, the* Rényi divergence *of order α between X and X' is defined by $D_\alpha(X||X') = \frac{1}{\alpha - 1} \cdot \ln\left(\mathrm{E}_{x \leftarrow X}\left[\left(\frac{P(x)}{P'(x)}\right)^{\alpha - 1}\right]\right)$, where $P(\cdot)$ and $P'(\cdot)$ are the probability mass/density functions of X and X', respectively.*

Definition 2.3 (zCDP Indistinguishability). *We say that two random variable X, X' over a domain $\mathcal{X}$ are ρ-*indistinguishable *(denote by $X \approx_\rho X'$), if for every $\alpha \in (1, \infty)$ it holds that $D_\alpha(X||X'), D_\alpha(X'||X) \leq \rho\alpha$. We say that X, X' are (ρ, δ)-*indistinguishable *(denote by $X \approx_{\rho,\delta} X'$), if there exist events $E, E' \subseteq \mathcal{X}$ with $\Pr[X \in E], \Pr[X' \in E'] \geq 1 - \delta$ such that $X|_E \approx_\rho X|_{E'}$.*

Definition 2.4 ((ρ, δ)-zCDP (Bun & Steinke, 2016)). *An algorithm A is δ-*approximate ρ-zCDP *(in short, (ρ, δ)-*zCDP*), if for any neighboring databases $\mathcal{D}, \mathcal{D}'$ it holds that $\mathsf{A}(\mathcal{D}) \approx_{\rho,\delta} \mathsf{A}(\mathcal{D}')$.*[3] *If the above holds for $\delta = 0$, we say that A is ρ-*zCDP.

Fact 2.5 (Composition (Bun & Steinke, 2016)). *If $\mathsf{A}\colon \mathcal{X}^* \to \mathcal{Y}$ is (ρ, δ)-*zCDP *and $\mathsf{A}'\colon \mathcal{X}^* \times \mathcal{Y} \to \mathcal{Z}$ is (ρ', δ')-*zCDP *(as a function of its first argument), then the algorithm $\mathsf{A}''(\mathcal{D}) := \mathsf{A}'(\mathcal{D}, \mathsf{A}(\mathcal{D}))$ is $(\rho + \rho', \delta + \delta')$-*zCDP.

Fact 2.6 (Gaussian Mechanism (Dwork et al., 2006a; Bun & Steinke, 2016)). *Let $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$ be vectors with $\|\boldsymbol{x} - \boldsymbol{x}'\| \leq \lambda$. Then for $\rho > 0$, $\sigma = \frac{\lambda}{\sqrt{2\rho}}$ and $Z \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \cdot \mathbb{I}_{d \times d})$ it holds that $\boldsymbol{x} + Z \approx_\rho \boldsymbol{x}' + Z$.*

3. Friendly zCDP

In this section we define a "friendly" relaxation of zCDP, and give an example of such an algorithm. We refer to Appendix B for the extended version of this section that includes full constructions, statements and proofs.

Definition 3.1 (Friendly zCDP). *An algorithm A is called f-*friendly (ρ, δ)-zCDP*, if for every neighboring databases $\mathcal{D}, \mathcal{D}'$ such that $\mathcal{D} \cup \mathcal{D}'$ is f-friendly, it holds that $\mathsf{A}(\mathcal{D}) \approx_{\rho,\delta} \mathsf{A}(\mathcal{D}')$.*

[3] We remark that our two parameters (ρ, δ)-zCDP has a different meaning than the two parameters definition (ξ, ρ)-zCDP of (Bun & Steinke, 2016). Throughout this work, we only consider the case $\xi = 0$ and therefore omit it from notation.

The following is a concrete example of such algorithm.

Algorithm 3.2 (FriendlyAvg, informal)**.**
Input: A database $\mathcal{D} \in (\mathbb{R}^d)^n$, privacy parameters $\rho, \delta > 0$, and $r \geq 0$. Operation:

1. *Compute an 0.1ρ-zCDP estimation $\hat{n}$ of n such that $\hat{n} \leq n$ with confidence $1-\delta$ (if $\hat{n} \leq 0$, abort).*
2. *Output $\mathrm{Avg}(\mathcal{D}) + \mathcal{N}(0, \sigma^2 \mathbb{I}_{d\times d})$ for $\sigma = \frac{2r}{\hat{n}} \cdot \frac{1}{\sqrt{1.8\rho}}$.*

We claim that FriendlyAvg is dist_r-friendly zCDP. Indeed, this holds since the ℓ_2-sensitivity of the function Avg is at most $2r/n \leq 2r/\hat{n}$ for neighboring databases with dist_r-friendly union. In order to see it, fix dist_r-friendly neighboring databases $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ and $\mathcal{D}' = \mathcal{D}_{-j}$, and note that by friendliness, for every $i \in [n] \setminus \{j\}$ there exists a point $\boldsymbol{y}_i \in \mathbb{R}^d$ such that $\|\boldsymbol{x}_i - \boldsymbol{y}_i\| \leq r$ and $\|\boldsymbol{x}_j - \boldsymbol{y}_i\| \leq r$, yielding that $\|\mathrm{Avg}(\mathcal{D}) - \mathrm{Avg}(\mathcal{D}')\| = \left\|\frac{(n-1)\cdot \boldsymbol{x}_j - \sum_{i\in[n]\setminus\{j\}} \boldsymbol{x}_i}{n(n-1)}\right\| \leq \frac{\sum_{i\in[n]\setminus\{j\}} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{n(n-1)} \leq \frac{\sum_{i\in[n]\setminus\{j\}} (\|\boldsymbol{x}_i - \boldsymbol{y}_i\| + \|\boldsymbol{x}_j - \boldsymbol{y}_i\|)}{n(n-1)} \leq \frac{2r}{n}$.

4. From Friendly to Standard zCDP

In this section we describe a paradigm for transforming any f-friendly zCDP algorithm A, for some $f\colon \mathcal{X}^2 \to \{0,1\}$, into a standard (end-to-end) zCDP one. We refer to Appendix C for the extended version of this section that includes also a paradigm for the DP model along with full constructions, statements, proofs, and comparison between the models.

The main component is algorithm FriendlyCore (described below) that outputs a core $\mathcal{C} \subseteq \mathcal{D}$:

Algorithm 4.1 (FriendlyCore)**.**
Input: A database $\mathcal{D} = (x_1, \ldots, x_n) \in \mathcal{X}^*$, a predicate $f\colon \mathcal{X}^2 \mapsto \{0,1\}$, and $\rho, \delta > 0$.
Operation:

i. *Let $\rho_1 = 0.1\rho$ and $\rho_2 = 0.9\rho$.*

ii. *Compute $\hat{n} = n + \sqrt{\frac{\ln(2/\delta)}{\rho_1}} + \mathcal{N}(0, \frac{1}{2\rho_1})$.*

iii. *For $i \in [n]$:*

   (a) *Let $z_i = \sum_{j=1}^n f(x_i, x_j) - n/2$, and let $\hat{z}_i = z_i + \mathcal{N}(0, \frac{\hat{n}}{8\rho_2})$.*

   (b) *If $\hat{z}_i < \sqrt{\frac{\hat{n}\cdot \ln(2\hat{n}/\delta)}{4\rho_2}} + \frac{1}{2}$, set $v_i = 0$. Otherwise, set $v_i = 1$.*

iv. *Output $\mathcal{C} = \mathcal{D}_{\{i\in[n]\colon v_i = 1\}}$.*

The intuition for FriendlyCore is the following: First, since we work in the insertion/deletion model, we first need to create a private estimation $\hat{n}$ of n. With confidence $1-\delta/2$, it holds that $\hat{n} \geq n$. In that case, we obtain with confidence $1-\delta/2$ that all elements x_i's that have no more than $n/2$ friends (and therefore with $z_i \leq 0$) are going to be out of the core $\mathcal{C}$. This yields that except with probability δ, the core only contains elements with more than $n/2$ friends. Therefore, for neighboring databases $\mathcal{D} = (x_1, \ldots, x_n)$ and $\mathcal{D}' = (x_1, \ldots, x_{n-1})$, we obtain with confidence $1-\delta$ that the database $\mathcal{C} \cup \mathcal{C}'$, for $\mathcal{C} = \mathsf{FriendlyCore}(\mathcal{D})$ and $\mathcal{C}' = \mathsf{FriendlyCore}(\mathcal{D}')$, is f-friendly. Furthermore, note that for every $i \in [n-1]$ it holds that $|z_i - z_i'| = |1/2 - f(x_i, x_n)| = 1/2$, where z_i' refers to the value of z_i in the execution $\mathsf{FriendlyCore}(\mathcal{D}')$. It follows by the properties of the Gaussian mechanism and composition that the values of $(\hat{z}_1, \ldots, \hat{z}_{n-1})$ are all together ρ_2-indistinguishable, yielding (by post-processing) that $\mathcal{C} \setminus \{x_n\} \approx_{\rho_2} \mathcal{C}'$.

These friendliness and privacy properties of FriendlyCore are formally stated in the following lemma.

Lemma 4.2. *Fix neighboring databases $\mathcal{D} = (x_1, \ldots, x_n)$ and $\mathcal{D}' = (x_1, \ldots, x_{n-1})$, and let $V = (V_1, \ldots, V_n)$ and $V' = (V_1', \ldots, V_{n-1}')$ be the (random variables of the) values of $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)$ in independent random executions of $\mathsf{FriendlyCore}(\mathcal{D}, f, \rho, \delta)$ and $\mathsf{FriendlyCore}(\mathcal{D}', f, \rho, \delta)$ (respectively). Then there exist events $E \subseteq \{0,1\}^n$ and $E' \subseteq \{0,1\}^{n-1}$ with $\Pr[V \in E], \Pr[V' \in E'] \geq 1-\delta$, such that the following holds:*

1. *Friendliness: For every $\boldsymbol{v} \in E$ and $\boldsymbol{v}' \in E'$, the database $\mathcal{C} \cup \mathcal{C}'$, for $\mathcal{C} = \mathcal{D}_{\{i\in[n]\colon v_i=1\}}$ and $\mathcal{C}' = \mathcal{D}'_{\{i\in[n-1]\colon v_i'=1\}}$, is f-friendly, and*
2. *Privacy: $(V_{-j})|_E \approx_\rho V'|_{E'}$.*

The following lemma states that whenever n is sufficiently large, then FriendlyCore also satisfies the following utility guarantee.

Lemma 4.3. *Let $f\colon \mathcal{X}^2 \to \{0,1\}$ and $\rho, \delta > 0$. For every $0 \leq \alpha < 1/2$, $n \geq \frac{-4\cdot\ln((1/2-\alpha)\rho\delta)}{(1/2-\alpha)^2\rho}$, and $\mathcal{D} \in \mathcal{X}^n$, with probability $1-\delta$ over a random execution of $\mathsf{FriendlyCore}(\mathcal{D}, f, \rho, \delta)$, the output core $\mathcal{C}$ contains all elements $x \in \mathcal{D}$ with $\sum_{y\in\mathcal{D}} f(x,y) \geq (1-\alpha)n$.*

4.1. Paradigm for zCDP

We now state our general paradigm for obtaining standard (end-to-end) zCDP.

Theorem 4.4 (Paradigm for zCDP)**.** *For every $\rho, \delta > 0$ and f-friendly (ρ', δ')-zCDP algorithm A, algorithm $\mathsf{B}(\mathcal{D}) := \mathsf{A}(\mathsf{FriendlyCore}(\mathcal{D}, f, \rho, \delta))$ is $(\rho + \rho', \delta + \delta')$-zCDP. Furthermore, for every $0 \leq \alpha < 1/2$, $n \geq \frac{-4\cdot\ln((1/2-\alpha)\rho\delta)}{(1/2-\alpha)^2\rho}$*

and $\mathcal{D} \in \mathcal{X}^n$, with probability $1-\delta$ over the execution $\mathsf{FriendlyCore}(\mathcal{D}, f, \rho, \delta)$*, the output includes all elements $x \in \mathcal{D}$ with $\sum_{y\in\mathcal{D}} f(x,y) \geq (1-\alpha)n$.*

4.2. Computation efficiency

FriendlyCore computes $f(x,y)$ for all pairs, that is, doing $O(n^2)$ applications of the predicate. However, using standard concentration bounds, it is possible to use a random sample of $O(\log(n/\delta))$ elements y for estimating with high accuracy the number of friends of each x. This provides very similar privacy guarantees, but is computationally more efficient for large n. See Appendix G for more details.

5. Applications

In this section we briefly present two applications of FriendlyCore: Averaging (Section 5.1) and Clustering (Section 5.2). We refer to Appendix D for a full descriptions of our algorithms, proofs of their privacy guarantees, and refer to Appendices H.2 and H.3 for missing utility statements and proofs. In Appendix F we also present a third application of learning a covariance matrix in the DP model.

5.1. Averaging

In this section we use FriendlyCore to compute a private average of points $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in (\mathbb{R}^d)^*$. In Section 5.1.1 we present a zCDP algorithm that given an (utility) advise of the effective diameter r of the points, estimates $\mathrm{Avg}(\mathcal{D})$ up to an additive ℓ_2 error of $O\left(\frac{r}{n}\cdot\sqrt{\frac{d}{\rho}}\right)$. In Section 5.1.2 we sketch the case where the effective diameter r is unknown, but only a segment that contains r is given. See Appendix D.1.3 for comparison with previous results.

5.1.1. Known Diameter

Algorithm 5.1 describes the algorithm for the known diameter case, and Theorem 5.2 states its privacy guarantee.

Algorithm 5.1 (FC_Avg)**.**
Input: A database $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in (\mathbb{R}^d)^*$, privacy parameters $\rho, \delta > 0$, and a diameter $r \geq 0$. Operation:

1. *Compute* $\mathcal{C} = \mathsf{FriendlyCore}(\mathcal{D}, \mathsf{dist}_r, 0.1\rho, \delta/2)$.
2. *Output* $\mathsf{FriendlyAvg}(\mathcal{C}, 0.9\rho, \delta/2, r)$.

Theorem 5.2. *Algorithm* FC_Avg$(\cdot, \rho, \delta, r)$ *is* (ρ, δ)-zCDP.

5.1.2. Unknown Diameter

Here we assume that the effective diameter r is unknown, but we get as input a (utility) advise of lower and upper bounds $r_{\min}, r_{\max}$ (respectively) on it. Informally, our algorithm FC_Avg_UnknownDiam searches for the effective diameter r using a private binary search over $\{r_{\min}, 2\cdot r_{\min}, 4\cdot r_{\min}, \ldots, r_{\max}\}$ along with a private algorithm that determines (with high confidence) if a given diameter is good or not. After the search, the algorithm applies FC_Avg on the chosen diameter. This process results with an additive ℓ_2 error of $O\left(\frac{r}{n}\sqrt{\frac{(d+\log\log(r_{\max}/r_{\min}))}{\rho}}\right)$.

5.2. Clustering

In this section we use FriendlyCore for constructing our private clustering algorithm FC_Clustering. Recently, (Cohen et al., 2021) identified a very simple clustering problem, called *unordered k-tuple clustering*, and reduced standard clustering tasks like k-means and k-GMM (under common separation assumptions) to this simple problem via the sample and aggregate framework of (Nissim et al., 2007). The idea is to split the database into random parts, and execute a non-private clustering algorithm on each part for obtaining an *unordered* k-tuples from each execution. Then the goal is to privately aggregate all the k-tuples for obtaining a new k-tuple that is close to them. See Figure 2 for a graphical illustration.

(Cohen et al., 2021) formalized the k-tuple clustering problem, described simple algorithms that privately solve this problem, and then provided proven utility guarantees for k-means and k-GMM using the above reduction. However, their algorithms do not perform well in practice (i.e., requires either too many tuples or an extremely large separation). In this section we show how to solve the unordered k-tuple clustering problem using FriendlyCore in a much more efficient way, yielding the first algorithm of this type that is also practical in many interesting cases (see Appendix E.2). In Algorithm 5.3 we sketch the steps of the algorithm, and refer to Figure 3 for a graphical illustration of the steps on synthetic data in dimension 2. A formal description of the algorithm appears in Appendix D.2.

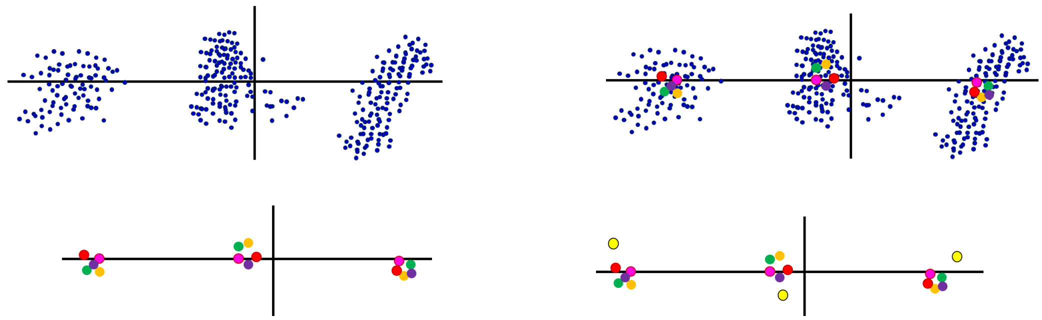

Figure 2: Top left: Database of points. Top right: Executing a non-private clustering algorithm over random parts of the data. Each execution returns an *unordered* k-tuple (e.g., the red points are the first tuple, the green points are the second tuple, etc.). Bottom left: The original points are ignored, and the focus is on a new database, where each element there is an unordered k-tuple (e.g., the tuple of red points is the first element in the new database). Bottom right: When the tuples are close to each other (as in the picture), the goal is to output a new k-tuple that is close to them (e.g., the yellow points). The challenge is to do it while preserving differential privacy (with respect to the new database of tuples).

Algorithm 5.3 (FC_Clustering, informal)**.**
Input: A database $\mathcal{D} \in (\mathbb{R}^d)^*$, *parameters* $\rho, \delta > 0$, *a bound* $\Lambda > 0$ *on the* ℓ_2 *norm of the points, and a parameter* $t \in \mathbb{N}$ *(number of tuples).*
Oracle: Non private clustering algorithm A.
Operation:

1. *Shuffle the order of the points in* $\mathcal{D}$. *Let* $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ *be the database after the shuffle.*
2. *For* $i \in [t]$: *Compute the* k*-tuple* $X^i = \mathsf{A}(\mathcal{D}^i)$ *where* $\mathcal{D}^i = (\boldsymbol{x}_{(i-1)\cdot m+1}, \ldots, \boldsymbol{x}_{i \cdot m})$ *for* $m = \lfloor n/t \rfloor$.
3. *Let* $\mathcal{T} = (X^1, \ldots, X^t)$.
4. *Compute* $\mathcal{C} = \mathsf{FriendlyCore}(\mathcal{T}, \mathsf{match}_{1/7}, \rho/3, \delta/3)$.
5. *Pick a tuple* $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k) \in \mathcal{T}$ *and split the set of all points of all the tuples in* $\mathcal{T}$ *into* k *parts* $\mathcal{Q}^1, \ldots, \mathcal{Q}^k$ *according to it (i.e., each point* $\boldsymbol{y}$ *is chosen to be in* $\mathcal{Q}^i$ *for* $i = \operatorname{argmin}_{j \in [k]} \|\boldsymbol{x}_i - \boldsymbol{y}\|$*).*
6. *For* $i \in [k]$: *Compute* $(\rho/3, \delta/3)$*-zCDP averages* $Y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k)$ *for* $\mathcal{Q}^1, \ldots, \mathcal{Q}^k$ *(respectively).*
7. *Perform a private Lloyd step over the entire database* $\mathcal{D}$ *with the centers* Y *(using privacy budget* $\rho/3, \delta/3$ *and radius* Λ*), and output the resulting centers.*

Remark 5.4. *Computing the averages in Step 6 can be done by applying* FC_Avg_UnknownDiam *on each of the* $\mathcal{Q}^i$*'s (i.e., additional* k *calls to* FriendlyCore*). But actually, we do that using a single call to* FriendlyCore *which applied with a special type of predicate over* ordered *tuples (see Appendix D.2.3 for more details).*

Theorem 5.5 (Privacy of FC_Clustering)**.** *Algorithm* $\mathsf{FC\_Clustering}^{\mathsf{A}}(\cdot, \rho, \delta, \Lambda, t)$ *is* (ρ, δ)*-zCDP (for any* A*).*

6. Empirical Results

In this section we present empirical results of our FriendlyCore based averaging and clustering algorithms. In all experiments we used privacy parameter $\rho = 1, \delta = 10^{-8}$, and all of them were tested on a MacBook Pro Laptop with 4-core Intel i7 CPU with 2.8GHz, and with 16GB RAM. We refer to Appendix E for the full details of our experiments.

6.1. Averaging

We tested mean estimation of samples from a Gaussian with *unknown* mean and *known* variance. We compared a Python implementation of our private averaging algorithm FC_Avg with the algorithm CoinPress of (Biswas et al., 2020). The implementations of CoinPress, and the experimental test bed, were taken from the publicly available code of (Biswas et al., 2020) provided at `https://github.com/twistedcubic/coin-press`. Following (Biswas et al., 2020), we generate a dataset of n samples from a d-dimensional Gaussian $\mathcal{N}(0, I_{d \times d})$. We ran FC_Avg with $r = \sqrt{2}(\sqrt{d} + \sqrt{\ln(100n)})$.[4]

Algorithm CoinPress requires a bound R on the ℓ_2 norm of the unknown mean. Both algorithms perform a similar final private averaging step that has dependence on $\sqrt{d}$. But they differ in the "preparation:" CoinPress has inherent dependence on d and R. FC_Avg preparation, on the other hand, has no dependence on d or R.

Following (Biswas et al., 2020) we perform 50 repetitions of each experiment and use the trimmed average of values between the 0.1 and 0.9 quantiles. We show the ℓ_2 error of our estimate on the Y-axis. Figure 4(1) reports

[4]This choice is for guaranteeing that almost all pairs of samples have ℓ_2 distance at most r from each other (computed according to the known variance)

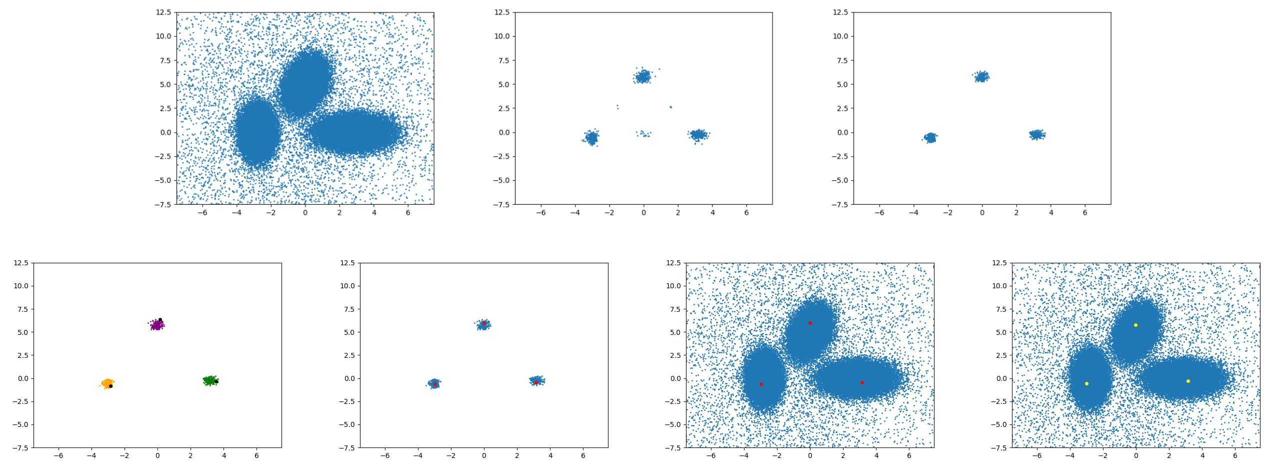

Figure 3: Top figures from left to right: (a) Database of size $n = 10^5$. (b) Points of 300 3-tuples that have been generated by (non-private) k-means++ on random parts of the database (Step 2). (c) Points of all the 202 3-tuples that were chosen to be in the core (Step 4). Bottom figures from left to right: (d) Picking the first tuple (black points) and splitting the points according to it (Step 5). (e) Privately estimating the averages of each part (red points, Step 6). (f) The private centers places on the entire data. (g) The centers after a private Lloyd step (yellow points, Step 7).

the effect of varying the bound R, with fixed $d = 1000$ and $n = 800$. We tested CoinPress with 4, 20 and 40 iterations. We observe that FC_Avg, that does not depend on R, outperforms CoinPress for $R > 10^7$. Figure 4(2) reports the effect of varying the dimension d, with fixed $n = 800$ and $R = 10\sqrt{d}$. We tested CoinPress with 2, 4 and 8 iterations. We observed that the performance of all algorithms deteriorates with increasing d, which is expected due to all algorithms using private averaging, but CoinPress deteriorates much faster in the large-d regime.

Finally we note that CoinPress slightly performs better than FC_Avg in the small-d small-R regime (shown in Figure 7(3), Appendix E.1). The reason is that FriendlyAvg (Algorithm 3.2), which is the last step of FC_Avg, uses noise of magnitude $\approx \frac{2r}{n\sqrt{2\rho}}$ which is far by a factor of 2 from the ideal magnitude that we could hope for.

6.2. Clustering

We tested the performance of our private clustering algorithm FC_Clustering with 200 tuples on a number of k-Means and k-GMM tasks. We compared a Python implementation of FC_Clustering with a recent algorithm of Chang & Kamath (2021) that is based on recursive locality-sensitive hashing (LSH). We denote their algorithm by LSH_Clustering. The implementations of LSH_Clustering, and the experimental test bed of Figure 4(3), were taken from the publicly available code of (Chang & Kamath, 2021) provided at `https://github.com/google/differential-privacy/tree/main/learning/clustering`. LSH_Clustering guarantees privacy in the DP model. Therefore, in order to compare it with our $(\rho = 1, \delta)$-zCDP guarantee, we chose to apply it with a $(\varepsilon = 2, \delta)$-DP guarantee, so that non of the guarantees implies the other. Furthermore, unlike FC_Clustering which may fail to produce centers in some cases (e.g., when the core of tuples is empty or close to be empty), LSH_Clustering always produces centers. Therefore, in order to handle failures of FC_Clustering, we used only $\rho = 0.99$ privacy budget, and on failures we executed LSH_Clustering for guaranteeing $\rho = 0.01$ zCDP.

We performed 30 repetitions of each experiment and present the medians (points) along with the 0.1 and 0.9 quantiles.

In Figure 4(3) we present a comparison in dimension $d = 2$ with $k = 8$ synthetic clusters, sampled equality from 8 separated Gaussians. We plotted the normalized k-means loss that is computed by $1 - X/Y$, where X is the cost of k-means++ on the entire data, and Y is the cost of the tested private algorithm. We observe that for small values of n, FC_Clustering fails often, which yield an inaccurate results. Yet, increasing n also increases the success probability of FC_Clustering which yields very accurate results, while LSH_Clustering stay behind. We refer to Figure 8 (Right) in Appendix E.2 for a graphical illustration of the centers in one of the iterations.

In Figure 4(4) we used the publicly available dataset of (Fonollosa & Huerta, 2015) that contains the acquired time series from 16 chemical gas sensors exposed to gas mixtures at varying concentration levels. The dataset contains $\approx 8M$ rows, where each row contains 16 sensors' measurements

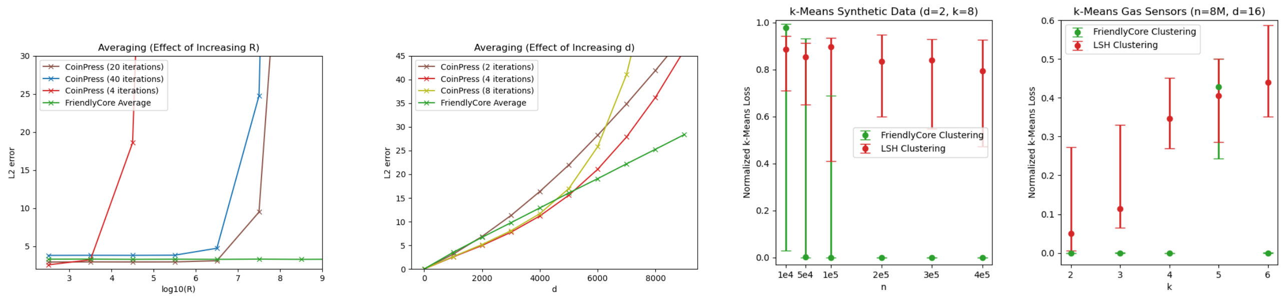

Figure 4: From Left to Right: (1) Averaging in $d = 1000$ and $n = 800$, varying R. (2) Averaging in $n = 800$, $R = 10\sqrt{d}$, varying d. (3) k-means in $d = 2$ and $k = 8$, for varying n. (4) k-means over Gas Sensors' measurements, varying k.

at a given point in time, so we translate each such row into a 16-dimensional point. We compared the clustering algorithms for varying k. We observed that FC_Clustering succeed well on various k's, except of $k = 5$ in which it fails due to instability of k-means++ on this database.[5]

In Appendix E.2, Figure 9 (Left), we present a comparison for separating $n = 2.5 \cdot 10^5$ samples from a uniform mixture of $k = 5$ Gaussians with separation $\approx \sqrt{d/2}$ between the centers, showing that using a (non-private) PCA-based clustering that easily separate between such Gaussians in high dimension, we gain a perfect labeling accuracy in contrast to LSH_Clustering.

In summary, we observed from the experiments that when FC_Clustering succeed, it outputs very accurate results. However, FC_Clustering may fail due to instability of the non-private algorithm on random parts of the database. Hence, it seems that in cases where we have a clear separation or many points (less separation requires more data points), we might gain by combining between FC_Clustering and LSH_Clustering. In this work we chose to spend 0.99 of the privacy budget on FC_Clustering, but other combinations might perform better on different cases.

7. Conclusion

We presented a general tool FriendlyCore for preprocessing metric data before privately aggregating it. The processed data is guaranteed to have some properties that can simplify or boost the accuracy of aggregation. Our tool is flexible, and in this work we illustrate it by presenting three different applications (averaging, clustering, and learning an unrestricted covariance matrix). We show the wide applicability of our framework by applying it to private mean estimation and clustering, and comparing it to private algorithms which are specifically tailored for those tasks. For private averaging, we presented a simple algorithm with dimension-independent preprocessing, that is also independent of the ℓ_2 norm of the points.[6] For private clustering, we presented the first practical algorithm that is based on the sample and aggregate framework of (Nissim et al., 2007), which has proven utility guarantees for easy instances (see Appendix H.3), and achieves very accurate results in practice when the data is either well separated or very large.

Acknowledgements

Edith Cohen is supported by Israel Science Foundation grant no. 1595-19.

Haim Kaplan is supported by Israel Science Foundation grant no. 1595-19, and the Blavatnik Family Foundation.

Yishay Mansour has received funding from the European Research Council (ERC) under the European Union'sHorizon 2020 research and innovation program (grant agreement No. 882396), by the Israel Science Foundation (grant number 993/17) and the Yandex Initiative for Machine Learning at Tel Aviv University.

Uri Stemmer is partially supported by the Israel Science Foundation (grant 1871/19) and by Len Blavatnik and the Blavatnik Family foundation.

References

Achlioptas, D. and McSherry, F. On spectral learning of mixtures of distributions. In *Learning Theory, 18th Annual Conference on Learning Theory, COLT 2005*, volume 3559, pp. 458–469, 2005.

Ashtiani, H. and Liaw, C. Private and polynomial time algorithms for learning gaussians and beyond. *CoRR*,

[5]There are two different solutions for $k = 5$ that have similar low cost but do not match, yielding that when splitting the data into random parts, k-means++ choose one of them in one set of parts and the other one in the other parts, and therefore fails.

[6]The latter property results with an optimal asymptotic that matches the histogram-based construction of (Karwa & Vadhan, 2018).

abs/2111.11320, 2021. URL https://arxiv.org/abs/2111.11320.

Biswas, S., Dong, Y., Kamath, G., and Ullman, J. R. Coinpress: Practical private mean and covariance estimation. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS 2020*, 2020.

Bun, M. and Steinke, T. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography - 14th International Conference, TCC 2016*, volume 9985, pp. 635–658, 2016.

Chang, A. and Kamath, P. Practical differentially private clustering, 2021. URL https://ai.googleblog.com/2021/10/practical-differentially-private.html.

Cohen, E., Kaplan, H., Mansour, Y., Stemmer, U., and Tsfadia, E. Differentially-private clustering of easy instances. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, volume 139, pp. 2049–2059, 2021. URL https://arxiv.org/abs/2112.14445.

Dwork, C. and Lei, J. Differential privacy and robust statistics. In *STOC*, pp. 371–380. ACM, May 31–June 2 2009.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, volume 4004, pp. 486–503, 2006a.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *TCC*, volume 3876, pp. 265–284, 2006b.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *TCC*, pp. 265–284. Springer, March 4-7 2006c.

Dwork, C., Rothblum, G. N., and Vadhan, S. P. Boosting and differential privacy. In *FOCS*, pp. 51–60. IEEE Computer Society, 2010.

Fonollosa, J. and Huerta, R. Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring, 2015. URL http://archive.ics.uci.edu/ml/datasets/Gas+sensor+array+under+dynamic+gas+mixtures.

Huang, Z., Liang, Y., and Yi, K. Instance-optimal mean estimation under differential privacy. In *Advances in Neural Information Processing Systems*, volume 34, pp. 25993–26004, 2021.

Kamath, G., Li, J., Singhal, V., and Ullman, J. Privately learning high-dimensional distributions. In Beygelzimer, A. and Hsu, D. (eds.), *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, volume 99 of *Proceedings of Machine Learning Research*, pp. 1853–1902. PMLR, 2019a.

Kamath, G., Sheffet, O., Singhal, V., and Ullman, J. Differentially private algorithms for learning mixtures of separated gaussians. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS 2019*, pp. 168–180, 2019b.

Kamath, G., Singhal, V., and Ullman, J. Private mean estimation of heavy-tailed distributions. *arXiv preprint arXiv:2002.09464*, 2020.

Kamath, G., Mouzakis, A., Singhal, V., Steinke, T., and Ullman, J. A private and computationally-efficient estimator for unbounded gaussians. *CoRR*, abs/2111.04609, 2021. URL https://arxiv.org/abs/2111.04609.

Karwa, V. and Vadhan, S. Finite sample differentially private confidence intervals. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, 2018.

Kothari, P. K., Manurangsi, P., and Velingker, A. Private robust estimation by stabilizing convex relaxations. *CoRR*, abs/2112.03548, 2021. URL https://arxiv.org/abs/2112.03548.

Levy, D., Sun, Z., Amin, K., Kale, S., Kulesza, A., Mohri, M., and Suresh, A. T. Learning with user-level privacy. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*, pp. 12466–12479, 2021.

Nissim, K., Raskhodnikova, S., and Smith, A. Smooth sensitivity and sampling in private data analysis. In *STOC*, pp. 75–84. ACM, 2007.

Nissim, K., Stemmer, U., and Vadhan, S. P. Locating a small cluster privately. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016*, pp. 413–427, 2016.

Ostrovsky, R., Rabani, Y., Schulman, L. J., and Swamy, C. The effectiveness of lloyd-type methods for the k-means problem. *J. ACM*, 59(6):28:1–28:22, 2012.

Rényi, A. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pp. 547–561, 1961.

Scala, M. Hypergeometric tail inequalities: ending the insanity. *arXiv preprint arXiv:1311.5939*, 2009.

Shechner, M. Differentially private algorithms for clustering with stability assumptions. Master's thesis, Ben-Gurion University, 2021.

Shechner, M., Sheffet, O., and Stemmer, U. Private k-means clustering with stability assumptions. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020*, volume 108 of *Proceedings of Machine Learning Research*, pp. 2518–2528, 2020.

Singhal, V. and Steinke, T. Privately learning subspaces. *Advances in Neural Information Processing Systems*, 34, 2021.

A. Preliminaries (Extended)

A.1. Notation

Throughout this work, a database $\mathcal{D}$ is an (ordered) vector over a domain $\mathcal{X}$. Given $\mathcal{D} = (x_1, \ldots, x_n) \in \mathcal{X}^n$, for $\mathcal{I} \subseteq [n]$ let $\mathcal{D}_{\mathcal{I}} := (x_i)_{i \in \mathcal{I}}$, let $\mathcal{D}_{-\mathcal{I}} := \mathcal{D}_{[n]\setminus\mathcal{I}}$, and for $i \in [n]$ let $\mathcal{D}_i := x_i$ and $\mathcal{D}_{-i} := \mathcal{D}_{-\{i\}}$ (i.e., $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$). For $\mathcal{D} = (x_1, \ldots, x_n)$ and $\mathcal{D}' = (x'_1, \ldots, x'_m)$ let $\mathcal{D} \cup \mathcal{D}' = (x_1, \ldots, x_n, x'_1, \ldots, x'_m)$. For $n \in \mathbb{N}$ we denote by 0^n the n-size all-zeros vector.

For $p \in [0,1]$ let $\mathrm{Bern}(p)$ be the Bernoulli distribution that outputs 1 w.p. p and 0 otherwise. For $\boldsymbol{p} = (p_1, \ldots, p_n) \in [0,1]^n$, we let $\mathrm{Bern}(\boldsymbol{p})$ be the distribution that outputs $(V_1, \ldots, V_n)$, where $V_i \leftarrow \mathrm{Bern}(p_i)$, and the V_i's are independent.

For $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$, we let $\|\boldsymbol{x}\| := \sqrt{\sum_{i=1}^d x_i^2}$ (i.e., the ℓ_2 norm of $\boldsymbol{x}$) and let $\|\boldsymbol{x}\|_1 := \sum_{i=1}^n |x_i|$ (the ℓ_1 norm of $\boldsymbol{x}$). For $\boldsymbol{c} \in \mathbb{R}^d$ and $r \geq 0$, we denote $B(\boldsymbol{c}, r) := \{\boldsymbol{x} \in \mathbb{R}^d \colon \|\boldsymbol{x} - \boldsymbol{c}\| \leq r\}$. For a database $\mathcal{D} \in (\mathbb{R}^d)^*$ we denote by $\mathrm{Avg}(\mathcal{D}) := \frac{1}{|\mathcal{D}|} \cdot \sum_{\boldsymbol{x} \in \mathcal{D}} \boldsymbol{x}$ the average of all points in $\mathcal{D}$. For $r \geq 0$ and $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ we denote $\mathsf{dist}_r(\boldsymbol{x}, \boldsymbol{y}) := \mathbb{1}_{\{\|\boldsymbol{x}-\boldsymbol{y}\| \leq r\}}$ (i.e., 1 if $\|\boldsymbol{x} - \boldsymbol{y}\| \leq r$ and 0 otherwise).

The support of a discrete random variable X over $\mathcal{X}$, denoted $\mathrm{Supp}(X)$, is defined as $\{x \in \mathcal{X} : P(x) > 0\}$, where $P(\cdot)$ is the probability mass/density function of X's distribution.

Throughout this paper, we define neighboring databases with respect to the insertion/deletion model, where one database is obtain by adding or removing an element from the other database. Formally,

Definition A.1 (Neighboring databases)**.** *Let $\mathcal{D}$ and $\mathcal{D}'$ be two databases over a domain $\mathcal{X}$. We say that $\mathcal{D}$ and $\mathcal{D}'$ are* neighboring, *if either there exists $j \in [|\mathcal{D}|]$ such that $\mathcal{D}_{-j} = \mathcal{D}'$, or there exists $j \in [|\mathcal{D}'|]$ such that $\mathcal{D} = \mathcal{D}'_{-j}$.*

A.2. Zero-Concentrated Differential Privacy (zCDP)

Definition A.2 (Rényi Divergence ((Rényi, 1961)))**.** *Let X and X' be random variables over $\mathcal{X}$. For $\alpha \in (1, \infty)$, the* Rényi divergence *of order α between X and X' is defined by*

$$D_\alpha(X||X') = \frac{1}{\alpha - 1} \cdot \ln\left(\mathrm{E}_{x \leftarrow X}\left[\left(\frac{P(x)}{P'(x)}\right)^{\alpha-1}\right]\right),$$

where $P(\cdot)$ and $P'(\cdot)$ are the probability mass/density functions of X and X', respectively.

Definition A.3 (zCDP Indistinguishability)**.** *We say that two random variable X, X' over a domain $\mathcal{X}$ are ρ-*indistinguishable *(denote by $X \approx_\rho X'$), iff for every $\alpha \in (1, \infty)$ it holds that*

$$D_\alpha(X||X'), D_\alpha(X'||X) \leq \rho\alpha.$$

*We say that X, X' are (ρ, δ)-*indistinguishable *(denote by $X \approx_{\rho,\delta} X'$), iff there exist events $E, E' \subseteq \mathcal{X}$ with $\Pr[X \in E], \Pr[X' \in E'] \geq 1 - \delta$ such that $X|_E \approx_\rho X|_{E'}$.*

Definition A.4 ((ρ, δ)-zCDP (Bun & Steinke, 2016))**.** *An algorithm A is δ-*approximate ρ-zCDP *(in short, (ρ, δ)-*zCDP*), if for any neighboring databases $\mathcal{D}, \mathcal{D}'$ it holds that $\mathsf{A}(\mathcal{D}) \approx_{\rho,\delta} \mathsf{A}(\mathcal{D}')$.*[7] *If the above holds for $\delta = 0$, we say that A is ρ-*zCDP.

A.3. (ε, δ)-Differential Privacy (DP)

Definition A.5 ((ε, δ)-DP-indistinguishable)**.** *Two random variable X, X' over a domain $\mathcal{X}$ are called (ε, δ)-DP-indistinguishable (in short, $X \approx^{\mathsf{DP}}_{\varepsilon,\delta} X'$), iff for any event $T \subseteq \mathcal{X}$, it holds that $\Pr[X \in T] \leq e^\varepsilon \cdot \Pr[X' \in T] + \delta$. If $\delta = 0$, we write $X \approx^{\mathsf{DP}}_{\varepsilon} X'$.*

Definition A.6 ((ε, δ)-DP (Dwork et al., 2006b))**.** *Algorithm A* is (ε, δ)-DP, *if for any two neighboring databases $\mathcal{D}, \mathcal{D}'$ it holds that $\mathsf{A}(\mathcal{D}) \approx^{\mathsf{DP}}_{\varepsilon,\delta} \mathsf{A}(\mathcal{D}')$. If $\delta = 0$ (i.e., pure privacy), we say that A* is ε-DP.

A.4. Properties of DP and zCDP

Fact A.7 (From DP to zCDP and vice versa ((Bun & Steinke, 2016)))**.** *Any (ε, δ)-DP mechanism is $(\frac{1}{2}\varepsilon^2, \delta)$-$\mathsf{zCDP}$. Any (ρ, δ)-zCDP mechanism is $(\rho + 2\sqrt{\rho \ln(1/\delta')},\ \delta + \delta')$-$\mathsf{DP}$ for every $\delta' > 0$.*

Fact A.8 (Group Privacy ((Bun & Steinke, 2016)))**.** *Let $\mathcal{D}$ and $\mathcal{D}'$ be a pair of databases that differ by k points (i.e., $\mathcal{D}$ is obtained by k operations of addition or deletion of points on $\mathcal{D}'$). If A is ρ-zCDP, then $\mathsf{A}(\mathcal{D}) \approx_{k^2\rho} \mathsf{A}(\mathcal{D}')$. If A is (ε, δ)-DP, then $\mathsf{A}(\mathcal{D}) \approx^{\mathsf{DP}}_{k\varepsilon,\ e^{k\varepsilon}k\delta} \mathsf{A}(\mathcal{D}')$.*

[7] We remark that our two parameters (ρ, δ)-zCDP has a different meaning than the two parameters definition (ξ, ρ)-zCDP of (Bun & Steinke, 2016). Throughout this work, we only consider the case $\xi = 0$ and therefore omit it from notation.

Fact A.9 (Post-processing)**.** *Let F be a (randomized) function. If A is (ρ, δ)-zCDP, then $F \circ \mathsf{A}$ is (ρ, δ)-zCDP. If A is (ε, δ)-DP, then $F \circ \mathsf{A}$ is (ε, δ)-DP.*

A.4.1. The Laplace Mechanism

Definition A.10 (Laplace distribution)**.** *For $\sigma \geq 0$, let $\mathrm{Lap}(\sigma)$ be the Laplace distribution over $\mathbb{R}$ with probability density function $p(z) = \frac{1}{2\sigma} \exp\left(-\frac{|z|}{\sigma}\right)$.*

Theorem A.11 (The Laplace Mechanism (Dwork et al., 2006c))**.** *Let $x, x' \in \mathbb{R}$ with $|x - x'| \leq \lambda$. Then for every $\varepsilon > 0$ it holds that $x + \mathrm{Lap}(\lambda/\varepsilon) \approx_{\varepsilon}^{\mathsf{DP}} x' + \mathrm{Lap}(\lambda/\varepsilon)$.*

A.4.2. The Gaussian Mechanism

Definition A.12 (Gaussian distributions)**.** *For $\mu \in \mathbb{R}$ and $\sigma \geq 0$, let $\mathcal{N}(\mu, \sigma^2)$ be the Gaussian distribution over $\mathbb{R}$ with probability density function $p(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)$. For higher dimension $d \in \mathbb{N}$, let $\mathcal{N}(\mathbf{0}, \sigma^2 \cdot \mathbb{I}_{d \times d})$ be the spherical multivariate Gaussian distribution with variance σ^2 in each axis. That is, if $Z \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot \mathbb{I}_{d \times d})$ then $Z = (Z_1, \ldots, Z_d)$ where $Z_1, \ldots, Z_d$ are i.i.d. according to $N(0, \sigma^2)$.*

Fact A.13 (Concentration of One-Dimensional Gaussian)**.** *If X is distributed according to $\mathcal{N}(0, \sigma^2)$, then for all $\beta > 0$ it holds that*

$$\Pr\left[X \geq \sigma\sqrt{2\ln(1/\beta)}\right] \leq \beta.$$

Theorem A.14 (The Gaussian Mechanism (Dwork et al., 2006a; Bun & Steinke, 2016))**.** *Let $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$ be vectors with $\|\boldsymbol{x} - \boldsymbol{x}'\|_2 \leq \lambda$. For $\rho > 0$, $\sigma = \frac{\lambda}{\sqrt{2\rho}}$ and $Z \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot \mathbb{I}_{d \times d})$ it holds that $\boldsymbol{x} + Z \approx_{\rho} \boldsymbol{x}' + Z$. For $\varepsilon, \delta > 0$, $\sigma = \frac{\lambda\sqrt{2\ln(1.5/\delta)}}{\varepsilon}$ and $Z \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot \mathbb{I}_{d \times d})$ it holds that $\boldsymbol{x} + Z \approx_{\varepsilon,\delta}^{\mathsf{DP}} \boldsymbol{x}' + Z$.*

We remark that zCDP is tailored for this mechanism, i.e. it achieves pure zCDP with relatively small noise (compared to the DP case).

A.4.3. Composition

Fact A.15 (Composition of DP and zCDP mechanisms (Dwork et al., 2010; Bun & Steinke, 2016))**.** *If $\mathsf{A}\colon \mathcal{X}^* \to \mathcal{Y}$ is (ρ, δ)-zCDP and $\mathsf{A}'\colon \mathcal{X}^* \times \mathcal{Y} \to \mathcal{Z}$ is (ρ', δ')-zCDP (as a function of its first argument), then the algorithm $\mathsf{A}''(\mathcal{D}) := \mathsf{A}'(\mathcal{D}, \mathsf{A}(\mathcal{D}))$ is $(\rho + \rho',\ \delta + \delta')$-zCDP. If A is (ε, δ)-DP and A' is (ε', δ')-DP then A'' is $(\varepsilon + \varepsilon', \delta + \delta')$-DP.*

We remark that Fact A.15 is optimal for the zCDP model, but not optimal for the DP model.

A.4.4. Other Facts

Fact A.16. *Let X, X' be random variables over a domain $\mathcal{X}$, and let $E, E \subseteq \mathcal{X}$ be events such that $X|_E \approx_{\rho,\delta} X'|_{E'}$ and $\Pr[X \in E], \Pr[X' \in E'] \geq 1 - \delta'$. Then $X \approx_{\rho,\ \delta+\delta'} X'|_{E'}$.*

Proof. By definition there exists $F \subseteq E$ and $F' \subseteq E'$ with $\Pr[X \in F | X \in E], \Pr[X \in F' | X \in E'] \geq 1 - \delta$ such that $X|_F \approx_{\rho} X'|_{F'}$. The proof now follows since $\Pr[X \in F] = \Pr[X \in E] \cdot \Pr[X \in F | X \in E] \geq (1 - \delta') \cdot (1 - \delta) \geq 1 - (\delta + \delta')$, and similarly $\Pr[X' \in F'] \geq 1 - (\delta + \delta')$. ☐

The following fact is proven in Appendix H.1.

Fact A.17. *Let X, X' be ρ-indistinguishable random variables over $\mathcal{X}$, and let $E \subseteq \mathcal{X}$ be an event with $\Pr[X \in E], \Pr[X' \in E] \geq q$. Then $X|_E \approx_{\rho/q} X'|_E$.*

B. Friendly Differential Privacy (Extended)

In this section we define a "friendly" relaxation of zCDP and DP, and give an example of such an algorithm. We start by defining an f-friendly database for a predicate f.

Definition B.1 (f-friendly)**.** *Let $\mathcal{D}$ be a database over a domain $\mathcal{X}$, and let $f\colon \mathcal{X}^2 \to \{0, 1\}$ be a predicate. We say that $\mathcal{D}$ is f-friendly if for every $x, y \in \mathcal{D}$, there exists $z \in \mathcal{X}$ (not necessarily in $\mathcal{D}$) such that $f(x, z) = f(y, z) = 1$.*

We next define the relaxation of zCDP and DP, where the privacy requirement must only hold for neighboring datasets that their union is f-friendly. Formally,

Definition B.2 (Friendly zCDP and DP)**.** *An algorithm A is called f-friendly (ρ, δ)-zCDP, if for every neighboring databases $\mathcal{D}, \mathcal{D}'$ such that $\mathcal{D} \cup \mathcal{D}'$ is f-friendly, it holds that $\mathsf{A}(\mathcal{D}) \approx_{\rho,\delta} \mathsf{A}(\mathcal{D}')$. If for every such $\mathcal{D}, \mathcal{D}'$ it holds that $\mathsf{A}(\mathcal{D}) \approx_{\varepsilon,\delta}^{\mathsf{DP}} \mathsf{A}(\mathcal{D}')$, we say that A is f-friendly (ε, δ)-DP.*

Note that nothing is guaranteed when $\mathcal{D} \cup \mathcal{D}'$ is not f-friendly. Intuitively, this allows us to focus the privacy requirement only on well-behaved inputs, potentially requiring significantly less noise to be added.

We next describe a concrete example of a friendly zCDP algorithm for estimating the average of points in $\mathbb{R}^d$, where the friendliness is with respect to the predicate $\mathsf{dist}_r(\boldsymbol{x}, \boldsymbol{y}) := \mathbb{1}_{\{\|\boldsymbol{x}-\boldsymbol{y}\| \leq r\}}$ for a given parameter $r \geq 0$.

Algorithm B.3 (FriendlyAvg)**.**

Input: A database $\mathcal{D} \in (\mathbb{R}^d)^*$, privacy parameters $\rho, \delta > 0$, and $r \geq 0$.

Operation:

1. *Let $n = |\mathcal{D}|$, $\rho_1 = 0.1(1 - \delta)\rho$ and $\rho_2 = 0.9\rho$.*

2. *Compute* $\hat{n} = n - \sqrt{\frac{\ln(1/\delta)}{\rho_1}} - 1 + \mathcal{N}(0, \frac{1}{2\rho_1})$.
3. *If* $n = 0$ *or* $\hat{n} \leq 0$, *output* $\perp$ *and abort.*
4. *Otherwise, output* $\mathrm{Avg}(\mathcal{D}) + \mathcal{N}(0, \sigma^2 \cdot \mathbb{I}_{d\times d})$ *for* $\sigma = \frac{2r}{\hat{n}} \cdot \frac{1}{\sqrt{2\rho_2}}$.

We remark that Step 4 of FriendlyAvg is the standard zCDP Gaussian Mechanism (Theorem A.14) that guarantees ρ_2-indistinguishably for two databases $\mathcal{D}$ and $\mathcal{D}'$ with $\|\mathrm{Avg}(\mathcal{D}) - \mathrm{Avg}(\mathcal{D}')\| \leq 2r/\hat{n}$. Steps 1 to 3 are for making the value of $\hat{n}$ indistinguishable between executions over neighboring databases (recall that we handle the insertion/deletion model).

We also remark that FriendlyAvg can be easily modified for the DP model: Given $\varepsilon > 0$ (instead of ρ), split it into $\varepsilon_1, \varepsilon_2$, compute $\hat{n} = n - \frac{\ln(1/\delta)}{\varepsilon_1} + \mathrm{Lap}(1/\varepsilon_1)$ (i.e., add laplace noise instead of Gaussian noise), and at the last step, use the Gaussian mechanism for DP with $\sigma' = \frac{2r}{\hat{n}} \cdot \frac{\sqrt{2\ln(2.5/\delta)}}{\varepsilon_2}$.

We next prove the properties of FriendlyAvg (in the zCDP model).

Claim B.4 (Privacy of FriendlyAvg)**.** *Algorithm* $\mathsf{FriendlyAvg}(\cdot, \rho, \delta, r)$ *is* dist_r*-friendly* (ρ, δ)-zCDP.

Proof. Let $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ and $\mathcal{D}' = \mathcal{D}_{-j}$ be two f_r-friendly neighboring databases, and let $n' = n-1$. Consider two independent random executions of $\mathsf{FriendlyAvg}(\mathcal{D})$ and $\mathsf{FriendlyAvg}(\mathcal{D}')$ (both with the same input parameters ρ, δ, r). Let $\widehat{N}$ and O be the (r.v.'s of the) values of $\hat{n}$ and the output in the execution $\mathsf{FriendlyAvg}(\mathcal{D})$, let $\widehat{N}'$ and O' be these r.v.'s w.r.t. the execution $\mathsf{FriendlyAvg}(\mathcal{D}')$, and let ρ_1, ρ_2 be as in Step 1.

If $n' = 0$ then $\Pr[O' = \perp] = 1$ and $n = 1$, and by a concentration bound of the normal distribution (Fact A.13) it holds that $\Pr[O = \perp] \geq 1 - \delta$. Therefore, we conclude that $O \approx_{0,\delta} O'$ in this case.

It is left to handle the case $n' \geq 1$. By Fact A.13 (concentration of one-dimensional Gaussian) it holds that $\Pr\left[\widehat{N} \leq n\right], \Pr\left[\widehat{N}' \leq n\right] \geq 1 - \delta$. It is left to prove that $O|_{\hat{N} \leq n} \approx_{\rho} O'|_{\hat{N}' \leq n}$.

Since $n - n' = 1$, then by the properties of the Gaussian Mechanism (Theorem A.14) it holds that $\widehat{N} \approx_{\rho_1} \widehat{N}'$. By Fact A.17 we deduce that $\widehat{N}|_{\widehat{N} \leq n} \approx_{\rho_1/(1-\delta)} \widehat{N}'|_{\widehat{N}' \leq n}$. Hence by composition (Fact A.15) it is left to prove that for every fixing of $\hat{n} \leq n$ it holds that $O|_{\widehat{N} = \hat{n}} \approx_{\rho_2} O'|_{\widehat{N}' = \hat{n}}$. For $\hat{n} \leq 0$ it is clear (both outputs are $\perp$). Therefore, we show it for $\hat{n} \in (0, n]$.

By the dist_r-friendly assumption, for every $i \in [n] \setminus \{j\}$ there exists a point $\boldsymbol{y}_i \in \mathbb{R}^d$ such that $\|\boldsymbol{x}_i - \boldsymbol{y}_i\| \leq r$ and $\|\boldsymbol{x}_j - \boldsymbol{y}_i\| \leq r$. Now, observe that $\|\mathrm{Avg}(\mathcal{D}) - \mathrm{Avg}(\mathcal{D}')\| = \left\| \frac{(n-1)\cdot \boldsymbol{x}_j - \sum_{i \in [n]\setminus\{j\}} \boldsymbol{x}_i}{n(n-1)} \right\| \leq \frac{\sum_{i \in [n]\setminus\{j\}} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{n(n-1)} \leq \frac{\sum_{i \in [n]\setminus\{j\}} (\|\boldsymbol{x}_i - \boldsymbol{y}_i\| + \|\boldsymbol{x}_j - \boldsymbol{y}_i\|)}{n(n-1)} \leq \frac{2r}{n}$. Namely, the ℓ_2-sensitivity of the function Avg is at most $2r/n \leq 2r/\hat{n}$ for neighboring and dist_r-friendly databases. The proof now follows by the guarantee of the Gaussian Mechanism (Theorem A.14). □

C. From Friendly to Standard Differential Privacy (Extended)

In this section we describe a paradigm for transforming any f-friendly zCDP (or DP) algorithm A, for some $f \colon \mathcal{X}^2 \to \{0,1\}$, into a standard zCDP (or DP) one. The main component is an algorithm F (called a "filter") that decides which elements to take into the core. Namely, given a database $\mathcal{D} = (x_1, \ldots, x_n)$, $\mathsf{F}(\mathcal{D})$ returns a vector $\boldsymbol{v} \in \{0,1\}^n$ such that the sub-database $\mathcal{C} = (x_i)_{\boldsymbol{v}_i = 1}$ (the "core") satisfies properties that are described below. We only focus on *product*-filters:

Definition C.1 (product-filter)**.** *We say that* $\mathsf{F} \colon \mathcal{X}^* \to \{0,1\}^*$ *is a* product-filter *if for every* n *and every* $\mathcal{D} \in \mathcal{X}^n$*, there exists* $\boldsymbol{p} = (p_1, \ldots, p_n) \in [0,1]^n$ *such that* $V = \mathsf{F}(\mathcal{D})$ *is distributed according to* $\mathrm{Bern}(\boldsymbol{p})$.

In this work we present two product-filters: BasicFilter (Appendix C.1) and zCDPFilter (Appendix C.2). The filters are slightly different, but follow the same paradigm: For every $i \in [n]$, compute $\sum_{j=1}^{n} f(x_i, x_j)$ (i.e., the number of x_i's friends). If this number is no more than $n/2$, then set $p_i = 0$ (or almost zero). If this number is high (i.e., close to n), then set $p_i = 1$ (or almost one). Between $n/2$ and n, use smooth low-sensitivity p_i's (i.e., probabilities that do not change by much if the number of friends is changed by one). As a result, we obtain in particular that all the elements in the core are guaranteed to have more than $n/2$ friends. It follows that if we look at executions on neighboring databases, then the resulting cores $\mathcal{C}$ and $\mathcal{C}'$ satisfy that $\mathcal{C} \cup \mathcal{C}'$ is f-friendly because for every $x_i, x_j \in \mathcal{C} \cup \mathcal{C}'$, the set of x_i's friends must intersect the set of x_j's friends.

The utility property (i.e., taking elements with many friends), is captured by the following definition.

Definition C.2 ((f, α, β, n)-complete filter)**.** *We say that a filter* $\mathsf{F} \colon \mathcal{X}^* \to \{0,1\}^*$ *is* (f, n, α, β)-complete*, if given a database* $\mathcal{D} = (x_1, \ldots, x_n) \in \mathcal{X}^n$, $\mathsf{F}(\mathcal{D})$ *outputs w.p.* $1 - \beta$ *a vector* $\boldsymbol{v} = (v_1, \ldots, v_n) \in \{0,1\}^n$ *s.t.* $v_i = 1$ *for every* $i \in [n]$ *with* $\sum_{j=1}^{n} f(x_i, x_j) \geq (1-\alpha)n$*. We omit the* n *if the above holds for every* $n \in \mathbb{N}$*, and omit the* β *if the above also holds for* $\beta = 0$.

Namely, with probability $1 - \beta$, such a filter gives us a "core" $\mathcal{C}$ which contains all elements $x_i \in \mathcal{D}$ that are friends of at least $1 - \alpha$ fraction of the elements in $\mathcal{D}$. For $\alpha = 0$ we obtain a filter that preserves a "complete" database $\mathcal{D}$: if for every $x_i, x_j \in \mathcal{D}$ it holds that $f(x_i, x_j) = 1$ (i.e., all the elements are friends of each other), then w.p. $1-\beta$ it holds that $\mathcal{C} = \mathcal{D}$ (i.e., no element is removed from the core).

C.1. Basic Filter

In the following we describe BasicFilter and prove its properties.

> **Algorithm C.3** (BasicFilter).
> *Input: A database $\mathcal{D} = (x_1, \ldots, x_n) \in \mathcal{X}^*$, a predicate $f\colon \mathcal{X}^2 \mapsto \{0,1\}$, and $0 \le \alpha < 1/2$.*
> *Operation:*
>
> i. *For $i \in [n]$:*
>   (a) *Let $z_i = \sum_{j=1}^n f(x_i, x_j) - n/2$.*
>   (b) *Sample $v_i \leftarrow \mathrm{Bern}(p_i)$ for $p_i = \begin{cases} 0 & z_i \le 0 \\ 1 & z_i \ge (1/2-\alpha)n. \\ \frac{z_i}{(1/2-\alpha)n} & o.w. \end{cases}$*
>
> ii. *Output $\boldsymbol{v} = (v_1, \ldots, v_n)$.*

Note that for every i, if x_i has at most $n/2$ friends, then $p_i = 0$, and if x_i has at least $(1-\alpha)n$ friends, then $p_i = 1$. We next state and prove the properties of BasicFilter.

Lemma C.4. *For any predicate $f\colon \mathcal{X}^2 \to \{0,1\}$ and $0 \le \alpha < 1/2$, $\mathsf{F} = \mathsf{BasicFilter}(\cdot, f, \alpha)$ is an (f,α)-complete product-filter. Furthermore, for every $n \in \mathbb{N}$ and every neighboring databases $\mathcal{D} \in \mathcal{X}^n$ and $\mathcal{D}' = \mathcal{D}_{-j}$, the following holds w.r.t. the random variables $V = \mathsf{F}(\mathcal{D})$ and $V' = \mathsf{F}(\mathcal{D}')$:*

1. *Friendliness: For every $\boldsymbol{v} \in \mathrm{Supp}(V)$ and $\boldsymbol{v}' \in \mathrm{Supp}(V')$, the database $\mathcal{C} \cup \mathcal{C}'$, for $\mathcal{C} = \mathcal{D}_{\{i \in [n]\colon v_i = 1\}}$ and $\mathcal{C}' = \mathcal{D}'_{\{i \in [n-1]\colon v'_i = 1\}}$, is f-friendly, and*
2. *Stability: Let $\boldsymbol{p} = (p_1, \ldots, p_n)$ and $\boldsymbol{p}' = (p'_1, \ldots, p'_{n-1})$ for $p_i = \Pr[V_i = 1]$ and $p'_i = \Pr[V'_i = 1]$. Then $\left\|\boldsymbol{p}_{-j} - \boldsymbol{p}'\right\|_1 \le 1/(1-2\alpha)$.*

Namely, apart of being a complete filter, BasicFilter preserves small ℓ_1 norm of the probabilities of the vectors up to the index j of the additional element. In addition, for any neighboring databases $\mathcal{D}$ and $\mathcal{D}'$, it guarantees that $\mathcal{C} \cup \mathcal{C}'$, for the resulting cores $\mathcal{C}$ and $\mathcal{C}'$, is f-friendly.

Proof of Lemma C.4. It is clear by construction that $\mathsf{F} = \mathsf{BasicFilter}(\cdot, f, \alpha)$ is a product-filter. Also, the (f,α)-complete property immediately holds by construction since for every database $\mathcal{D} = (x_1, \ldots, x_n)$, each element x_i with $\sum_{j=1}^n f(x_i, x_j) \ge (1-\alpha)n$ has $z_i \ge (1/2-\alpha)n$ and therefore $p_i = 1$ (i.e., $v_i = 1$ w.p. 1). We next prove the friendliness and stability properties.

Fix neighboring databases $\mathcal{D} = (x_1, \ldots, x_n)$ and $\mathcal{D}' = \mathcal{D}_{-j}$, let $V = \mathsf{F}(\mathcal{D})$ and $V' = \mathsf{F}(\mathcal{D}')$, let z_i, p_i be the values in the execution $\mathsf{F}(\mathcal{D})$ and let z'_i, p'_i be these values in the execution $\mathsf{F}(\mathcal{D}')$. For proving the friendliness property, we fix $i \in [n]$ with $p_i > 0$ and $k \in [n-1]$ with $p'_k > 0$, and show that there exists y with $f(x_i, y) = f(x_k, y) = 1$. Since $p_i > 0$ it holds that $\sum_{\ell \in [n]} f(x_i, x_\ell) \ge \lfloor n/2 \rfloor + 1$ and therefore $\sum_{\ell \in [n] \setminus \{j\}} f(x_i, x_\ell) \ge \lfloor n/2 \rfloor$. In addition, since $p'_k > 0$ it holds that $\sum_{\ell \in [n] \setminus \{j\}} f(x_k, x_\ell) \ge \lfloor (n-1)/2 \rfloor + 1$. Since $\lfloor n/2 \rfloor + (\lfloor (n-1)/2 \rfloor + 1) = n > n-1$, there must exists $\ell \in [n] \setminus \{j\}$ with $f(x_i, x_\ell) = f(x_k, x_\ell) = 1$, as required.

For proving the stability property, note that for every $i \in [n] \setminus \{j\}$ it holds that $\frac{z_i}{n} - \frac{z'_i}{n-1} = \frac{\sum_{\ell \in [n]} f(x_i, x_\ell)}{n} - \frac{\sum_{\ell \in [n] \setminus \{j\}} f(x_i, x_\ell)}{n-1} = \frac{f(x_i, x_j)}{n} - \frac{\sum_{\ell \in [n] \setminus \{j\}} f(x_i, x_\ell)}{n(n-1)}$. Since the above belongs to $[-1/n, 1/n]$, we deduce that $|p_i - p'_i| \le \frac{1}{(1-2\alpha)n}$ and conclude that $\left\|\boldsymbol{p}_{-j} - \boldsymbol{p}'\right\|_1 \le \frac{n-1}{(1-2\alpha)n} < \frac{1}{1-2\alpha}$. □

C.2. zCDP Filter

We next describe our filter zCDPFilter that is tailored for the zCDP model and is better in practice.

> **Algorithm C.5** (zCDPFilter).
> *Input: A database $\mathcal{D} = (x_1, \ldots, x_n) \in \mathcal{X}^*$, a predicate $f\colon \mathcal{X}^2 \mapsto \{0,1\}$, and $\rho, \delta > 0$.*
> *Operation:*
>
> i. *Let $\rho_1 = 0.1\rho$ and $\rho_2 = 0.9\rho$.*
>
> ii. *Compute $\hat{n} = n + \sqrt{\frac{\ln(2/\delta)}{\rho_1}} + \mathcal{N}(0, \frac{1}{2\rho_1})$.*
>
> iii. *For $i \in [n]$:*
>   (a) *Let $z_i = \sum_{j=1}^n f(x_i, x_j) - n/2$, and let $\hat{z}_i = z_i + \mathcal{N}(0, \frac{\hat{n}}{8\rho_2})$.*
>   (b) *If $\hat{z}_i < \sqrt{\frac{\hat{n} \cdot \ln(2\hat{n}/\delta)}{4\rho_2}} + \frac{1}{2}$, set $v_i = 0$. Otherwise, set $v_i = 1$.*
>
> iv. *Output $\boldsymbol{v} = (v_1, \ldots, v_n)$.*

Note that zCDPFilter differs from BasicFilter in the way it uses the $\{z_i\}$'s. BasicFilter use them directly to compute low-sensitivity probabilities $\{p_i\}$'s such that each v_i is sampled from $\mathrm{Bern}(p_i)$. zCDPFilter, on the other hand, does not compute the $\{p_i\}$'s explicitly. Rather, it creates noisy versions $\{\hat{z}_i\}$ of the $\{z_i\}$'s that preserve indistinguishability between neighboring executions, and therefore guarantees that the $\{v_i\}$'s are also indistinguishable by post-processing.

This and all other properties of zCDPFilter are stated in the following theorem.

Lemma C.6. *Let $f\colon \mathcal{X}^2 \to \{0,1\}$ and $\rho, \delta > 0$. $\mathsf{F} = \mathsf{zCDPFilter}(\cdot, f, \rho, \delta)$ is a product-filter that is (f, α, β, n)-complete for every $0 \le \alpha < 1/2$, $\beta > 0$, and $n \ge \frac{-4\cdot\ln((1/2-\alpha)\rho\cdot\min\{\beta,\delta\})}{(1/2-\alpha)^2\rho}$. Furthermore, for every $n \in \mathbb{N}$ and every neighboring databases $\mathcal{D} = (x_1,\ldots,x_n)$ and $\mathcal{D}' = \mathcal{D}_{-j}$, there exist events $E \subseteq \{0,1\}^n$ and $E' \subseteq \{0,1\}^{n-1}$ with $\Pr[\mathsf{F}(\mathcal{D}) \in E], \Pr[\mathsf{F}(\mathcal{D}') \in E'] \ge 1-\delta$, such that the following holds w.r.t. the random variables $V = \mathsf{F}(\mathcal{D})$ and $V' = \mathsf{F}(\mathcal{D}')$:*

1. *Friendliness: For every $\boldsymbol{v} \in E$ and $\boldsymbol{v}' \in E'$, the database $\mathcal{C} \cup \mathcal{C}'$, for $\mathcal{C} = \mathcal{D}_{\{i\in[n]\colon v_i=1\}}$ and $\mathcal{C}' = \mathcal{D}'_{\{i\in[n-1]\colon v'_i=1\}}$, is f-friendly, and*
2. *Privacy: $(V_{-j})|_E \approx_\rho V'|_{E'}$.*

The proof of Lemma C.6 appears at Appendix H.4 and sketched below.

Proof Sketch. Fix two neighboring databases $\mathcal{D} = (x_1,\ldots,x_n)$ and $\mathcal{D}' = (x_1,\ldots,x_{n-1})$, and consider two independent executions of $\mathsf{F}(\mathcal{D})$ and $\mathsf{F}(\mathcal{D}')$ for $\mathsf{F} = \mathsf{zCDPFilter}(\cdot, f, \rho, \delta)$. For simplicity, we assume that both executions use the same value $\hat{n}$ at Step ii. For utility, we use the fact $\hat{n} \le n + \sqrt{\frac{\ln(2/\delta)}{\rho_1}} + \sqrt{\frac{\ln(2/\beta)}{\rho_1}}$ with confidence $1-\beta/2$. By the lower bound on n, it follows that $(1/2-\alpha)n \ge \sqrt{\frac{\hat{n}\cdot\ln(2\hat{n}/\delta)}{4\rho_2}} + \sqrt{\frac{\hat{n}\cdot\ln(2\hat{n}/\beta)}{4\rho_2}} + \frac{1}{2}$, yielding that all elements with $(1-\alpha)n$ friends are added to the core with confidence $1-\beta/2$.

For proving friendliness and privacy, we define $E \subseteq \{0,1\}^n$ to be the subset of all vectors $\boldsymbol{v} \in \{0,1\}^n$ that does not include "bad" coordinates $i \in [n]$. Namely, $\boldsymbol{v}_i = 0$ for $i \in [n]$ with $\sum_{j=1}^{n-1} f(x_i, x_j) \le (n-1)/2$. Event $E' \subseteq \{0,1\}^{n-1}$ is defined by $\{\boldsymbol{v}_{-n}\colon \boldsymbol{v} \in E\}$ (i.e., the vectors in E without the n'th coordinate).

Note that $\hat{n} \ge n$ with confidence $1-\delta/2$. In that case it follows that in both executions $\mathsf{F}(\mathcal{D})$ and $\mathsf{F}(\mathcal{D}')$, all the bad elements are removed with confidence $1-\delta/2$, yielding that outputs are in E and E' (respectively).

The friendliness property now follows since for every $\boldsymbol{v} \in E$ and $\boldsymbol{v}' \in E'$ and for every $i, j \in [n-1]$ such that $\boldsymbol{v}_i = 1$ and $\boldsymbol{v}'_j = 1$, there exists $\ell \in [n-1]$ such that $f(x_i, x_\ell) = f(x_j, x_\ell) = 1$.

For proving the privacy guarantee, note that for every $i \in [n-1]$ it holds that $|z_i - z'_i| = |1/2 - f(x_i, x_n)| = 1/2$, yielding that $\widehat{Z}_i \approx_{\rho/n} \widehat{Z}'_i$. Therefore, by composition and post-processing, we deduce that $V_{-n} \approx_\rho V'$. Now note that when conditioning V on the event E, the "bad" coordinates become 0, and the distribution of the other coordinates remain the same (this is because the V_i's are independent, and E only fixes the bad i's to zero). Similarly, the same holds when conditioning V' on the event E', and therefore we conclude that $(V_{-n})|_E \approx_\rho V'|_{E'}$. □

Note that unlike BasicFilter, zCDPFilter has restrictions on n and β in the utility guarantee (i.e., β cannot be 0, and there is also a lower bound on n). Also, the friendliness and privacy properties only hold together with high probability, and not with probability 1 as in BasicFilter. Still, zCDPFilter is preferable in the zCDP model since its privacy guarantee is stronger than the stability guarantee (i.e., bound on the ℓ_1 norm) of BasicFilter. Another advantage of zCDPFilter is that it does not need to get the utility parameter α as input. Rather, it guarantees utility for any value α that preserves the lower bound on n.

C.3. Paradigm for zCDP

We next define FriendlyCore and state the general paradigm for obtaining (standard) end-to-end zCDP.

Definition C.7. *Define* $\mathsf{FriendlyCore}(\mathcal{D}, f, \rho, \delta) := \mathcal{D}_{\{i\colon v_i=1\}}$ *for* $\boldsymbol{v} = \mathsf{zCDPFilter}(\mathcal{D}, f, \rho, \delta)$.

Theorem C.8 (Paradigm for zCDP)**.** *For every $\rho, \delta > 0$ and f-friendly (ρ', δ')-zCDP algorithm A, algorithm $\mathsf{B}(\mathcal{D}) := \mathsf{A}(\mathsf{FriendlyCore}(\mathcal{D}, f, \rho, \delta))$ is $(\rho+\rho',\ \delta+\delta')$-zCDP. Furthermore, for every $0 \le \alpha < 1/2$, $\beta > 0$, $n \ge \frac{-4\cdot\ln((1/2-\alpha)\rho\min\{\beta,\delta\})}{(1/2-\alpha)^2\rho}$ and $\mathcal{D} \in \mathcal{X}^n$, with probability $1-\beta$ over the execution $\mathsf{FriendlyCore}(\mathcal{D}, f, \rho, \delta)$, the output includes all elements $x \in \mathcal{D}$ with $\sum_{y\in\mathcal{D}} f(x,y) \ge (1-\alpha)n$.*

For proving the privacy guarantee, we use the following lemma (proven in Appendix H.5) that bounds the zCDP-indistinguishability loss between two executions of a zCDP mechanism over random databases R, R' that are "almost indistinguishable" from being neighboring.

Lemma C.9. *Let $\mathcal{D} = (x_1,\ldots,x_n)$ and $\mathcal{D}' = \mathcal{D}_{-j}$ be neighboring databases, let V, V' be random variables over $\{0,1\}^n$ and $\{0,1\}^{n-1}$ (respectively) such that $V_{-j} \approx_{\rho,\delta} V'$, and define the random variables $R = \mathcal{D}_{\{i\in[n]\colon V_i=1\}}$ and $R' = \mathcal{D}'_{\{i\in[n-1]\colon V'_i=1\}}$. Let A be an algorithm such that for any neighboring $\mathcal{C} \in \mathrm{Supp}(R)$ and $\mathcal{C}' \in \mathrm{Supp}(R')$ satisfy $\mathsf{A}(\mathcal{C}) \approx_{\rho',\delta'} \mathsf{A}(\mathcal{C}')$. Then $\mathsf{A}(R) \approx_{\rho+\rho',\,\delta+\delta'} \mathsf{A}(R')$.*

Note that the requirement from algorithm A in Lemma C.9 is weaker than being (fully) (ρ', δ')-zCDP since it only guarantees indistinguishability for pairs of neighboring databases $(\mathcal{C}, \mathcal{C}') \in \mathrm{Supp}(R) \times \mathrm{Supp}(R')$, and not necessarily for all neighboring pairs in $\mathcal{X}^* \times \mathcal{X}^*$. This weaker requirement takes a crucial part in proving the privacy guarantee in Theorem C.8, since we apply the lemma with the algorithm A

which is only f-friendly zCDP, and use the fact that we are certified that every $\mathcal{C}$ and $\mathcal{C}$ in the support satisfy that $\mathcal{C} \cup \mathcal{C}'$ is f-friendly. The proof of Lemma C.9 basically follows by composition, but is slightly subtle. See the proof at Appendix H.5. We now prove Theorem C.8 using Lemma C.9.

Proof of Theorem C.8. The utility guarantee immediately follows since $\mathsf{zCDPFilter}(\cdot, f, \rho, \delta, \beta)$ is an (f, n, α, β)-complete database for such values of n (Lemma C.6). In the following we prove the privacy guarantee of B.

Let $\mathcal{D} = (x_1, \ldots, x_n)$ and $\mathcal{D}' = \mathcal{D}_{-j}$ be two neighboring databases. Consider two independent executions $\mathsf{B}(\mathcal{D})$ and $\mathsf{B}(\mathcal{D}')$. Let V be the (r.v. of the) value of $\boldsymbol{v}$ in the execution $\mathsf{B}(\mathcal{D})$ (the output of zCDPFilter that is computed internally in FriendlyCore), and let V' this r.v. w.r.t. the execution $\mathsf{B}(\mathcal{D}')$. By Lemma C.6, there exist events $E \subseteq \{0,1\}^n$ and $E' \subseteq \{0,1\}^{n-1}$ with $\Pr[V \in E], \Pr[V' \in E'] \geq 1 - \delta$ that satisfy Item 1 (friendliness) and Item 2 (privacy). The friendliness property implies that for every $\boldsymbol{v} \in E$ and $\boldsymbol{v}' \in E'$, the database $\mathcal{C} \cup \mathcal{C}'$, for $\mathcal{C} = \mathcal{D}_{\{i \in [n] \colon v_i = 1\}}$ and $\mathcal{C}' = \mathcal{D}'_{\{i \in [n-1] \colon v'_i = 1\}}$, is f-friendly. Therefore, in case $\mathcal{C}$ and $\mathcal{C}'$ are neighboring, we deduce that $\mathsf{A}(\mathcal{C}) \approx_{\rho', \delta'} \mathsf{A}(\mathcal{C}')$ since A is f-friendly (ρ', δ')-zCDP. The privacy guarantee of zCDPFilter implies that $V_{-j}|_E \approx_\rho V'|'_E$. Hence, by Lemma C.9 we deduce that $\mathsf{A}(R)|_{V \in E} \approx_{\rho+\rho', \delta'} \mathsf{A}(R')|_{V' \in E'}$ for the random variables $R = \mathcal{D}_{\{i \in [n] \colon V_i = 1\}}$ and $R' = \mathcal{D}'_{\{i \in [n-1] \colon V'_i = 1\}}$. We now conclude by Fact A.16 that $\mathsf{A}(R) \approx_{\rho+\rho', \delta+\delta'} \mathsf{A}(R')$, as required since $\mathsf{A}(R) \equiv \mathsf{B}(\mathcal{D})$ and $\mathsf{A}(R') \equiv \mathsf{B}(\mathcal{D}')$. □

C.4. Paradigm for DP

We next define FriendlyCoreDP and state the general paradigm for obtaining (standard) end-to-end DP.

Definition C.10. *Define* $\mathsf{FriendlyCoreDP}(\mathcal{D}, f, \alpha) := \mathcal{D}_{\{i \colon v_i = 1\}}$ *for* $\boldsymbol{v} = \mathsf{BasicFilter}(\mathcal{D}, f, \alpha)$.

Theorem C.11 (Paradigm for DP). *For every* $0 \leq \alpha < 1/2$ *and every f-friendly (ε, δ)-DP algorithm* A*, algorithm* $\mathsf{B}(\mathcal{D}) := \mathsf{A}(\mathsf{FriendlyCoreDP}(\mathcal{D}, f, \alpha))$ *is* $(\gamma(e^\varepsilon - 1),\ \gamma\delta e^{\varepsilon + \gamma(e^\varepsilon - 1)})$-DP *for* $\gamma = \frac{1}{(1-2\alpha)} + 1$. *Furthermore, the output of* $\mathsf{FriendlyCoreDP}(\mathcal{D}, f, \rho, \delta)$ *includes all elements* $x \in \mathcal{D}$ *with* $\sum_{y \in \mathcal{D}} f(x, y) \geq (1 - \alpha)n$.

We remark that for small values of ε and $\alpha = 0$, Theorem C.11 yields that if A is f-friendly (ε, δ)-DP, then B is $\approx (2\varepsilon,\ 2e^{3\varepsilon}\delta)$-DP, and in general for $\varepsilon = O(1)$ and $1/2 - \alpha = \Omega(1)$ we obtain $(O(\varepsilon), O(\delta))$-DP. Namely, the paradigm is optimal (up to constant factors) for transforming an f-friendly (ε, δ)-DP, for $\varepsilon = O(1)$, into a standard DP one.

For proving the privacy guarantee of FriendlyCoreDP, we use the following lemma (proven in Appendix H.6) that bounds the DP-indistinguishability loss between two executions over "ℓ_1-close" random databases.

Lemma C.12. *Let* $\mathcal{D} \in \mathcal{X}^n$ *and let* $\boldsymbol{p}, \boldsymbol{p}' \in [0,1]^n$ *with* $\|\boldsymbol{p} - \boldsymbol{p}'\|_1 \leq \gamma$. *Let V and V' be two random variables, distributed according to* $\mathrm{Bern}(\boldsymbol{p})$ *and* $\mathrm{Bern}(\boldsymbol{p}')$*, respectively, and define the random variables* $R = \mathcal{D}_{\{i \colon V_i = 1\}}$ *and* $R' = \mathcal{D}_{\{i \colon V'_i = 1\}}$. *Let* A *be an algorithm that for every neighboring databases* $\mathcal{C} \in \mathrm{Supp}(R)$ *and* $\mathcal{C}' \in \mathrm{Supp}(R')$ *satisfy* $\mathsf{A}(\mathcal{C}) \approx^{\mathsf{DP}}_{\varepsilon, \delta} \mathsf{A}(\mathcal{C}')$. *Then* $\mathsf{A}(R) \approx^{\mathsf{DP}}_{\gamma(e^\varepsilon - 1),\ \gamma\delta e^{\varepsilon + \gamma(e^\varepsilon - 1)}}$ $\mathsf{A}(R')$.

We now prove Theorem C.11 using Lemma C.12.

Proof of Theorem C.11. The utility guarantee immediately holds since $\mathsf{BasicFilter}(\cdot, f, \alpha)$ is (f, α)-complete (Lemma C.4). We next focus on proving the privacy guarantee.

Fix two neighboring databases $\mathcal{D} \in \mathcal{X}^n$ and $\mathcal{D}' = \mathcal{D}_{-j}$. Consider two independent executions of $\mathsf{B}(\mathcal{D})$ and $\mathsf{B}(\mathcal{D}')$. Let V be the (r.v. of the) value of $\boldsymbol{v}$ in the execution $\mathsf{B}(\mathcal{D})$ (the output of BasicFilter that is computed internally in FriendlyCoreDP), and let V' this r.v. w.r.t. the execution $\mathsf{B}(\mathcal{D}')$. By the stability property (Lemma C.4), there exist $\boldsymbol{p}, \boldsymbol{p}' \in [0,1]^n$ such that $V \leftarrow \mathrm{Bern}(\boldsymbol{p})$ and $V' \leftarrow \mathrm{Bern}(\boldsymbol{p}')$ and it holds that $\left\|\boldsymbol{p}_{-j} - \boldsymbol{p}'\right\|_1 \leq 1/(1 - 2\alpha)$. In order to apply Lemma C.12, we need to extend V' to be an n-size vector. Let $\tilde{V}'$ be the n-size vector that is obtained by adding 0 to the j'th location in V' (i.e., $\tilde{V}'_j = 0$ and $\tilde{V}'_{-j} = V'_{-j}$), and let $\tilde{\boldsymbol{p}}' \in \{0,1\}^n$ be the vector such that $\tilde{V}' \leftarrow \mathrm{Bern}(\tilde{\boldsymbol{p}}')$ (obtained by adding 0 to the j'th location in $\boldsymbol{p}'$). So it holds that $\left\|\boldsymbol{p} - \tilde{\boldsymbol{p}}'\right\|_1 \leq 1 + 1/(1 - 2\alpha)$. Let $R = \mathcal{D}_{\{i \colon V_i = 1\}}$ and $R' = \mathcal{D}_{\{i \colon \tilde{V}'_i = 1\}}$. By the friendliness property (Lemma C.4), for every $\mathcal{C} \in \mathrm{Supp}(R)$ and $\mathcal{C}' \in \mathrm{Supp}(R')$ it holds that $\mathcal{C} \cup \mathcal{C}'$ is f-friendly. We now conclude the proof by Lemma C.12 since A is f-friendly (ε, δ)-DP and it holds that $\mathsf{A}(R) \equiv \mathsf{B}(\mathcal{D})$ and $\mathsf{A}(R') \equiv \mathsf{B}(\mathcal{D}')$. □

C.5. Comparison Between the Paradigms

Up to constant factors, the paradigm for DP is optimal, since we transform an f-friendly (ε, δ)-DP algorithm into a $\approx (2\varepsilon, 2e^{3\varepsilon}\delta)$-DP one. However, in the zCDP model, when n is sufficiently large, we can use most of the privacy budget (say, 0.9 of it) for the friendly algorithm A, and use the rest for FriendlyCore (i.e., we do not have to lose significant constant factors). The zCDP model has also advantage of tight composition, and whenever the friendly algorithm A relies on the Gaussian Mechanism (i.e., for averaging and clustering problems), which is tailored for zCDP, we gain in accuracy compared to the DP model.

C.6. Computation efficiency

Our filters BasicFilter and zCDPFilter computes $f(x, y)$ for all pairs, that is, doing $O(n^2)$ applications of the predicate. However, using standard concentration bounds, it is possible to use a random sample of $O(\log(n/\delta))$ elements y for estimating with high accuracy the number of friends of each x. This provides very similar privacy guarantees, but is computationally more efficient for large n. See Appendix G for more details.

D. Applications (Extended)

In this section we present two applications of FriendlyCore: Averaging (Appendix D.1) and Clustering (Appendix D.2). These applications are described in the zCDP model, but can easily be adopted to the DP model as well. In Appendix F we present a third application of Covariance Matrix Estimation in the DP model, which relies on the tools that have been recently developed by (Ashtiani & Liaw, 2021). In this section we only describe the algorithms and prove their privacy guarantees, where we refer to Appendix H for the missing statements and proofs of the utility guarantees.

D.1. Averaging

In this section we use FriendlyCore to compute a private average of points $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in (\mathbb{R}^d)^*$. In Appendix D.1.1 we present a zCDP algorithm that given an (utility) advise of the effective diameter r of the points, estimates $\mathrm{Avg}(\mathcal{D})$ up to an additive ℓ_2 error of $O\left(\frac{r}{n} \cdot \sqrt{\frac{d}{\rho}}\right)$. In Appendix D.1.2 we present the case where the effective diameter r is unknown, but only a segment $[r_{\min}, r_{\max}]$ that contains r is given. Throughout this section, we remind the reader that we denote $\mathsf{dist}_r(\boldsymbol{x}, \boldsymbol{y}) := \mathbb{1}_{\{\|\boldsymbol{x}-\boldsymbol{y}\| \leq r\}}$.

D.1.1. Known Diameter

In the following we describe the algorithm for the known diameter case.

Algorithm D.1 (FC_Avg)**.**
Input: A database $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in (\mathbb{R}^d)^*$*, privacy parameters* $\rho, \delta > 0$ *and a diameter* $r \geq 0$.
Operation:

1. *Let* $\rho_1 = 0.1\rho$ *and* $\rho_2 = 0.9\rho$.
2. *Compute* $\mathcal{C} = \mathsf{FriendlyCore}(\mathcal{D}, \mathsf{dist}_r, \rho_1, \delta/2)$.
3. *Output* $\mathsf{FriendlyAvg}(\mathcal{C}, \rho_2, \delta/2, r)$ *(Algorithm B.3).*

Claim D.2 (Privacy of FC_Avg)**.** *Algorithm* FC_Avg$(\cdot, \rho, \delta, r)$ *is* (ρ, δ)-zCDP

Proof. Claim B.4 implies that FriendlyAvg$(\cdot, \rho_2, \delta/2, r)$ is dist$_r$-friendly $(\rho_2, \delta/2)$-zCDP. Therefore, we conclude by the privacy guarantee of the FriendlyCore paradigm (Theorem C.8) that FC_Avg$(\cdot, \rho, \delta, \beta, r)$ is $(\rho = \rho_1 + \rho_2, \delta)$-zCDP. □

D.1.2. Unknown Diameter

In the following we describe the algorithm FC_Avg_UnknownDiam for the unknown diameter case, where we are only given a lower and upper bound $r_{\min}, r_{\max}$ (respectively) on the effective diameter r. This is done by first searching for the diameter r using a private binary search FindDiam, and then apply our known diameter algorithm FC_Avg, which results with an additive ℓ_2 error of $O\left(\frac{r}{n}\sqrt{\frac{(d+\log\log(r_{\max}/r_{\min}))}{\rho}}\right)$ (proven in Appendix H). The following algorithm is the basic component of our binary search which checks (privately) whether a parameter r is a good diameter.

Algorithm D.3 (CheckDiam)**.**
Input: A database $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in (\mathbb{R}^d)^*$*, a privacy parameter* $\rho > 0$*, a confidence parameter* $\beta > 0$*, and a diameter* $r \geq 0$.
Operation:

i. *For* $i \in [n]$*: Compute* $s_i = |\{j \in [n] \colon \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \leq r\}|$.

ii. *Let* $a = (\sum_{i=1}^n s_i)/n$ *and let* $\hat{a} = a + \mathcal{N}(0, 2/\rho)$.

iii. *Output* $\begin{cases} 1 & \hat{a} \geq n - \sqrt{\frac{4\ln(1/\beta)}{\rho}} \\ 0 & o.w. \end{cases}$.

Claim D.4 (Privacy of CheckDiam)**.** *Algorithm* CheckDiam$(\cdot, \rho, \beta, r)$ *is* ρ-zCDP.

Proof. Fix two neighboring databases $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ and $\mathcal{D}' = \mathcal{D}_{-j}$, where we assume w.l.o.g. that $j = n$. i.e., $\mathcal{D}' = (x_1, \ldots, x_{n-1})$. Let $a, \{s_i\}_{i=1}^n$ and $a', \{s'_i\}_{i=1}^{n-1}$ be the values from Step ii in the executions CheckDiam$(\mathcal{D})$ and CheckDiam$(\mathcal{D}')$, respectively, and note that for every $i \in [n-1]$ is holds that $s'_i \leq s_i \leq s'_i + 1$. Therefore, it holds that

$$a \geq \frac{\sum_{i=1}^{n-1} s'_i}{n} = a' - \frac{\sum_{i=1}^{n-1} s'_i}{n(n-1)} \geq a' - 1,$$

and

$$a \leq \frac{\sum_{i=1}^{n-1}(s'_i + 1) + s_n}{n} \leq a' + \frac{n-1}{n} + \frac{s_n}{n} \leq a' + 2.$$

The privacy guarantee now follows by the Gaussian mechanism (Theorem A.14) and post-processing (Fact A.9). □

We next describe our private binary search for the diameter r.

Algorithm D.5 (FindDiam).
Input: A database $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in (\mathbb{R}^d)^n$*, a privacy parameter* $\rho > 0$*, a confidence parameter* $\beta > 0$*, lower and upper bounds* $r_{\min}, r_{\max} \geq 0$ *on the diameter (respectively), and a base* $b > 1$*.*
Operation:

i. *Let* $t = \log_b(r_{\max}/r_{\min})$*.*
ii. *Perform a binary search over* $x \in \{b^0, b^1, \ldots, b^t\}$*, each step of the search is done by calling to* CheckDiam$(\mathcal{D}, \frac{\rho}{\log_2(t)}, \frac{\beta}{\log_2(t)}, r = x \cdot r_{\min})$*.*
iii. *Output* $r = x \cdot r_{\min}$ *where* x *is the outcome of the above binary search.*

Claim D.6 (Privacy of FindDiam). *Algorithm* FindDiam$(\cdot, \rho, \beta, r_{\min}, r_{\max}, b)$ *is* ρ-zCDP.

Proof. Immediately holds by the privacy guarantee of CheckDiam (Claim D.4) and basic composition of $\log_2(\ell)$ iterations of the binary search. □

We now ready to fully describe our algorithm for estimating the average of points where the effective diameter is unknown.

Algorithm D.7 (FC_Avg_UnknownDiam).
Input: A database $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in (\mathbb{R}^d)^n$*, privacy parameters* $\rho, \delta > 0$*, a confidence parameter* $\beta > 0$*, and lower and upper bounds* $r_{\min}, r_{\max} > 0$ *on the diameter (respectively).*
Operation:

1. *Let* $\rho_1 = 0.1\rho$ *and* $\rho_2 = 0.9\rho$*.*
2. *Compute* $r =$ FindDiam$(\mathcal{D}, \rho_1, \beta/2, r_{\min}, r_{\max}, b = 1.5)$*.*
3. *Output* FC_Avg$(\mathcal{D}, \rho_2, \delta, \beta/2, r)$*.*

Claim D.8 (Privacy of FC_Avg_UnknownDiam). *Algorithm* FC_Avg_UnknownDiam$(\cdot, \rho, \delta, \beta, r_{\min}, r_{\max})$ *is* (ρ, δ)-zCDP

Proof. Immediately follows by composition (Fact A.15) of the ρ_1-zCDP mechanism FindDiam (Claim D.6) and the (ρ_2, δ)-zCDP mechanism FC_Avg (Claim D.2). □

D.1.3. Comparison with Previous Results

There are many previous results about private averaging in various settings that, like our averaging algorithms, attempt to add additive noise that is proportional to the effective diameter of the points (e.g., see (Nissim et al., 2016; Karwa & Vadhan, 2018; Kamath et al., 2019a; 2020; Biswas et al., 2020; Huang et al., 2021; Levy et al., 2021)). All these results (including ours) have a preprocessing step in which "outliers" are clipped or trimmed, and then it becomes "privacy safe" to add a small noise. Our FriendlyCore-based preprocessing step has two main advantages compared to the other methods: (1) It is dimension-independent, and (2) It is independent of the ℓ_2-norm of the points. These two advantages are illustrated in the experiments of Appendix E.1. As far as we know, all previous results do not satisfy (1), and most of them do not satisfy (2) either (the histogram-based construction of (Karwa & Vadhan, 2018) is the only result which is also independent of the ℓ_2-norm of the points, but is very dependent in the dimension). In addition, we remark that the additive error of our algorithms match the $\tilde{O}\left(\frac{r}{n} \cdot \sqrt{\frac{d}{\rho}}\right)$ optimal upper bound of (Huang et al., 2021). Actually, we even provide an asymptotical improvement compared to (Huang et al., 2021), because their approach requires an assumed bound Λ on the ℓ_2 norm of all the data points, even when the effective diameter r is known (a logarithmic dependency on Λ is hidden inside the $\tilde{O}$). In contrast, our approach does not need such a bound in the known r case (Claim H.4), and in the unknown r case we only require rough bounds $r_{\min}, r_{\max}$ on it (Claim H.8).

D.2. Clustering

In this section we use FriendlyCore for constructing our private clustering algorithm FC_Clustering. Recently, (Cohen et al., 2021) identified a very simple clustering problem, called *unordered k-tuple clustering*, and reduced standard clustering tasks like k-means and k-GMM (under common separation assumptions) to this simple problem via the sample and aggregate framework of (Nissim et al., 2007). The idea is to split the database into random parts, and execute a non-private clustering algorithm on each part for obtaining an *unordered* k-tuples from each execution. Then the goal is to privately aggregate all the k-tuples for obtaining a new k-tuple that is close to them. See Figure 5 for a graphical illustration.

(Cohen et al., 2021) formalized the k-tuple clustering problem, described simple algorithms that privately solve this problem, and then provided proven utility guarantees for k-means and k-GMM using the above reduction. However, their algorithms do not perform well in practice (i.e., requires either too many tuples or an extremely large separation). In this section we show how to solve the unordered k-tuple clustering problem using FriendlyCore in a much more efficient way, yielding the first algorithm of this type that is also practical in many interesting cases (see Appendix E.2). In this section we only describe our algorithms and prove

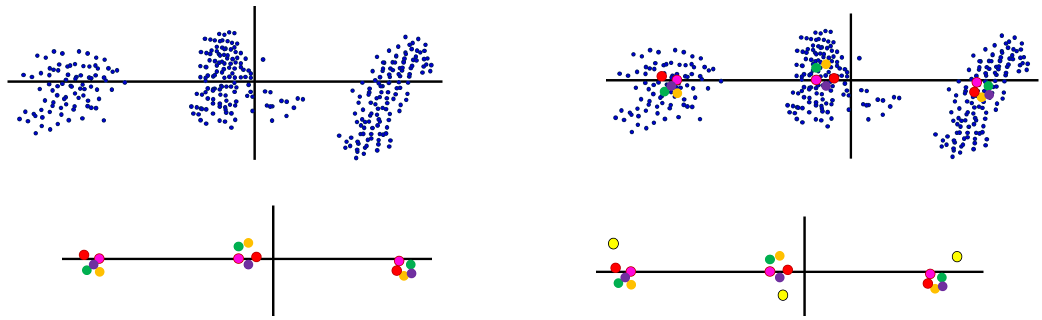

Figure 5: Top left: Database of points. Top right: Executing a non-private clustering algorithm over random parts of the data. Each execution returns an *unordered* k-tuple (e.g., the red points are the first tuple, the green points are the second tuple, etc.). Bottom left: The original points are ignored, and the focus is on a new database, where each element there is an unordered k-tuple (e.g., the tuple of red points is the first element in the new database). Bottom right: When the tuples are close to each other (as in the picture), the goal is to output a new k-tuple that is close to them (e.g., the yellow points). The challenge is to do it while preserving differential privacy (with respect to the new database of tuples).

their privacy guarantees. We refer to Appendix H.3 for proven utility guarantees that use the tools and formalization of (Cohen et al., 2021).

In Appendix D.2.1 we define a predicate match_γ for unordered k-tuples (where γ is a matching quality parameter), and prove properties of this predicate, where the main property is Claim D.15 which states that a match_γ-friendly database is $\mathsf{match}_{2\gamma/(1-\gamma)}$-complete. In Appendix D.2.2 we present a reduction FriendlyReorder from unordered to ordered tuples, that is privacy safe for databases that are $\mathsf{match}_{1/7}$-friendly. In Appendix D.2.3 we present the ordered tuples problem, and solve it again using a special specification of FriendlyCore. In Appendix D.2.4 we combine the reduction from unordered to ordered tuples, along with the algorithm for ordered one, and present our end-to-end zCDP algorithm FC_kTupleClustering for unordered k-tuple clustering. Finally, in Appendix D.2.5 we are going back to the original clustering problems that we are interested in (e.g., k-means and k-GMM) and present our main clustering algorithm FC_Clustering that combines between our algorithm FC_kTupleClustering for unordered k-tuple clustering to the reduction of (Cohen et al., 2021) from standard clustering problems into the unordered tuples problem.

While FC_Clustering consists of several components, the algorithm itself is not very complicated. For making the presentation more accessible, in Algorithm D.9 we give an informal description of FC_Clustering, and in Figure 3 we present a graphical illustration of the steps on synthetic data.

Algorithm D.9 (FC_Clustering, informal)**.**
Input: A database $\mathcal{D} \in (\mathbb{R}^d)^*$, parameters $\rho, \delta > 0$, a bound $\Lambda > 0$ on the ℓ_2 norm of the points, and a parameter $t \in \mathbb{N}$ (number of tuples).
Oracle: Non private clustering algorithm A.
Operation:

1. *Shuffle the order of the points in $\mathcal{D}$. Let $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ be the database after the shuffle.*
2. *For $i \in [t]$: Compute the k-tuple $X^i = \mathsf{A}(\mathcal{D}^i)$ where $\mathcal{D}^i = (\boldsymbol{x}_{(i-1)\cdot m+1}, \ldots, \boldsymbol{x}_{i\cdot m})$ for $m = \lfloor n/t \rfloor$.*
3. *Let $\mathcal{T} = (X^1, \ldots, X^t)$ (a database of unordered tuples).*
4. *Compute $\mathcal{C} = \mathsf{FriendlyCore}(\mathcal{T}, \mathsf{match}_{1/7}, \rho/3, \delta/3)$ ($\mathsf{match}_{1/7}$ is defined in Definition D.16).*
5. *Pick a tuple $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k) \in \mathcal{T}$ and split the set of all points of all the tuples in $\mathcal{T}$ into k parts $\mathcal{Q}^1, \ldots, \mathcal{Q}^k$ according to it (i.e., each point $\boldsymbol{y}$ is chosen to be in $\mathcal{Q}^i$ for $i = \mathrm{argmin}_{j\in[k]} \|\boldsymbol{x}_i - \boldsymbol{y}\|$).*
6. *For $i \in [k]$: Compute $(\rho/3, \delta/3)$-zCDP averages $Y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k)$ for $\mathcal{Q}^1, \ldots, \mathcal{Q}^k$ (respectively).*
7. *Perform a private Lloyd step over the entire database $\mathcal{D}$ with the centers Y (using privacy budget $\rho/3, \delta/3$ and radius Λ), and output the resulting centers.*

Theorem D.10 (Privacy of FC_Clustering)**.** *Algorithm* $\mathsf{FC\_Clustering}^{\mathsf{A}}(\cdot, \rho, \delta, \Lambda, t)$ *is* (ρ, δ)-zCDP *(for any* A*).*

The proof of Theorem D.10, along with the formal construction, appears at Appendix D.2.5.

Remark D.11. *Steps 4 to 6 of Algorithm 5.3 are actually an* *informal description of our algorithm* FC_kTupleClustering, *which is formally described in Appendix D.2.4. Step 5, which also can be seen as "ordering" the unordered tuples, is an informal description of our algorithm* FriendlyReorder *which is described in Appendix D.2.2. Note that computing the averages in Step 6 can be done by applying* FC_Avg_UnknownDiam *on each of the Q^i's (i.e., additional k calls to* FriendlyCore*). But actually, we do that by a new algorithm* FC_AvgOrdTup *that only uses a single call to* FriendlyCore *which is applied with a special type of predicate over* ordered *tuples. Algorithm* FC_AvgOrdTup *is described in Appendix D.2.3.*

D.2.1. Unordered k-Tuple Clustering

In this section we are given a database $\mathcal{D} \in ((\mathbb{R}^d)^k)^*$, where each element $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k) \in \mathcal{D}$ is a k-tuple of points in $\mathbb{R}^d$. In case all tuples are close to each other (up to reordering), the goal is to privately determine a new k-tuple that is close to them.

We start by defining a predicate over such tuples that captures the "closeness" property.

Definition D.12 (Predicate match_γ)**.** *For $\gamma \in [0,1]$, a permutation $\pi\colon [k] \to [k]$ and $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$, $Y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k) \in (\mathbb{R}^d)^k$, let $\mathsf{match}_\gamma^\pi(X,Y) = 1$ iff for every $i \in [k]$ it holds that*

$$\left\|\boldsymbol{x}_i - \boldsymbol{y}_{\pi(i)}\right\| < \gamma \cdot \min_{j \neq i}\{\min\{\left\|\boldsymbol{x}_i - \boldsymbol{y}_{\pi(j)}\right\|, \left\|\boldsymbol{x}_j - \boldsymbol{y}_{\pi(i)}\right\|\}\}.$$

We let $\mathsf{match}_\gamma(X,Y) = 1$ iff there exists a permutation π such that $\mathsf{match}_\gamma^\pi(X,Y) = 1$ (otherwise, $\mathsf{match}_\gamma(X,Y) = 0$).

Namely, for small constant γ, $\mathsf{match}_\gamma(X,Y) = 1$ means that there is a clear one-to-one matching between the points in $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ and the points in $Y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k)$ (see Figure 6 for an illustration).

In the following we prove key properties of this predicate. We start by stating an approximate triangle inequality with respect to this predicate for the case of the identity permutation.

Claim D.13. *Let $X, Y, Z \in (\mathbb{R}^d)^k$ such that $\mathsf{match}_\gamma^{\mathsf{id}}(X,Z) = \mathsf{match}_\gamma^{\mathsf{id}}(Y,Z) = 1$, where* id *is the identify permutation. Then $\mathsf{match}_{2\gamma/(1-\gamma)}^{\mathsf{id}}(X,Y) = 1$.*

Proof. Fix $i \in [k]$ and $j \in [k] \setminus \{i\}$, and note that

1. $\mathsf{match}_\gamma^{\mathsf{id}}(X,Z) = 1$ implies $\|\boldsymbol{x}_i - \boldsymbol{z}_i\| < \gamma \cdot \min\{\|\boldsymbol{x}_i - \boldsymbol{z}_j\|, \|\boldsymbol{x}_j - \boldsymbol{z}_i\|\}$.
2. $\mathsf{match}_\gamma^{\mathsf{id}}(Y,Z) = 1$ implies $\|\boldsymbol{y}_i - \boldsymbol{z}_i\| < \gamma \cdot \min\{\|\boldsymbol{y}_i - \boldsymbol{z}_j\|, \|\boldsymbol{y}_j - \boldsymbol{z}_i\|\}$.

We prove the claim by showing that $\|\boldsymbol{x}_i - \boldsymbol{y}_i\| < \frac{2\gamma}{1-\gamma}\left\|\boldsymbol{x}_i - \boldsymbol{y}_j\right\|$ (and by symmetry between X and Y we also deduce that $\|\boldsymbol{x}_i - \boldsymbol{y}_i\| < \frac{2\gamma}{1-\gamma}\|\boldsymbol{x}_j - \boldsymbol{y}_i\|$). Using triangle inequality multiple times, it holds that

$$\begin{aligned}\|\boldsymbol{x}_i - \boldsymbol{y}_i\| &\leq \|\boldsymbol{x}_i - \boldsymbol{z}_i\| + \|\boldsymbol{y}_i - \boldsymbol{z}_i\| \\ &< \gamma(\|\boldsymbol{x}_i - \boldsymbol{z}_j\| + \left\|\boldsymbol{y}_j - \boldsymbol{z}_i\right\|) \\ &\leq \gamma(2\left\|\boldsymbol{x}_i - \boldsymbol{y}_j\right\| + \|\boldsymbol{x}_i - \boldsymbol{z}_i\| + \left\|\boldsymbol{y}_j - \boldsymbol{z}_j\right\|). \qquad (1)\end{aligned}$$

We next bound $\|\boldsymbol{x}_i - \boldsymbol{z}_i\| + \left\|\boldsymbol{y}_j - \boldsymbol{z}_j\right\|$ as a function of $\left\|\boldsymbol{x}_i - \boldsymbol{y}_j\right\|$. Observe that

$$\|\boldsymbol{x}_i - \boldsymbol{z}_i\| < \gamma\|\boldsymbol{x}_i - \boldsymbol{z}_j\| \leq \gamma(\left\|\boldsymbol{x}_i - \boldsymbol{y}_j\right\| + \left\|\boldsymbol{y}_j - \boldsymbol{z}_j\right\|)$$

and

$$\left\|\boldsymbol{y}_j - \boldsymbol{z}_j\right\| < \gamma\left\|\boldsymbol{y}_j - \boldsymbol{z}_i\right\| \leq \gamma(\left\|\boldsymbol{x}_i - \boldsymbol{y}_j\right\| + \|\boldsymbol{x}_i - \boldsymbol{z}_i\|).$$

By summing the above two inequalities we deduce that

$$\|\boldsymbol{x}_i - \boldsymbol{z}_i\| + \left\|\boldsymbol{y}_j - \boldsymbol{z}_j\right\| < \frac{2\gamma}{1-\gamma}\left\|\boldsymbol{x}_i - \boldsymbol{y}_j\right\|. \qquad (2)$$

We now conclude by Equations (1) and (2) that

$$\|\boldsymbol{x}_i - \boldsymbol{y}_i\| < \left(2\gamma + \frac{2\gamma^2}{1-\gamma}\right)\left\|\boldsymbol{x}_i - \boldsymbol{y}_j\right\| = \frac{2\gamma}{1-\gamma}\left\|\boldsymbol{x}_i - \boldsymbol{y}_j\right\|.$$

□

We next extend Claim D.13 for arbitrary permutations.

Claim D.14. *Let $X, Y, Z \in (\mathbb{R}^d)^k$ such that $\mathsf{match}_\gamma^{\pi_1}(X,Z) = \mathsf{match}_\gamma^{\pi_2}(Y,Z) = 1$. Then $\mathsf{match}_{2\gamma/(1-\gamma)}^{\pi_2 \circ \pi_1^{-1}}(X,Y) = 1$.*

Proof. Let $X' = (\boldsymbol{x}_{\pi_1^{-1}(i)})_{i=1}^k$ and $Y' = (\boldsymbol{y}_{\pi_2^{-1}(i)})_{i=1}^k$. Then it holds that $\mathsf{match}_\gamma^{\mathsf{id}}(X',Z) = \mathsf{match}_\gamma^{\mathsf{id}}(Y',Z) = 1$, where id is the identity permutation. By Claim D.13 we deduce that $\mathsf{match}_{2\gamma/(1-\gamma)}^{\mathsf{id}}(X',Y') = 1$, yielding that $\mathsf{match}_{2\gamma/(1-\gamma)}^{\pi_2 \circ \pi_1^{-1}}(X,Y) = 1$. □

The following claim states how much we loss by moving from a friendly database into a complete one (in which there is a match between every pair of tuples).

Claim D.15. *If $\mathcal{D} \in ((\mathbb{R}^d)^k)^*$ is match_γ-friendly, then it is $\mathsf{match}_{2\gamma/(1-\gamma)}$-complete.*

Proof. Immediately follows by Claim D.14 since the match_γ-friendly assumption implies that for every $X, Y \in \mathcal{D}$ there exists $Z \in (\mathbb{R}^d)^k$ such that $\mathsf{match}_\gamma(X,Z) = \mathsf{match}_\gamma(Y,Z) = 1$.

□

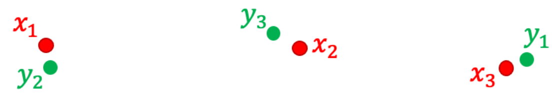

Figure 6: A graphical illustration of tuples $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$ and $Y = (\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3)$ with $\mathsf{match}_{1/7}(X, Y) = 1$.

D.2.2. From Unordered to Ordered Tuples

The main component of our clustering algorithm is to reorder the unordered tuples in a way that is not influenced by adding or removing a single tuple. Note that without privacy, such a reordering can be easily done by picking an arbitrary tuple X, and reorder every tuple Y according to it, as describe in the following definition.

Definition D.16. *For* $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k), Y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k) \in (\mathbb{R}^d)^k$ *with* $\mathsf{match}_1(X, Y) = 1$*, define* $\mathsf{ord}_X(Y) := (\boldsymbol{y}_{\pi(1)}, \ldots, \boldsymbol{y}_{\pi(k)})$*, where* $\pi\colon [k] \to [k]$ *is the (unique) permutation such that* $\mathsf{match}_1^{\pi}(X, Y) = 1$ *(i.e.,* $\forall i \in [k] : \pi(i) = \operatorname{argmin}_{j \in [k]}\{\left\|\boldsymbol{x}_i - \boldsymbol{y}_j\right\|\}$*).*

The following claim implies that picking one of the tuples and ordering the others according to it, is actually safe when the database is friendly. In other words, the claim states that for a $\mathsf{match}_{1/7}$-friendly database, every two tuples must induce the same reordering of the other tuples (up to a permutation).

Claim D.17. *For any* $\mathsf{match}_{1/7}$*-friendly* $\mathcal{S} \in ((\mathbb{R}^d)^k)^*$ *and any* $X, Y \in \mathcal{S}$*, there exists a permutation* $\pi\colon [k] \to [k]$ *(depends only on* X, Y*) such that for all* $Z \in \mathcal{S}$*, the tuples* $\widetilde{Z} = \mathsf{ord}_X(Z)$ *and* $\widetilde{Z}' = \mathsf{ord}_Y(Z)$ *satisfy for all* $i \in [k]$ *that* $\widetilde{Z}_{\pi(i)} = \widetilde{Z}'_i$*.*

Proof. Fix $X, Y, Z \in \mathcal{S}$. By Claim D.15 it holds that $\mathcal{D}$ is $\mathsf{match}_{1/3}$-complete. In particular, there exists permutations π_1, π_2, π_3 such that $\mathsf{match}_{1/3}^{\pi_1}(X, Z) = \mathsf{match}_{1/3}^{\pi_2}(Y, Z) = \mathsf{match}_{1/3}^{\pi_3}(X, Y) = 1$. First, this implies that $\mathsf{ord}_X(Z) = (Z_{\pi_1(i)})_{i=1}^k$ and $\mathsf{ord}_Y(Z) = (Z_{\pi_2(i)})_{i=1}^k$. Second, by applying Claim D.14 on the fact that $\mathsf{match}_{1/3}^{\pi_1}(X, Z) = \mathsf{match}_{1/3}^{\pi_2}(Y, Z) = 1$, we obtain that $\mathsf{match}_1^{\pi_2 \circ \pi_1^{-1}}(X, Y) = 1$. Since it also holds that $\mathsf{match}_{1/3}^{\pi_3}(X, Y) = 1$, we conclude that $\pi_3 = \pi_2 \circ \pi_1^{-1}$, and the claim follows by setting $\pi = \pi_3$ (which only depends on X, Y). $\square$

We now use Claim D.17 in order to construct an $\mathsf{match}_{1/7}$-friendly zCDP algorithm for unordered tuples that applies a zCDP algorithm for ordered tuples (i.e., it reduces the unordered tuples problem to the ordered ones).

Algorithm D.18 (FriendlyReorder)**.**
Input: A database $\mathcal{D} = (X^1, \ldots, X^n) \in ((\mathbb{R}^d)^k)^*$.
Operation:

1. *If* $\mathcal{D}$ *is empty, output* $\mathsf{A}(D)$*. Otherwise:*
2. *Sample a uniformly random permutation* $\pi\colon [k] \to [k]$.
3. *For* $i \in [n]$ *let* $(\boldsymbol{y}_1^i, \ldots, \boldsymbol{y}_k^i) = \mathsf{ord}_{X^1}(X^i)$ *and let* $\widetilde{Y}^i = (\boldsymbol{y}_{\pi(1)}^i, \ldots, \boldsymbol{y}_{\pi(k)}^i)$.
4. *Output* $\widetilde{\mathcal{D}} = (\widetilde{Y}^1, \ldots, \widetilde{Y}^n)$.

Claim D.19 (Privacy of FriendlyReorder)**.** *If* A *(algorithm for ordered tuples) is* (ρ, δ)*-zCDP then* $\mathsf{B}(\mathcal{D}) := \mathsf{A}(\mathsf{FriendlyReorder}(\mathcal{D}))$ *is* $\mathsf{match}_{1/7}$*-friendly* (ρ, δ)*-zCDP.*

Proof. Fix neighboring databases $\mathcal{D} = (X^1, \ldots, X^n) \in ((\mathbb{R}^d)^k)^*$ and $\mathcal{D}' = \mathcal{D}_{-j}$ such that $\mathcal{D} \cup \mathcal{D}'$ is $\mathsf{match}_{1/7}$-friendly. For a permutation $\pi\colon [k] \to [k]$ let $\mathsf{FriendlyReorder}_{\pi}$ be algorithm $\mathsf{FriendlyReorder}$ where the permutation chosen in Step 2 is set to π (and not chosen uniformly at random). We prove the claim by showing that for every permutation π there exists a permutation π' such that $\mathsf{A}(\mathsf{FriendlyReorder}_{\pi}(\mathcal{D})) \approx_{\rho,\delta} \mathsf{A}(\mathsf{FriendlyReorder}_{\pi'}(\mathcal{D}'))$.

If $j \neq 1$ (i.e., the first tuple in $\mathcal{D}$ and $\mathcal{D}'$ is X^1), then for every permutation π, the resulting database $\widetilde{\mathcal{D}}$ in $\mathsf{FriendlyReorder}_{\pi}(\mathcal{D})$ and the corresponding database $\widetilde{\mathcal{D}}'$ in $\mathsf{FriendlyReorder}_{\pi}(\mathcal{D}')$ are neighboring (in particular, $\widetilde{\mathcal{D}}' = \widetilde{\mathcal{D}}_{-j}$), and we deduce that the outputs (after applying A) are (ρ, δ)-indistinguishable since A is (ρ, δ)-zCDP.

Otherwise, $\mathcal{D}' = (X^2, \ldots, X^n)$. Since $\mathcal{D}$ is $\mathsf{match}_{1/7}$-friendly, Claim D.17 implies that there exists a permutation $\sigma\colon [k] \to [k]$ such that for all $i \in [n] \setminus \{1\}$, the tuple $(\boldsymbol{y}_1^i, \ldots, \boldsymbol{y}_k^i) = \mathsf{ord}_{X^1}(X^i)$ satisfy $(\boldsymbol{y}_{\sigma(1)}^i, \ldots, \boldsymbol{y}_{\sigma(k)}^i) = \mathsf{ord}_{X^2}(X^i)$. In the following, fix a permutation $\pi\colon [k] \to [k]$, and define $\pi' = \pi \circ \sigma^{-1}$. Then it holds that the resulting database $\widetilde{\mathcal{D}}$ in $\mathsf{FriendlyReorder}_{\pi}(\mathcal{D})$ and the corresponding database $\widetilde{\mathcal{D}}'$ in $\mathsf{FriendlyReorder}_{\pi'}(\mathcal{D}')$ are neighboring (in particular, $\widetilde{\mathcal{D}}' = \widetilde{\mathcal{D}}_{-1}$), and conclude that $\mathsf{A}(\mathsf{FriendlyReorder}_{\pi}(\mathcal{D})) \approx_{(\rho,\delta)} \mathsf{A}(\mathsf{FriendlyReorder}_{\pi'}(\mathcal{D}'))$. $\square$

D.2.3. Ordered k-Tuple Clustering

In this section we are given a database $\mathcal{D} = (X^1, \ldots, X^n) \in ((\mathbb{R}^d)^k)^*$ where each $X^i = (\boldsymbol{x}_1^i, \ldots, \boldsymbol{x}_k^i)$ is a k-tuple, and the goal is to estimate the averages in each coordinates of the tuples. That is, to estimate $(\mathrm{Avg}(\mathcal{D}^1), \ldots, \mathrm{Avg}(\mathcal{D}^k))$ where $\mathcal{D}^j = (\boldsymbol{x}_j^i)_{i=1}^n$. We present an algorithm that given an (utility) advice of values $r_1, \ldots, r_k \geq 0$ such that for all $j \in [k]$ and $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{D}^j$ it holds that $\|\boldsymbol{x} - \boldsymbol{y}\| \leq r_j$, it estimate each $\mathrm{Avg}(\mathcal{D}^j)$ up to an additive error of $\tilde{O}\left(\frac{r_j}{n} \cdot \sqrt{\frac{d}{\rho}}\right)$. The diameters advice are computed in a private preprocessing step.

Note that this problem can be trivially solved by applying our average algorithm (Appendix D.1) on each set $\mathcal{D}^j$. This however, requires k invocations of FriendlyCore (one per average), which requires $n = \Omega(k \log(1/\min\{\beta, \delta\})/\rho)$ (i.e., n is linearly dependent in k). In this section we show how to solve it using a single invocation of FriendlyCore with the following extension of the predicate dist_r for pairs over $\mathbb{R}^d$ to $\mathsf{dist}_{r_1,\ldots,r_k}$ for pairs over $(\mathbb{R}^d)^k$.

Definition D.20 (Predicate $\mathsf{dist}_{r_1,\ldots,r_k}$)**.** *For $r_1, \ldots, r_k$ and $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k), Y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k) \in (\mathbb{R}^d)^k$, we let $\mathsf{dist}_{r_1,\ldots,r_k}(X, Y) = \prod_{i=1}^k \mathsf{dist}_r(\boldsymbol{x}_i, \boldsymbol{y}_i)$.*

Algorithm D.21 (FriendlyOrdTupAvg)**.**
Input: A database $\mathcal{D} = (X^i = (\boldsymbol{x}_1^i, \ldots, \boldsymbol{x}_k^i))_{i=1}^n$ of ordered *tuples, privacy parameters $\rho, \delta > 0$, and diameters $r_1, \ldots, r_k \geq 0$.*
Operation:

1. *Let $\rho_1 = 0.1(1-\delta)\rho$ and $\rho_2 = 0.9\rho$.*
2. *Compute $\hat{n} = n - \sqrt{\frac{\ln(1/\delta)}{\rho_1}} - 1 + \mathcal{N}(0, \frac{1}{2\rho_1})$, where $n = |\mathcal{D}|$.*
3. *If $n = 0$ or $\hat{n} \leq 0$, output $\perp$ and abort.*
4. *Otherwise, for $j \in [k]$:*
   - *Let $\mathcal{D}^j = (\boldsymbol{x}_j^i)_{i=1}^n$.*
   - *Compute $\hat{\boldsymbol{a}}^j = \mathrm{Avg}(\mathcal{D}^j) + \mathcal{N}(0, \sigma^2 \cdot \mathbb{I}_{d \times d})$, for $\sigma = \frac{2r_j}{n} \cdot \sqrt{\frac{k}{2\rho_2}}$.*
5. *Output $(\hat{\boldsymbol{a}}^1, \ldots, \hat{\boldsymbol{a}}^n)$.*

Claim D.22 (Privacy of FriendlyOrdTupAvg)**.** *Algorithm FriendlyOrdTupAvg$(\cdot, \rho, \delta, r_1, \ldots, r_k)$ is $\mathsf{dist}_{r_1,\ldots,r_k}$-friendly (ρ, δ)-zCDP.*

Proof. Let $\mathcal{D} = (X_1, \ldots, X_n)$ and $\mathcal{D}' = \mathcal{D}_{-j}$ be two $\mathsf{dist}_{r_1,\ldots,r_k}$-friendly neighboring databases, and let $n' = n - 1$. Consider two independent random executions of FriendlyAvg$(\mathcal{D})$ and FriendlyAvg$(\mathcal{D}')$ (both with the same input parameters $\rho, \delta, r_1, \ldots, r_k$). Let $\widehat{N}$ and $\widehat{A} = (\widehat{A}^1, \ldots, \widehat{A}^k)$ be the (r.v.'s of the) values of $\hat{n}$ and $(\hat{\boldsymbol{a}}^1, \ldots, \hat{\boldsymbol{a}}^k)$ in the execution FriendlyAvg$(\mathcal{D})$, let $\widehat{N}'$ and $\widehat{A}'$ be these r.v.'s w.r.t. the execution FriendlyAvg$(\mathcal{D}')$, and let ρ_1, ρ_2 be as in Step 1. As done in the proof of Claim B.4, it is enough to prove that $\widehat{A}|_{\widehat{N}=\hat{n}} \approx_{\rho_2} \widehat{A}'|_{\widehat{N}'=\hat{n}}$ for every $\hat{n} \leq n$. In particular, it is enough to prove that for every $j \in [k]$ it holds that $\widehat{A}^j|_{\widehat{N}=\hat{n}} \approx_{\rho_2/k} \widehat{A}'^j|_{\widehat{N}'=\hat{n}}$. Since $\mathcal{D} \cup \mathcal{D}'$ is $\mathsf{dist}_{r_1,\ldots,r_k}$-friendly, for every j it holds that $\mathcal{D}^j \cup (\mathcal{D}^j)'$ is dist_{r_j}-friendly. Hence, using the same arguments as in the proof of Claim B.4, it holds that $\left\|\mathrm{Avg}(\mathcal{D}^j) - \mathrm{Avg}((\mathcal{D}^j)')\right\| \leq 2r_j/n \leq 2r_j/\hat{n}$. Hence, by the properties of the Gaussian mechanism (Theorem A.14) we conclude that $\widehat{A}^j|_{\widehat{N}=\hat{n}} \approx_{\rho_2/k} \widehat{A}'^j|_{\widehat{N}'=\hat{n}}$, as required. □

We now present our main zCDP algorithm for averaging ordered k-tuples, that is based on finding a friendly core of such tuples, and applying the friendly algorithm FriendlyOrdTupAvg.

Algorithm D.23 (FC_AvgOrdTup)**.**
Input: A database $\mathcal{D} = (X^i = (\boldsymbol{x}_1^i, \ldots, \boldsymbol{x}_k^i))_{i=1}^n \in ((\mathbb{R}^d)^k)^*$, privacy parameters $\rho, \delta > 0$, a confidence parameter $\beta > 0$ and lower and upper bounds $r_{\min}, r_{\max} > 0$ on the diameters (respectively).
Operation:

1. *Let $\rho_1 = \rho_2 = 0.05\rho$, $\rho_3 = 0.9\rho$ and $b = 1.5$.*
2. *For $j \in [k]$:*
   - *Let $\mathcal{D}^j = (\boldsymbol{x}_j^i)_{i=1}^n$.*
   - *Compute r_j = FindDiam$(\mathcal{D}^j, \rho_1/k, \beta/(2k), r_{\min}, r_{\max}, b)$ (Algorithm D.5).*
3. *Compute $\mathcal{C}$ = FriendlyCore$(\mathcal{D}, \mathsf{dist}_{r_1,\ldots,r_k}, \rho_2, \delta/2, \beta/2)$.*
4. *Output FriendlyOrdTupAvg$(\mathcal{C}, \rho_3, \delta/2, r_1, \ldots, r_k)$ (Algorithm D.21).*

Claim D.24 (Privacy of FC_AvgOrdTup)**.** *Algorithm FC_AvgOrdTup$(\cdot, \rho, \delta, \beta, r_{\min}, r_{\max})$ is (ρ, δ)-zCDP.*

Proof. By Claim D.6, each execution of FindDiam$(\cdot, \rho_1/k, \beta/(2k), r_{\min}, r_{\max}, b = 1.5)$ is ρ_1/k-zCDP, and therefore the computation of $r_1, \ldots, r_k$ is $(\rho_1, \delta/2)$-zCDP. Since FriendlyOrdTupAvg$(\cdot, \rho_3, \delta/2, r_1, \ldots, r_k)$ is $\mathsf{dist}_{r_1,\ldots,r_k}$-friendly ρ_3-zCDP (Claim D.22), we deduce by the privacy guarantee of the FriendlyCore paradigm (Theorem C.8) that Steps 3+4 are $(\rho_2 + \rho_3, \delta)$-zCDP. We now conclude by composition that the entire computation is

$(\rho = \rho_1 + \rho_2 + \rho_3, \delta)$-zCDP. □

D.2.4. Unordered k-Tuple Clustering: Putting All Together

Now that we have the reduction FriendlyReorder from unordered to ordered k-tuples (for friendly databases), and given our algorithm FC_AvgOrdTup for ordered k-tuple clustering, we describe the fully end-to-end zCDP algorithm for unordered k tuple clustering.

Algorithm D.25 (FC_kTupleClustering)**.**
Input: A database $\mathcal{D} = (X^i = (\boldsymbol{x}_1^i, \ldots, \boldsymbol{x}_k^i))_{i=1}^n \in ((\mathbb{R}^d)^k)^*$*, privacy parameters* $\rho, \delta > 0$*, a confidence parameter* $\beta > 0$ *and lower and upper bounds* $r_{\min}, r_{\max} > 0$ *on the diameters (respectively).*
Operation:

- *Compute* $\mathcal{C} = \mathsf{FriendlyCore}(\mathcal{D}, \mathsf{match}_{1/7}, \rho/2, \delta/2, \beta/2)$.
- *Compute* $\tilde{\mathcal{C}} = \mathsf{FriendlyReorder}(\mathcal{C})$ *(Algorithm D.18).*
- *Output* $\mathsf{FC\_AvgOrdTup}(\tilde{\mathcal{C}}, \rho/2, \delta/2, \beta/2, r_{\min}, r_{\max})$ *(Algorithm D.23).*

Claim D.26 (Privacy of FC_kTupleClustering)**.** *Algorithm* $\mathsf{FC\_kTupleClustering}(\cdot, \rho, \delta, \beta, r_{\max}, r_{\min})$ *is* (ρ, δ)-zCDP.

Proof. Since $\mathsf{A} = \mathsf{FC\_AvgOrdTup}(\cdot, \rho/2, \delta/2, \beta, r_{\max}, r_{\min})$ is $(\rho/2, \delta/2)$-zCDP (Claim D.24), we deduce by Claim D.19 that $\mathsf{A}(\mathsf{FriendlyReorder}(\cdot))$ is $\mathsf{match}_{1/7}$-friendly $(\rho/2, \delta/2)$-zCDP. Hence, we conclude by Theorem C.8 that the output is (ρ, δ)-zCDP. □

D.2.5. FriendlyCore Clustering

Given algorithm FC_kTupleClustering, we now can plug it into the reduction of (Cohen et al., 2021) from standard clustering problems into the k tuple clustering, for obtaining our final clustering method FC_Clustering (described below). In this section we only prove its privacy guarantee, where we refer to Appendix H.3 for the utility guarantees of FC_kTupleClustering and of FC_Clustering for k-means and k-GMM under common separation assumptions (which follow by the tools of (Cohen et al., 2021)).

Algorithm D.27 (NoisyLloydStep)**.**
Input: A database $\mathcal{D} \in (\mathbb{R}^d)^*$*, a* k*-tuple* $Y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k) \in (\mathbb{R}^d)^k$*, privacy parameters* $\rho, \delta > 0$*, and a bound* Λ *on the* ℓ_2 *norm of the points.*
Operation:

1. *Remove all* $\boldsymbol{x} \in \mathcal{D}$ *with* $\|\boldsymbol{x}\| > \Lambda$ *.*
2. *For* $i \in [k]$*:*
   (a) *Let* $\mathcal{D}^i = (\boldsymbol{x} \in \mathcal{D} \colon i = \operatorname{argmin}_{j \in [k]} \|\boldsymbol{x} - \boldsymbol{y}_j\|)$.
   (b) *Compute* $\hat{a}_i = \mathsf{FriendlyAvg}(\mathcal{D}^i, \rho, \delta, r = 2\Lambda)$ *(Algorithm B.3).*
3. *Output* $(\hat{a}_1, \ldots, \hat{a}_k)$.

Algorithm D.28 (FC_Clustering)**.**
Input: A database $\mathcal{D} \in (\mathbb{R}^d)^*$*, privacy parameters* $\rho, \delta > 0$*, a confidence parameter* $\beta > 0$*, a lower bound* $r_{\min} > 0$ *on the diameters of the clusters, a bound* $\Lambda > 0$ *on the* ℓ_2 *norm of the points, and a parameter* $t \in \mathbb{N}$ *(number of tuples).*
Oracle: Non private clustering algorithm A.
Operation:

1. *Shuffle the order of the points in* $\mathcal{D}$*. Let* $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ *be the database after the shuffle.*
2. *Let* $m = \lfloor n/t \rfloor$.
3. *For* $i \in [t]$*: Compute the* k*-tuple* $X^i = \mathsf{A}(\mathcal{D}^i)$ *for* $\mathcal{D}^i = (\boldsymbol{x}_{(i-1)\cdot m+1}, \ldots, \boldsymbol{x}_{i \cdot m})$.
4. *Let* $\mathcal{T} = (X^1, \ldots, X^t)$.
5. *Compute* $Y = \mathsf{FC\_kTupleClustering}(\mathcal{T}, \rho/2, \delta/2, \beta, r_{\min}, 2\Lambda)$.
6. *Output* $\mathsf{NoisyLloydStep}(\mathcal{D}, Y, \rho/2, \delta/2, \Lambda)$.

Claim D.29 (Privacy of FC_Clustering)**.** *Algorithm* $\mathsf{FC\_Clustering}^{\mathsf{A}}(\cdot, \rho, \delta, \beta, r_{\min}, \Lambda, t)$ *is* (ρ, δ)-zCDP *(for any* A*).*

Proof. First, note that for every $Y \in (\mathbb{R}^d)^k$, algorithm $\mathsf{NoisyLloydStep}(\cdot, Y, \rho, \delta, \beta, r_{\min}, r_{\max})$ is ρ-zCDP. This is because $\mathsf{FC\_Avg\_UnknownDiam}(\cdot, \rho, \delta, \beta, r_{\min}, r_{\max})$ is (ρ, δ)-zCDP and for every neighboring databases $\mathcal{D}$ and $\mathcal{D}'$, there is only a single i such that the databases $\mathcal{D}^i$ and $\mathcal{D}'^i$ from Step 2a of $\mathsf{NoisyLloydStep}(\mathcal{D}, \ldots)$ and $\mathsf{NoisyLloydStep}(\mathcal{D}', \ldots)$ (respectively) are neighboring, and the others equal to each other.

Back to FC_Clustering, we obtain the required privacy by composition of FC_kTupleClustering and NoisyLloydStep. □

E. Empirical Results (Extended)

In this section we present empirical results of our FriendlyCore based averaging and clustering algorithms. In all experiments we used privacy parameter $\rho = 1$, $\delta = 10^{-8}$, and all of them were tested in a MacBook Pro Laptop with 4-core Intel i7 CPU with 2.8GHz, and with 16GB RAM.

E.1. Averaging

We tested mean estimation of samples from a Gaussian with *unknown* mean and *known* variance. We compared a Python implementation of our private averaging algorithm FC_Avg with the algorithm CoinPress of (Biswas et al., 2020). The implementations of CoinPress, and the experimental test bed, were taken from the publicly available code of (Biswas et al., 2020) provided at https://github.com/twistedcubic/coin-press. Following (Biswas et al., 2020), we generate a dataset of n samples from a d-dimensional Gaussian $\mathcal{N}(0, I_{d\times d})$. We ran FC_Avg with $r = \sqrt{2}(\sqrt{d} + \sqrt{\ln(100n)})$ for guaranteeing that almost all pairs of samples have ℓ_2 distance at most r from each other (computed according to the known variance).

Algorithm CoinPress requires a bound R on the ℓ_2 norm of the unknown mean. Both algorithms perform a similar final private averaging step that has dependence on $\sqrt{d}$. But they differ in the "preparation:" CoinPress has inherent dependence on d and R. FC_Avg preparation, on the other hand, has no dependence on d or R.

Following (Biswas et al., 2020) we perform 50 repetitions of each experiment and use the trimmed average of values between the 0.1 and 0.9 quantiles. We show the ℓ_2 error of our estimate on the Y-axis. Figure 7(1) reports the effect of varying the bound R, with fixed $d = 1000$ and $n = 800$. We tested CoinPress with 4, 20 and 40 iterations. We observe that FC_Avg, that does not depend on R, outperforms CoinPress for $R > 10^7$. Figure 7(2) reports the effect of varying the dimension d, with fixed $n = 800$ and $R = 10\sqrt{d}$. We tested CoinPress with 2, 4 and 8 iterations. We observed that the performance of all algorithms deteriorates with increasing d, which is expected due to all algorithms using private averaging, but CoinPress deteriorates much faster in the large-d regime.

Finally we note that CoinPress slightly performs better than FC_Avg in the small-d small-R regime (see Figure 7(3) that includes also a comparison to the algorithm of (Karwa & Vadhan, 2018)). The reason is that FriendlyAvg (Algorithm B.3), which is the last step of FC_Avg, uses noise of magnitude $\approx \frac{2r}{n\sqrt{2\rho}}$ which is far by a factor of 2 from the ideal magnitude that we could hope for.

E.2. Clustering

We tested the performance of our private clustering algorithm FC_Clustering with $t = 200$ tuples on a number of k-Means and k-GMM tasks. We compared a Python implementation of FC_Clustering with a recent algorithm of Chang & Kamath (2021) that is based on recursive locality-sensitive hashing (LSH). We denote their algorithm by LSH_Clustering. The implementations of LSH_Clustering, and the experimental test bed of Figure 8, were taken from the publicly available code of (Chang & Kamath, 2021) provided at https://github.com/google/differential-privacy/tree/main/learning/clustering. LSH_Clustering guarantees privacy in the DP model. Therefore, in order to compare it with our $(\rho = 1, \delta)$-zCDP guarantee, we chose to apply it with a $(\varepsilon = 2, \delta)$-DP guarantee, so that non of the guarantees implies the other. Furthermore, unlike FC_Clustering which may fail to produce centers in some cases (e.g., when the core of tuples is empty or close to be empty), LSH_Clustering always produces centers. Therefore, in order to handle failures of FC_Clustering, we used only $\rho = 0.99$ privacy budget, and on failures we executed LSH_Clustering with $\varepsilon = \sqrt{0.02}$ (which implies $\rho = 0.01$ zCDP) as backup.

We performed 30 repetitions of each experiment and present the medians (points) along with the 0.1 and 0.9 quantiles.

In Figure 8 (Left) we present a comparison in dimension $d = 2$ with $k = 8$ clusters. In each repetition, we sampled eight random centers $\{\boldsymbol{c}_i\}_{i=1}^8$ from the unit ball, and the database was obtained by collecting $n/8$ samples from each Gaussian $\mathcal{N}(\boldsymbol{c}_i,\ 0.0221 I_{2\times 2})$, where the samples were clipped to ℓ_2 norm of 1. For FC_Clustering we used an oracle access to k-means++ provided by the KMeans algorithm of the Python library sklearn, and used $r_{\min} = 0.001$ and radius $\Lambda = 1$. We set the radius parameter of LSH_Clustering to 1. We plotted the normalized k-means loss that is computed by $1 - X/Y$, where X is the cost of k-means++ on the entire data, and Y is the cost of the tested private algorithm. From this experiment we observed that for small values of n, FC_Clustering fails often, which yield an inaccurate results. Yet, increasing n also increases the success probability of FC_Clustering which yields very accurate results, while LSH_Clustering stay behind. See Figure 8 (Right) for a graphical illustration of the centers in one of the iterations for $n = 2e5$.

In Figure 9 (Left) we present a comparison for separating $n = 2.5 \cdot 10^5$ samples from a uniform mixture of $k = 5$ Gaussians $\mathcal{N}(\boldsymbol{c}_i,\ I_{d\times d})$ for varying d. In each repetition, each of the $\boldsymbol{c}_i$'s was chosen uniformly from $\{1, 2\}^d$, yielding that the distance between each pair of centers is $\approx \sqrt{d/2}$.

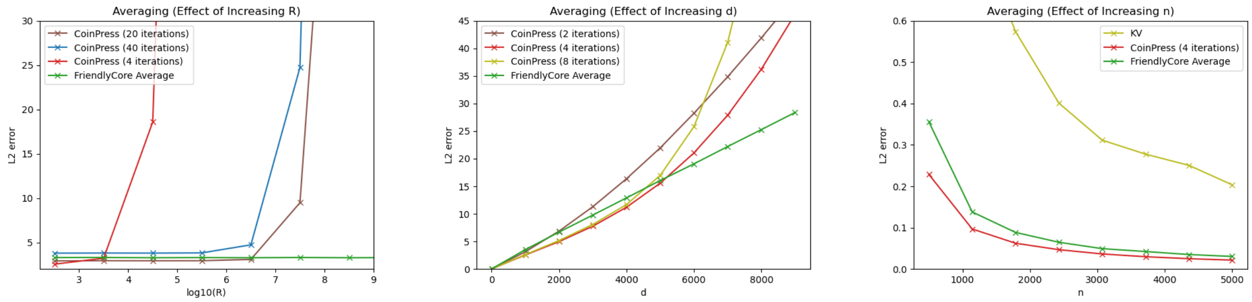

Figure 7: From Left to Right: (1) Averaging in $d = 1000$ and $n = 800$, varying R. (2) Averaging in $n = 800$, $R = 10\sqrt{d}$, varying d. (3) Averaging in $d = 50$ and $R = 10\sqrt{d}$, varying n.

We analyze the labeling accuracy, which is computed by finding the best permutation that fits between the true labeling and the induced clustering, and plotted the labeling failure of the best fit. Here, we used $r_{\min} = 0.1$, and radius $\Lambda = 10\sqrt{d}$ for FC_Clustering and LSH_Clustering. For the non-private oracle access of FC_Clustering, we used a PCA-based clustering that easily separate between such Gaussians in high dimension.[8] From this experiment we observed that FC_Clustering takes advantage of the PCA method and succeed well when d is large, while LSH_Clustering is almost oblivious to the increasing d here.

At that point, we showed that FC_Clustering succeed well on well-separated databases, since the results of the non-private algorithm (each is executed on a random piece of data) are very similar to each other in such cases. We next show that such stability can also be achieved on large enough real-world datasets, even when there is no clear separation into k clusters.

In Figure 9 (Right) we used the publicly available dataset of (Fonollosa & Huerta, 2015) that contains the acquired time series from 16 chemical gas sensors exposed to gas mixtures at varying concentration levels. The dataset contains $\approx 8M$ rows, where each row contains 16 sensors' measurements at a given point in time, so we translate each such row into a 16-dimensional point. We compared the clustering algorithms for varying k, where we used $r_{\min} = 0.1$, and radius $\Lambda = 10^5$ for FC_Clustering and LSH_Clustering. We observed that FC_Clustering, with k-means++ as the non-private oracle, succeed well on various k's, except of $k = 5$ in which it fails due to instability of the non-private algorithm.[9]

[8]The algorithm project the points into the k principal components, cluster the points in that low dimension, and then translate the clustering back to the original points and perform a Lloyd step.

[9]There are two different solutions for $k = 5$ that have similar low cost but do not match, yielding that when splitting the data into random pieces, the non-private KMeans choose one of them in one set of part and the other one in the other pieces, and therefore fails.

In summary, we observed from the experiments that when FC_Clustering succeed, it outputs very accurate results. However, FC_Clustering may fail due to instability of the non-private algorithm on random pieces of the database. Hence, it seems that in cases where we have a clear separation or many points, we might gain by combining between FC_Clustering and LSH_Clustering. In this work we chose to spend 0.99 of the privacy budget on FC_Clustering, but other combinations might perform better on different cases.

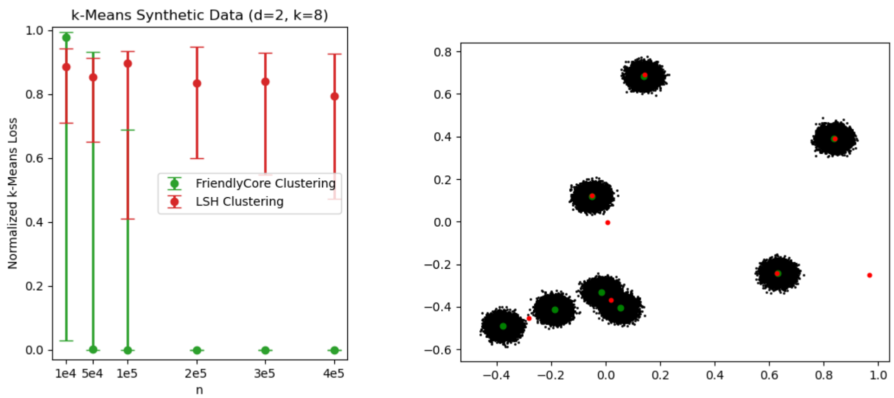

Figure 8: Left: k-means results in $d = 2$ and $k = 8$, for varying n. Right: A graphical illustration of the centers in one of the iterations for $n = 2e5$. Green points are the centers of FC_Clustering and the red points are the centers of LSH_Clustering.

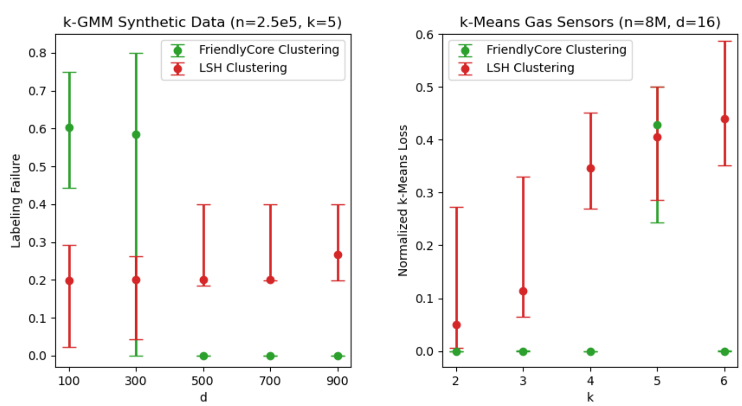

Figure 9: Left: Labeling Failure of samples from a uniform mixture of $k = 5$ Gaussians, varying d. Right: k-means results on Gas Sensors' measurements over time, varying k.

F. Learning a Covariance Matrix

In this section, we are given a database that consists of independent samples from a Gaussian $\mathcal{N}(0, \Sigma)$ where the covariance matrix $\Sigma \succeq 0$ is unknown, no bounds on $\|\Sigma\|$ (the operator norm) are given, and the goal is to privately estimate Σ. Without privacy, it can just be estimated by the empirical covariance of the samples: $\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i \cdot \boldsymbol{x}_i^T$. Recently, three independent and concurrent works of Kamath et al. (2021); Ashtiani & Liaw (2021); Kothari et al. (2021) gave a polynomial-time algorithm for this problem (all the three works were published after the first version of our work that did not include the covariance matrix application). The core of Ashtiani & Liaw (2021)'s construction consists of a framework in the DP model for privately learning average-based aggregation tasks, that has the same flavor of FriendlyCore. Their tool does not output a subset $\mathcal{C} \subseteq \mathcal{D}$ as FriendlyCore. Rather, it outputs a weighted average of the elements, where the weights are chosen in a way that makes the task of privately estimating it to be simpler than its unrestricted counterpart (in particular, their framework guarantees that "outliers" receive weight 0, and by that it is certified that the weighted average has only low sensitivity).[10] For learning a covariance matrix, they implicitly use a special "friendliness" predicate between covariance matrices, and apply their tool on the empirical covariance matrices, each is computed (non-privately) from a different part of the data points.

We next show how to apply FriendlyCore along with the tools of Ashtiani & Liaw (2021) in order to privately learn an unrestricted covariance matrix. Similarly to (Ashtiani & Liaw, 2021), we only handle the case that $\Sigma \succ 0$ (i.e., where Σ has full rank).[11] In addition, following the main step of (Ashtiani & Liaw, 2021), we only show the reduction to the restricted covariance case that is well studied (e.g., see (Biswas et al., 2020; Kamath et al., 2019a)).

We start by defining a predicate and stating key properties from (Ashtiani & Liaw, 2021).

Definition F.1 (Predicate $\mathsf{matrixDist}_\gamma$ (Ashtiani & Liaw, 2021))**.** *For $d \times d$ matrices $\Sigma_1, \Sigma_2 \succeq 0$, let* $\mathsf{matrixDist}(\Sigma_1, \Sigma_2) := \max\left(\left\|\Sigma_2^{-1/2}\Sigma_1\Sigma_2^{-1/2} - I_d\right\|, \left\|\Sigma_1^{-1/2}\Sigma_2\Sigma_1^{-1/2} - I_d\right\|\right)$ *if $\Sigma_1, \Sigma_2 \succ 0$, and ∞ otherwise. For $\gamma \geq 0$ let* $\mathsf{matrixDist}_\gamma(\Sigma_1, \Sigma_2) := \mathbb{1}_{\{\mathsf{matrixDist}(\Sigma_1,\Sigma_2) \leq \gamma\}}$.

The intuition behind the distance measure matrixDist is that it does not scale with the norms of the matrices, i.e., $\mathsf{matrixDist}(\Sigma_1, \Sigma_2) = \mathsf{matrixDist}(\lambda\Sigma_1, \lambda\Sigma_2)$ for any $\lambda > 0$. Therefore, it is useful in our case where we do not have any bounds on the norm of the covariance matrix.

In addition, matrixDist satisfies an approximate triangle inequality (stated bellow).

Lemma F.2 (Approximate triangle inequality (Lemma 7.2 in (Ashtiani & Liaw, 2021)))**.** *If* $\mathsf{matrixDist}(\Sigma_1, \Sigma_2) \leq 1$ *and* $\mathsf{matrixDist}(\Sigma_2, \Sigma_3) \leq 1$ *then* $\mathsf{matrixDist}(\Sigma_1, \Sigma_3) \leq \frac{3}{2} \cdot (\mathsf{matrixDist}(\Sigma_1, \Sigma_2) + \mathsf{matrixDist}(\Sigma_2, \Sigma_3))$.

Note that Lemma F.2 implies that if $\mathcal{D}$ is $\mathsf{matrixDist}_\gamma$-friendly (for $\gamma \leq 1$), then it is $\mathsf{matrixDist}_{3\gamma}$-complete.

The idea now is to apply FriendlyCoreDP with this predicate (using a small constant γ, say 0.1) in order to privately estimate the covariance matrix. At the high-level, this is done by the following process: (1) Split the samples into equal-size parts and compute the empirical covariance matrix of each part. (2) On the resulting database of matrices $\mathcal{T} = (\Sigma_1, \ldots, \Sigma_t)$, apply FriendlyCoreDP for obtaining a core $\mathcal{C} \subseteq \mathcal{T}$ that is certified to be $\mathsf{matrixDist}_{0.1}$-friendly. Then, execute an $\mathsf{matrixDist}_{0.1}$-friendly DP algorithm over the core $\mathcal{C}$.

It is left to design an $\mathsf{matrixDist}_{0.1}$-friendly DP algorithm. The first step is to use the following fact which states that if $\mathcal{T} \cup \mathcal{T}'$ is $\mathsf{matrixDist}_{0.1}$-friendly (and therefore, $\mathsf{matrixDist}_{0.3}$-complete), then $\mathsf{matrixDist}(\mathrm{Avg}(\mathcal{T}), \mathrm{Avg}(\mathcal{T}')) \leq O(1/|\mathcal{T}|)$.

Lemma F.3 (Implicit in (Ashtiani & Liaw, 2021))**.** *There exists a constant $c_1 > 0$ such that the following holds: Let $\gamma > 0$ and let $\Sigma_1, \ldots, \Sigma_n \succ 0$ such that* $\mathsf{matrixDist}(\Sigma_i, \Sigma_j) \leq 0.3$ *for every $i, j \in [n]$. Assuming that $n \geq c_1/\gamma$, then it holds that* $\mathsf{matrixDist}(\frac{1}{n}\sum_{i=1}^{n}\Sigma_i, \frac{1}{n-1}\sum_{i=2}^{n}\Sigma_i) \leq \gamma$.

Next, we define a mechanism B_η such that for any two matrices Σ_1, Σ_2 with $\mathsf{matrixDist}(\Sigma_1, \Sigma_2) \leq \widetilde{O}\left(\frac{\varepsilon\eta}{\sqrt{d\ln(2/\delta)}}\right)$, it holds that $\mathsf{B}_\eta(\Sigma_1) \approx^{\mathsf{DP}}_{\varepsilon,\delta} \mathsf{B}_\eta(\Sigma_2)$.

Lemma F.4 (Lemma 9.1 in (Ashtiani & Liaw, 2021))**.** *For a matrix $\Sigma \succ 0$ and $\eta > 0$, define $\mathsf{B}_\eta(\Sigma) := \Sigma^{1/2}(I + \eta G)(I + \eta G)^T\Sigma^{1/2}$, where G is a $d \times d$ matrix with independent $\mathcal{N}(0, 1)$ entries. Then for every $\eta > 0$, $\varepsilon, \delta \in (0, 1]$, and every matrices $\Sigma_1, \Sigma_2 \succ 0$ such that* $\mathsf{matrixDist}(\Sigma_1, \Sigma_2) \leq \gamma$ *for* $\gamma = \min\{\sqrt{\frac{\varepsilon}{2d(d+1/\eta^2)}}, \frac{\varepsilon}{8d\sqrt{\ln(1/\delta)}}, \frac{\varepsilon}{8\ln(2/\delta)}, \frac{\varepsilon\eta}{12\sqrt{d\ln(2/\delta)}}\}$, *it holds that* $\mathsf{B}_\eta(\Sigma_1) \approx^{\mathsf{DP}}_{\varepsilon,\delta} \mathsf{B}_\eta(\Sigma_2)$.

We now describe our friendly DP algorithm for estimating the mean of covariance matrices.

[10] The framework of Ashtiani & Liaw (2021) has similar ideas to FriendlyCore. However, it is wrapped by a more complicated abstraction, and it is not clear how to apply it for tasks like k-tuple clustering, in which a weighted average is not meaningful. We therefore believe that FriendlyCore, apart of being practical, is also simpler, more intuitive and more general.

[11] The general case can be done by first privately computing the exact subspace using propose-test-release or a private histogram (see (Singhal & Steinke, 2021; Ashtiani & Liaw, 2021)), and then working on the resulting subspace with a full rank matrix.

Algorithm F.5 (FriendlyCovariance).
Input: A database $\mathcal{D} = (\Sigma_1, \ldots, \Sigma_n) \in (\mathbb{R}^{d\times d})^*$ *and parameters* $\varepsilon, \delta, \eta > 0$.
Operation:

1. *Let* $n = |\mathcal{D}|$, *let* $\varepsilon_1 = 0.1\varepsilon$ *and* $\varepsilon_2 = 0.9\varepsilon$, *let* $\gamma = \gamma(\eta, \varepsilon_2, \delta e^{-\varepsilon_1})$ *be the value from Lemma F.4, and let* c_1 *be the constant from Lemma F.3.*
2. *Compute* $\hat{n} = n - \frac{\ln(1/\delta)}{\varepsilon_1} + \mathrm{Lap}(1/\varepsilon_1)$.
3. *If* $n = 0$ *or* $\hat{n} \leq c_1/\gamma$, *output* $\perp$ *and abort.*
4. *Output* $\mathsf{B}_\eta\left(\frac{1}{n}\sum_{i=1}^n \Sigma_i\right)$, *where* B_η *is the algorithm from Lemma F.4.*

Claim F.6 (Privacy of FriendlyCovariance). *Algorithm* $\mathsf{FriendlyCovariance}(\cdot, \varepsilon, \delta, \gamma)$ *is* $\mathsf{matrixDist}_{0.1}$*-friendly* (ε, δ)-DP.

For proving Claim F.6 we use the following simple fact.

Fact F.7. *Let* X, X' *be two random variables over* $\mathcal{X}$. *Assume there exist events* $E, E' \subseteq \mathcal{X}$ *such that: (1)* $\Pr[X \in E] \in e^{\pm\varepsilon_1} \cdot \Pr[X' \in E']$, *and (2)* $X|_E \approx^{\mathsf{DP}}_{\varepsilon_2,\delta} X'|_{E'}$, *and (3)* $X|_{\neg E} \approx^{\mathsf{DP}}_{\varepsilon_2,\delta} X'|_{\neg E'}$. *Then* $X \approx^{\mathsf{DP}}_{\varepsilon_1+\varepsilon_2,\, \delta e^{\varepsilon_1}} X'$.

Proof. Fix an event $T \subseteq \mathcal{X}$ and compute

$$
\begin{aligned}
&\Pr[X \in T]\\
&= \Pr[X \in T \mid E] \cdot \Pr[X \in E] + \Pr[X \in T \mid \neg E] \cdot \Pr[X \notin E]\\
&\leq (e^{\varepsilon_2} \cdot \Pr[X' \in T \mid E'] + \delta) \cdot e^{\varepsilon_1} \cdot \Pr[X' \in E']\\
&+ (e^{\varepsilon_2} \cdot \Pr[X' \in T \mid \neg E'] + \delta) \cdot e^{\varepsilon_1} \cdot \Pr[X' \notin E']\\
&= e^{\varepsilon_1+\varepsilon_2} \cdot \Pr[X' \in T] + \delta e^{\varepsilon_1}.
\end{aligned}
$$

□

We now prove Claim F.6 using Fact F.7.

Proof. Let $\mathcal{D} = (\Sigma_1, \ldots, \Sigma_n)$ and $\mathcal{D}' = \mathcal{D}_{-j}$ be two $\mathsf{matrixDist}_{0.1}$-friendly neighboring databases, and let $n' = n - 1$. Consider two independent random executions of $\mathsf{FriendlyCovariance}(\mathcal{D})$ and $\mathsf{FriendlyCovariance}(\mathcal{D}')$ (both with the same input parameters $\varepsilon, \delta, \eta$). Let $\widehat{N}$ and O be the (r.v.'s of the) values of $\hat{n}$ and the output in the execution $\mathsf{FriendlyAvg}(\mathcal{D})$, let $\widehat{N}'$ and O' be these r.v.'s w.r.t. the execution $\mathsf{FriendlyCovariance}(\mathcal{D}')$, and let $\varepsilon_1, \varepsilon_2$ be as in Step 1. If $n \leq c_1/\gamma$ then $\Pr[O = \perp], \Pr[O' = \perp] \geq 1 - \delta$ and we conclude that $O \approx^{\mathsf{DP}}_{0,\delta} O'$ in this case.

It is left to handle the case $n \geq c_1/\gamma$. Let E be the event $\{O \neq \perp\}$ and E' be the event $\{O' \neq \perp\}$. By construction it is clear that $\Pr[E] \in e^{\pm\varepsilon_1}\Pr[E']$ and that $O|_{\neg E} \equiv O|_{\neg E'}$ (under the conditioning on $\neg E$ and $\neg E'$, it holds that $O = O' = \perp$). By Fact F.7, it suffices to prove that $O|_E \approx^{\mathsf{DP}}_{\varepsilon_2,\, \delta e^{-\varepsilon_1}} O'|_{E'}$ for showing that $O \approx^{\mathsf{DP}}_{\varepsilon,\delta} O'$.

Since $\mathcal{D} \cup \mathcal{D}'$ is $\mathsf{matrixDist}_{0.1}$-friendly, we deduce by approximate triangle inequality (Lemma F.2) that it is $\mathsf{matrixDist}_{0.3}$-complete. Let $\Sigma = \frac{1}{n}\sum_{i\in[n]} \Sigma_i$ and $\Sigma' = \frac{1}{n-1}\sum_{i\in[n]\setminus\{j\}} \Sigma_i$. By Lemma F.3 it holds that $\mathsf{matrixDist}(\Sigma, \Sigma') \leq \gamma$. Since $O|_E = \mathsf{B}_\eta(\Sigma)$ and $O'|_{E'} = \mathsf{B}_\eta(\Sigma')$, we conclude by Lemma F.4 that $O|_E \approx^{\mathsf{DP}}_{\varepsilon_2,\, \delta e^{-\varepsilon_1}} O'|_{E'}$, as required.

□

We now formally describe the end-to-end DP algorithm for covariance estimation:

Algorithm F.8 (FC_Covariance).
Input: A database $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in (\mathbb{R}^d)^*$, *privacy parameters* $\varepsilon, \delta \in (0, 1]$, *and parameters* $t \in \mathbb{N}$ *and* $\eta > 0$.
Operation:

1. *Let* $m = \lfloor n/t \rfloor$.
2. *For* $i \in [t]$: *Compute* $\Sigma_i = \frac{1}{m} \cdot \sum_{j=(i-1)\cdot m+1}^{i\cdot m} \boldsymbol{x}_j \cdot \boldsymbol{x}_j^T$.
3. *Let* $\mathcal{T} = (\Sigma_1, \ldots, \Sigma_t)$.
4. *Compute* $\mathcal{C} = \mathsf{FriendlyCoreDP}(\mathcal{T}, \mathsf{matrixDist}_{0.1}, \alpha = 0)$.
5. *Output* $\mathsf{FriendlyCovariance}(\mathcal{C}, \varepsilon, \delta, \eta)$.

Theorem F.9 (Privacy of FC_Covariance). $\mathsf{FC\_Covariance}(\cdot, \varepsilon, \delta, t)$ *is* $(2(e^\varepsilon - 1), 2\delta e^{\varepsilon+2(e^\varepsilon-1)})$-DP.

Proof. Immediately follows by the privacy guarantee of FriendlyCoreDP (Theorem C.11) because FriendlyCovariance is $\mathsf{matrixDist}_{0.1}$-friendly (ε, δ)-DP (Claim F.6). □

F.1. Utility of FC_Covariance

The following key lemma (Lemma F.10) implies that it is enough to take $n = \tilde{\Omega}(t \cdot (d + \ln(1/\beta)))$ samples in order to guarantee with confidence $1 - \beta/3$ that the database $\mathcal{T}$ is $\mathsf{matrixDist}_{0.1}$-complete, yielding that FriendlyCoreDP takes all matrices into the core.

Lemma F.10 (Lemma 9.3 in (Ashtiani & Liaw, 2021)). *There exists a constant* $c_2 > 0$ *such that the following holds: Let* $m \geq c_2(d + \ln(6/\beta))$, *let* $X_1, \ldots, X_m$ *be i.i.d. samples from* $\mathcal{N}(0, \Sigma)$ *for* $\Sigma \succ 0$, *and let* $\widehat{\Sigma} = \frac{1}{m} \cdot \sum_{i=1}^m X_i X_i^T$. *Then* $\mathsf{matrixDist}(\Sigma, \widehat{\Sigma}) \leq 1/30$ *with probability* $1 - \beta/3$.

In addition, (Ashtiani & Liaw, 2021) proved that choosing $\eta = \Theta\left(\frac{1}{\sqrt{d}+\sqrt{\ln(1/\beta)}}\right)$ suffices for making algorithm B_η accurate, as stated below.

Lemma F.11 (Lemma 9.2 in (Ashtiani & Liaw, 2021))**.** *There exists a constant $c_3 > 0$ such that the following holds: Let $\eta = \frac{1}{c_3(\sqrt{d}+\sqrt{\ln(6/\beta)})}$ and let B_η be the algorithm from Lemma F.4. Then for any $\Sigma \succ 0$ it holds that* $\mathsf{matrixDist}(\mathsf{B}_\eta(\Sigma), \Sigma) \leq 1/30$ *with probability* $1-\beta/3$.

Now note that by Step 3 of FriendlyCovariance, it is required to create at least $t = \Omega\left(\frac{1}{\gamma} + \frac{\ln(1/(\beta\delta))}{\varepsilon}\right)$ matrices in order to fail with probability at most $\beta/3$.

Putting all together, we obtain the following utility guarantee.

Theorem F.12 (Utility of FC_Covariance)**.** *Let c_1, c_2 and c_3 be the constants from Lemmas F.3, F.10 and F.11 (respectively), Let $m = c_2(d + \ln(6/\beta))$, let $\eta = \frac{1}{c_3(\sqrt{d}+\sqrt{\ln(6/\beta)})}$, and let $t = c_1/\gamma + \ln(1/\delta) + \ln(3/\beta)$ where $\gamma = \gamma(\eta, 0.9\varepsilon, \delta e^{-0.1\varepsilon}) = \widetilde{O}\left(\frac{\varepsilon}{d\sqrt{\ln(1/\beta)\ln(1/\delta)}}\right)$ is the value from Lemma F.4. Consider a random execution of* $\mathsf{FC\_Covariance}(\mathcal{D} = (X_1, \ldots, X_n), \varepsilon, \delta, t, \eta)$ *where $n = m \cdot t$ and $X_1, \ldots, X_n$ are i.i.d. samples from $\mathcal{N}(0, \Sigma)$ for $\Sigma \succ 0$. Then with probability $1-\beta$, the output $\widehat{\Sigma}$ satisfy* $\mathsf{matrixDist}(\widehat{\Sigma}, \Sigma) \leq 0.1$.

Note that the overall sample complexity that Theorem F.12 requires is $n = \widetilde{\Omega}\left(\frac{d^2 \ln(1/\beta)^{3/2}\ln(1/\delta)^{1/2}}{\varepsilon}\right)$ which matches the sample complexity of (Ashtiani & Liaw, 2021) (Theorem 9.4).

Proof. Let C and $T = (\Sigma_1, \ldots, \Sigma_t)$ be the (r.v.'s of the) values of $\mathcal{C}$ and $\mathcal{T}$ in the execution. Let E_1 be the event that $\forall i \in [t]:\ \mathsf{matrixDist}(\Sigma_i, \Sigma) \leq 1/30$, and let E_2 be the event that FriendlyCovariance outputs a matrix $\widehat{\Sigma}$ (and not $\perp$). By Lemma F.10 we obtain that $\Pr[E_1] \geq 1-\beta/3$, and in the following we assume it occurs. Note that by approximate triangle inequality (Lemma F.2) we obtain that T is $\mathsf{matrixDist}_{0.1}$-complete, and therefore $C = T$ by the utility of FriendlyCoreDP (Theorem C.11). By definition of t (the size of C in this case) and concentration bound of the Laplace distribution, it holds that $\Pr[E_2 \mid E_1] \geq 1-\beta/3$, and in the following we assume it occurs. Let E_3 be the event that the output $\widehat{\Sigma}$ satisfy $\mathsf{matrixDist}(\widehat{\Sigma}, \mathrm{Avg}(T)) \leq 1/30$. By Lemma F.11 it holds that $\Pr[E_3 \mid E_1 \wedge E_2] \geq 1-\beta/3$, and in the following we assume it occurs. By the convexity of matrixDist (Lemma 7.2 in (Ashtiani & Liaw, 2021)), we deduce that $\mathsf{matrixDist}(\mathrm{Avg}(T), \Sigma) \leq 1/30$. Hence by applying again approximate triangle inequality (Lemma F.2) we conclude that $\mathsf{matrixDist}(\widehat{\Sigma}, \Sigma) \leq 0.1$ whenever $E_1 \wedge E_2 \wedge E_3$ occurs, and the proof of the theorem follows since this event happens with probability $1-\beta$. □

We note that by Theorem F.12, with probability $1-\beta$, FC_Covariance outputs a matrix $\widehat{\Sigma}$ such that $0.9I_d \preceq \widehat{\Sigma}^{-1/2}\Sigma\widehat{\Sigma}^{1/2} \preceq 1.1I_d$. In addition, note that if $X \sim \mathcal{N}(0, \Sigma)$, then $\widehat{\Sigma}^{-1/2}X \sim \mathcal{N}(0, \widehat{\Sigma}^{-1/2}\Sigma\widehat{\Sigma}^{1/2})$. Therefore, we reduced the problem to the restricted covariance case.

G. Computational Efficiency of FriendlyCore

Recall that FriendlyCore computes $f(x, y)$ for all pairs, that is, doing $O(n^2)$ applications of the predicate. However, using standard concentration bounds, it is possible to use a random sample of $O(\log(n/\delta))$ elements y for estimating with high accuracy the number of friends of each x.

In more detail, given a database $\mathcal{D} = (x_1, \ldots, x_n)$, the goal is to efficiently estimate $w_i := \sum_{i=j}^{n} f(x_i, x_j)$ for each $i \in [n]$. For that, we can sample a random m-size subset $\mathcal{S} = (y_1, \ldots, y_m)$ of $\mathcal{D}$ (without replacement). Then, we use the estimations $\tilde{w}_i = \frac{n}{m} \cdot \sum_{j=1}^{m} f(x_i, y_j)$. For making the privacy analysis go through, we need to replace each $z_i = w_i - n/2$ with a value $\tilde{z}_i$ such that for all $i \in [n]$, if $z_i \leq 0$ then $\tilde{z}_i \leq 0$ (except with probability δ). Note that if $z_i \leq 0$ (i.e., $w_i \leq n/2$), then $\sum_{j=1}^{m} f(x_i, y_j)$ (where the y_j's are the random variables) is distributed according to the Hypergeometric distribution $\mathcal{HG}(n, w_i, m)$ which is defined by the number of ones in an m-size random subset of an n-size binary vector with w_i ones. We now use the following tail inequality for Hypergeometric distribution:

Fact G.1 ((Scala, 2009), Equation 10)**.** *Let S be distributed according to $\mathcal{HG}(n, w, m)$. Then*

$$\forall \xi \geq 0: \quad \Pr[S \geq (w/n + \xi) \cdot m] \leq e^{-2\xi^2 m}$$

By Fact G.1, if $w_i \leq n/2$ then $\Pr[\tilde{w}_i \geq (1/2+\xi)n] \leq e^{-2\xi^2 m}$. By taking m of the form $m = \frac{1}{2\xi^2} \cdot \ln(n/\delta)$ we obtain that with probability $1-\delta$, for every $i \in [n]$ it holds that $z_i \leq 0 \implies \tilde{w}_i \leq (1/2+\xi)n$. Therefore, the only change we should do is to replace the value $z_i = w_i - n/2$ (computed in BasicFilter and zCDPFilter) with the value $\tilde{z}_i := \tilde{w}_i - (1/2+\xi)n$. This results with a privacy guarantee that is equivalent to Theorems C.8 and C.11 up to an additional δ in the privacy approximation error. Regarding utility guarantees, in Theorem C.8 (zCDP model), this change means that using the same lower bound on n implies now a utility guarantee for elements with $(1-\alpha+\xi)n$ friends (rather than $(1-\alpha)n$).[12] The parameter ξ determines the trade-of between utility and efficiency: Smaller ξ

[12] In the DP model we slightly need to change the probabilities p_i's in BasicFilter such that $\tilde{z}_i \geq (1/2-\alpha+\xi)n \implies p_i = 1$ rather than $z_i \geq (1/2-\alpha)n \implies p_i = 1$.

requires more computations, but results with a better utility guarantee.

H. Missing Proofs

H.1. Proving Fact A.17

Fact A.17 is an immediate corollary of the following fact.

Fact H.1. *Let $\alpha \in (1, \infty)$, let P and Q be probability distributions over $\mathcal{X}$ with $D_\alpha(P||Q) < \infty$, and let $E \subseteq \mathcal{X}$ be an event. Then it holds that*

$$D_\alpha(P|_E||Q|_E) \leq \frac{1}{P[E]} \cdot D_\alpha(P||Q).$$

Proof. For simplicity we only present the proof for the case that P and Q are discrete, but it can easily be extended to the continuous case as well. Compute

$$\begin{aligned}
&D_\alpha(P||Q)\\
&= \frac{1}{\alpha-1}\ln\left(\sum_{x\in E}\frac{P(x)^\alpha}{Q(x)^{\alpha-1}} + \sum_{x\notin E}\frac{P(x)^\alpha}{Q(x)^{\alpha-1}}\right)\\
&\geq \frac{1}{\alpha-1}P[E]\cdot\ln\left(\frac{1}{P[E]}\cdot\sum_{x\in E}\frac{P(x)^\alpha}{Q(x)^{\alpha-1}}\right)\\
&+ \frac{1}{\alpha-1}P[\neg E]\cdot\ln\left(\frac{1}{P[\neg E]}\cdot\sum_{x\notin E}\frac{P(x)^\alpha}{Q(x)^{\alpha-1}}\right)\\
&= P[E]\cdot D_\alpha(P|_E||Q|_E) + P[E]\cdot\ln\left(\frac{P[E]}{Q[E]}\right)\\
&+ P[\neg E]\cdot D_\alpha(P|_{\neg E}||Q|_{\neg E}) + P[\neg E]\cdot\ln\left(\frac{P[\neg E]}{Q[\neg E]}\right)\\
&= P[E]\cdot D_\alpha(P|_E||Q|_E) + P[\neg E]\cdot D_\alpha(P|_{\neg E}||Q|_{\neg E})\\
&+ D_{KL}(\mathrm{Bern}(P[E])||\,\mathrm{Bern}(Q[E]))\\
&\geq P[E]\cdot D_\alpha(P|_E||Q|_E),
\end{aligned}$$

where the first inequality holds by Jensen's inequality, and D_{KL} denotes the KL-divergence. □

H.2. Utility of Averaging Algorithms

Throughout this section, we use the following definition.

Definition H.2 ((f, α, ℓ)-complete)**.** *A database $\mathcal{D}$ is called (f, α, ℓ)-complete iff there exist at least $n - \ell$ elements $x \in \mathcal{D}$ such that $|\{y \in \mathcal{D} \colon f(x, y) = 1\}| \geq (1 - \alpha)n$. If $\alpha = \ell = 0$ (meaning that $f(x, y) = 1$ for all $x, y \in \mathcal{D}$), we say that $\mathcal{D}$ is f-complete.*

H.2.1. Utility of FriendlyAvg

Claim H.3 (Utility of FriendlyAvg)**.** *The following holds for any $\rho, \beta, \delta > 0$: Let $\mathcal{D} \in (\mathbb{R}^d)^n$ for $n = \Omega\left(\sqrt{\frac{\ln(1/(\beta\delta))}{\rho}}\right)$. Then with probability $1 - \beta$, $\mathsf{FriendlyAvg}(\mathcal{D}, \rho, \delta, r)$ (Algorithm B.3) outputs $\hat{\boldsymbol{a}} \in \mathbb{R}^d$ with $\|\hat{\boldsymbol{a}} - \mathrm{Avg}(\mathcal{D})\| \leq O\left(\frac{r}{n}\cdot\sqrt{\frac{d\ln(1/\beta)}{\rho}}\right)$.*

Proof. Consider a random execution of $\mathsf{FriendlyAvg}(\mathcal{D}, \rho, \delta, r)$, let $\hat{N}$ be the value of $\hat{n}$ in the execution, and let $\hat{A}$ be its output. Assuming that $n \geq 2\cdot\frac{\sqrt{\ln(1/\delta)}+\sqrt{\ln(2/\beta)}}{\sqrt{\rho_1}} + 2$, it holds that $\hat{N} \geq n/2$ with probability $1 - \beta/2$ (holds by Fact A.13). Given that $\hat{N} \geq n/2$, we obtain by Fact A.13 that $\left\|\hat{A} - \mathrm{Avg}(\mathcal{D})\right\| \leq \frac{2r}{n/2}\cdot\sqrt{\frac{d\ln(2/\beta)}{\rho_2}}$ with probability $1 - \beta/2$, as required. □

H.2.2. Utility of FC_Avg_KnownDiam

Claim H.4. *Let $\mathcal{D} \in (\mathbb{R}^d)^n$ be dist_r-complete, for $n \geq \frac{16\cdot\ln(4/(\rho\cdot\max\{\beta,\delta\}))}{\rho}$. Then w.p. $1-\beta$, $\mathsf{FC\_Avg\_KnownDiam}(\mathcal{D}, \rho, \delta, \beta, r)$ estimates $\mathrm{Avg}(\mathcal{D})$ up to an additive error of $O\left(\frac{r}{n}\cdot\sqrt{\frac{d\ln(1/\beta)}{\rho}}\right)$.*

Proof. By applying the utility guarantee of FriendlyCore with $\alpha = 0$ (Theorem C.8) it holds that with probability $1 - \beta/2$, the core $\mathcal{C}$ that FriendlyCore forwards to FriendlyAvg is all $\mathcal{D}$. The proof then follows by the utility guarantee of FriendlyAvg (Claim H.3). □

Claim H.4 can be extended to cases where the database $\mathcal{D}$ is only close to be dist_r-complete, i.e., cases in which we are only given r' that is smaller than the effective diameter r of the database, but still most of the points are close by ℓ_2 distance of r'.

Lemma H.5. *Let $\mathcal{D} \in (\mathbb{R}^d)^n$ be an dist_r-complete database for $\mathsf{dist}_r(\boldsymbol{x}, \boldsymbol{y}) := \mathbb{1}_{\{\|\boldsymbol{x}-\boldsymbol{y}\|\leq r\}}$, and let $r' \leq r$ be such that $\mathcal{D}$ is $(\mathsf{dist}_{r'}, \alpha, \beta)$-complete for $0 \leq \alpha < 1/2$ and $\ell < n/2$. If $n \geq \frac{-4\cdot\ln(1/2\cdot(1/2-\alpha)\rho\max\{\beta,\delta\})}{(1/2-\alpha)^2\rho}$, then w.p. $1 - \beta$ over $\mathsf{FC\_Avg\_KnownDiam}(\mathcal{D}, \rho, \delta, \beta, r')$, the output $\hat{\boldsymbol{a}}$ satisfy $\|\hat{\boldsymbol{a}} - \mathrm{Avg}(\mathcal{D})\| \leq \frac{\ell r}{n} + O\left(\frac{r'}{n}\cdot\sqrt{\frac{d\ln(1/\beta)}{\rho}}\right)$.*

Proof. By applying the utility guarantee of FriendlyCore (Theorem C.8) it holds that with probability $1-\beta/2$, the core $\mathcal{C} \subseteq \mathcal{D}$ that FC_Paradigm forwards to FriendlyAvg in the execution $\mathsf{FC\_Avg\_KnownDiam}(\mathcal{D}, \rho, \delta, \beta, r')$ contains all points $\boldsymbol{x} \in \mathcal{D}$ with $|\{\boldsymbol{y} \in \mathcal{D} \colon \|\boldsymbol{x} - \boldsymbol{y}\| \leq r'\}| \geq (1 - \alpha)n$, which in particular yields that $|\mathcal{C}| \geq n - \ell$. Along with the utility guarantee of FriendlyAvg (Claim H.3), we obtain that the output $\hat{\boldsymbol{a}}$ satisfy $\|\hat{\boldsymbol{a}} - \mathrm{Avg}(\mathcal{C})\| \leq O\left(\frac{r'}{n}\cdot\sqrt{\frac{d\ln(1/\beta)}{\rho}}\right)$. We conclude the proof since $\|\mathrm{Avg}(\mathcal{D}) - \mathrm{Avg}(\mathcal{C})\| \leq \frac{(n-|\mathcal{C}|)\cdot r}{n} \leq \frac{\ell r}{n}$, where the first inequality since $\mathcal{D}$ is dist_r-complete. □

H.2.3. Utility of FC_Avg_UnknownDiam

Claim H.6 (Utility of CheckDiam (Algorithm D.3)). *If $\mathcal{D}$ is dist_r-complete, then* $\mathsf{CheckDiam}(\mathcal{D},\rho,\beta,r)$ *outputs* 1 *w.p.* $1-\beta$. *If $\mathcal{D}$ is* **not** $(\mathsf{dist}_r,\alpha,\ell := \frac{2}{\alpha}\sqrt{\frac{4\ln(1/\beta)}{\rho}})$*-complete for some $\alpha > 0$, then* $\mathsf{CheckDiam}(\mathcal{D},\rho,\beta,r)$ *outputs* 0 *w.p.* $1-\beta$.

Proof. Let $\mathcal{D} = (\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n) \in (\mathbb{R}^d)^*$ and let $\widehat{A}$ be the value of $\hat{a}$ in a random execution of $\mathsf{CheckDiam}(\mathcal{D},\rho,\beta,r)$. If $\mathcal{D}$ is dist_r-complete, then $a = n$, and therefore we deduce by Fact A.13 that $\Pr\left[\widehat{A} \geq n - \sqrt{\frac{4\ln(1/\beta)}{\rho}}\right] = \Pr\left[\mathcal{N}(0,2/\rho) \geq -\sqrt{\frac{4\ln(1/\beta)}{\rho}}\right] \geq 1-\beta$. If $\mathcal{D}$ is not $(\mathsf{dist}_r,\alpha,\ell)$-complete, then there are more than ℓ points $\boldsymbol{x}_i \in \mathcal{D}$ with $s_i < (1-\alpha)n$. Therefore $a = (\sum_{i=1}^n s_i)/n < \frac{(n-\ell)n+\ell\cdot(1-\alpha)n}{n} = n - \alpha\ell = n - 2\sqrt{\frac{4\ln(1/\beta)}{\rho}}$. Hence, we conclude by Fact A.13 that $\Pr\left[\widehat{A} \geq n - \sqrt{\frac{4\ln(1/\beta)}{\rho}}\right] = \Pr\left[\mathcal{N}(0,2/\rho) \geq \sqrt{\frac{4\ln(1/\beta)}{\rho}}\right] \leq \beta$. □

Claim H.7 (Utility of FindDiam (Algorithm D.5)). *Let $\mathcal{D} \in (\mathbb{R}^d)^*$ be an $\mathsf{dist}_{r_{\max}}$-complete database. Then for every $\alpha,\beta > 0$, with probability $1-\beta$ over a random execution of* $\mathsf{FindDiam}(\mathcal{D},\rho,\beta,r_{\max},r_{\min},b)$*, the output r of the execution satisfies that $\mathcal{D}$ is $(\mathsf{dist}_r,\alpha,\ell)$-complete for* $\ell = O\left(\frac{1}{\alpha}\cdot\sqrt{\frac{\log(1/\beta)\log\log(r_{\max}/r_{\min})}{\rho}}\right)$.

Proof. The binary search performs at most $\log_2(t)$ calls to CheckDiam, each returns a "correct" result with probability $1-\beta/\log_2(t)$ (follows by Claim H.6), where "correct" means that if the output for r is 1 then $\mathcal{D}$ is $(\mathsf{dist}_r,\alpha,\ell(r) := \frac{2}{\alpha}\sqrt{\frac{4\ln(1/\beta)\log_2\log_b(r_{\min}/r_{\max})}{\rho}})$-complete, and if the output for r is 0 then $\mathcal{D}$ is not dist_r-complete. Overall, all calls are "correct" with probability $1-\beta$, yielding that the resulting r of the binary search satisfy that $\mathcal{D}$ is $(\mathsf{dist}_r,\alpha,\ell(r))$-complete, as required. □

Claim H.8 (Utility of FC_Avg_UnknownDiam (Algorithm D.7)). *Let $\mathcal{D} \in (\mathbb{R}^d)^n$ be an dist_r-complete database for $r \in [r_{\min}, r_{\max}]$ and* $n = \Omega\left(\frac{\log(1/\min\{\beta,\delta\})}{\rho} + \sqrt{\frac{\log(1/\beta)\log\log(r_{\max}/r_{\min})}{\rho}}\right)$. *Then with probability $1-\beta$ over the execution* $\mathsf{FC\_Avg\_UnknownDiam}(\mathcal{D},\rho,\delta,\beta,r_{\min},r_{\max})$*, the output $\hat{\boldsymbol{a}}$ satisfy* $\|\hat{\boldsymbol{a}} - \mathrm{Avg}(\mathcal{D})\| \leq O\left(\frac{r}{n}\sqrt{\frac{\log(1/\beta)(d+\log\log(r_{\max}/r_{\min}))}{\rho}}\right)$.

Proof. Let $\mathcal{D}$ as in the theorem statement. By the utility guarantee of FindDiam (Claim H.6) it holds that with probability $1-\beta/2$, the resulting r' (the value of r that is computed in Step 2 of FC_Avg_UnknownDiam) satisfy that $\mathcal{D}$ is $(\mathsf{dist}_{r'},0.1,\ell)$-complete for $\ell = O\left(\sqrt{\frac{\log(1/\beta)\log\log(r_{\max}/r_{\min})}{\rho}}\right)$. Given that, we apply the extended utility guarantee of FC_Avg_KnownDiam (Lemma H.5) which yields that with probability $1-\beta/2$, the additive error is at most $\frac{\ell r}{n} + O\left(\frac{r'}{n}\sqrt{\frac{d\log(1/\beta)}{\rho}}\right) = O\left(\frac{r}{n}\sqrt{\frac{\log(1/\beta)(d+\log\log(r_{\max}/r_{\min}))}{\rho}}\right)$, as required. □

H.2.4. Utility of FC_AvgOrdTup

Claim H.9 (Utility of FC_AvgOrdTup (Algorithm D.23)). *Let $\mathcal{D} = (X^i = (\boldsymbol{x}_1^i,\ldots,\boldsymbol{x}_k^i)) \in ((\mathbb{R}^d)^k)^n$ be an $\mathsf{dist}_{r_1,\ldots,r_k}$-complete database for $r_1,\ldots,r_k \in [r_{\min},r_{\max}]$ where* $n = \Omega\left(\frac{\log(1/\min\{\beta,\delta\})}{\rho} + \sqrt{\frac{k\log(k/\beta)\log\log(r_{\max}/r_{\min})}{\rho}}\right)$, *and for $j \in [k]$ let $\mathcal{D}^j = (\boldsymbol{x}_j^i)_{i=1}^n$. Then w.p. $1-\beta$ over the execution* $\mathsf{FC\_AvgOrdTup}(\mathcal{D},\rho,\delta,\beta,r_{\min},r_{\max})$*, the output $(\hat{\boldsymbol{a}}^1,\ldots,\hat{\boldsymbol{a}}^k)$ satisfy for all $j \in [k]$:* $\left\|\hat{\boldsymbol{a}}^j - \mathrm{Avg}(\mathcal{D}^j)\right\| \leq O\left(\frac{r}{n}\sqrt{\frac{k\log(k/\beta)(d+\log\log(r_{\max}/r_{\min}))}{\rho}}\right)$.

The proof holds similarly to Claim H.8 up to the factor k that we loose in the privacy parameter and the confidence parameter, except for the first term in the lower bound on n that does not need to be multiply by k since we only apply FriendlyCore twice and not k times.

H.3. Utility of Clustering Algorithms

In this section we state the utility of our main clustering algorithm FC_Clustering for k-means and k-GMM under common separation assumption, using the reductions of (Cohen et al., 2021) to k-tuple clustering.

H.3.1. Definitions from (Cohen et al., 2021)

We recall from (Cohen et al., 2021) the property of a collection of unordered k-tuples $(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_k) \in (\mathbb{R}^d)^k$, which we call *partitioned by Δ-far balls*.

Definition H.10 (Δ-far balls). *A set of k balls $\mathcal{B} = \{B_i = B(\boldsymbol{c}_i, r_i)\}_{i=1}^k$ over $\mathbb{R}^d$ is called* ***Δ-far balls****, if for every $i \in [k]$ it holds that $\|\boldsymbol{c}_i - \boldsymbol{c}_j\| \geq \Delta\cdot\max\{r_i,r_j\}$ (i.e., the balls are relatively far from each other).*

Definition H.11 (partitioned by Δ-far balls). *A k-tuple $X \in (\mathbb{R}^d)^k$ is partitioned by a given set of k Δ-far balls $\mathcal{B} = \{B_1,\ldots,B_k\}$, if for every $i \in [k]$ it holds that $|X \cap B_i| = 1$. A database k-tuples $\mathcal{D} \in ((\mathbb{R}^d)^k)^*$ is* partitioned by

$\mathcal{B}$, if each $X \in \mathcal{D}$ is partitioned by $\mathcal{B}$. We say that $\mathcal{D}$ is partitioned by Δ-far balls *if such a set $\mathcal{B}$ of k Δ-far balls exists.*

For a database of k-tuples $\mathcal{D} \in ((\mathbb{R}^d)^k)^*$, we let $\text{Points}(\mathcal{D})$ be the collection of all the points in all the k-tuples in $\mathcal{D}$.

Definition H.12 (The points in a collection of k-tuples)**.** *For $\mathcal{D} = ((\boldsymbol{x}_{1,j})_{j=1}^k, \ldots, (\boldsymbol{x}_{n,j})_{j=1}^k) \in ((\mathbb{R}^d)^k)^n$, we define $\text{Points}(\mathcal{D}) = (\boldsymbol{x}_{i,j})_{i\in[n],j\in[k]} \in (\mathbb{R}^d)^{kn}$.*

We now formally define the partition of a database $\mathcal{D} \in ((\mathbb{R}^d)^k)^*$ which is partitioned by Δ-far balls for $\Delta > 3$.

Definition H.13 ($\text{Partition}(\mathcal{D})$)**.** *Given a database $\mathcal{D} \in ((\mathbb{R}^d)^k)^*$ which is partitioned by Δ-far balls for $\Delta > 3$, we define the partition of $\mathcal{D}$, which we denote by $\text{Partition}(\mathcal{D}) = \{\mathcal{P}^1, \ldots, \mathcal{P}^k\}$, by fixing an (arbitrary) k-tuple $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k) \in \mathcal{D}$ and setting $\mathcal{P}^i = (\boldsymbol{x} \in \text{Points}(\mathcal{D})\colon i = \text{argmin}_{j\in[k]}\|\boldsymbol{x} - \boldsymbol{x}_j\|)$.*

Definition H.14 (good-averages solutions)**.** *Let $\mathcal{D} \in ((\mathbb{R}^d)^k)^n$, let $\{\mathcal{P}^1, \ldots, \mathcal{P}^k\} = \text{Partition}(\mathcal{D})$, let $\boldsymbol{a}_i = \text{Avg}(\mathcal{P}^i)$, and let $\alpha, r_{\min} \geq 0$. We say that a k-tuple $Y = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k\} \in (\mathbb{R}^d)^k$ is an* $(\alpha, r_{\min})$-good-averages solution *for clustering $\mathcal{D}$, if there exist radii $r_1, \ldots, r_k \geq 0$ such that $\mathcal{B} = \{B_i = B(\boldsymbol{a}_i, r_i)\}_{i=1}^k$ are Δ-far balls (for $\Delta > 3$) that partitions $\mathcal{D}$, and for every $i \in [k]$ it holds that:*

$$\|\boldsymbol{y}_i - \boldsymbol{a}_i\| \leq \alpha \cdot \max\{r_i, r_{\min}\}$$

For applications, (Cohen et al., 2021) focused on a specific type of algorithms for the k-tuple clustering problems, that outputs a good-averages solution.

Definition H.15 (averages-estimator for k-tuple clustering)**.** *Let A be an algorithm that gets as input a database of unordered tuples in $((\mathbb{R}^d)^k)^*$. We say that A is an* $(n, \alpha, r_{\min}, \beta, \Lambda, \Delta)$-averages-estimator for k-tuple clustering, *if for every $\mathcal{D} \in (B(\mathbf{0}, \Lambda)^k)^* \subseteq ((\mathbb{R}^d)^k)^n$ that is partitioned by Δ-far balls, $\mathsf{A}(\mathcal{D})$ outputs w.p. $1 - \beta$ an $(\alpha, r_{\min})$-good-averages solution $Y \in (\mathbb{R}^d)^k$ for clustering $\mathcal{D}$.*

H.3.2. Utility of FC_kTupleClustering

We next prove that FC_kTupleClustering (Algorithm D.25) is a good averages-estimator for k-tuple clustering.

Claim H.16 (Utility of FC_kTupleClustering)**.** *Algorithm $\mathsf{FC\_kTupleClustering}(\cdot, \rho, \delta, \beta, r_{\min}, r_{\max})$ is an $(n, \alpha = 1, r_{\min}, \beta, \Lambda = r_{\max}/2, \Delta = 10)$-averages-estimator for k-tuple clustering, for*
$$n = \Omega\left(\frac{\log(1/\min\{\beta,\delta\})}{\rho} + \sqrt{\frac{k\log(k/\beta)(d+\log\log(r_{\max}/r_{\min}))}{\rho}}\right).$$

Proof. If $\mathcal{D}$ is partitioned by 10-far balls, then in particular it is $\mathsf{match}_{1/7}$-complete (the predicate from Definition D.16). Therefore, at the first step of FC_kTupleClustering, the core of tuples contains all of $\mathcal{D}$. The proof now immediately follow by the utility guarantee of algorithm FC_AvgOrdTup (Claim H.9). □

H.3.3. Utility of FC_Clustering for k-Means

In the k-means problem, we are given a database $\mathcal{D} \in (\mathbb{R}^d)^*$ and a parameter $k \in \mathbb{N}$, the goal is to compute k centers $C = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k) \in (\mathbb{R}^d)^k$ that minimize $\text{COST}_{\mathcal{D}}(C) := \sum_{\boldsymbol{x}\in\mathcal{D}} \min_{i\in[k]}\|\boldsymbol{x} - \boldsymbol{c}_i\|$ as possible, i.e. close as possible to $\text{OPT}_k(\mathcal{D}) := \min_{C\in(\mathbb{R}^d)^k} \text{COST}_{\mathcal{D}}(C)$.

We state our utility guarantee for databases that are separated according to Ostrovsky et al. (2012).

Definition H.17 ((ϕ, ξ)-separated (Ostrovsky et al., 2012; Cohen et al., 2021))**.** *A database $\mathcal{D} \in (\mathbb{R}^d)^*$ is (ϕ, ξ)-separated for k-means if $\text{OPT}_k(\mathcal{D}) + \xi \leq \phi^2 \cdot \text{OPT}_{k-1}(\mathcal{D})$.*

As shown by (Ostrovsky et al., 2012), for such database with sufficiently small ϕ, any set of centers C that well approximate the k-means cost, must be close in distance to the optimal centers (i.e., there must be a match between the centers). Therefore, by using a good approximation k-means algorithm as an oracle for FC_Clustering, we obtain a guarantee that FC_kTupleClustering succeed to compute a tuple Y that is close to all other non-private algorithm. This property has been used by (Cohen et al., 2021; Shechner et al., 2020) for constructing private clustering for such databases. Here we state the properties of our construction, which follows from Theorem 5.11 in (Cohen et al., 2021) (reduction to k-tuple clustering).

Claim H.18 (Utility of FC_Clustering for k-Means)**.** *Let A be a (non-private) ω-approximation algorithm for k-means (i.e., that always returns centers with cost $\leq \omega\text{OPT}_k$), and let*
$$t = \Omega\left(\frac{\log(1/\min\{\beta,\delta\})}{\rho} + \sqrt{\frac{k\log(k/\beta)(d+\log\log(r_{\max}/r_{\min}))}{\rho}}\right)$$
(the number of tuples that are required by Claim H.16). Then for any $\mathcal{D} \in \mathcal{B}(\mathbf{0}, \Lambda)^n$ that is (ϕ, ξ)-separated for k-means for $\phi \leq \frac{1}{\sqrt{17(1+\omega)}}$ and $\xi = \tilde{\Omega}(\Lambda^2 kdt + \Lambda\sqrt{kdt\omega \cdot \text{OPT}_k(\mathcal{D})})$, algorithm $\mathsf{FC\_Clustering}(\mathcal{D}, \rho, \delta, \beta, r_{\min} = \gamma/n, \Lambda, t)$ outputs with probability $1 - \beta$ centers $C \in (\mathbb{R}^d)^k$ such that $\text{COST}_{\mathcal{D}}(C) \leq (1 + 64\gamma)\text{OPT}_k(\mathcal{D}) + O\big(\Lambda^2 k(d + \log(k/\beta))/\rho\big)$, for $\gamma = 2 \cdot \frac{\omega\phi^2+\phi}{1-\phi}$.

We remark that additive errors in the cost is independent of n, and the additive term ξ in the separation is only logarithmic in n (hidden inside the $\tilde{\Omega}$).

H.3.4. Utility of FC_Clustering for k-GMM

In this section we state the utility guarantee for learning a mixture of well separated and bounded k Gaussians.

The setting is that we are given n samples from a mixture $\{(\mu_1, \Sigma_1, w_1), \ldots, (\mu_k, \Sigma_k, w_k)\}$, i.e., for each sample, one of the Gaussians is chosen w.p. propositional to its weight (the i'th Gaussian is chosen w.p. $w_i / \sum_{j=1}^k w_j$), and then the sample is taken from $\mathcal{N}(\mu_i, \Sigma_i)$ for the chosen i. The goal here is to output a set of k centers $C = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k) \in (\mathbb{R}^d)^k$ which is a perfect classifier: Up to reordering of the $\boldsymbol{c}_i$'s, for every sample $\boldsymbol{x}$ that was drawn from the i'th Gaussian in the mixture, it holds that $i = \operatorname{argmin}_{j \in [k]} \|\boldsymbol{x} - \boldsymbol{c}_i\|$.

As done in previous works (Cohen et al., 2021; Kamath et al., 2019b), we assume that we are given a lower bound $w_{\min}$ on the weights, and a lower and upper bounds $\sigma_{\min}, \sigma_{\max}$ on the norm of each covariance matrix Σ_i. Unlike those works, we do not need to assume a bound R on the ℓ_2 norms of each μ_i.

We use the PCA-based algorithm of (Achlioptas & McSherry, 2005) as the non-private oracle access for $\mathsf{FC\_Clustering}$ given $s = \Omega(k(d + \log(k/\beta))/w_{\min})$ samples from a mixture that has assumed separation

$$\|\mu_i - \mu_j\| \geq \Omega\left(\sqrt{k \log(nk)} + 1/\sqrt{w_i} + 1/\sqrt{w_j}\right) \cdot \max\{\|\Sigma_i\|, \|\Sigma_j\|\}, \tag{3}$$

for every distinct $i, j \in [k]$, outputs a perfect classifier with confidence $1 - \beta$ (note that the separation is independent of d). We now state the utility guarantee of $\mathsf{FC\_Clustering}$ that follows (implicitly) by the proof of Theorem 6.12 in (Cohen et al., 2021) (reduction to k-tuple clustering).

Claim H.19 (Utility of $\mathsf{FC\_Clustering}$ for k-GMM)**.** *Let $\mathcal{D}$ be a set of $n = s \cdot t$ samples from a mixture $\{(\mu_1, \Sigma_1, w_1), \ldots, (\mu_k, \Sigma_k, w_k)\}$ for $t = \Omega\left(\frac{\log(1/\min\{\beta,\delta\})}{\rho} + \sqrt{\frac{k \log(k/\beta)(d + \log\log(r_{\max}/r_{\min}))}{\rho}}\right)$ (the number of tuples that are required by Claim H.16) and $s = \Omega(k(d + \log(kt/\beta))/w_{\min})$ (the numer of samples required by (Achlioptas & McSherry, 2005) for confidence $1 - \beta/t$). Assume that the mixture is separated according to Equation (3), and for each i: $w_i \geq w_{\min}$ and $\sigma_{\min} \leq \|\Sigma_i\| \leq \sigma_{\max}$. Then with probability $1 - 2\beta$, the output of $\mathsf{FC\_Clustering}^{\mathsf{A}}(\mathcal{D}, \rho, \delta, \beta, r_{\min} = 0.1\sigma_{\min}, \Delta = 10\sigma_{\max})$, for A being (Achlioptas & McSherry, 2005)'s algorithm, outputs a perfect classifier.*

H.4. Proving Lemma C.6

In this section we prove the properties of $\mathsf{zCDPFilter}$ (Algorithm C.5), restated below.

Lemma H.20 (Restatement of Lemma C.6)**.** *Let $f \colon \mathcal{X}^2 \to \{0,1\}$ and $\rho, \delta > 0$. $\mathsf{F} = \mathsf{zCDPFilter}(\cdot, f, \rho, \delta)$ is a product-filter that is (f, α, β, n)-complete for every $0 \leq \alpha < 1/2$, $\beta > 0$, and $n \geq \frac{-4 \cdot \ln((1/2-\alpha)\rho \cdot \min\{\beta,\delta\})}{(1/2-\alpha)^2 \rho}$. Furthermore, for every $n \in \mathbb{N}$ and every neighboring databases $\mathcal{D} = (x_1, \ldots, x_n)$ and $\mathcal{D}' = \mathcal{D}_{-j}$, there exist events $E \subseteq \{0,1\}^n$ and $E' \subseteq \{0,1\}^{n-1}$ with $\Pr[\mathsf{F}(\mathcal{D}) \in E], \Pr[\mathsf{F}(\mathcal{D}') \in E'] \geq 1 - \delta$, such that the following holds w.r.t. the random variables $V = \mathsf{F}(\mathcal{D})$ and $V' = \mathsf{F}(\mathcal{D}')$:*

1. *Friendliness: For every $\boldsymbol{v} \in E$ and $\boldsymbol{v}' \in E'$, the database $\mathcal{C} \cup \mathcal{C}'$, for $\mathcal{C} = \mathcal{D}_{\{i \in [n] \colon v_i = 1\}}$ and $\mathcal{C}' = \mathcal{D}'_{\{i \in [n-1] \colon v'_i = 1\}}$, is f-friendly, and*

2. *Privacy: $(V_{-j})|_E \approx_\rho V'|_{E'}$.*

Proof. Fix two neighboring databases $\mathcal{D} = (x_1, \ldots, x_n)$ and $\mathcal{D}' = \mathcal{D}_{-k}$. For simplicity and without loss of generality, we assume that $k = n$, i.e., $\mathcal{D}' = (x_1, \ldots, x_{n-1})$. Consider two independent executions $\mathsf{F}(\mathcal{D})$ and $\mathsf{F}(\mathcal{D}')$ for $\mathsf{F} = \mathsf{zCDPFilter}(\cdot, f, \rho, \delta)$ (Algorithm C.5). Let ρ_1, ρ_2 be as in Step i, let $\boldsymbol{z} = (z_1, \ldots, z_n)$ be the values of these variables in the execution $\mathsf{F}(\mathcal{D})$, and let $\{z'_i\}_{i=1}^{n-1}$ be these values in the execution $\mathsf{F}(\mathcal{D}')$. In addition, let $\widehat{N}, \{\widehat{Z}_i, V_i\}_{i=1}^n$ be the (r.v.'s of) the values of $\hat{n}, \{\hat{z}_i, v_i\}_{i=1}^n$ in the execution $\mathsf{F}(\mathcal{D})$, and let $\widehat{N}', \{\widehat{Z}'_i, V'_i\}_{i=1}^{n-1}$ be these r.v.'s w.r.t. $\mathsf{F}(\mathcal{D}')$.

We first prove that F is (f, n, α, β)-complete (Definition C.2) for every n that satisfy

$$(1/2 - \alpha)n \geq \left(\sqrt{\frac{\tilde{n} \cdot \ln(2\tilde{n}/\delta)}{4\rho_2}} + \sqrt{\frac{\tilde{n} \cdot \ln(2n/\beta)}{4\rho_2}} + \frac{1}{2}\right), \tag{4}$$

for $\tilde{n} = n + \sqrt{\frac{\ln(2/\delta)}{\rho_1}} + \sqrt{\frac{\ln(2/\beta)}{\rho_1}}$ (In particular, this holds for $n \geq \frac{-4 \cdot \ln((1/2-\alpha)\rho \cdot \max\{\beta,\delta\})}{(1/2-\alpha)^2 \rho}$). Fix n that satisfy Equation (4). First, note that by a concentration bound of Gaussians (Fact A.13) it holds that

$$\Pr\left[\widehat{N} > \tilde{n}\right] \leq \beta/2 \tag{5}$$

Second, note that for every i with $\sum_{j=1}^n f(x_i, x_j) = 1 \geq (1 - \alpha)n$ it holds that $z_i \geq (1/2 - \alpha)n$. We deduce that for every such i

$$\begin{aligned}
&\Pr\left[\mathrm{V}_i = 0 \mid \widehat{N} \leq \tilde{n}\right] \\
&= \Pr\left[\widehat{Z}_i < \sqrt{\frac{\widehat{N} \cdot \ln(2\widehat{N}/\delta)}{4\rho_2}} + \frac{1}{2} \mid \widehat{N} \leq \tilde{n}\right] \\
&\leq \Pr\left[\mathcal{N}\left(0, \frac{\widehat{N}}{8\rho_2}\right) < -\sqrt{\frac{\tilde{n} \cdot \ln(2n/\beta)}{4\rho_2}} \mid \widehat{N} \leq \tilde{n}\right] \\
&\leq \beta/2n,
\end{aligned} \tag{6}$$

where the penultimate inequality holds by Equation (4). Hence, by the union bound, we deduce that w.p. $1-\beta$, for all these i's it holds that $V_i = 1$, as required.

We next define the events E and E' for the friendliness and privacy properties.

First, note that by Fact A.13 it holds that

$$\Pr\left[\widehat{N} < n\right] \leq \delta/2 \tag{7}$$

In the following, let $\mathcal{I} = \{i \in [n] \colon \sum_{j=1}^{n-1} f(x_i, x_j) \leq (n-1)/2\}$ and let $E \subseteq \{0,1\}^n$ be the event $\{\boldsymbol{v} \in \{0,1\}^n \colon \boldsymbol{v}_{\mathcal{I}} = 0^{|\mathcal{I}|}\}$. In addition, let $\mathcal{I}' = \mathcal{I} \setminus \{n\}$ and let $E' \subseteq \{0,1\}^{n-1}$ be the event $\{\boldsymbol{v}' \in \{0,1\}^{n-1} \colon \boldsymbol{v}'_{\mathcal{I}'} = 0^{|\mathcal{I}'|}\}$. Note that for every $i \in \mathcal{I}$ it holds that $z_i \leq -1/2$ and $z'_i \leq 1/2$, and therefore $\Pr\left[\boldsymbol{V}_i = 1 \mid \widehat{N} \geq n\right] = \Pr\left[\widehat{Z}_i > \sqrt{\frac{\widehat{N}\cdot\ln(2\widehat{N}/\delta)}{4\rho_2}} + \frac{1}{2} \mid \widehat{N} \geq n\right] \leq \frac{\delta}{2n}$, where the last inequality holds by Fact A.13. Therefore, by the union bound we deduce that

$$\Pr[\boldsymbol{V} \notin E] \leq \delta/2 + \Pr\left[\boldsymbol{V} \notin E \mid \widehat{N} \geq n\right] \leq \delta.$$

A similar calculation also yields that $\Pr\left[\boldsymbol{V}' \notin E'\right] \leq \delta$. It remains to prove friendliness and privacy w.r.t. the events E and E'.

To prove friendliness, fix $\boldsymbol{v} \in E$ and $\boldsymbol{v}' \in E'$. By definition of E, for every $i \in [n]$ s.t. $\boldsymbol{v}_i = 1$ it holds that $\sum_{j=1}^{n-1} f(x_i, x_j) > (n-1)/2$, and for every $i' \in [n-1]$ s.t. $\boldsymbol{v}'_{i'} = 1$ it holds that $\sum_{j=1}^{n-1} f(x_{i'}, x_j) > (n-1)/2$. This yields that there exists at least one $j \in [n-1]$ such that $f(x_i, x_j) = f(x_{i'}, x_j) = 1$. We therefore conclude that $\mathcal{D}_{\{i \colon v_i = 1\}} \cup \mathcal{D}'_{\{i \colon v'_i = 1\}}$ is f-friendly.

We now prove privacy. Note that for every $i \in [n-1]$ it holds that $|z_i - z'_i| = |1/2 - f(x_i, x_n)| = 1/2$. By the properties of the Gaussian Mechanism for zCDP (Theorem A.14) we obtain that $\widehat{Z}_i \approx_{\rho/n} \widehat{Z}'_i$. By composition of zCDP mechanisms (Fact A.15) we obtain that $(\widehat{Z}_1, \ldots, \widehat{Z}_{n-1}) \approx_\rho (\widehat{Z}'_1, \ldots, \widehat{Z}'_{n-1})$. Hence, by post-processing, it holds that $\boldsymbol{V}_{-n} \approx_\rho \boldsymbol{V}'$. Now note that when conditioning $\boldsymbol{V}$ on the event E, the coordinates in $\mathcal{I}$ become 0, and the distribution of the coordinates outside $\mathcal{I}$ remain the same, i.e. $\boldsymbol{V}_{-\mathcal{I}}|_E \equiv \boldsymbol{V}_{-\mathcal{I}}$ (this is because the $\boldsymbol{V}_i$'s are independent, and E is only an event on the coordinates in $\mathcal{I}$). Similarly, the same holds when conditioning $\boldsymbol{V}'$ on the event E'. Since $\mathcal{I}' = \mathcal{I} \setminus \{n\}$, we conclude that $(\boldsymbol{V}_{-n})|_E \approx_\rho (\boldsymbol{V}')|_{E'}$. □

H.5. Proving Lemma C.9

In this section we prove Lemma C.9, restated below.

Lemma H.21 (Restatement of Lemma C.9)**.** *Let $\mathcal{D} = (x_1, \ldots, x_n)$ and $\mathcal{D}' = \mathcal{D}_{-j}$ be neighboring databases, let V, V' be random variables over $\{0,1\}^n$ and $\{0,1\}^{n-1}$ (respectively) such that $V_{-j} \approx_{\rho,\delta} V'$, and define the random variables $R = \mathcal{D}_{\{i\in[n] \colon V_i = 1\}}$ and $R' = \mathcal{D}'_{\{i\in[n-1] \colon V'_i = 1\}}$. Let A be an algorithm such that for any neighboring $\mathcal{C} \in \operatorname{Supp}(R)$ and $\mathcal{C}' \in \operatorname{Supp}(R')$ satisfy $\mathsf{A}(\mathcal{C}) \approx_{\rho',\delta'} \mathsf{A}(\mathcal{C}')$. Then $\mathsf{A}(R) \approx_{\rho+\rho',\, \delta+\delta'} \mathsf{A}(R')$.*

We use the following fact about Rényi divergence.

Fact H.22 (Quasi-Convexity)**.** *[Lemma 2.2 in (Bun & Steinke, 2016)] Let P_0, P_1 and Q_0, Q_1 be two distributions, and let $P = tP_0 + (1-t)P_1$ and $Q = tQ_0 + (1-t)Q_1$ for $t \in [0,1]$. Then for any $\alpha > 1$:*

$$D_\alpha(P||Q) \leq \max\{D_\alpha(P_0||Q_0), D_\alpha(P_1||Q_1)\}$$

The following fact is an immediate corollary of Fact H.22.

Fact H.23. *Let $X = tX_0 + (1-t)X_1$ for $t \in [0,1]$. If $X_0 \approx_{\rho,\delta} Y$ and $X_1 \approx_{\rho,\delta} Y$, then $X \approx_{\rho,\delta} Y$.*

The composition proof for zCDP mechanisms immediately follows by the composition property of Rényi divergence (see (Bun & Steinke, 2016)), and can straightforwardly be extended to the following fact.

Fact H.24. *Let $Y \approx_{\rho,\delta} Y'$, and let F and F' be two (randomized) functions such that $\forall y \in \operatorname{Supp}(Y) \cup \operatorname{Supp}(Y')$: $\mathsf{F}(y) \approx_{\rho',\delta'} \mathsf{F}'(y)$. Then $\mathsf{F}(Y) \approx_{\rho+\rho',\, \delta+\delta'} \mathsf{F}'(Y')$.*

We now use Facts H.23 and H.24 to prove Lemma C.9 which handles specific cases where the input databases that we consider are random variables which are only "close" to being neighboring.

proof of Lemma C.9. Let $\mathcal{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ and $\mathcal{D}' = \mathcal{D}_{-j} = (\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_{n-1})$. The proof holds by Fact H.24 for the following choices of Y, Y', F, F': Let $Y := V_{-j}$ and $Y' := V'$. For $\boldsymbol{y} \in \operatorname{Supp}(Y) \cup \operatorname{Supp}(Y') \subseteq \{0,1\}^{n-1}$, define $F'(\boldsymbol{y}) := \mathsf{A}(\mathcal{C}')$ for $\mathcal{C}' = (x'_i)_{i\in[n-1] \colon \boldsymbol{y}_i = 1}$, and define $F(\boldsymbol{y})$ as the output of the following process: (1) Sample $v_j \leftarrow V_j|_{V_{-j}=\boldsymbol{y}}$ and let $\boldsymbol{v}_{-j} := \boldsymbol{y}$, (2) Output $\mathsf{A}(\mathcal{C})$ for $\mathcal{C} = (x_i)_{i\in[n] \colon \boldsymbol{v}_i = 1}$. By definition, $\mathsf{A}(R) \equiv F(Y)$ and $\mathsf{A}(R') \equiv F'(Y')$. Since $Y \approx_{\rho,\delta} Y'$, it is left to prove that $F(\boldsymbol{y}) \approx_{\rho',\delta'} F'(\boldsymbol{y})$ for every $\boldsymbol{y} \in \operatorname{Supp}(Y) \cup \operatorname{Supp}(Y')$. Fix such $\boldsymbol{y}$, let $\mathcal{C}' = (x'_i)_{\{i\in[n-1] \colon \boldsymbol{y}_i = 1\}}$ and let $\mathcal{C}$ be the database that is obtained by adding x_j to the j'th location in $\mathcal{C}'$ (i.e., $\mathcal{C}_j = x_j$ and $\mathcal{C}_{-j} = \mathcal{C}'$). Note that $F'(\boldsymbol{y}) \equiv \mathsf{A}(\mathcal{C}')$, and $F(\boldsymbol{y})$ depends on the value of the sample v_j: If $v_j = 0$ then it outputs $\mathsf{A}(\mathcal{C}')$ (same output as $F'(\boldsymbol{y})$), and if $v_j = 1$ then it outputs $\mathsf{A}(\mathcal{C})$ which is (ρ', δ')-indistinguishable from $\mathsf{A}(\mathcal{C}')$ since $\mathcal{C}, \mathcal{C}'$ are neighboring databases in $\operatorname{Supp}(R), \operatorname{Supp}(R')$ (respectively). In particular, $F(\boldsymbol{y})$ is a convex combination of random variables that are (ρ', δ')-indistinguishable from $F'(\boldsymbol{y})$. Hence, we deduce by Fact H.23 that $F(\boldsymbol{y}) \approx_{\rho',\delta'} F'(\boldsymbol{y})$, as required.

□

H.6. Proof of Lemma C.12

Claim H.25 (Restatement of Lemma C.12). *Let $\mathcal{D} \in \mathcal{X}^n$ and let $\boldsymbol{p}, \boldsymbol{p}' \in [0,1]^n$ with $\|\boldsymbol{p} - \boldsymbol{p}'\|_1 \leq \gamma$. Let V and V' be two random variables, distributed according to $\mathrm{Bern}(\boldsymbol{p})$ and $\mathrm{Bern}(\boldsymbol{p}')$, respectively, and define the random variables $R = \mathcal{D}_{\{i:\, V_i=1\}}$ and $R' = \mathcal{D}_{\{i:\, V_i'=1\}}$. Let A be an algorithm that for every neighboring databases $\mathcal{C} \in \mathrm{Supp}(R)$ and $\mathcal{C}' \in \mathrm{Supp}(R')$ satisfy $\mathsf{A}(\mathcal{C}) \approx^{\mathsf{DP}}_{\varepsilon,\delta} \mathsf{A}(\mathcal{C}')$. Then $\mathsf{A}(R) \approx^{\mathsf{DP}}_{\gamma(e^\varepsilon-1),\, \gamma\delta e^{\varepsilon+\gamma(e^\varepsilon-1)}} \mathsf{A}(R')$.*

Proof. We assume w.l.o.g. that V and V' are jointly distributed in the following probability space: For each $i \in [n]$, we draw $T_i \leftarrow U[0,1]$, and set $V_i = \mathbb{1}_{\{T_i \leq p_i\}}$ and $V_i' = \mathbb{1}_{\{T_i \leq p_i'\}}$. Note that with this choice,

$$\Pr[V_i \neq V_i'] = \Delta_i := |p_i - p_i'|. \tag{8}$$

In the following, for $i \in [n]$ define

$$\tau_i := \frac{\min\{p_i, p_i'\}}{1 - |p_i - p_i'|},$$

where we let $\tau_i = 0$ in case $|p_i - p_i'| = 1$. Consider a partition of the support of this joint probability space as a product over i of two parts for each i: Let $E_{i,0}$ be the event $\{T_i \leq \tau_i\}$ and let $E_{i,1}$ be the event $\{T_i \geq \tau_i\}$.

This partition has the following structure. First note that $\min\{p_i, p_i'\} \leq \tau_i \leq \max\{p_i, p_i'\}$. The first inequality is immediate. The second inequality follows since $\tau_i(1 - \Delta_i) = \min\{p_i, p_i'\}$ implies that $\tau_i = \min\{p_i, p_i'\} + \tau_i\Delta_i \leq \min\{p_i, p_i'\} + \Delta_i = \max\{p_i, p_i'\}$. Therefore, under $E_{i,z}$ (for each $z \in \{0,1\}$), at least one of V_i or V_i' is fixed.

We use the following claim.

Claim H.26. *For every $i \in [n]$ and $z \in \{0,1\}$ it holds that*

$$\Pr[V_i \neq V_i' \mid E_{i,z}] = \Delta_i,$$

Proof. By Equation (8), it suffices to establish the claim for $E_{i,0}$ ($T_i \leq \tau_i$). Assume without loss of generality that $p_i \leq p_i'$. Since $T_i \leq \tau_i \leq p_i'$, we have $V_i' = 1$. For outcomes $T_i \leq p_i$ we have $V_i = V_i'$. For outcomes $T_i \in (p_i, \tau_i)$ we have $V_i \neq V_i'$. The conditional probability is

$$\frac{\tau_i - p_i}{\tau_i} = \left(\frac{p_i}{1 - \Delta_i} - p_i\right)\frac{1 - \Delta_i}{p_i} = \Delta_i\ .$$

□

As a corollary, due to the joint space being a product space, we have that this also holds in each part $F_{\boldsymbol{z}} = \bigcap_i E_{i,z_i}$, for $\boldsymbol{z} = (z_1, \ldots, z_n) \in \{0,1\}^n$ of the joint space. That is,

$$\forall \boldsymbol{z} \in \{0,1\}^n, \forall i \in [n]: \quad \Pr[V_i \neq V_i' \mid F_{\boldsymbol{z}}] = \Delta_i. \tag{9}$$

We now get to the group privacy analysis. For possible outputs $\mathcal{S}$ of Algorithm A, we relate the probabilities that $\mathsf{A}(R) \in \mathcal{S}$ and that of $\mathsf{A}(R') \in \mathcal{S}$ (recall that $R = \mathcal{D}_{\{i:\, V_i=1\}}$ and $R' = \mathcal{D}_{\{i:\, V_i'=1\}}$).

Note that for the random variables R and R' we have

$$\Pr[\mathsf{A}(R) \in \mathcal{S}] = \sum_{\boldsymbol{z} \in \{0,1\}^n} \Pr[F_{\boldsymbol{z}}] \cdot \Pr[\mathsf{A}(R) \in T \mid F_{\boldsymbol{z}}] \tag{10}$$

$$\Pr[\mathsf{A}(R') \in \mathcal{S}] = \sum_{\boldsymbol{z} \in \{0,1\}^n} \Pr[F_{\boldsymbol{z}}] \cdot \Pr[\mathsf{A}(R') \in T \mid F_{\boldsymbol{z}}]\ . \tag{11}$$

In the following, recall that by definition of Δ_i it holds that $\sum_{i=1}^n \Delta_i = \|\boldsymbol{p} - \boldsymbol{p}'\| \leq \gamma$. The following Claim will complete the proof.

Claim H.27. *For $z \in \{0,1\}^n$,*

$$Pr[\mathsf{A}(R') \in T \mid F_{\boldsymbol{z}}] \leq e^{\gamma(e^\varepsilon-1)}\Pr[\mathsf{A}(R) \in T \mid F_{\boldsymbol{z}}] + \gamma e^{\varepsilon+\gamma(e^\varepsilon-1)}\delta.$$

Proof. Let V^* be the center vector of part $F_{\boldsymbol{z}}$, that is, for each i, if V_i is fixed on the support of E_{i,z_i} to a value $b \in \{0,1\}$ then $V_i^* = b$ and otherwise, if V_i' is fixed to $b \in \{0,1\}$ let $V_i^* = b$. Define the random variable $R^* = \mathcal{D}_{\{i:\, V_i^*=1\}}$, let $I \subseteq [n]$ be the positions i where V_i is fixed on the support of E_{i,z_i}, let $I' = [n] \setminus I$.

We now relate the two probabilities $\Pr[\mathsf{A}(R) \in T \mid F_{\boldsymbol{z}}]$ and $\Pr[\mathsf{A}(R^*) \in T \mid F_{\boldsymbol{z}}]$. Note that for every $i \in I$ we have $V_i = V_i^*$. It is only possible to have $V_i \neq V_i^*$ for $i \in I'$. Let H_k be the event in $F_{\boldsymbol{z}}$ that V is different than V^* in k coordinates. This event is a sum of $|I'|$ Bernoulli random variables with probabilities $\{\Delta_i\}_{i\in I'}$. Let $\Delta_{I'} = \sum_{i\in I'} \Delta_i$ and let $\Delta_I = \sum_{i\in I} \Delta_i$. Compute

$$\begin{aligned}
&\Pr[\mathsf{A}(R) \in \mathcal{S} \mid F_{\boldsymbol{z}}] \qquad (12)\\
&= \sum_{k=0}^n \Pr[H_k \mid F_{\boldsymbol{z}}] \cdot \Pr[\mathsf{A}(R) \in \mathcal{S} \mid H_k \cap F_{\boldsymbol{z}}]\\
&\leq \sum_{k=0}^n \Pr[H_k \mid F_{\boldsymbol{z}}] \cdot e^{k\varepsilon}\Pr[\mathsf{A}(R^*) \in \mathcal{S} \mid F_{\boldsymbol{z}}]\\
&+ \sum_{k=0}^n \Pr[H_k \mid F_{\boldsymbol{z}}] \cdot k e^{k\varepsilon}\delta\\
&= \Pr[\mathsf{A}(R^*) \in \mathcal{S} \mid F_{\boldsymbol{z}}] \sum_{k=0}^n \Pr[H_k \mid F_{\boldsymbol{z}}] \cdot e^{k\varepsilon}\\
&+ \delta \sum_{k=0}^n \Pr[H_k \mid F_{\boldsymbol{z}}] \cdot k e^{k\varepsilon}\\
&\leq \Pr[\mathsf{A}(R^*) \in \mathcal{S} \mid F_{\boldsymbol{z}}] e^{\Delta_{I'}(e^\varepsilon-1)} + \Delta_{I'}\delta e^{\varepsilon+\Delta_{I'}(e^\varepsilon-1)}
\end{aligned}$$

The first inequality holds by group privacy and by the fact that R^* is fixed under $F_{\boldsymbol{z}}$. The last inequality holds by Claim H.28 and Claim H.29.

Similarly, let H'_k be the event in $F_{\boldsymbol{z}}$ that V' is different than V^* in k coordinates. The probability of H'_k is according to a sum of $|I|$ Bernoulli random variables with probabilities $\{\Delta_i\}_{i\in I}$.

$$
\begin{aligned}
&\Pr[\mathsf{A}(R') \in T \mid F_{\boldsymbol{z}}] \\
&= \sum_{k=0}^{n} \Pr[H'_k \mid F_{\boldsymbol{z}}] \cdot \Pr[\mathsf{A}(R') \in \mathcal{S} \mid H_k \cap F_{\boldsymbol{z}}] \\
&\geq \sum_{k=0}^{n} \Pr[H'_k \mid F_{\boldsymbol{z}}] \cdot e^{-k\varepsilon}\big(\Pr[\mathsf{A}(R^*) \in \mathcal{S} \mid F_{\boldsymbol{z}}] - ke^{k\varepsilon}\delta\big) \\
&= \Pr[\mathsf{A}(R^*) \in \mathcal{S} \mid F_{\boldsymbol{z}}] \sum_{k=0}^{n} \Pr[H'_k \mid F_{\boldsymbol{z}}] \cdot e^{-k\varepsilon} \\
&- \delta \sum_{k=0}^{n} \Pr[H'_k \mid F_{\boldsymbol{z}}] k \\
&= \Pr[\mathsf{A}(R^*) \in \mathcal{S} \mid F_{\boldsymbol{z}}] \sum_{k=0}^{n} \Pr[H'_k \mid F_{\boldsymbol{z}}] \cdot e^{-k\varepsilon} - \delta\Delta_I \\
&\geq \Pr[\mathsf{A}(R^*) \in \mathcal{S} \mid F_{\boldsymbol{z}}] e^{-\Delta_I(e^{\varepsilon}-1)} - \delta\Delta_I
\end{aligned}
$$

The first inequality holds by group privacy. The last inequality holds by an adaptation of Claim H.28. Rearranging, we obtain

$$
\begin{aligned}
&\Pr[\mathsf{A}(R^*) \in \mathcal{S} \mid F_{\boldsymbol{z}}] \\
&\leq \Pr[\mathsf{A}(R') \in \mathcal{S} \mid F_{\boldsymbol{z}}] e^{\Delta_I(e^{\varepsilon}-1)} + \delta\Delta_I e^{\Delta_I(e^{\varepsilon}-1)}
\end{aligned}
\tag{13}
$$

The claim follows by combining (12) and (13), noting that $\gamma = \sum_{i=1}^{n} \Delta_i = \Delta_I + \Delta_{I'}$. ☐

The proof follows using (10) and (11) by substitution the claim for each $F_{\boldsymbol{z}}$. ☐

Claim H.28. *Let $X = X_1 + \ldots + X_n$, where the X_i's are independent, and each X_i is distributed according to* $\mathrm{Bern}(p_i)$*, and let $\alpha = \sum_{i=1}^{n} p_i$. Then for every $\varepsilon > 0$ it holds that* $\mathrm{E}\big[e^{\varepsilon X}\big] \leq e^{(e^{\varepsilon}-1)\alpha}$.

Proof. The proof holds by the following calculation

$$
\begin{aligned}
\log(\mathrm{E}\big[e^{\varepsilon X}\big]) &= \log(\prod_{i=1}^{n}(1 - p_i + p_i e^{\varepsilon})) \\
&= \sum_{i=1}^{n} \log(1 - p_i + p_i e^{\varepsilon}) \\
&\leq n \cdot \log\left(1 - \frac{\sum_{i=1}^{n} p_i}{n} + \frac{\sum_{i=1}^{n} p_i}{n} e^{\varepsilon}\right) \\
&\leq n \cdot \log\left(e^{(e^{\varepsilon}-1)\frac{\sum_{i=1}^{n} p_i}{n}}\right) \\
&= (e^{\varepsilon} - 1)\alpha.
\end{aligned}
$$

The first inequality holds by Jensen's inequality since the function $x \mapsto \log(1 - x + xe^{\varepsilon})$ is concave. The second inequality holds since $1 - x + xe^{\varepsilon} = 1 + (e^{\varepsilon} - 1)x \leq e^{(e^{\varepsilon}-1)x}$ for every x. ☐

Claim H.29. *Let $X = X_1 + \ldots + X_n$, where the X_i's are independent, and each X_i is distributed according to* $\mathrm{Bern}(p_i)$*, and let $\alpha = \sum_{i=1}^{n} p_i$. Then for all $\varepsilon > 0$ it holds that*
$\mathrm{E}\big[X \cdot e^{\varepsilon X}\big] \leq \alpha \cdot e^{\varepsilon + (e^{\varepsilon}-1)\alpha}$.

Proof. Compute

$$
\begin{aligned}
\mathrm{E}\big[X \cdot e^{\varepsilon X}\big] &= \sum_{i=1}^{n} \mathrm{E}\big[X_i \cdot e^{\varepsilon X}\big] \\
&= \sum_{i=1}^{n} \mathrm{E}\big[X_i \cdot e^{\varepsilon X_i}\big] \cdot \mathrm{E}\Big[e^{\varepsilon(X - X_i)}\Big] \\
&\leq \sum_{i=1}^{n} p_i e^{\varepsilon} \cdot e^{(e^{\varepsilon}-1)\alpha} = \alpha \cdot e^{\varepsilon + (e^{\varepsilon}-1)\alpha},
\end{aligned}
$$

where the inequality holds by Claim H.28. ☐