Sampling and Partitioning for Differential Privacy

Hamid Ebadi
Chalmers University of Technology
Göteborg, Sweden
hamid.ebadi@chalmers.se

Thibaud Antignac
Chalmers University of Technology
Göteborg, Sweden
thibaud.antignac@chalmers.se

David Sands
Chalmers University of Technology
Göteborg, Sweden
dave@chalmers.se

Abstract—**Differential privacy enjoys increasing popularity thanks to both a precise semantics for privacy and effective enforcement mechanisms. Many tools have been proposed to spread its use and ease the task of the concerned data scientist. The most promising among them completely discharge the user of the privacy concerns by transparently taking care of the privacy budget. However, their implementation proves to be delicate, and introduce flaws by falsifying some of the theoretical assumptions made to guarantee differential privacy. Moreover, such tools rely on assumptions leading to over-approximations which artificially reduce utility. In this paper we focus on a key mechanism that tools do not support well: sampling. We demonstrate an attack on PINQ (McSherry, SIGMOD 2009), one of these tools, relying on the difference between its internal mechanics and the formal theory for the sampling operation, and study a range of sampling methods and show how they can be correctly implemented in a system for differential privacy.**

1. Introduction

Sampling is a procedure in statistics that involves studying only a subset of a population to estimate properties and quantities of the entire population when access to the entire dataset is costly or practically infeasible. Several sampling methods have been introduced aiming to give statisticians an unbiased representative dataset model with properties generalisable to the entire population.

Regardless of the method, the main reasons to use sampling/partitioning, especially in the context of privacy preserving analysis, can be summarised in the following points.

- **Environment-driven.** Stream processing or real time systems usually suffer from limited computing and storage capabilities. Fitting data into memory is not always possible and the computation power is not always sufficient for real time response requirements. To overcome these limitations, parallel and distributed computing (e.g. Map-Reduce) and computations on samples are frequently used.
- **Application-driven.** Machine learning algorithms [14] usually train an algorithm on a subset of a dataset called *training set* and then use the rest of the dataset to evaluate the effectiveness of the algorithm in the *validation/test* phase before it is used for the real world application. Keeping these two subsets separated is one of the main requirements otherwise it is not clear whether the algorithm generalizes or just remembers (memorises) the answers.
- **Utility-driven.** In the differential privacy setting, as explained in several papers [6], [16], when a dataset D is decomposed into arbitrary disjoint subsets D_i, the queries M_{ϵ_i} on these disjoint partitions provides less differential privacy cost than when these queries are executed in sequence. Sharing of a privacy budget between two analyses is an important outcome of parallel composition. This property, known as parallel composition, is particularly useful to efficiently build histograms.
- **Privacy-driven.** Intuitively, working on only a subset of sensitive data might result in less leakage of sensitive information. Random sampling has the potential to amplify privacy.

Studying the literature and tools that employ sampling and partitioning in differential privacy context we noticed informal, incomplete, and sometimes incorrect arguments on implication of sampling and partitioning on the privacy guarantees of an analysis. As an example we show that operators in PINQ are incorrectly handled and demonstrate an attack which reliably reveals the presence of an individual item (Section 3 and Appendix A.1).

The main aim of this paper is to examine the range of sampling and partitioning operators supporting the mentioned goals and their differential privacy implications. We further study how PINQ and similar frameworks can be extended to support a range of operators for sampling and partitioning (Section 4).

Before we go further explaining how these operators positively or negatively affect the privacy of an analysis, let's have an overview on the sampling methods and the primary and differential privacy principles that PINQ and similar frameworks are built upon (Section 2).

2. Preliminaries

We begin by recapping the basic definition of differential privacy, and some standard terminology for sampling that we will use in the remainder of the paper.

Differential Privacy. Differential privacy is generally presented as a way to perturbate replies to queries made over a private dataset in such a way that every individual can plausibly deny being a member of this dataset. The noise added to this effect has to be carefully controlled to reach an acceptable trade-off between privacy and utility preservation. This control is made through the parameter ϵ; hence the name of ϵ-differential privacy. The definition sets a bound on the ratio between the result of two queries made on datasets differing on only one element.

Definition 1 (Differential privacy)**.** *Let two datasets A and A' be called* neighbours *if their symmetric difference is a singleton – i.e. $A = A' \cup \{a\}$ or $A' = A \cup \{a\}$ for some a,i.e., $|A \: \Delta \: A'| = 1$ (also known as the Hamming distance, where Δ is the symmetric difference operation). A randomized computation M provides ϵ-differential privacy if for any neighbours A and A' and any possible output $S \subseteq \mathrm{Rng}(M)$,*

$$\frac{\Pr[M(A) \in S]}{\Pr[M(A') \in S]} \leq e^{\epsilon}.$$

with $\mathrm{Rng}(f)$ the range of a function f.

This definition of differential privacy is the strongest among many variants, and is the version studied in this paper. It is worth mentioning one close relative, which also goes by the same name but is subtly different. The variant, which for the present discussion we will call ***weak differential privacy***[1] is obtained by considering two datasets A and A' to be neighbours if one can be obtained from the other by *changing* exactly one element. The general relation between the two definitions is that if M is ϵ-differentially private then it will be (at least) 2ϵ-weakly differentially private. But the converse does not hold, since differential privacy protects small changes in the size of the dataset, whereas weak differential privacy does not.

The use of Definition 1 is important in the design of tools such as PINQ because queries target not only the input dataset but also datasets constructed by transforming the input with relational algebra operations. Weak differential privacy does not compose well with many standard and useful data transformations. To see this we make the following observation: the size operation which (deterministically) returns the size of its dataset argument is 0-weakly differentially private. I.e., you can return the size of the dataset at zero privacy cost. But suppose that the dataset was built by first selecting all individuals with a specific social security number. Under (strong) differential privacy this is not a problem since selection preserves the neighbourhood property. However, it does not preserve the corresponding neighbourhood relation under weak differential privacy. This behaviour cannot be allowed because the size operation would reveal the presence of a particular individual.

Definition 2 (Function sensitivity)**.** *Function sensitivity is the maximum change of a function's resulting value caused by any single item change. For a function $f\colon \mathcal{A} \to \mathbb{R}$ and any two neighbouring datasets $A, A' \subseteq \mathcal{A}$, sensitivity is defined as* $\mathrm{sensitivity}(f) = \max_{A,A' \subseteq \mathcal{A}} |f(A) - f(A')|$.

1. We are not aware of a standard name when discussed in the context of the definition above, although in early work it was referred to as ϵ-indistinguishability [6].

Compositions. Many statistical analyses can be expressed as a chain of relational algebra transformations of some input data followed by an aggregation function. *Stability*, closely related to sensitivity, is a measure of how much a data transformation might scale up the sensitivity of a query.

Definition 3 (Transformation stability)**.** *A transformation T, function from datasets to datasets, is said to be c-stable if for any two neighbouring datasets A and A', we have $|T(A) \,\Delta\, T(A')| \leq c$.*

The result of composing a function f with a c-stable transformation has the sensitivity of $c \times \mathrm{sensitivity}(f)$. As a result the composition theorem of differential privacy is immediate.

Theorem 2.1 (Transformation composition [16][Theorem 2)**.** *] Let M provide ϵ-differential privacy, and T be an arbitrary c-stable transformation. The composite computation $M \circ T$ provides $\epsilon \times c$-differential privacy.*

The relation between stability and differential privacy parameters lies at the heart of the formal theory behind differential privacy tools. These tools take advantage of some simple composition properties of queries themselves, namely that composite queries have additive privacy cost – or even better when the data is partitioned into disjoint sets.

Theorem 2.2 (Sequential composition)**.** *Composition of n queries $M_1, M_2, \ldots, M_n$ that are executed sequentially can be seen as one query with $\sum_{1 \leq i \leq n} \epsilon_i$-differential privacy.*

A "better" result [16, Theorem 4] is obtained if queries are executed on disjoint subsets (partitions) of the dataset. In that case the differential privacy bound of $\max_{1 \leq i \leq n}(\epsilon_i)$ is achieved. We will state this in a more general form, but to do so we need to lift the definition of differential privacy to pairs of datasets, by defining (D_1, D_2) and (D_1', D_2') to be neighbours if D_1 and D_1' are neighbours and $D_2 = D_2'$ or vice-versa (see [22]). If M_1 and M_2 are queries on D, let $\langle M_1, M_2 \rangle$ denote the parallel query which when given a pair of datasets, returns a pair of query results, i.e., $\langle M_1, M_2 \rangle(D_1, D_2) = (M_1(D_1), M_2(D_2))$. Now the parallel composition can be stated as follows:

Theorem 2.3 (Parallel composition)**.** *If M_i is ϵ_i-differentially private, $i \in \{1, 2\}$, then $\langle M_1, M_2 \rangle$ is $\max(\epsilon_1, \epsilon_2)$-differentially private.*

Note that the parallel composition theorem as we have stated it, unlike [16, Theorem 4], does not mention partitioning; the benefit of the theorem comes by creating the input to $\langle M_1, M_2 \rangle$ by a partitioning operation. Also, not all strategies for partitioning are useful here. Indeed, the key is to have a partitioning operation of stability one, and thus we can apply the transformation composition theorem favourably. Fortunately, partitioning using a fixed partition on the whole domain has this property. The potential benefit of parallel composition motivates our study not only of random sampling but also of random partitioning.

Sampling. In the following, we will consider a variety of data sampling methods, so we also fix the terminology we will use. A *sampling method* is a probabilistic function which, given an input set D, produces a multiset over D, known as a sample. This very general concept includes deterministic sampling operations (e.g., where D is an indexed set, and the sample is a fixed number of elements based on the index) – for example what is sometimes called *convenience sampling* which simply selects the most conveniently accessible elements.

It also includes the case where the sample is a simple subset of D – and in this case the sampling method is said to be *without replacement*.

Definition 4 (Uniform sampling)**.** *If RS is a sampling method, define $\overrightarrow{RS(D)}$ to be the set of possible outcomes – i.e. $\{K \mid \Pr[RS(D) = K] > 0\}$ Then RS is a uniform sampling method if the probability of obtaining any two samples in $\overrightarrow{RS(D)}$ is the same – i.e. the possible samples are all equally likely.*

Uniform sampling includes the standard notion of *simple random sampling*, with or without replacement, as well as fully deterministic sampling methods.

Sampling methods are invariably parameterised over the desired "size", which indicates, for example, the number of elements to be taken in the sample, or the probability that each element is selected. As a consequence, strictly speaking, we will be dealing with families of sampling functions $\{RS_k\}_{k \in \text{size}}$.

Note that any sampling method without replacement can be viewed as a partition of the input into two disjoint subsets, the sample and the rest.

We are now ready to have a look at the state of affairs to perform sampling in the wild and to develop the limitations mentioned in introduction.

3. Sampling in PINQ

The basic principles highlighted in the theorems of the previous section, and generalisations thereof, form a basis for general purpose programming frameworks that allow construction of analyses which are differentially private by construction [9], [10], [16], [17], [21], [25]. These frameworks use a similar method to construct compound queries, but for the purpose of this paper we focus on PINQ [16].

This tool is a platform to perform differential privacy-preserving analysis on datasets. It is based on the LINQ interface from the Microsoft .NET framework. Differential privacy properties are automatically and transparently enforced by PINQ when the naive user only expresses the queries of interest.

The aim in this section is to analyse the status of sampling in PINQ to motivate the extension of the framework with sampling and partitioning operations. As mentioned, PINQ is an API for enforcing differential privacy, and is built from deterministic transformations T_i, which are variants of the well-known SQL-like operations, and primitive differentially-private aggregation operations M_ϵ, parameterised by the intended privacy level ϵ. To get an idea of the way PINQ works, suppose that we take the original data, perform a sequence of transformations $T_1, \ldots T_n$ on it, apply a primitive aggregation operation M_ϵ, and publish the result. PINQ counts the overall privacy cost of this by deducting $\epsilon \times \Pi_{i \in \{1,..n\}} \text{stability}(T_i)$ from the budget (if the budget is insufficient, an exception is thrown).

Take and Skip. Since the theory of PINQ is based on non-probabilistic transformations, PINQ supports just two sampling operations which allow the selection of (at most) a fixed number of elements based on the underlying ordering of the elements in the database.

PINQ supports sampling as a stand-alone transformation but not with the purpose of amplifying privacy. The two complimentary transformation methods `Take(n)` and `Skip(n)` are based on their equivalent methods in LINQ. `Take(n)`, as explained in Microsoft Software Developer Network documentation [2] "returns a specified number of contiguous elements from the start of a sequence" and `Skip(n)` [1] "bypasses a specified number of elements in a sequence and then returns the remaining elements". Since these two methods work similarly, albeit in a dual manner, we only discuss the issue with the `Take(n)` method in this paper. A similar argument holds for the `Skip(n)` method.

The sensitivity of Take. Discussing the sensitivity of Take is tricky because it treats the data A as a sequence and not a set. In the following we assume that the index is part of the data. Let $\text{Take}(A, n)$ be the mathematical function corresponding to the `A.Take(n)` PINQ operation. When `n` $\leq$ |`A`| the difference between results of $\text{Take}(A, n)$ and $\text{Take}(A \cup \{a_x\}, n)$ is assumed to be zero, and when $n > |A|$, the difference is one as it returns all the elements. Consequently, in PINQ, the stability of Take transformations is assumed to be one.

However, the ordering of elements influences the stability. As an example, in the case of union of item a and dataset A, there is no guarantee that element a takes the highest ordering value and is placed at the end of the list A. In the current implementation the ordering of resulting dataset follows the following rules:

- the relative ordering of items within a set remains the same, and
- the order of sets passed as arguments to union determines the order of sets.

To be more specific, when a_x is added to the end of the list (the item a_x appears as the second argument) we have a stability of zero: $\text{Take}(\{a_1 \ldots a_n\}, 1) \;\Delta\; \text{Take}(\{a_1 \ldots a_n\} \cup \{a_1\}, 1) = \{a_1\} \;\Delta\; \{a_1\} = \emptyset$. When the item is added to the beginning of the list we observe a stability of two: $\text{Take}(\{a_1 \ldots a_n\}, 1) \;\Delta\; \text{Take}(\{a_x\} \cup \{a_1 \ldots a_n\}, 1) = \{a_1\} \;\Delta\; \{a_x\} = \{a_1, a_x\}$. This demonstrates that PINQ's assumption that Take has stability of 1 is incorrect. The problem with the incorrect sensitivity of Take is not immediately visible when applying aggregation operations to the result. This is because many of the aggregation operations supported by PINQ, e.g. `NoisyCount`, provide the same ϵ of privacy for both variants of differential privacy, since they are no more sensitive to removing an element (a symmetric difference of one) than they are to changing an element (a symmetric difference of two).

An attack that can scale up the difference by any arbitrary factor is demonstrated and explained in Listing 1 from Appendix A.1. The proof of concept code determines the existence of a specific element (i.e., number 7) in the dataset with higher probability than the limited privacy bound ϵ. The idea behind the proof of concept is to apply some transformation that iteratively adds random items to the dataset if an individual item, here number 7, is not present in the dataset. With an actual dataset of size 9, we got 9867.4529 ($\approx 10000 + 9$) as the randomized size of the protected dataset object in one experiment after 10000 iterations. Running the same program, by adding the number 7 results in -20.0768 (≈ 9) as the randomized size. The large distance between the two possible outcomes can reveal the presence of any item in the database.

4. Uniform Sampling and Partitioning

The previous section highlights some shortcomings in PINQ:

- sampling for the purpose of scaling data to available resources is the only supported operation, and
- the privacy cost (the sensitivity) is worse than assumed, and thus taking elements halves the available privacy budget.

In this section we consider a variety of sampling methods with a view to including them in a PINQ-like framework. We wish to explore sampling for the various purposes listed in the introduction, and answer the following questions:

1) From a privacy perspective, can we do (significantly) better than Take for selecting a specific number of elements by using random sampling?
2) Can we partition the unknown-sized data into portions of known relative size, to share the data between different analyses and to take advantage of the parallel composition benefits of analyses on disjoint data?
3) For which kinds of sampling can we achieve amplification of privacy?
4) Is it possible to do partitioning that is parameterised on the size of the intended sample?
5) What are the stability properties of a well-behaved partitioning transformation?

Methods that are used for sampling data from a dataset is generally categorised into fixed size and unfixed size sampling methods. Fixed size sampling methods, given the sample size as a numerical number or a fraction of dataset, return a fixed number of elements every time the sampling is performed. Common fixed size sampling methods, including sampling without replacement, sampling with replacement, and fraction sampling are studied in this paper. For the unfixed size sampling we study Bernoulli sampling in which, given the sample inclusion probability, each element of the population is subject to an independent Bernoulli trial which decides about its inclusion.

Random uniform sampling *with* and *without* replacement are used when the algorithm, the system capacity, or the real-time requirements enforces a specific number of items for the system input (which are all environment-driven limitations). Randomized fraction sampling guarantees a precise ratio between the sample size and the population. Knowing the population size, one can compute the sample size. However when the size of population is not known (in the case of differential privacy) the sample size remains unknown. The size of samples from Bernoulli sampling is dynamic and follows the Bernoulli distribution. All the sampling methods can be used to reduce the size of the database but it is the application that influences the choice of the sampling method. In the following, we focus on utility and privacy-driven goals. Before this, we need to revisit the notion of stability.

4.1. From Deterministic to Probabilistic Transformations

In this subsection we look at general considerations in the analysis of sampling methods. We generalise the notion of stability and the associated composition methods from the deterministic to the probabilistic case, introduce random partitioning, and outline a simple strategy for reasoning about the differential privacy of queries applied after sampling.

Probabilistic Stability. The definition of transformation stability (Definition 3) is not useful when lifted naively to non-deterministic transformations, since in the worst case the Hamming distance between two samples of size n is $2n$ when the samples are selected from two disjoint subsets. While the stability of $2n$ is an obvious upper-bound for PINQ's randomized `Take(n)` method, we investigate the possibility of introducing a lower upper bound that can be plugged into differential privacy frameworks.

To introduce uniform sampling, we propose a generalisation for definition of stability that allows us to reason about uniform sampling and its composition with differential private mechanisms. This generalised notion of stability has to be independent of the content and of the size of the database and to be pluggable as a function of the privacy cost in these frameworks.

Definition 5 (Probabilistic stability). *Let σ be a function in $\mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$. We say a randomized transformation RS is σ-probabilistically stable if for any ϵ-differentially private mechanism M, $M \circ RS$ is $\sigma(\epsilon)$-differentially private.*

We observe that c-stability can be expressed as a σ-probabilistic stability as follows:

Proposition 4.1. *If T is c-stable, then T is σ-probabilistically stable with $\sigma(\epsilon) = c \times \epsilon$.*

Thus this simple generalisation provides a drop-in replacement for the composition principle of Theorem 2.1 since it is compositional by construction; the simple accounting mechanism of PINQ can be generalised from deterministic transformations with known c-stability, to randomised transformations with known σ-probabilistic stability:

Proposition 4.2. *Suppose that $RS_1, \dots, RS_n$ is a sequence of randomised transformations for which RS_i is σ_i-probabilistically stable, and M is ϵ-differentially private. The mechanism which computes $M \circ RS_1 \circ \cdots \circ RS_n$ is ϵ'-differentially private, where $\epsilon' = (\sigma_1 \circ \cdots \circ \sigma_n)(\epsilon)$.*

This follows easily from Proposition 4.1, the associativity of function composition, and a simple induction on n.

Partitioning and Parallel Queries. The sampling algorithms that are considered in this paper (except sampling with replacement), can also be seen as a partitioning algorithm. Selected items for the sample and the excluded items construct the two partitions that cover all the items.Let RS be a sampling method without replacement. Let $\hat{RS}$ denote the partitioning operation defined as

$$\Pr[\hat{RS}(D) = (K, \bar{K})]$$
$$= \begin{cases} \Pr[RS(D) = K] & \text{if } K \cup \bar{K} = D \text{ and } K \cap \bar{K} = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

It turns out that for just one sampling method we get a low σ-sensitivity (i.e. where $\sigma(\epsilon) < \epsilon$, but for others the results are more expansive ($\sigma(\epsilon) > \epsilon$).

Reasoning About Sampling Methods. Probabilistic stability measures the effect of a transformation on the privacy that a differentially private algorithm enjoys. We observe that any transformation that is expressed in a primitive recursive form keeps the amount of changes limited in each iteration, and have a bounded and easily understood stability. This is the case, for example, for *map* functions (such as the projection operation of relational algebra), recursively defined for some mapping function f as $\text{map}_f\,(D \cup \{a\}) = \text{map}_f\,(D) \cup \{f(a)\}$. In this form it is immediate to see that it has stability of one. In general a recursive equation which allows us to compare the operator's action on $D \cup \{a\}$ and D respectively is precisely what we need to reason about (probabilistic) stability.

Thus, in the following subsections we take the same approach for reasoning about sampling. It turns out that we have a useful generic theorem for recursively specified sampling:

Theorem 4.3. *Let RS be a uniform sampling function. If RS is characterised by a recursive equation $RS(D+\{e\}) = \psi_D(RS(D), e)$ for some randomised function ψ_D then:*

- *The probabilistic stability of uniform sampling RS is $\sigma(\epsilon) = \ln \sum_{i \in \mathbb{N}} e^{i\epsilon}. \Pr[|\psi_D(K, e) \,\Delta\, K| = i]$.*
- *The probabilistic stability of uniform partitioning $\hat{RS}$, built based on uniform sampling RS, is*

$\hat{\sigma}(\epsilon) = \ln \sum_{i \in \mathbb{N}} e^{(i+|i-1|)\epsilon}. \Pr[|\psi_D(K, e) \,\Delta\, K| = i]$.

The proof of this theorem is given in Appendix B.

These theorems extract the common part of the reasoning about sampling methods in this paper. But how easy is it to find a (nontrivial) recursive form for a sampling algorithm? Some methods are trivial to phrase in this form (e.g. Bernoulli sampling, for example). For others we can find a recursive form by looking to the literature on *online* (stream) versions of sampling methods. In a streaming version of a sampling method a new sample is computed from the previous sample and the newly arrived datum. As a consequence, this can be viewed as a recursive specification of the underlying sampling method.

4.2. Uniform (Fixed Size) Sampling Without Replacement

The first method we consider is fixed size uniform sampling without replacement which we denote by the size-indexed family of functions $\{R_n\}_{n \in \mathbb{N}}$. This method is potentially useful when a specific maximum sized target set is desired. Let n be the (maximum) number of elements to be sampled, and D the dataset. When $n < |D|$, n non-distinct items are randomly chosen from the dataset D. When $n \geq |D|$ the entire dataset D is returned. Each item can be selected at most once, which means a sample larger than the dataset itself is not possible.

Definition 6 (Probabilistic behaviour of R_n)**.** *If K is any random sample in the range of $R_n(D)$ (shown as $K \in \overrightarrow{R_n(D)}$), the probabilistic behaviour of random uniform sampling without replacement when $n < |D|$ is $\Pr[R_n(D) = K] = \binom{|D|}{n}^{-1}$. When $n \geq |D|$, the probability doesn't play any role and the function behaves deterministically as:* $\Pr[R_n(D) = D] = 1$.

This follows from the simple fact that R_n is a uniform sampling method, so the probability of any sample is just one over the number of possible samples of that size.

One common implementation of sampling without replacement is based on the Fisher-Yates shuffle [8] with the complexity of $O(n)$ to randomly pick n indices from the array that is holding the item set. This algorithm relies on the presence of all elements in the system and knowing the exact size of the dataset. However in the streaming setting where the size of dataset is not known in advance or when the dataset is larger than the available memory, Vitter [28] introduced the *Reservoir sampling algorithm* with a complexity of $O(|D|)$. An inductive proof for this algorithm can be found in [3] and [27].

The simplified Algorithm 1, known as reservoir sampling, produces a sample with the required probabilistic property. In each iteration the algorithm maintains a reservoir sample that keeps the sample collected so far. When a new data item is introduced, the algorithm inductively construct the new sample by using the sample constructed from the previous step [4].

The algorithm can be explained as follows. For the first n items introduced (one by iteration), the items are directly put into the reservoir. Once the reservoir is full and cannot get the i^{th} item ($i > n$), the algorithm generates a random number r between 1 and i. If r is less than or equal to n, it replaces the element stored in the index r of the reservoir with the new item. This determines whether an item should be included or excluded.

Algorithm 1 Uniform Sampling Without Replacement

```
function $R_n$(D)
    for $i \leftarrow 1, |D|$ do
        if $i \leq n$ then
            $K[i] \leftarrow D[i]$          ▷ Deterministically add D[i]
        else
            $r \leftarrow$ RANDOM$(1, i)$
            if $r \leq n$ then
                $K[r] \leftarrow D[i]$          ▷ Include D[i]
                                          ▷ Otherwise exclude D[i]
    return $K[1..n]$
```

The array D in the Algorithm 1 can be a stream of data items processed one at the time (D[i]). This algorithm provides a recursive specification of random sampling without replacement with the base case $R_n(D) = D$ when $|D| \leq n$ and the recursive case in the form of $R_n(D \cup \{e\}) = \psi_{n,|D|}(R_n(D), e)$. The family of probabilistic functions $\psi_{n,|D|}$ models inner iterations of the Algorithm 1. When the reservoir is not full ($i \leq n$) all items are added to the reservoir deterministically. For $i > n$, the incoming element replaces one item of the reservoir with the probability of $\frac{n}{|i|}$.

Theorem 4.4. *The following is a recursive characterisation of R_n:*

$$\begin{array}{lll} R_n(D) & = D & \text{if } |D| \leq n \\ R_n(D \cup \{e\}) & = \psi_{n,|D|}(R_n(D), e) & \text{otherwise} \end{array}$$

such that for all samples K of D of size n, $\psi_{n,|D|}$ is the probabilistic function defined by the following:

$$\begin{array}{rcl} \Pr[\psi_{n,m}(K, e) = K] & = & \frac{m+1-n}{m+1} \\ \Pr[\psi_{n,m}(K, e) = K^{+e}_{-f}] & = & \frac{1}{m+1} \quad (\text{for any } f \in K) \\ \Pr[\text{otherwise}] & = & 0 \end{array}$$

where K^{+e}_{-f} stands for the dataset K to which the element e has been added and the element f removed.

A proof of this theorem is provided in Appendix B.1.

Theorem 4.5 (Probabilistic stability of R_n)**.** *Random sampling of n elements without replacement, R_n, has a probabilistic stability $\sigma(\epsilon) = \ln(\frac{ne^{2\epsilon}+1}{n+1})$.*

The proof is outlined as follows. To find this probabilistic stability σ for sampling without replacement, we instantiate Theorem 4.3 for different value of D:

Case $n \geq |D \cup \{e\}|$: using $\psi_{n,|D|}$ from Theorem 4.4 we have the upper bound of $\sigma(\epsilon) = \epsilon$.

Case $n < |D \cup \{e\}|$: using $\psi_{n,|D|}$ from Theorem 4.4 we have the upper-bound of $\ln(\frac{ne^{2\epsilon}+1}{n+1}) < 2\epsilon$.

The maximum value appears in the case $n < |D \cup \{e\}|$. For practical purposes (small ϵ, large n), the value of $\ln(\frac{ne^{2\epsilon}+1}{n+1})$ doesn't lead to a significant improvement of the privacy cost compared to the deterministic case 2ϵ.

Partitioning property. We now consider the case of a parallel query of a partition using R_n.

Theorem 4.6. *Let $\hat{M} = \langle M_1, M_2 \rangle$ where M_1 and M_2 are ϵ-differentially private queries. Then $\langle M_1, M_2 \rangle \circ \hat{R}_n$ is $\hat{\sigma} = 3\epsilon$-differentially private.*

The proof is as follows: consider the two possible cases and, by using Theorem 4.3:

Case $n \geq |D| + 1$: $\hat{\sigma(\epsilon)} = \epsilon$.

Case $n < |D| + 1$: $\hat{\sigma}(\epsilon) = \ln(\frac{ne^{3\epsilon}+1}{n+1}) < 3\epsilon$.

In total we have a probabilistic stability of $\hat{\sigma}(\epsilon) = \ln(\frac{ne^{3\epsilon}+1}{n+1}) < 3\epsilon$.

4.3. Uniform (Fixed Size) Sampling with Replacement

This method of sampling takes n elements from the dataset D. Each selection is independently done from the entire dataset D, meaning an item may appear more than once in the sample or the sample may be larger than the dataset itself. We have analysed this sampling method using ideas from Park et. al [19], [20] who propose several algorithms with different complexities and prove their correctness. The algorithm closest to the recursive form is the *RSWR-naive* algorithm [19]. We have been able to show that the probabilistic stability of the recursive step is $\sigma(\epsilon) = \ln \sum_{0 \leq l \leq n} e^{2l\epsilon}.\binom{n}{l}(\frac{1}{n})^l(\frac{n-1}{n})^{n-l}$. We omit the details since the result is not really useful; it is easy to see that we have a lower bound of $n \times \epsilon$ which is already not practical; this bound is achieved by comparing the two neighbours $\emptyset$ and $\{a\}$. In the former case we must get an empty sample, but in the latter case a sample of n copies of a.

4.4. Fraction Sampling

Fraction sampling randomly chooses precisely $\lfloor |D| \times p \rfloor$ items (with $\lfloor . \rfloor$ being the floor function) from dataset D, which corresponds to a fraction of items of approximately p. While in statistics this method and fixed size sampling are treated similarly (a sample size can be expressed as a fraction/ratio), the strong notion of differential privacy where the exact size of sample is secret makes this distinction useful in our case. The advantage of using this method of sampling over fixed size sampling is that the analyst doesn't need to have any information about the number of elements in the database while still getting precise partitioning.

One can see that fraction sampling, which samples a fraction p of elements from the dataset, is similar to sampling without replacement in which the size of reservoir is dynamic and increases over time. *Pareto πps schema* – or *order sampling schemes* – introduced by Rosén [23], [24] is referred to as the class of algorithms for sampling with probability proportional to size (πps). The basic form of these algorithms, with equal inclusion probability, assigns each item a unique random value chosen from a certain range (0,1) and calculate the Pareto ranking variable that is basically just an artificial random ordering. A similar algorithm for a distributed setting that can also be used to achieve fraction sampling is explained in [5]. The authors state that:

"The core insight behind reservoir sampling is that picking a random sample of size k is equivalent to generating a random permutation (ordering) of the elements and picking the top k elements."

For the purpose of this research, centralised system with equal probabilities for each item, the algorithm can be simplified as Algorithm 2. We claim that fraction sampling,

Algorithm 2 Fraction Sampling

```
function P(D, p)
    for $i \leftarrow 1, |D|$ do
        $r \leftarrow$ RANDOM$(1, i)$
        $R[i] \leftarrow R[r]$
        $R[r] \leftarrow D[i]$
    return $R[1..\lfloor p \times |D| \rfloor]$
```

P_p, with $K \in \overrightarrow{P_p(D)}$, $lb = \lfloor p.|D| \rfloor$ (the current sample size) and $ub = \lfloor p.(|D|+1) \rfloor$ (possibly the new sample size after addition of e), has the following recursive characterisation:

$$\begin{array}{lll} P_p(\emptyset) & = \emptyset & \text{(base case)} \\ P_p(D \cup \{e\}) & = \psi_{p,D}(P_p(D), e) & \text{(recursive step)} \end{array}$$

if K is a valid sample of $P_p(D)$, for the case $ub - lb = 1$ when we increase the size of reservoir, the probabilistic function $\psi_{p,D}$ is defined as

$$\begin{array}{rcl} \Pr[\psi_{p,D}(K,e) = K \cup \{e\}] & = & \frac{1}{|D|+1} \\ \Pr[\psi_{p,D}(K,e) = K \cup \{e\} \setminus \{f\} \cup \{x\}] & = & \frac{ub}{|D|+1} \\ \Pr[\psi_{p,D}(K,e) = K \cup \{x\}] & = & 1 - \frac{ub+1}{|D|+1} \end{array}$$

For the case $ub - lb = 0$, when we have no change in the size of reservoir, it is defined as:

$$\begin{array}{rcl} \Pr[\psi_{p,D}(K,e) = K] & = & 1 - \frac{lb}{|D|+1} \\ \Pr[\psi_{p,D}(K,e) = K \cup \{e\} \setminus \{f\}] & = & \frac{lb}{|D|+1} \end{array}$$

Theorem 4.7. *The probabilistic stability of fraction sampling, P_p, is $\sigma(\epsilon) = \ln(\max(e^{2\epsilon}p + 1 - p, e^{3\epsilon}p + e^{\epsilon}(1-p)))$.*

The probabilistic stability is the maximum of the upper bound for the case $(ub - lb = 1)$ and the case $(ub - lb = 0)$.

Partitioning Property. Similarly considering the maximum of both cases, we get a probabilistic stability of $\hat{\sigma}(\epsilon) = \ln \max(e^{3\epsilon}p + (1-p)e^{\epsilon}, e^{5\epsilon}p + 1 - p)$.

4.5. Bernoulli Sampling

The differential privacy of Bernoulli Sampling has been previously considered in the literature [15], [26] with the aim of amplifying privacy. Bernoulli Sampling independently decides about inclusion of each item from the population. Given inclusion probability of b, iterating over all members of a population, Bernoulli trials with probability b decides about the inclusion of each item.

Definition 7. *In Bernoulli sampling, a sample $K \subseteq D$ of size n has a probability of $\Pr[B_b(D) = K] = b^n(1-b)^{|D|-n}$.*

The following is a recursive characterisation of B_b:

$$\begin{array}{lll} B_b(\emptyset) & = \emptyset & \text{(base case)} \\ B_b(D \cup \{e\}) & = \psi_b(B_b(D), e) & \text{(recursive step)} \end{array}$$

where $\psi_b(B_b(D), e)$ is the probabilistic function defined for all samples K of $B_b(D)$ as :

$$\begin{array}{rcl} \Pr[\psi_b(K,e) = K] & = & 1 - b \\ \Pr[\psi_b(K,e) = K \cup \{e\}] & = & b \end{array}$$

As sketched in [26] and proved in the more general case of (ϵ, δ)-differential privacy [15], we benefit from improved privacy of $\sigma(\epsilon) = \ln(b.e^{\epsilon} + 1 - b)$.

Partitioning Property. The result is the same privacy cost as for parallel composition of a deterministic partitioning method.

Theorem 4.8. *Bernoulli partitioning has the probabilistic stability of $\hat{\sigma}(\epsilon) = \epsilon$.*

Proof. Instantiating Theorem 4.3 with ψ_b from the recursive definition, we reach the bound of ϵ as follows.

$$\begin{array}{rl} \hat{\sigma}(\epsilon) & = \ln(e^{(0+|0-1|)\epsilon}. \Pr[|\psi_b(K,e) \,\Delta\, K| = 0] \\ & \quad + e^{(1+|1-1|)\epsilon}. \Pr[|\psi_b(K,e) \,\Delta\, K| = 1]) \\ & = \ln e^{\epsilon}(b + (1-b)) = \epsilon \end{array}$$

□

It is reasonable not to expect a better result. Indeed, the privacy amplification of this sampling method occurs when b is small. Alas, in the context of partitioning, that means that the largest partition will be a sample of $1 - b$. As a consequence, whatever privacy amplification was achieved for the small half is lost on the big half.

5. Related work

Sampling and partitioning is explicitly used in several frameworks. Airavat [25] uses partitioning in its *MapReduce* framework. In the *sample and aggregation* framework used for computing *smooth sensitivity*, Nissim et al. [18] uses random partitioning to divide the database into smaller databases. For simplification, instead of partitioning, the databases is constructed by taking random samples that are chosen independently of each other. When the number of clusters is low, each data item has the chance to appear in multiple samples. However the weak variant of differential privacy is used (indistinguishability). As mentioned earlier, the privacy amplification effects of sampling are discussed in [15], [26]. In earlier work, Kasiviswanathan et al. [12] used this amplification effect to build private learners. Some works focus on data dependant partitioning of data to build histograms, Xiao et al. [29] research two partitioning strategies, cell-based and kd-tree based partitioning for building histograms and Kellaris and Papadopoulos [13] use a grouping technique based on sampling.

A non uniform sampling is used by Jorgensen et al. [11] which assigns a numerical value corresponding to the personal privacy preference for each database item. The personalised privacy then influences the inclusion probability of that individual in the sample used for differentially private analyses.

6. Conclusion

In this paper we have explored a variety of sampling methods and their effect on differential privacy in the strong case where the size of the dataset is private.

In particular we demonstrated a simple and practical attack on PINQ's deterministic sampling transformation which surprisingly shows that a sampling procedure may have a negative effect on the differential privacy cost of an analysis.

Further we investigated whether introducing randomness in selection of samples and partitions can significantly improve the differential privacy bounds compared to what their deterministic alternatives promise.

Technically, we have also shown how the concept of probabilistic stability provides a suitable generalisation of stability in systems that take advantage of composition principles to guarantee differential privacy.

While the choice of a sampling method is mainly driven by the environment and the application, we showed that, among common probability sampling and partitioning methods studied in this paper, (i) only Bernoulli sampling has the privacy amplification property and (ii) the corresponding partitioning algorithm fits perfectly with parallel query composition.

Further work includes the generalisation to approximate (ϵ, δ)-differential privacy and the implementation of the sampling mechanisms in a PINQ-style framework.

References

[1] Microsoft .NET Framework 4. Queryable.skip<tsource>method. https://msdn.microsoft.com/en-us/library/bb357513%28v=vs.100%29.aspx, 2016.

[2] Microsoft .NET Framework 4. Queryable.take<tsource>method. https://msdn.microsoft.com/en-us/library/bb300906%28v=vs.100%29.aspx, 2016.

[3] D. R. Bellhouse A. I. McLeod. A convenient algorithm for drawing a simple random sample. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 32(2):182–184, 1983.

[4] Ivan Rezucha C. T. Fan, Mervin E. Muller. Development of sampling plans by using sequential (item by item) selection techniques and digital computers. *J. of the Am. Stat. Assoc.*, 57(298):387–402, 1962.

[5] Had00b Big data made simple. Reservoir sampling in mapreduce. http://had00b.blogspot.se/2013/07/random-subset-in-mapreduce.html, 2013.

[6] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proc. of the 3rd Conference on Th. of Crypt.*, TCC'06, pages 265–284, Berlin, Heidelberg, 2006. Springer-Verlag.

[7] Úlfar Erlingsson, Aleksandra Korolova, and Vasyl Pihur. RAPPOR: randomized aggregatable privacy-preserving ordinal response. *CoRR*, abs/1407.6981, 2014.

[8] Ronald Aylmer Sir Fisher and Frank Yates. *Statistical tables for biological, agricultural and medical research*. London : Oliver and Boyd, 1938.

[9] Marco Gaboardi, Andreas Haeberlen, Justin Hsu, Arjun Narayan, and Benjamin Pierce. Linear dependent types for differential privacy, 2013.

[10] Marco Gaboardi, Andreas Haeberlen, Justin Hsu, Arjun Narayan, and Benjamin C. Pierce. Linear dependent types for differential privacy. In *Proc. of the 40th Annual ACM SIGPLAN-SIGACT Symp. on Princ. of Prog. Lang.*, POPL '13, pages 357–370, New York, NY, USA, 2013. ACM.

[11] Z. Jorgensen, T. Yu, and G. Cormode. Conservative or liberal? personalized differential privacy. In *2015 IEEE 31st Int. Conf. on Data Eng.*, pages 1023–1034, April 2015.

[12] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. What can we learn privately? *SIAM J. Comput*, 40(3):793–826, 2011.

[13] Georgios Kellaris and Stavros Papadopoulos. Practical differential privacy via grouping and smoothing. In *Proceedings of the 39th international conference on Very Large Data Bases*, PVLDB'13, pages 301–312. VLDB Endowment, 2013.

[14] Igor Kononenko and Matjaz Kukar. *Machine Learning and Data Mining: Introduction to Principles and Algorithms*. Horwood Publishing Limited, 2007.

[15] Ninghui Li, Wahbeh Qardaji, and Dong Su. On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy. In *Proc. of the 7th ACM Symp. on Inf., Comp. and Comm. Sec.*, ASIACCS '12, pages 32–33, New York, NY, USA, 2012. ACM.

[16] Frank McSherry. Privacy integrated queries. In *Proc. of the 2009 ACM SIGMOD Int. Conf. on Manag. of Data (SIGMOD)*. Association for Computing Machinery, Inc., June 2009.

[17] Prashanth Mohan, Abhradeep Thakurta, Elaine Shi, Dawn Song, and David Culler. Gupt: privacy preserving data analysis made easy. In *Proc. of the 2012 ACM SIGMOD Int. Conf. on Manag. of Data*, pages 349–360. ACM, 2012.

[18] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proc. of the 39th Annual ACM Symp. on Th. of Comp.*, STOC '07, pages 75–84, New York, NY, USA, 2007. ACM.

[19] Byung-Hoon Park, George Ostrouchov, and Nagiza F. Samatova. Sampling streaming data with replacement. *Computational Statistics Data Analysis*, 52(2):750 – 762, 2007.

[20] Byung-Hoon Park, George Ostrouchov, Nagiza F Samatova, and Al Geist. Reservoir-based random sampling with replacement from data stream. In *SDM*, pages 492–496. SIAM, 2004.

[21] Davide Proserpio, Sharon Goldberg, and Frank McSherry. Calibrating data to sensitivity in private data analysis: A platform for differentially-private analysis of weighted datasets. *Proc. VLDB Endow.*, 7(8):637–648, April 2014.

[22] Jason Reed and Benjamin C. Pierce. Distance makes the types grow stronger: A calculus for differential privacy. In *ACM SIGPLAN International Conference on Functional Programming (ICFP)*, 2010.

[23] Bengt Rosén. Asymptotic theory for order sampling. *Journal of Statistical Planning and Inference*, 62(2):135 – 158, 1997.

[24] Bengt Rosén. On sampling with probability proportional to size. *Journal of Statistical Planning and Inference*, 62(2):159 – 191, 1997.

[25] Indrajit Roy, Srinath T. V. Setty, Ann Kilzer, Vitaly Shmatikov, and Emmett Witchel. Airavat: Security and privacy for mapreduce. In *Proc. of the 7th USENIX Conf. on Networked Sys. Design and Impl.*, NSDI'10, pages 20–20, Berkeley, CA, USA, 2010. USENIX Association.

[26] Adam Smith. Differential privacy and the secrecy of the sample. https://adamdsmith.wordpress.com/2009/09/02/sample-secrecy/, 2009.

[27] Y. Tillé. *Sampling Algorithms*. Springer Series in Statistics. Springer, 2006.

[28] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, March 1985.

[29] Yonghui Xiao, Li Xiong, and Chun Yuan. *Differentially Private Data Release through Multidimensional Partitioning*, pages 150–168. Springer Berlin Heidelberg, 2010.

Appendix A.

A.1. Proof of Concept to Demonstrate the PINQ's Weakness

The programming frameworks introduced in this paper comes with the guarantee that an analyst with the permission to run arbitrary analyses on a database will not learn more than the limited private budget specified by the administrator. An attack that can break any of the following guarantees, beyond the limited probability of e^{ϵ}, is considered a successful privacy breach:

- Determine whether an individual is in the database or not.
- Extract the individuals properties (age, salary, ...).

Given information about an item i (here number 7), the goal of proof of concept code in Listing 1 (simplified in Algorithm 3) is to determine the presence of the item in the dataset D. It is easy to show that the second attack is a form of the first attack if the attacker enumerate over possible value of the property to find which value in the domain of the private property appears in the dataset.

The first step in the proof of concept is to partition the dataset D into two parts according to satisfying the equality with the target element i. As a result $K = \{x | x \in D, \lambda x \rightarrow x == i\}$ and $\bar{K} = D \setminus K$. Note that K and $\bar{K}$ share ϵ budget since they are disjoint subsets and PINQ's budget management system request only ϵ when two ϵ-differential private query are executed on K and $\bar{K}$. Figure 1 demonstrate how request for privacy budget flows between intermediate protected objects of Algorithm 3 when a query with ϵ cost is executed on the protected object containing the result of $\bar{K} \cup \mathrm{Take}(K \cup r, 1)$.

The following transformations that are used in this sample code don't scale up the stability since they are all 1-stable over their arguments:

- addition of r to the K using union
- taking the first element
- merging the result with the partition $\bar{K}$ using union

If item i is present in D, the partition K has only the item i. In this case taking the first element of $K \cup r$ gives i and when we merge $\bar{K}$ and $\mathrm{Take}(K \cup r, 1)$ together using union transformation, we end up with the starting dataset D. The interesting case happens when i is not in D, in this case K is empty and take returns the random item r. The final merge (union) of $\bar{K}$ and $\mathrm{Take}(K \cup r, 1)$ results in $D \cup r$. Performing this procedure several times to add several arbitrary random items makes the outcomes so different that they are distinguishable even after a differentially private counting query with a low value of ϵ.

Listing 1. Proof of concept code

```
// Secure initialisation
double[] rawData =
  new double[] {1,2,3,4,5,6,8,9,10};
var sourceD = rawData.ToArray().AsQueryable();
var agent= new PINQAgentBudget(1.0);
var D = new
  PINQueryable<double> (sourceD, agent);
```

Algorithm 3 Proof of concept code that reveals the existence of the item i in the protected dataset D

```
procedure REVEAL($i, D$)
    for $s \leftarrow 1, scale$ do
        $(K, \bar{K}) \leftarrow$ Partition$(D, \lambda x \rightarrow (x == i))$
        $r \leftarrow$ RandomItem()            ▷ random $r$ $(r \neq i)$
        $D \leftarrow$ Take$(K \cup r, 1) \cup \bar{K}$
    return NoisyCount$_{\epsilon}(D)$
```

$$\begin{tikzcd}
 & \boxed{K} \arrow[r, bend left, "\epsilon"] & \boxed{K \cup r} \arrow[dr, "\epsilon"] & \\
\boxed{D} \arrow[ur, "\epsilon"] \arrow[dr, "\epsilon"'] & & & \boxed{\mathrm{Take}(K \cup r, 1)} \arrow[d, "\epsilon"] \\
 & \boxed{\bar{K}} \arrow[rr, "\epsilon"'] & & \boxed{\bar{K} \cup \mathrm{Take}(K \cup r, 1)}
\end{tikzcd}$$

Figure 1. privacy budget request between protected objects

```
double[] invalid = new double[] { 999 };
double i = 7; // The item we are looking for
int scale = 1000;
bool[] keys = new bool[] {true,false} ;
for (int s = 0; s < scale; s++) {
  invalid[0] += 0.0001; // a unique item (random)
  var r = invalid.ToArray().AsQueryable();
  var parts = D.Partition
   (keys, x => (x==i)? true :false);
  var K = parts [true];
  var K_ = parts [false];
  D = K_.Union(
    K.Union(r).Take (1)
  );
}
Console.WriteLine("Noisy result:\t\t{0:F4}",
  D.NoisyCount(0.01));
```

Appendix B.

Probabilistic Stability of Uniform Sampling (Without Replacement)

Proof. We want to find the probabilistic stability $\sigma(\epsilon)$ such that $e^{-\sigma(\epsilon)} \leq \frac{\Pr[M(RS(D \cup \{e\})) \subseteq S]}{\Pr[M(RS(D)) \subseteq S]} \leq e^{\sigma(\epsilon)}$. We fix S, an arbitrary query result in the range of M. By using the recursive form of $\overrightarrow{RS(D \cup \{e\})}$ (Theorem 4.4) and the law of total probability over $\overrightarrow{RS(D)}$, we have:

$$\frac{\Pr[M(RS(D \cup \{e\})) \subseteq S]}{\Pr[M(RS(D)) \subseteq S]}$$
$$= \frac{\Pr[M(\psi_D(RS(D), e)) \subseteq S]}{\Pr[M(RS(D)) \subseteq S]}$$
$$= \frac{\sum_{K \in \overrightarrow{RS(D)}} \Pr[M(\psi_D(K, e)) \subseteq S]. \Pr[RS(D) = K]}{\sum_{K \in \overrightarrow{RS(D)}} \Pr[M(K) \subseteq S]. \Pr[RS(D) = K]}$$

Then, by equality of all samples probability (Definition 6), we derive:

$$= \frac{\alpha. \sum_{K \in \overrightarrow{RS(D)}} \Pr[M(\psi_D(K, e)) \subseteq S]}{\alpha. \sum_{K \in \overrightarrow{RS(D)}} \Pr[M(K) \subseteq S]}$$

Let us define $K^{\pm i} = \{K' \big| |K \, \Delta \, K'| = i\}$ a function defining Hamming distance-based equivalence classes on neighbourhoods. Thus, by law of total probability over $\mathbb{N}$, we have:

$$= \frac{\sum_{K\in\overrightarrow{RS(D)}} \sum_{i\in\mathbb{N}} \begin{matrix}\Pr[\psi_D(K,e)\in K^{\pm i}]. \\ \Pr[M(K^{\pm i})\subseteq S]\end{matrix}}{\sum_{K\in\overrightarrow{RS(D)}} \Pr[M(K)\subseteq S]}$$

We now give an over-approximation by relying on the fact that all probabilities are greater than zero. By differential privacy (Definition 1), we have $\Pr[M(K^{\pm i})\subseteq S] \leq e^{i\epsilon}\Pr[M(K)\subseteq S]$. Similarly for the lower-bound, we have $e^{-i\epsilon}\Pr[M(K)\subseteq S] \leq \Pr[M(K^{\pm i})\subseteq S]$.

Thus, for any i,

$$\leq \frac{\sum_{i\in\mathbb{N}} \sum_{K\in\overrightarrow{RS(D)}} \Pr[M(K)\subseteq S].e^{i\epsilon}.\beta_i}{\sum_{K\in\overrightarrow{RS(D)}} \Pr[M(K)\subseteq S]}$$

with $\beta_i = \Pr[\psi_D(K,e)\in K^{\pm i}]$.

Finally, by Observation 1 in [7] and simplification, we get the following upper-bound:

$$\leq \sum_{i\in\mathbb{N}} e^{i\epsilon}.\Pr[|\psi_D(K,e)\,\Delta\,K| = i]$$

We have a similar lower-bound by the same approximation.

This allows us to conclude the following result: $\sigma(\epsilon) \leq \ln \sum_{i\in\mathbb{N}} e^{i\epsilon}.\Pr[|\psi_D(K,e)\,\Delta\,K| = i]$ for any K, D, e, and ψ. □

Probabilistic Stability of Uniform Partitioning (Without Replacement)

Proof. We follow a similar proof technique and almost identical steps as the previous part, to find probabilistic stability $\hat{\sigma}(\epsilon)$ such that $e^{-\hat{\sigma}(\epsilon)} \leq \frac{\Pr[\hat{M}(\hat{RS}(D\cup\{e\}))\subseteq S]}{\Pr[\hat{M}(\hat{RS}(D))\subseteq S]} \leq e^{\hat{\sigma}(\epsilon)}$ we fix $S = \langle S_1, S_2\rangle$ to be a set of pairs of results in the range of $\hat{M}$ [2]. Using recursive form of RS and law of total probability.

$$\frac{\Pr[\hat{M}(\hat{RS}(D\cup\{e\}))\subseteq S]}{\Pr[\hat{M}(\hat{RS}(D))\subseteq S]}$$

$$= \frac{\sum_{\langle K,\overline{K}\rangle\in\overrightarrow{\hat{RS}(D)}} \begin{bmatrix}\Pr[\hat{RS}(D) = \langle K,\overline{K}\rangle]. \\ \Pr[\hat{M}(\hat{\psi_D}(\langle K,\overline{K}\rangle, e)\subseteq S]\end{bmatrix}}{\sum_{\langle K,\overline{K}\rangle\in\overrightarrow{\hat{RS}(D)}} \begin{bmatrix}\Pr[\hat{RS}(D) = \langle K,\overline{K}\rangle]. \\ \Pr[\hat{M}(\langle K,\overline{K}\rangle)\subseteq S]\end{bmatrix}}$$

Knowing $K\cup\bar{K} = D$, after introducing e, $K'\cup\bar{K}' = D\cup\{e\}$ such that K' is the result of function $\psi_D(K,e)$, if $|K\,\Delta\,K'| = i$ then $|\bar{K}\,\Delta\,\bar{K}'| = |i-1|$.

$$\frac{\alpha.\sum_{K\in\overrightarrow{RS(D)}} \left[\sum_{i\in\mathbb{N}} \begin{bmatrix}\Pr[\hat{\psi_D}(\langle K,\overline{K}\rangle, e)\in\langle K^{\pm i},\overline{K^{\pm i}}\rangle]. \\ \Pr[\langle M_1(K^{\pm i}), M_2(\overline{K^{\pm i}})\rangle\subseteq S]\end{bmatrix}\right]}{\alpha.\sum_{K\in\overrightarrow{RS(D)}} \Pr[\langle M_1(K), M_2(\overline{K})\rangle\subseteq S]}$$

2. $\hat{RS}$ is lifted function from sampling method RS that creates two disjoint partitions K and $\overline{K}$. Similarly $\hat{M}$ is lifted function that run two queries M_1 and M_2 on these two partitions.

$$= \frac{\sum_{K\in\overrightarrow{RS(D)}} \sum_{i\in\mathbb{N}} \beta_i.\Pr[\langle M_1(K^{\pm i}), M_2(\overline{K^{\pm i}})\rangle\subseteq S]}{\sum_{K\in\overrightarrow{RS(D)}} \Pr[\langle M_1(K), M_2(\overline{K})\rangle\subseteq S]}$$

$$\leq \frac{\sum_{i\in\mathbb{N}} \sum_{K\in\overrightarrow{RS(D)}} \beta_i.\begin{bmatrix}e^{i\epsilon}.\Pr[M_1(K)\subseteq S_1] \\ e^{|i-1|\epsilon}.\Pr[M_2(\overline{K})\subseteq S_2]\end{bmatrix}}{\sum_{K\in\overrightarrow{RS(D)}} \Pr[M_1(K)\subseteq S_1].\Pr[M_2(\overline{K})\subseteq S_2]}$$

with $\beta_i = \Pr[\psi_D(K,e)\in K^{\pm i}]$.

Simplifying the expression we have probabilistic stability of $\hat{\sigma}(\epsilon) = \ln\sum_{i\in\mathbb{N}} e^{(i+|i-1|)\epsilon}.\Pr[|\psi_D(K,e)\,\Delta\,K| = i]$ □

B.1. Recursive (Fixed Sized) Uniform Sampling (Without Replacement)

Proof. In the case where $n \geq |D\cup\{e\}|$, the probability of the sample to be $D\cup\{e\}$ is 1.

For the case where $n < |D\cup\{e\}|$, we rely on the inductive definition of $R_n(D\cup\{e\})$. Two cases are possible:

Case $e\in X : \Pr[R_n(D\cup\{e\}) = X]$

$$= \sum_{a\in D\cup\{e\}\setminus X} \begin{bmatrix}\Pr[R_n(D) = X\cup\{a\}\setminus\{e\}] \\ \Pr[\psi_{n,|D|}(X\cup\{a\}\setminus\{e\}, e) = X]\end{bmatrix}$$

$$= \underbrace{(|D|+1-n)}_{\text{from summation}}.\binom{|D|}{n}^{-1}.(|D|+1)^{-1}$$

$$= \binom{|D|+1}{n}^{-1}$$

Case $e\notin X : \Pr[R_n(D\cup\{e\}) = X]$

$$= \Pr[R_n(D) = X].\Pr[\psi_{n,|D|}(X,e) = X]$$

$$= \binom{|D|}{n}^{-1}.\frac{|D|+1-n}{|D|+1}$$

$$= \binom{|D|+1}{n}^{-1}$$

In both cases, $|D|+1-n$ makes match the expected probabilities. □