Implementing Differentially Private Integer Partitions via the Exponential Mechanism (Working Paper)

Christina Ilvento

1 Introduction

In addition to the core use-cases of releasing differentially private estimates of database statistics and counting queries, significant progress has been made in releasing differentially private integer partitions or frequency lists. A frequency list is a list of frequencies in sorted order, e.g., an ordered list of number of visitors to the top 10 most popular websites. Although a list of frequencies alone may seem less useful than a histogram including cell labels, this type of data summary is highly useful if the cell labels themselves are sensitive, e.g. passwords. Recent work of Blocki, Datta and Bonneau [1] proposed an elegant solution to the problem of releasing differentially private frequency lists based on the exponential mechanism. By using differential privacy and giving a formal guarantee on the privacy loss due to releasing frequency statistics, they were able to get access to and publish statistics for a Yahoo! password dataset that had previously been inaccessible to researchers. Their solution includes two key insights: (1) The weight of any potential outcome can be split into a prefix and a suffix, which allows for efficient computation of weights via dynamic programming and an iterative sampling procedure. (2) The infinite outcome space of all integer partitions can be scoped down by realizing that the probability of selection of any potential outcome with utility cost larger than d is small and can be translated into an acceptable value of δ for approximate differential privacy.

In this work, we examine the problem of extending and implementing the method of Blocki et al. Our contributions include: (1) A pure $\varepsilon-$DP extension of the method that allows for domain-specific adaptations to conserve privacy budget and improve accuracy. (2) A construction for sampling differentially private integer partitions of size exactly n. (3) An efficient method to compute the bias of the procedure, which can assist in answering meta-queries such as, how many unique passwords are in the dataset? (4) We demonstrate the pitfalls of inexact implementation (in the case of approximate DP) and show how to implement the mechanism exactly based on the base-2 DP exponential mechanism of [2]. We also provide a reference implementation in Rust.[1]

2 Preliminaries and terminology

Base-2 Differential Privacy. Base-2 differential privacy ([2]) re-casts differential privacy in terms of 2^η rather than e^ε, and there is a straightforward equivalence in the definitions for $\eta \ln(2) = \varepsilon$.

Definition 2.1. A randomized mechanism $\mathcal{M}$ is said to be $\eta|_{\mathbf{2}}$**-differentially private** if for every pair of adjacent databases D and D', $\Pr[\mathcal{M}(D) \in O] \leq 2^\eta \Pr[\mathcal{M}(D') \in O]$.

Although the definition is theoretically the same as base-e DP, by using base-2 and making appropriate choices for η, i.e., $\eta = -z \log_2(\frac{x}{2^y})$ for positive integer x, y, z such that $\frac{x}{2^y} < 1$, 2^η can be evaluated exactly with limited precision, which greatly aids in exact implementation of differentially private mechanisms.

[1] The original mechanism implementation of [1] in C# was graciously shared with us during the preparation of this work.

Integer Partitions. We follow the terminology and naming conventions of [1] wherever possible. An integer partition of n is any set of integers $x = \{x_1, x_2, ..., x_n\}$ such that $\sum_i x_i = n$, and $x_i \geq x_{i+1} \geq 0$. The set of integer partitions of n is written $\mathcal{P}(n)$ and the set of all partitions is written $\mathcal{P}$. We use x, y or f to denote integer partitions, $\tilde{f}$ to denote a differentially private version of f, f_i to indicate the i^{th} entry in f and f^i to indicate the prefix of f up to and including the i^{th} entry. Given a private integer partition, the goal is to output an integer partition that is "close" to the original but is not influenced too much by any single individual in the underlying dataset. Blocki, et al. defined "close" via the $\mathbf{dist}$ function: given $x \in \mathcal{P}(n)$ $x' \in \mathcal{P}(n')$, $\mathbf{dist}(x, x') := \sum_{i=1}^{\max(n,n')} |x_i - x'_i|$, where x_i, x'_i are defined to be zero for any $i > n$ or n'. We say that two partitions are adjacent (written $x \sim x'$) if at most one individual is added or removed from the dataset.

3 The Integer Partition Exponential Mechanism

The primary challenge with sampling integer partitions based on their $\mathbf{dist}$ from the target private partition using the exponential mechanism is the infinite outcome space. Instead, Blocki et al. proposed sampling only from partitions within $\mathbf{dist} = d$ of the private partition, such that the total weight of all partitions further than d is at most δ, resulting in an approximate-DP mechanism.[2] To characterize this set of partitions, it's very convenient to think in terms of upper and lower bounds on the partition. In particular, we determine the maximum and minimum possible values at index j for any partition within distance d of the target x, and define the set $\mathcal{P}_{L^d}^{U^d}$ of allowed partitions:

$$U_j^d := \max_{\substack{x \in \mathcal{P} \\ \mathbf{dist}(f,x) \leq d}} x_j \qquad L_j^d := \min_{\substack{x \in \mathcal{P} \\ \mathbf{dist}(f,x) \leq d}} x_j \qquad \mathcal{P}_{\mathbb{L}}^{\mathbb{U}} := \{x \in \mathcal{P} \mid \nexists x_i \notin [\mathbb{L}_i, \mathbb{U}_i]\} \tag{1}$$

We can now give a more formal definition of the $\delta-$negligible integer partition exponential mechanism.

Mechanism 1 ($\delta-$negligible Integer Partition Exponential Mechanism [1])**.** Given a private integer partition x, a distance d, and privacy parameter ε, compute the bounds U^d and L^d as specified in Equation 1. Release an output y sampled according to the probability distribution induced by the exponential mechanism over the outcome space $\mathcal{P}_{L^d}^{U^d}$, i.e., $\Pr[\mathcal{E}^{\varepsilon}_{\mathbf{dist}(f\cdot)}(\mathcal{P}_{L^d}^{U^d}) = y] = e^{-\frac{\varepsilon}{2}\mathbf{dist}(x,y)}(\sum_{y \in \mathcal{P}_{L^d}^{U^d}} e^{-\frac{\varepsilon}{2}\mathbf{dist}(x,y)})^{-1}$.

Dynamic programming can be used to sample efficiently from this distribution by pre-computing the cumulative weight $w_{i,Q,\varepsilon}$ of all valid completions at index i that begin with value $Q \in [L_i^d, U_i^d]$. This naturally implies an iterative sampling procedure which chooses Q as the next value with probability proportional to the cumulative weight of all valid completions starting with Q in the next index. These quantities are summarized in Equation 2.

$$w_{i,Q,\varepsilon} := \sum_{\substack{x \in \mathcal{P} \setminus S_{f^i} \\ x_1 = Q}} e^{-\frac{\varepsilon}{2}\mathbf{dist}(f^i,x)} \qquad \Pr[\tilde{f}_i = Q \mid \tilde{f}_{i-1} = T] = \frac{w_{i,Q,\varepsilon}}{\sum_{Q'=\mathbb{L}_i}^{\min\{T,\mathbb{U}_i\}} w_{i,Q',\varepsilon}} \tag{2}$$

We omit the full details due to lack of space, but we remark that it is possible to sample restricted to partitions of exactly size n (if the size of the partition is public) by expanding the dynamic programming table to include the required suffix sum to achieve the desired count. This comes at a cost of a factor of n in pre-computation time.

[2] Detailed discussion of $\delta-$negligible set families as well as proofs of the stated properties can be found in Theorem 2 and Lemma 1 of [1].

Pitfalls of inexact implementation. A critical issue facing implementations of the approximately differentially private mechanism is the question of how inexact implementation affects the weights in the "negligible" (δ) region. For example, the weights of partitions with very large distances from the target partition may be rounded to zero. On the other hand, rounding weights up or setting a fixed minimum weight will impact the total weight in the "negligible" part of the outcome space. While we demonstrate that rounding errors in the inexact implementation may result in different privacy guarantees, we stress that we have no evidence to suggest that any prior work would have resulted in significant differences in privacy loss under any reasonable threat model. To address this problem, we implement the mechanism exactly based on the base-2 exponential mechanism of [2]. The logic of the mechanism remains unchanged, but we switch to privacy parameters for base-2, enforce exact arithmetic for cumulative weight computations, and use the NormalizedSample method of [2] to sample from the cumulative weights at each index. By implementing mechanism exactly, we simplify the proof of privacy for the implementation itself, as privacy follows only from correctness of the implementation, and does not rely on any system dependent approximations. This is also critical for achieving pure DP implementations, as discussed in Section 4.

Pure $\varepsilon-$DP Integer Partitions It was stated informally in [3] that the approximately private method could be extended to pure $\varepsilon-$DP. The most straightforward way to get pure DP is to use naive data independent bounds, i.e., by setting $\mathbb{U}_i = n$ and $\mathbb{L}_i = 0$ for all i. While this extension suffices for proving ε-DP, it comes at the cost of a running time of $O(n^3)$. We can get tighter bounds by observing that the maximum value for x_i for any $x \in \mathcal{P}(n)$ is $\frac{n}{i}$, and in fact we show that *any* bounds derived independently of the private partition will result in a pure $\varepsilon-$DP algorithm.

Mechanism 2 (Fixed Bound Integer Partition Exponential Mechanism)**.** Given a private integer partition x, privacy parameter η and bounds $\mathbb{U}$ and $\mathbb{L}$ st $\mathbb{U}_i \geq \mathbb{L}_i \geq 0$, release an output y sampled according to the probability distribution induced by the exponential mechanism over the outcome space $\mathcal{P}^{\mathbb{U}}_{\mathbb{L}}$, i.e., $\Pr[\mathcal{E}^{2\eta}_{\mathbf{dist}(x,\cdot)}(\mathcal{P}^{\mathbb{U}}_{\mathbb{L}}) = y] = 2^{-\eta\mathbf{dist}(x,y)}(\sum_{y\in\mathcal{P}^{\mathbb{U}}_{\mathbb{L}}} 2^{-\eta\mathbf{dist}(x,y)})^{-1}$.

Theorem 3.1. *Given $\mathbb{U}$ and $\mathbb{L}$ which are considered public information, the Fixed Bound Integer Partition Exponential Mechanism (Mechanism 2) is $2\eta|_2-$DP (or $2\ln(2)\eta-$DP).*

Proof. Mechanism 2 is an instance of the exponential mechanism with outcome space $O = \mathcal{P}^{\mathbb{U}}_{\mathbb{L}}$ and utility function $\mathbf{dist}$ where $\Delta\mathbf{dist} = 1$. □

The choice of bounds can improve running time, privacy budget usage and accuracy. For example, upper and lower bounds can be computed by using a small amount of the privacy budget to draw very noisy estimates of a subset of the partition, e.g., every k^{th} index. Alternatively, if a reference or historical partition is already public, fixed width bounds centered on the public partition or bound computed by setting a distance d from the *public* partition, where d is either informed by common sense or estimated with a small amount of privacy budget can be used. Furthermore, in cases when the partition is known to be sparse, i.e., most values are zero, a small amount of privacy budget can be used to truncate the bounds to reduce the running time. If the bounds can be constructed with a maximum width w, the running time of the mechanism becomes dependent on the number of cells scaled by a factor of w, which is significantly better than $O(n^3)$ for the naive data-independent bounds.

Computing mechanism bias Although the integer partition exponential mechanism has good accuracy properties, it is generally going to be biased and inconsistent. That is, $\mathbb{E}[\tilde{f}_i] \neq f_i$. Why? Integer partitions are non-negative, and as such values of zero must have positively biased noise to maintain differential privacy.

Algorithm 1

***Inputs**: target partition f, privacy parameter η, upper and lower bounds st* $\mathsf{len}(\mathbb{U}) = \mathsf{len}(\mathbb{L}) = n$.
***Outputs**:* $y \in \mathcal{P}_{\mathbb{L}}^{\mathbb{U}}$

procedure SAMPLE($f, \eta, \mathbb{U}, \mathbb{L}$)
  Initialize empty tables to store Weights
  $y_0 = \mathbb{U}_1$ ▷ Maximum possible value of y_1.
  $n \leftarrow \mathsf{len}(\mathbb{U})$
  for $i = 1, \ldots, n$ **do**
    $Q_{max} = \min\{y_{i-1}, \mathbb{U}_i\}$
    Initialize an empty list of weights W
    for $S = \mathbb{L}_i, \ldots, Q_{max}$ **do**
      $W_S \leftarrow$ COMPWEIGHTS(S, i, f, η)
      Append W_S to W
    $j \leftarrow$ NORMALIZEDSAMPLE(W)
    $y_i \leftarrow \mathbb{L}_i + j$ ▷ Value for index j.
  $y \leftarrow (y_1, \ldots, y_n)$
  return y

***Inputs**: The candidate size Q, the index i, upper and lower bounds $\mathbb{U}$ and $\mathbb{L}$, target integer partition x, and privacy parameter η. **Output**: Cumulative weight of all valid completions with Q at index i ($w_{i,Q,\eta}$).*

function COMPWEIGHTS(Q, i, x, η)
  $key \leftarrow (Q, i)$
  if weights.$Contains(key)$ **then**
    return weights.$GetValue(key)$
  if $Q \notin [\mathbb{L}_i, \mathbb{U}_i]$ **then**
    return 0
  else if $\mathbb{U}_i = 0$ & $Q = 0$
    return 1
  $val \leftarrow 0$
  if $Q \geq \mathbb{L}_i + 1$ **then** ▷ Optimize.
    $\nu \leftarrow 1$ if $Q \geq x_i + 1$; -1 otherwise.
    $val \leftarrow val + 2^{-\eta\nu} *$
      COMPWEIGHTS($Q - 1, i, x, \eta$)
    if $Q \leq \mathbb{U}_{i+1}$ **then**
      $val \leftarrow val + 2^{-\eta|Q - x_i|} *$
        COMPWEIGHTS($Q, i + 1, x, \eta$)
  else
    for $Q' = \mathbb{L}_{i+1}, \ldots, \min\{Q, \mathbb{U}_{i+1}\}$ **do**
      $val \leftarrow val + 2^{-\eta|Q - x_i|} *$
        COMPWEIGHTS($Q', i+1, x, \eta$)
  weights.$Add(key, val)$
  return val

Fortunately, we can write a simple expression for this bias: $B(f)_i = f_i - \mathbb{E}[\tilde{f}_i] = \sum_{Q \in [\mathbb{L}_i, \mathbb{U}_i]} \Pr[\tilde{f}_i = Q] \cdot (f_i - Q)$. This bias can be computed efficiently via dynamic programming, by noticing that $\Pr[\tilde{f}_i = Q]$ can be written as a sum over the prefixes ending in $T \geq Q$, i.e., $\Pr[\tilde{f}_i = Q] = \sum_{T=\max(\mathbb{L}_{i-1}, U)}^{\mathbb{U}_{i-1}} \Pr[\tilde{f}_{i-1} = T] \cdot \Pr[\tilde{f}_i = Q | \tilde{f}_{i-1} = T]$. As $w_{i,Q,\varepsilon}$ can be computed for all relevant Q, i pairs, we can substitute $\Pr[\tilde{f}_i = Q | \tilde{f}_{i-1} = T] = \frac{w_{i,Q,\varepsilon}}{\sum_{V=\mathbb{L}_i}^{\min(\mathbb{U}_i, T)} w_{i,V,\varepsilon}}$ yielding the recursive formula $\Pr[\tilde{f}_i = Q] = \sum_{T=\max(\mathbb{L}_{i-1}, Q)}^{\mathbb{U}_{i-1}} \Pr[\tilde{f}_{i-1} = T] \cdot \frac{w_{i,Q,\varepsilon}}{\sum_{V=\mathbb{L}_i}^{\min(\mathbb{U}_i, T)} w_{i,V,\varepsilon}}$.

4 Implementation Details

A reference implementation of Algorithm 1 as well as bias computation in Rust is available on github. We also implement methods for computing U^d and L^d as specified in Equation 1, with a log factor improvement over the versions specified in [1]. We also implement methods for computing bounds from noisy estimates as well as other strategies. The core mechanism implementation relies on NORMALIZEDSAMPLE and privacy parameter construction of [2], and is reasonably efficient. (Please see github for benchmarks.) Necessary precision is determined by trial and error, which can introduce a timing channel. Given that most applications of this mechanism are likely to be run in a batch or offline mode and thus won't leak fine-grained timing information, we did not prioritize timing channel prevention in this implementation.

5 Discussion and Future Work

In this work, we have presented extensions to the integer partition exponential mechanism proposed by Blocki, Datta and Bonneau and have given exact reference implementations via base-2 DP. By implementing exactly and relying on fixed, data-independent (or privately computed) bounds, we can achieve an efficient pure DP implementation without concerns of privacy loss due to approximation. Integer partitions are useful in many settings, and concurrent work depends on the pure DP version of this mechanism for the construction of differentially private integer histograms.

References

[1] Jeremiah Blocki, Anupam Datta, and Joseph Bonneau. 2016. Differentially Private Password Frequency Lists.. In *NDSS*, Vol. 16. 153.

[2] Christina Ilvento. 2019. Implementing the Exponential Mechanism Exactly with Base-2 Differential Privacy. (2019). https://arxiv.org/abs/1912.04222

[3] Jeremiah Blocki Microsoft. 2016. Differentially Private Integer Partitions and their Applications.