Implementing the sparse vector technique and variants

Vulnerabilities and mitigation via exact implementation (Working Paper)

Christina Ilvento

Harvard University, cilvento@g.harvard.edu

Abstract

The "sparse vector" or "above threshold" technique is extremely useful for answering a large number of threshold queries when very few are expected to be positive. Unfortunately, there have been a number of issues proving the correctness of proposed variants of sparse vector. Significant progress has been made recently in automated validation of proofs of correctness for the sparse vector technique, but these proofs do not directly address the issues of implementing sparse vector in practice. In particular, several proofs based on "randomness alignment," rely on continuous or infinite support discrete random variables and care must be taken to translate these techniques to finite precision implementations.

In this work, we present a purely differentially private discrete version of sparse vector and the discrete Laplace mechanism which can be implemented exactly using open-source multiple-precision arithmetic libraries. In addition to algorithms and privacy analysis, we provide a Rust reference implementation.

1 Introduction

In general, differentially private mechanisms must make a trade-off between the number of queries that can be answered and the accuracy of those answers. The sparse vector technique is extremely useful in that it allows answering an arbitrarily large number of threshold queries, so long as only a fixed number exceed the threshold. Although widely studied, the proof of privacy for sparse vector is subtle and many incorrect or unnecessarily conservative versions of the proof or extensions of the technique have been proposed. See Lyu, Su and Li [6] for a detailed survey of variants and proof issues.

In this work, we examine the practical problem of implementing the sparse vector technique. Although proofs of correctness have been validated using automated theorem proving (e.g., [9, 8]) translating the mechanism from theory to practice is non-trivial. Our contributions include:

1. An explanation of the practical issues with infinite support randomness alignment, a popular technique for proving privacy for sparse vector.

2. Discrete versions of the sparse vector and clamped Laplace mechanisms which can be implemented exactly using base-2 differential privacy, preventing privacy loss due to approximation.

3. A Rust reference implementation of the discrete sparse vector and clamped Laplace mechanisms (as well as building blocks including noisy threshold).

2 Preliminaries

Differential privacy

It's well understood that "anonymization" techniques which remove so-called "personally identifiable information" are woefully inadequate for privacy protection. Differential privacy, in contrast, is a mathematically rigorous formulation for privacy preserving data analysis which requires that the outcome of any analysis cannot change "too much" based on the data from a single individual [2]. More formally:

Definition 2.1. A randomized mechanism $\mathcal{M}$ is said to be ε-**differentially private** if for every pair of adjacent databases D and D', i.e., databases differing in a single row, $\Pr[\mathcal{M}(D) \in O] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in O]$.

For example, suppose we have a database D containing medical records for the students at a given university, and we want to publish the number of students who have tested positive for a virus in a privacy preserving manner. That is, we want to release an output which is informative, i.e., it closely approximates the true number of students who have tested positive, but observing the output should not give away too much information about whether any specific student has tested positive. To do this, we might use the Laplace Mechanism:

Mechanism 1 (Laplace Mechanism)**.** Given a query $q : \mathcal{D} \to \mathbb{R}$ with sensitivity $\Delta q := \max_{D \sim D' \in \mathcal{D}} |q(D) - q(D')|$, a privacy parameter ε and a database D, release $q(D) + \xi$ where ξ is drawn from the sensitivity scaled Laplace distribution, given by probability density $\mathsf{Lap}(\mathsf{t}|\frac{\Delta \mathsf{q}}{\varepsilon}) = \frac{\varepsilon}{2\Delta q} e^{-\frac{\varepsilon |t|}{\Delta q}}$. The Laplace Mechanism is $\varepsilon-$DP.

In our example, the query "how many students have tested positive" has sensitivity 1, as adding or removing a single student from the database can impact the total count by at most 1.[1] One nice property of the Laplace mechanism is that its error is constant, i.e., the magnitude of the noise doesn't depend

[1] Note, in some cases, we may want to protect groups of individuals, e.g., pairs of roommates, families, etc. This method is easily extended to cover these cases by scaling the sensitivity. Please see Dwork and Roth for additional discussion of group privacy [3].

on the size of the database, rather it depends on the sensitivity of the query. Furthermore, differentially private (DP) mechanisms are robust to post-processing, (e.g., rounding or clamping), arbitrary side information and compose with additive privacy loss, (i.e., running an ε_1-DP mechanism followed by an ε_2-DP mechanism has total privacy loss $\varepsilon_1 + \varepsilon_2$).

In practice, implementing differentially private mechanisms that match proven theoretical privacy guarantees can be problematic, as evaluations of e^ε or $\ln(x)$ will necessarily be approximate. Mironov demonstrated an attack on the Laplace mechanism based on inexact floating point implementations, and proposed the snapping mechanism as an alternative implementation [7]. Attacks based on inexact implementation of the exponential mechanism have also been demonstrated, and base-2 differential privacy was proposed to allow for exact implementations to circumvent these issues [4]. Base-2 differential privacy re-casts privacy loss in terms of 2^η rather than e^ε, and there is a straightforward equivalence in the definitions for $\eta \ln(2) = \varepsilon$.

Definition 2.2. A randomized mechanism $\mathcal{M}$ is said to be **$\eta|_2$-differentially private** if for every pair of adjacent databases D and D', $\Pr[\mathcal{M}(D) \in O] \leq 2^\eta \Pr[\mathcal{M}(D') \in O]$.

For appropriate choices of η, namely $\eta = -z \log_2(\frac{x}{2^y})$ for positive integer x, y, z such that $\frac{x}{2^y} < 1$, 2^η can be evaluated exactly. Please see [4] for details of required precision, etc.

By considering privacy directly in terms of 2^η, it's clearer to see how we might propose mechanisms which can be implemented exactly. For example, consider the Discrete Laplace mechanism:

Mechanism 2 (Discrete Laplace)**.** Given a query $q : \mathcal{D} \to \gamma\mathbb{N}$, where $\gamma\mathbb{N}$ denotes all integer multiples of γ, with sensitivity $\Delta q := \max_{D \sim D' \in \mathcal{D}} |q(D) - q(D')|$, a privacy parameter η and a database D, release $q(D) + \xi$ where ξ is drawn from the discrete Laplace distribution

$$\Pr[\mathsf{DiscLap}^\gamma(\frac{1}{\eta}) = k\gamma \mid k \in \mathbb{N}] = \frac{2^{-\eta\gamma|k|}}{\sum_{k\gamma \in \gamma\mathbb{N}} 2^{-\eta\gamma|k|}} \quad (1)$$

The Discrete Laplace Mechanism is $\eta\Delta q|_2-$DP.

Notice that if γ is an integer, then sampling from the Discrete Laplace mechanism consists of computing powers of $2^{-\eta}$ and normalized sampling. (See [4] for details of implementation of exact normalized sampling without division.) Although DiscLap as defined in Mechanism 2 requires a random variable with infinite support, we'll see how the *clamped* distribution can be simulated with finite precision in Section 3. The clamped discrete Laplace distribution is described by the probability distribution:

$$\Pr[\mathsf{ClampedDiscLap}^\gamma(\frac{1}{\eta}, A, B) = k\gamma \mid k \in \mathbb{N}] \quad (2)$$

$$= \frac{1}{\sum_{k\gamma \in \gamma\mathbb{N}} 2^{-\eta\gamma|k|}} \begin{cases} 0 \text{ for } k\gamma \notin [A, B] \cap \gamma\mathbb{N} \\ 2^{-\eta\gamma|k|} \text{ for } k\gamma \in (A, B) \cap \gamma\mathbb{N} \\ \sum_{k=-\infty}^{k\gamma = A} 2^{-\eta\gamma|k|} \text{ for } k\gamma = A \\ \sum_{k\gamma = B}^{\infty} 2^{-\eta\gamma|k|} \text{ for } k\gamma = B \end{cases}$$

Sparse Vector and Randomness Alignment

The sparse vector technique compares a stream of queries to a noisy threshold, and outputs $\perp$ or $\top$ depending on whether each (noisy) query exceeds the noisy threshold. The key insight to this technique is that by adding noise to the threshold, the mechanism only has to "pay" for queries that output $\top$. Algorithm 1, reproduced from [6], outlines the procedure.

Algorithm 1 Sparse Vector of [6]

Inputs*: private database D, stream of queries $Q = q_1, q_2, \ldots$ each with sensitivity bounded by Δ, a set of thresholds $\mathbf{T} = T_1, T_2, \ldots$, maximum number of $\perp$ output(s) c, and privacy parameter ε.* ***Outputs****: stream of answers $a_1, a_2, \ldots$ where $a_i \in \{\perp, \top\}$.*

```
1:  procedure SparseVector($D, Q, \Delta, \mathbf{T}, c, \varepsilon$)
2:      $\varepsilon_1 = \varepsilon/2$; $\varepsilon_2 = \varepsilon - \varepsilon_1$; $count = 0$
3:      $\rho \sim \mathsf{Lap}(\frac{\Delta}{\varepsilon_1})$
4:      for $q_i \in Q$ do
5:          $\nu_i \sim \mathsf{Lap}(\frac{2c\Delta}{\varepsilon_2})$
6:          if $q_i(D) + \nu_i \geq T_i + \rho$ then
7:              output $a_i = \top$
8:              $count = count + 1$
9:              if $count \geq c$ then
10:                 abort
11:         else
12:             output $a_i = \perp$
```

Lyu, Su and Li [6] provide a detailed description of the technique as well as a survey of incorrect or overly conservative variations on the technique. Their proof follows the general strategy of arguing that the difference in behavior of the mechanism on a database D given the noisy threshold z is not too different from the behavior of the mechanism on an adjacent database D' for the noisy threshold $z + \Delta$, where Δ is a bound on the sensitivity of the queries. This proof technique is also described as a general "template" for randomness alignment by Ding, Wang, Zhang and Kifer [1].

Loosely speaking, proofs based on randomness alignment first specify an alignment, that is a mapping $\phi_{D,D'} : \mathbb{R}^\infty \to \mathbb{R}^\infty$ which matches the randomness H which results in an output ω for the database D to the randomness H' which results in the same output for D'. Informally, given such an alignment, it suffices to show that the alignment is one-to-one and the "cost", i.e., the ratio of the probability of observing randomness H versus H', is bounded to prove privacy.

The challenge with translating these proofs to practical implementation is that good choices for alignments in theory may require that random variables have infinite support, which cannot always be directly implemented on finite computers. For example, there is a maximum value of the noisy threshold ρ_{max} that can be evaluated on a given system, and so mapping ρ_{max} to $\rho_{max} + \Delta$ is impossible, as $\rho_{max} + \Delta$ will never be observed for the neighboring database. While it's possible to treat the probability of sampling an extreme value of ρ as a failure case and achieve approximate (ε, δ) differential privacy, there are two significant caveats. (1) In the case of ρ_{max}, privacy loss can be arbitrarily large, and if proper clamping ordering is not enforced between $\nu_i + q_i(D)$ and $T_i + \rho$, an attacker can also achieve arbitrary privacy loss **for any** ρ via adversarial choice of threshold or queries. (Please see the accompanying github repository for an example of such an attack.) (2) Inexact implementation may incur additional privacy loss.

3 Discrete Sparse Vector

To produce a pure ε-DP implementation and avoid issues of approximation, we propose a discrete version of the sparse vector technique that can be implemented exactly with base-2 DP (Algorithm 2). We take all thresholds to be zero, which can be viewed as modifying each query by taking $\hat{q}_i(D) = q_i(D) - T_i$. The construction of the discrete version can be thought of in three steps: (1) switching to base-2 DP and discrete random variables with no logic changes, (2) adjusting the effective outcome space for ρ by applying Clamp, (3) sampling techniques to simulate sampling from the infinite discrete distributions with limited precision (Algorithm 3).

Algorithm 2 Discrete Sparse Vector

Inputs: *database D, query stream $Q = q_1, q_2, \dots$ each with sensitivity $\leq \Delta$ with values clamped to $[Q_{min}, Q_{max}]$, a maximum positive query count c, base-2 privacy parameter η, a width parameter w, and a granularity γ. All values are assumed to be integer multiples of γ.*

Outputs: *a stream of answers $a_1, a_2, \dots$ where each $a_i \in \{\bot, \top\}$.*

```
1:  procedure DiscSV($D, Q, Q_{min}, Q_{max}, \Delta, c, \eta, w, \gamma$)
2:      $\eta_1 \leftarrow \eta/2$; $\eta_2 \leftarrow \eta - \eta_1$
3:      $\hat{\rho} \leftarrow$ ClampedDiscLap($\frac{\eta_1}{\Delta}, \gamma, Q_{min} - w, Q_{max} + w$)
4:      $count \leftarrow 0$
5:      for $q_i \in Q$ do
6:          $\hat{\rho}_i =$ SymmetricClamp($\hat{\rho}, q_i(D), w$)
7:          $a_i \leftarrow$ LazyThreshold($\frac{\eta_2}{2c\Delta}, \hat{\rho}_i - q_i(D), \gamma$)
8:          output $a_i$
9:          if $a_i = \top$ then
10:             $count = count + 1$
11:             if $count \geq c$ then abort

    The symmetric clamping function, which clamps the value
    $\rho$ to the range $[x - w, x + w]$.
12: function SymmetricClamp($\rho, x, w$)
13:     if $\rho \in [x - w, x + w]$ then
14:         return $\rho$
15:     else if $\rho < x - w$ then
16:         return $x - w$
17:     else return $x + w$
```

Switching to discrete noise base-2. In each case where e^{ε} would have been used, we instead use 2^{η}. We fix a granularity γ and sample ρ and ν_i from the discrete Laplace mechanism at this granularity. That is, we sample from $\mathcal{O} = \gamma\mathbb{N}$ with probabilities according to the distribution in Equation 1 for privacy parameter η_1 or η_2. To keep the algorithm simple, we assume that $Q : D \to \gamma\mathbb{N}$, w, Q_{max}, Q_{min}, etc. are all in $\gamma\mathbb{N}$.

Modifying the outcome space of ρ. The key problem with the proof of privacy when only finite values of ρ are available is that we cannot match up ρ for the database D to $\rho + \Delta$ for D' for values of ρ near the maximum and minimum allowed magnitudes. To get around this issue, we use the Clamp procedure, which treats all values of $\rho < q_i(D) - w$ as $q_i(D) - w$ and all values of $\rho > q_i(D) + w$ as $q_i(D) + w$, i.e.

$$\textsc{Clamp}(\rho, x, w) := \begin{cases} \rho \text{ if } \rho \in [x - w, x + w] \\ x - w \text{ if } \rho < x - w \\ x + w \text{ if } \rho > x + w \end{cases} \tag{3}$$

Clamping based on the value of $q_i(D)$ may initially seem counter-intuitive, as we generally do *not* want outcome spaces to rely on private data. However, as we'll see below, centering $\hat{\rho}$ on $q_i(D)$ allows for simpler alignment.

Simulating sampling. To sample ρ (ClampedDiscLap), we first consider which values of ρ are actually meaningful to the mechanism: for $\rho \in (-\infty, q_i(D) - w) \cup (q_i(D) + w, \infty)$, Clamp reassigns ρ to $q_i(D) - w$ or $q_i(D) + w$, respectively, regardless of the exact value. Thus, to sample ρ, we first sample a *region* (either $(-\infty, q_i(D) - w)$, $(q_i(D) + w, \infty)$, or $[q_i(D) - w, q_i(D) + w]$) based on the sum of the weights of the elements of $\gamma\mathbb{N}$ in each region. If ρ falls in either extreme region, we simply assign the value Clamp would assign, and we don't bother to sample the specific value of ρ. Only if ρ falls in the middle region do we sample a specific value for ρ. Notice that the infinite sums needed to compute these weights have nice closed-form solutions, which are easily evaluated given a base $2^{-\eta\gamma}$. ClampedDiscLap is identical to sampling from $\mathsf{DiscLap}^{\gamma}$ and then applying Clamp.[2]

Furthermore, given ρ, sampling ν_i is also a matter of considering the relevant regions. We don't need the specific value of ν_i, only whether it is greater than or less than $\hat{\rho}_i - q_i(D)$. Applying the NoisyThreshold procedure, which samples either $\bot$ or $\top$ based on the total weight of the elements of $\gamma\mathbb{N}$ above or below the threshold, is identical to sampling from $\mathsf{DiscLap}^{\gamma}$ and then post-processing to determine whether $\nu_i \geq \hat{\rho}_i - q_i(D)$. Note that NoisyThreshold is unbiased, which makes it a useful component for many settings.

We rely on the implementation of NormalizedSample as described in [4] to perform weighted sampling without division. NormalizedSample takes a set of outcomes and associated weights, and samples an outcome according to the normalized weights.

Proof of Privacy

We follow the proof strategy of [6] closely to make comparisons to a proof of privacy for the original mechanism

[2] Strictly speaking, we guarantee sufficient information for all subsequent clamping procedures based on $q_i(D)$ by using Q_{min} and Q_{max} as proxies for the largest and smallest values of $q_i(D)$.

Algorithm 3 Discrete Sampling Functions

```
Notation: we use NormalizedSample$(O, W)$ to indicate using NormalizedSample to sample from outcome set $O = \{o_1, o_2, \ldots, o_{|O|}\}$ with associated weights $W = \{w_1, w_2, \ldots, w_{|O|}\}$ for brevity.

Inputs: privacy parameter $\eta$, granularity $\gamma$, allowed range $[w_{min}, w_{max}]$ where $w_{min}, w_{max} \in \gamma\mathbb{Z}$.
Outputs: $\rho$ sampled according to $\rho \sim \mathsf{DiscLap}^\gamma(\frac{1}{\eta})$ clamped to $[w_{min}, w_{max}]$, as specified in Equation 2.
1:  function ClampedDiscLap$(\eta, \gamma, w_{min}, w_{max})$
2:      $p_\ell \leftarrow$ GetSum$(2^{-\eta\gamma}, -\infty, w_{min}/\gamma - 1)$
3:      $p_u \leftarrow$ GetSum$(2^{-\eta\gamma}, w_{max}/\gamma + 1, \infty)$
4:      $p_t \leftarrow$ GetSum$(2^{-\eta\gamma}, -\infty, \infty)$
5:      $p_m \leftarrow p_t - p_\ell - p_u$
6:      $r \leftarrow$ NormalizedSample$(\{-, \cdot, +\}, \{p_\ell, p_m, p_u\})$
7:      if $r = +$ then
8:          return $w_{max}$
9:      else if $r = -$ then
10:         return $w_{min}$
11:     else
12:         $\mathcal{O} \leftarrow \gamma\mathbb{Z} \cap [w_{min}, w_{max}]$
13:         $\rho \sim$ NormalizedSample$(\mathcal{O}, \{2^{-\eta|o|} \mid o \in \mathcal{O}\})$
14:         return $\rho$
Inputs: privacy parameter $\eta$; threshold $\tau = \hat{\rho} - q_i(D)$, where $\tau$ is an integer multiple of the granularity parameter $\gamma$.
Outputs: $o \in \{\perp, \top\}$ sampled according to $\Pr[\nu \sim \mathsf{DiscLap}^\gamma(\frac{1}{\eta}) \geq \tau]$.
15: function LazyThreshold$(\eta, \tau, \gamma)$
16:     $p_\top \leftarrow$ GetSum$(2^{-\eta\gamma}, \tau/\gamma, \infty)$
17:     $p_\perp \leftarrow$ GetSum$(2^{-\eta\gamma}, -\infty, \infty) - p_\top$
18:     return NormalizedSample$(\{\top, \perp\}, \{p_\top, p_\perp\})$
Inputs: privacy parameter $\eta$; threshold $\tau = \hat{\rho} - q_i(D) + g$, conditional threshold $\tau^C = \hat{\rho} - q_i(D)$, where $\tau > \tau^C$ are integer multiples of the granularity parameter $\gamma$.
Outputs: $o \in \{\perp, \top\}$ sampled according to $\Pr[\nu \sim \mathsf{DiscLap}^\gamma(\frac{1}{\eta}) \geq \tau \mid \nu_i \geq \tau^C]$.
19: function CondLazyThreshold$(\eta, \tau, \tau^C, \gamma)$
20:     $p_\top \leftarrow$ GetSum$(2^{-\eta\gamma}, \tau/\gamma, \infty)$
21:     $p_\perp \leftarrow$ GetSum$(2^{-\eta\gamma}, \tau^C/\gamma, \infty) - p_\top$
22:     return NormalizedSample$(\{\top, \perp\}, \{p_\top, p_\perp\})$
Inputs: the "base" $B < 1$, starting index $\ell$, ending index $u$.
Outputs: $(1 - B)\sum_{k=\ell}^{u} B^{|k|}$.
23: function GetSum$(B, \ell, u)$
24:     if $\ell = -\infty$ and $u = \infty$ then
25:         return $1 + B$.
26:     else if $\ell = -\infty$ and $u$ is finite then
27:         return $\begin{cases} B^{|u|} \text{ if } u \leq 0 \\ 1 + B - B^{|u|+1} \text{ if } u > 0 \end{cases}$
28:     else if $u = \infty$ and $\ell$ is finite then
29:         return $\begin{cases} B^{|\ell|} \text{ if } \ell \geq 0 \\ 1 + B - B^{|\ell|+1} \text{ if } \ell < 0 \end{cases}$
30:     else
31:         return $(1 + B) -$ GetSum$(B, -\infty, \ell - 1) -$ GetSum$(B, u + 1, \infty)$
```

straightforward. We first briefly state a lemma concerning the increasing behavior of Clamp.

Lemma 3.1. *Given q_i such that $|q_i(D) - q_i(D')| \leq \Delta$, Clamp$(z, q_i(D), w) - q_i(D) \leq$ Clamp$(z + \Delta, q_i(D'), w) - q_i(D')$.*

The proof consists of the following steps. Suppose that $|q_i(D) - q_i(D')| = \alpha$. We show that Clamp$(z, q_i(D), w) - q_i(D) \leq$ Clamp$(z + \alpha, q_i(D'), w) - q_i(D') \leq$ Clamp$(z + \Delta, q_i(D'), w) - q_i(D')$, where the final inequality follows from the fact that Clamp is increasing in z. The complete proof of Lemma 3.1 appears in the Appendix.

Next, we prove the core lemma relating the probability of a set of $\perp$ or $\top$ outcomes for D given ρ to the probability for adjacent database D' given $\rho + \Delta$ (similar to Lemma 1 of [6]).

Lemma 3.2. *Let $\mathcal{A}$ be Algorithm 2. Given an output a of $\mathcal{A}$, let $\mathbf{I}_\perp$ indicate the indices of $\perp$ and $\mathbf{I}_\top$ the indices of $\top$, and let*

$$f_D(z) = \prod_{i \in \mathbf{I}_\perp} \Pr[q_i(D) + \nu_i < \text{Clamp}(z, q_i(D), w)]$$
$$g_D(z) = \prod_{i \in \mathbf{I}_\top} \Pr[q_i(D) + \nu_i \geq \text{Clamp}(z, q_i(D), w)]$$

Then $f_D(z) \leq f_{D'}(z + \Delta)$ and $g_D(z) \leq 2^{\eta_2} g_{D'}(z + \Delta)$.

Proof. **Part 1: f. Show that $f_D(z) \leq f_{D'}(z + \Delta)$.** We'll show that $\Pr[q_i(D) + \nu_i < \text{Clamp}(z, q_i(D), w)] \leq \Pr[q_i(D') + \nu_i < \text{Clamp}(z + \Delta, q_i(D'), w)]$ which implies that the product $f_D(z) \leq f_{D'}(z + \Delta)$. Consider the two quantities we want to relate. Recall from Equation 1 that $\Pr[\mathsf{DiscLap}^\gamma(\frac{1}{\eta}) = t\gamma] = \frac{2^{-\eta\gamma|t|}}{\sum_{t\gamma \in \mathcal{O}} 2^{-\eta\gamma|t|}}$. For simplicity, write $\sum_{t=-\infty}^{\infty} 2^{|-t|\gamma\frac{\eta_2}{2\Delta c}} = \mu^{-1}$.

$$\begin{aligned} &\Pr[q_i(D) + \nu_i < \text{Clamp}(z, q_i(D), w)] \\ &\quad = \Pr[\nu_i < \text{Clamp}(z, q_i(D), w) - q_i(D)] \\ &\quad = \mu \sum_{t=-\infty}^{\frac{1}{\gamma}(\text{Clamp}(z, q_i(D), w) - q_i(D)) - 1} 2^{-|t|\gamma\frac{\eta_2}{2\Delta c}} \end{aligned}$$

That is, the probability that $q_i(D) + \nu_i$ doesn't exceed the noisy threshold given z is the the sum of the probabilities that $\nu_i \sim \mathsf{DiscLap}$ is equal to $t\gamma$ for integer t such that $t\gamma < \text{Clamp}(z, q_i(D), w) - q_i(D)$), and likewise

$$\begin{aligned} &\Pr[q_i(D') + \nu_i < \text{Clamp}(z + \Delta, q_i(D'), w)] \\ &\quad = \Pr[\nu_i < \text{Clamp}(z + \Delta, q_i(D'), w) - q_i(D')] \\ &\quad = \mu \sum_{t=-\infty}^{\frac{1}{\gamma}(\text{Clamp}(z+\Delta, q_i(D'), w) - q_i(D')) - 1} 2^{-|t|\gamma\frac{\eta_2}{2\Delta c}} \end{aligned}$$

Given that Clamp$(z + \Delta, q_i(D'), w) - q_i(D') \geq$

$\textsc{Clamp}(z, q_i(D), w) - q_i(D)$ from Lemma 3.1, it follows that

$$\begin{aligned} f_D(z) &= \prod_{i \in \mathbf{I}_\perp} \Pr[q_i(D) + \nu_i < \textsc{Clamp}(z, q_i(D), w)] \\ &= \prod_{i \in \mathbf{I}_\perp} \Pr[\nu_i < \textsc{Clamp}(z, q_i(D), w) - q_i(D)] \\ &\leq \prod_{i \in \mathbf{I}_\perp} \Pr[\nu_i < \textsc{Clamp}(z+\Delta, q_i(D'), w) - q_i(D')] \\ &= f_{D'}(z+\Delta) \end{aligned}$$

Part 2: g**. Show** $g_D(z) \leq 2^{\eta_2} g_{D'}(z+\Delta)$. The proof follows from three steps. (A) We will first show that $\Pr[q_i(D')+\nu_i \geq \textsc{Clamp}(z-\alpha, q_i(D'), w)] \geq \Pr[q_i(D') + \nu_i \geq \textsc{Clamp}(z - \Delta, q_i(D'), w)]$ as $\textsc{Clamp}$ is increasing in z. (B) Then we will show that $\Pr[q_i(D') + \nu_i \geq \textsc{Clamp}(z - \Delta, q_i(D'), w)] \leq 2^{\eta_2/c} \Pr[q_i(D') + \nu_i \geq \textsc{Clamp}(z + \Delta, q_i(D'), w)]$. (C) And finally we will conclude that $g_D(z) \leq e^{\varepsilon_2} g_{D'}(z + \Delta)$ as $|\mathbf{I}_\top| \leq c$ based on the size of $\mathbf{I}_\top$.

(A) It follows that $\Pr[q_i(D') + \nu_i \geq \textsc{Clamp}(z - \alpha, q_i(D'), w)] \geq \Pr[q_i(D) + \nu_i \geq \textsc{Clamp}(z, q_i(D), w)]$ as $\textsc{Clamp}(z, q_i(D), w) - q_i(D) \geq \textsc{Clamp}(z - \alpha, q_i(D'), w) - q_i(D')$ per Lemma 3.1.

(B) Next, we'll show that $\Pr[q_i(D') + \nu_i \geq \textsc{Clamp}(z - \Delta, q_i(D'), w)] \leq 2^{\eta_2/c} \Pr[q_i(D') + \nu_i \geq \textsc{Clamp}(z + \Delta, q_i(D'), w)]$. Recall that

$$\begin{aligned} &\Pr[q_i(D') + \nu_i \geq \textsc{Clamp}(z+\Delta, q_i(D'), w)] \\ &\quad = \mu \sum_{t=\frac{1}{\gamma}(\textsc{Clamp}(z+\Delta, q_i(D'), w) - q_i(D'))}^{\infty} 2^{-|t|\gamma \frac{\eta_2}{2\Delta c}} \end{aligned}$$

Shift z by 2Δ:

$$\begin{aligned} &\geq \mu \sum_{t=\frac{1}{\gamma}(\textsc{Clamp}(z-\Delta, q_i(D'), w) - q_i(D'))}^{\infty} 2^{-|t+\frac{2\Delta}{\gamma}|\gamma \frac{\eta_2}{2\Delta c}} \\ &\geq 2^{\frac{-\eta_2}{c}} \mu \sum_{t=\frac{1}{\gamma}(\textsc{Clamp}(z-\Delta, q_i(D'), w) - q_i(D'))}^{\infty} 2^{-|t|\gamma \frac{\eta_2}{2\Delta c}} \\ &= 2^{-\frac{\eta_2}{c}} \Pr[q_i(D') + \nu_i \geq \textsc{Clamp}(z - \Delta, q_i(D'), w)] \end{aligned}$$

Where the first inequality follows from $|\textsc{Clamp}(z - \Delta, q_i(D'), w) - \textsc{Clamp}(z + \Delta, q_i(D'), w)| \leq 2\Delta$. Therefore

$$\begin{aligned} 2^{\frac{\eta_2}{c}} &\Pr[q_i(D') + \nu_i \geq \textsc{Clamp}(z + \Delta, q_i(D'), w)] \\ &\geq \Pr[q_i(D') + \nu_i \geq \textsc{Clamp}(z - \Delta, q_i(D'), w)] \\ &\geq \Pr[q_i(D) + \nu_i \geq \textsc{Clamp}(z, q_i(D), w)] \end{aligned}$$

where the second inequality follows from part (A). Thus $\Pr[q_i(D') + \nu_i \geq \textsc{Clamp}(z - \Delta, q_i(D'), w)] \leq 2^{\eta_2/c} \Pr[q_i(D') + \nu_i \geq \textsc{Clamp}(z + \Delta, q_i(D'), w)]$.

(C) Given $|\mathbf{I}_\top| \leq c$,

$$\begin{aligned} g_D(z) &= \prod_{i \in \mathbf{I}_\top} \Pr[q_i(D) + \nu_i \geq \textsc{Clamp}(z, q_i(D), w)] \\ &\leq \prod_{i \in \mathbf{I}_\top} 2^{\frac{\eta_2}{c}} \Pr[q_i(D') + \nu_i \geq \textsc{Clamp}(z + \Delta, q_i(D'), w)] \\ &= 2^{\eta_2} \prod_{i \in \mathbf{I}_\top} \Pr[q_i(D') + \nu_i \geq \textsc{Clamp}(z + \Delta, q_i(D'), w)] \\ &= 2^{\eta_2} g_{D'}(z+\Delta) \end{aligned}$$

This completes the proof of Part 2.

□

We can now state and prove the main theorem, closely following the logic of Theorem 2 of [6].

Theorem 3.3. *Let $\mathcal{A}$ be Algorithm 2. $\mathcal{A}$ is $(\eta_1 + \eta_2)|_2$-DP.*

Proof. Let $\mathcal{A}$ be Algorithm 2. Given an output a of $\mathcal{A}$, define $\mathbf{I}_\perp$, $\mathbf{I}_\top$, $f_D(z)$ and $g_D(z)$ as in Lemma 3.2. We write $\sum_{z=m;\gamma}^{n}$ to indicate that z is incremented by γ. Thus for any possible output a of $\mathcal{A}$:

$$\begin{aligned} &\Pr[\mathcal{A}(D) = a] \\ &\quad = \sum_{z=-\infty;\gamma}^{\infty} \Pr[\rho = z] \Pr[\mathcal{A}(D) = a \mid \rho = z] \\ &\quad = \sum_{z=-\infty;\gamma}^{\infty} \Pr[\rho = z] f_D(z) g_D(z) \\ &\quad \leq \sum_{z=-\infty;\gamma}^{\infty} 2^{\eta_1} \Pr[\rho = z+\Delta] f_{D'}(z+\Delta) g_D(z) \\ &\quad \leq \sum_{z=-\infty;\gamma}^{\infty} 2^{\eta_1+\eta_2} \Pr[\rho = z+\Delta] f_{D'}(z+\Delta) g_{D'}(z+\Delta) \end{aligned}$$

Where the first inequality follows from the definition of the discrete Laplace mechanism and Lemma 3.2 and the second inequality follows from Lemma 3.2. Take $z' = z + \Delta$:

$$\begin{aligned} &= 2^{\eta_1+\eta_2} \sum_{z=-\infty;\gamma}^{\infty} \Pr[\rho = z'] f_{D'}(z') g_{D'}(z') \\ &= 2^{\eta_1+\eta_2} \Pr[\mathcal{A}(D') = a] \end{aligned}$$

□

Caveat: Setting w to be small won't impact the privacy of the mechanism, but it will significantly impact accuracy. For example, suppose we took $w = 1$, and that $q_i(D) = -10$ and $q_i(D'') = 10$. Any $\rho \geq -9$ will be treated as -9 for D and 9 for $q_i(D'')$, and $q_i(D'')$ will report $\perp$ nearly as often as $q_i(D)$, which is not desirable from an accuracy perspective.

4 Implementation Details

Our reference Rust implementation of Algorithms 2 and 3 is available on github. The implementation reuses the $\textsc{NormalizedSample}$ method and other privacy parameter construction and convenience methods from [4]. The code primarily consists of exact translations of the pseudocode of Algorithms 2 and 3 with two additional important pieces: η "adjustment" and exact arithmetic enforcement.

If γ is an integer, $2^{\eta\gamma}$ can be computed exactly, and once we can compute $2^{\eta\gamma}$ exactly, raising to integer powers of k to compute sums is trivial.[3] The main difficulty in exact implementation is ensuring that we can exactly compute $2^{\eta\gamma}$ for non-integer γ. To do this, we require $\eta = -z\log_2(\frac{x}{2^y})$ such that either z is an integer multiple of γ or y is an integer multiple of γ and x^γ can be computed exactly. We then adjust η directly rather than trying to raise 2^η to a fractional power.

We require the caller to provide η_1 and η_2 directly in a form which can be adjusted, as the caller is in a better position to decide how they would like budget allocated between each step. We also require the caller to adjust η_1 and η_2 given the sensitivity of the queries and the count c, as the caller is in the best position to construct η in the required form. This corresponds to modifying Algorithm 2 by replacing Line 3 with $\hat{\rho} \leftarrow \textsc{ClampedDiscLap}(\eta_1^*, Q_{min} - w, Q_{max} + w)$ and Line 7 with $a_i \leftarrow \textsc{NoisyThreshold}(\eta_2^*, \hat{\rho} - q_i(D), \gamma)$, where $\eta_1^* = \eta_1/\Delta$ and $\eta_2^* = \eta_2/(2\Delta c)$ are provided by the caller.

To enforce exact arithmetic, we determine the required precision by computing adjustments of η and determining the precision required to compute noisy thresholds at the given γ within the range $[Q_{min} - w, Q_{max} + w]$. This allows us to determine whether we have sufficient precision to exactly execute the mechanism without any influence from the specific queries or the private data, eliminating a potential timing channel. **Timing channels are particularly problematic for sparse vector, because as long as the adversary can construct a query that is likely to output $\perp$ that yields timing information, it can be repeated to amplify the timing channel.** Furthermore, we enforce that all data-independent parameters are integer multiples of γ before evaluating any queries, and any q_i which are not integer multiples are rounded. This does potentially degrade the privacy guarantee if the queries don't have the appropriate outcome granularity, but by at most a sensitivity increase of 2γ, rather than allowing for an error channel with arbitrary privacy loss. That is, if the provided implementation of any q_i results in a value outside $\gamma\mathbb{N}$, rounding to $\gamma\mathbb{N}$ results in at most a 2γ increase in the sensitivity of the query. If, on the other hand, an error were returned, an adversary could construct a query which used the error condition as a side channel.

5 Randomness Alignment

In Section 3, we presented the proof of privacy for a particular variant of discrete sparse vector (i.e., all zero thresholds) for the purposes of exact implementation. A natural question to ask is how difficult it is to extend this proof for other variants of sparse vector, e.g., top k, releasing gap information. Fortunately, this question has already been largely addressed by the randomness alignment "proof template" of Ding, Wang, Zhang and Kifer [1]. This proof template essentially requires that one constructs an alignment function mapping the randomness which produces outcomes of the mechanism given D to the randomness which produces those same outcomes given D'. Their proof template is highly convenient and less error prone than proving privacy from scratch. In this section, we briefly summarize the concept of randomness alignment and the proof template given in [1][4], and then discuss how to extend or apply their proof template to exact implementations.

Definition 5.1 (Randomness Alignment [1]). Let $\mathcal{M}$ be a randomized algorithm. Let $D \sim D'$ be two adjacent databases. A *randomness alignment* is a function $\phi_{D,D'} : \mathbb{R}^\infty \to \mathbb{R}^\infty$ such that for all H on which $\mathcal{M}(D, H)$ terminates, we have that $\mathcal{M}(D, H) = \mathcal{M}(D', \phi_{D,D'}(H))$.

To closely match [1], we always consider a randomized algorithm $\mathcal{M} : \mathcal{D} \times \mathcal{H} \to \Omega$ which maps (database, randomness) pairs to the outcome space Ω. If left unstated, $D, D' \in \mathcal{D}$, $H \in \mathcal{H}$, etc. The proof template considers both continuous and discrete random variables; we focus on the discrete case, as that is most relevant to exact implementation. Specifically, condition 4 of Theorem 5 of [1] is omitted in the proof template, as it is only relevant for continuous random variables. The requirement that the ψ_i are differentiable is likewise dropped. We use ξ rather than η to denote random variables, so as not to overload the privacy parameter notation for base-2 DP.

In many cases, the most convenient way to specify an alignment is to specify a *local alignment* for each possible outcome $\omega \in \Omega$.

Definition 5.2 (local alignment [1]). Let $\mathcal{M}$ be a randomized algorithm that takes randomness input $H \in \mathcal{H}$ with outcome space Ω. Let $D \sim D'$ be a pair of adjacent databases and ω a possible output of $\mathcal{M}$. A *local alignment* for $\mathcal{M}$ is a function $\phi_{D,D',\omega} : \{H \in \mathcal{H} \mid \mathcal{M}(D, H) = \omega\} \to \{H \in \mathcal{H} \mid \mathcal{M}(D', H) = \omega\}$.

These local alignments can then be stitched together to form a single alignment by taking $\phi_{D,D'}(H) = \phi_{D,D',\mathcal{M}(D,H)}(H)$.

The primary requirement for the proof template for bounding privacy loss is that the cost of an alignment is bounded. That is, the ratio of the probabilities of the randomness required to produce ω for D versus D' is bounded.[5]

Definition 5.3 (Alignment cost [1]). Let $\mathcal{M}$ be a randomized algorithm that uses H, a set of discrete random variables, as its source of randomness. Let $\phi_{D,D',\omega}$ be a local alignment for $\mathcal{M}$. Let f be the probability mass function of H. The cost of the alignment is defined as $cost(\phi_{D,D'}) := \max_{H \in \mathcal{H}} \frac{f(H)}{f(\phi_{D,D'}(H))}$.

[3] Multiplying by $(1 - B)$ prevents division in the sum closed forms.

[4] Please see Sections 4 and 8 of [1] for more complete details, illustrative examples and proofs.

[5] There is not a universal formal definition of alignment cost given in [1], so we adapt the definition from Condition 3 of Theorem 5 of [1], and state the specialized version given in Definition 6 of [1] as an example.

For example, let $\mathcal{M}$ be a randomized algorithm that uses H, a set of discrete random variables, as its source of randomness. Let $\phi_{D,D',\omega}$ be a local alignment for $\mathcal{M}$. For any $H = (\xi_1, \xi_2, \ldots)$ let $H' = (\xi'_1, \xi'_2, \ldots)$ denote $\phi_{D,D',\omega}(H)$. Suppose each ξ_i is generated independently from a distribution f_i with the property that $\log(f_i(x)/f_i(y)) \leq |x - y|/\alpha_i$ for all x, y in the domain of f_i. For instance, ξ_i drawn from the discrete Laplace distribution have this property. The the cost of $\phi_{D,D',\omega}$ is defined as $cost(\phi_{D,D',\omega}) = \sum_i |\xi_i - \xi'_i|/\alpha_i$.[6]

Another key requirement for the proof template is that $\phi_{D,D'}$ is one-to-one. It was shown in [1] that if an alignment is constructed from *acyclic* local alignments, then it is one-to-one:

Definition 5.4 (acyclic [1]). Let $\mathcal{M}$ be a randomized algorithm operating on discrete random variables, and let $\phi_{D,D',\omega}$ be a local alignment for $\mathcal{M}$. For any $H = (\xi_1, \xi_2, \ldots)$ let $H' = (\xi'_1, \xi'_2, \ldots)$ denote $\phi_{D,D',\omega}(H)$. We say that $\phi_{D,D',\omega}$ is *acyclic* if there exists a permutation π and functions $\psi^j_{D,D',\omega}$ such that:

$$\xi'_{\pi(1)} = \xi_{\pi(1)} + \text{number that only depends on } D, D', \omega$$

$$\xi'_{\pi(j)} = \xi_{\pi(j)} + \psi^j_{D,D',\omega}(\xi_{\pi(1)}, \ldots, \xi_{\pi(j-1)}) \text{ for } j \geq 2$$

Loosely speaking, a local alignment is acyclic if there is some ordering of the random variables so that $\xi'_{\pi(j)}$ is completely determined by $\xi_{\pi(\leq j)}$. Given that a local alignment is acyclic, $\xi_{\pi(1)}$ is uniquely determined by H', and thus each $\xi_{\pi(i>1)}$ are uniquely determined by H', which yields the desired one-to-one property.

We now state the proof template of [1] for discrete random variables.

Theorem 5.5 (Proof Template). *Let $\mathcal{M}$ be a randomized algorithm with randomness source H, a set of discrete random variables, that terminates with probability* 1 *and suppose the number of random variables used by $\mathcal{M}$ can be determined from its output.*[7] *If, for all pairs of adjacent databases $D \sim D'$, there exists a randomness alignment function $\phi_{D,D'}$ (or local alignment functions $\phi_{D,D',\omega}$ for all $\omega \in \Omega$ and $D \sim D'$) that satisfies conditions 1 through 3 below, then $\mathcal{M}$ is $\ln(a)-$differentially private.*

1. *The alignment is properly specified, i.e., $\phi_{D,D'} : \mathbb{R}^\infty \to \mathbb{R}^\infty$ such that if $\mathcal{M}(D, H)$ terminates, $\mathcal{M}(D, H) = \mathcal{M}(D', \phi_{D,D'}(H))$ or for each local alignment and all $\omega \in \Omega$, if $\mathcal{M}(D, H) = \omega$ then $\mathcal{M}(D', \phi_{D,D',\omega}(H)) = \omega$.*

2. *The alignment (or all local alignments $\phi_{D,D',\omega}$ for all $\omega \in \Omega$) is one-to-one. That is, given D, D', ω and $\phi_{D,D'}(H)$ (or $\phi_{D,D',\omega}(H)$) the alignment is invertible and H can be uniquely determined.*

3. *For each pair of adjacent databases $D \sim D'$, the alignment* cost *of $\phi_{D,D'}$ (or all $\phi_{D,D',\omega}$) is bounded by a, i.e., given the probability mass function f of H, $\frac{f(H)}{f(\phi_{D,D'}(H))} \leq a$ (or for all ω, $\frac{f(H)}{f(\phi_{D,D',\omega}(H))} \leq a$) for all H (except on a set of measure* 0*).*

We refer the reader to [1] for the proof of Theorem 5.5 for continuous random variables[8], and give a brief outline of the proof for discrete random variables (closely following [1]) below.

Proof. To prove privacy, we must show that for all $E \subseteq \Omega$ that $\Pr[\mathcal{M}(D, H) \in E] \leq a \Pr[\mathcal{M}(D', H) \in E]$. Notice that any randomness alignment can be decomposed into (identical) corresponding local alignment functions, i.e. $\phi_{D,D',\omega}(H) = \phi_{D,D'}(H)$, so we only concern ourselves with local alignments. Let $\phi_1, \phi_2, \ldots$ be the distinct local alignment functions, and take $E_i = \{\omega \in E \mid \phi_{D,D',\omega} = \phi_i\}$. From Condition 1, we have that

$$\phi_i(\{H \mid \mathcal{M}(D, H) = \omega\}) \subseteq \{H' \mid \mathcal{M}(D', H') = \omega\}$$

From Condition 2, we have that each ϕ_i is one-to-one.

Note that

$$\{H \mid \mathcal{M}(D, H) \in E_i\} = \cup_{\omega \in E_i}\{H \mid \mathcal{M}(D, H) = \omega\}$$

and

$$\{H' \mid \mathcal{M}(D', H') = E_i\} = \cup_{\omega \in E_i}\{H' \mid \mathcal{M}(D', H') = \omega\}$$

Furthermore, the sets $\{H \mid \mathcal{M}(D, H) = \omega\}$ are pairwise disjoint for different ω and likewise $\{H' \mid \mathcal{M}(D', H') = \omega\}$ are pairwise disjoint for different ω, (i.e., one H or H' can't lead to more than one ω for a given database). Thus it follows that ϕ_i is one-to-one on $\{H \mid \mathcal{M}(D, H) = E_i\}$ and $\phi(\{H \mid \mathcal{M}(D, H) = E_i\}) \subseteq \{H' \mid \mathcal{M}(D', H') = E_i\}$, and for any $H' \in \{H' \mid \mathcal{M}(D', H') = E_i\}$ there exists $H \in \{H \mid \mathcal{M}(D, H) = E_i\}$ such that $H = \phi_i^{-1}(H')$.

From Condition 3, we have that $\frac{f(H)}{f(\phi_i(H))} = \frac{f(\phi_i^{-1}(H'))}{f(H')} \leq a$ for all $H \in \{H' \mid \mathcal{M}(D', H') = E_i\}$, (except on a set of measure 0). Then the following is true:

$$\begin{aligned}
\Pr[\mathcal{M}(D, H) \in E_i] &= \sum_{\{H \mid \mathcal{M}(D,H)=E_i\}} f(H) \\
&= \sum_{\phi_i(\{H \mid \mathcal{M}(D,H)=E_i\})} f(\phi_i^{-1}(H')) \\
&\leq \sum_{\phi_i(\{H \mid \mathcal{M}(D,H)=E_i\})} af(H') \\
&\leq \sum_{\{H' \mid \mathcal{M}(D',H')=E_i\}} af(H') \\
&= a \Pr[\mathcal{M}(D', H') \in E_i]
\end{aligned}$$

Where the final inequality follows from the fact that $\phi(\{H \mid \mathcal{M}(D, H) = E_i\}) \subseteq \{H' \mid \mathcal{M}(D', H') = E_i\}$ and that f is non-negative. Finally, since $E = \cup_i E_i$ and E_i are pairwise disjoint, we can conclude that

$$\Pr[\mathcal{M}(D, H) \in E] \leq a \Pr[\mathcal{M}(D', H') \in E]$$

□

[6] This example was adapted from Definition 6 of [1].

[7] Intuitively, knowing the number of random variables used is related to inverting the alignment to show one-to-one.

[8] Please see the proof of Theorem 5 and Lemma 1 in Section 8 of [1].

Algorithm 4 Discrete Sparse Vector with Gap

Inputs: *database D, query stream $Q = q_1, q_2, \dots$ each with sensitivity $\leq \Delta$ with values clamped to $[Q_{min}, Q_{max}]$, a set of gaps G such that $g \in G \geq 0$, a maximum positive query count c, base-2 privacy parameters η_1 and η_2, a width parameter w, and a granularity γ. All values are assumed to be integer multiples of γ.* ***Outputs***: *a stream of answers $a_1, a_2, \dots$ where each $a_i \in \{\perp, \top\}$.*

```
procedure GapSV($D, Q, G, \Delta, c, \eta_1, \eta_2, w, \gamma$)
    $\rho \sim \mathsf{DiscLap}^{\gamma}(\frac{\Delta}{\eta_1})$
    $count \leftarrow 0$
    for $q_i \in Q$ do
        $\nu_i \leftarrow \mathsf{DiscLap}(\frac{2\Delta c}{\eta_2})$
        $a_i^0 \leftarrow \begin{cases} \perp \text{ if } \nu_i + q_i(D) < \textsc{Clamp}(\rho, q_i(D), w) \\ \top \text{ if } \nu_i + q_i(D) \geq \textsc{Clamp}(\rho, q_i(D), w) \end{cases}$
        if $a_i^0 = \top$ then
            $\rho_i \leftarrow \textsc{Clamp}(\rho, q_i(D), w)$
            for $j = 1, 2, \dots,$ do
                $a_i^j \leftarrow \begin{cases} \perp \text{ if } \nu_i + q_i(D) < \rho_i + G_j \\ \top \text{ if } \nu_i + q_i(D) \geq \rho_i + G_j \end{cases}$
            output $(a_i^0, a_i^1, a_i^2, \dots)$
            $count = count + 1$
            if $count \geq c$ then abort
        else
            output $\perp$
```

Algorithm 5 Simulation of Discrete Sparse Vector with Gap

Inputs: *database D, query stream $Q = q_1, q_2, \dots$ each with sensitivity $\leq \Delta$ with values clamped to $[Q_{min}, Q_{max}]$, a set of gaps G such that $g \in G \geq 0$, a maximum positive query count c, base-2 privacy parameters η_1 and η_2, a width parameter w, and a granularity γ. All values are assumed to be integer multiples of γ.* ***Outputs***: *a stream of answers $a_1, a_2, \dots$ where each $a_i \in \{\perp, \top\}$.*

```
procedure SimGapSV($D, Q, G, \Delta, c, \eta_1, \eta_2, w, \gamma$)
    $\hat{\rho} \leftarrow \textsc{BoundedLap}(\frac{\eta_1}{\Delta}, Q_{min} - w, Q_{max} + w)$
    $count \leftarrow 0$
    for $q_i \in Q$ do
        $\hat{\rho}_i \leftarrow \textsc{Clamp}(\hat{\rho}, q_i(D), w)$
        $a_i \leftarrow \textsc{LazyThreshold}(\frac{\eta_2}{2c\Delta}, \hat{\rho}_i - q_i(D), \gamma)$
        if $a_i^0 = \top$ then
            $g_0 \leftarrow 0$
            for $j = 1, 2, \dots$ do
                if $a_i^{j-1} = \top$ then
                    $\tau^C \leftarrow \hat{\rho}_i - q_i(D); \tau \leftarrow g_j + \tau^C$
                    $a_i^j \leftarrow \textsc{CondLazyThreshold}(\frac{\eta_2}{2c\Delta}, \tau, \tau^C, \gamma)$
                else $a_i^j \leftarrow \perp$
            output $(a_i^0, a_i^1, a_i^2, \dots)$
            $count = count + 1$
            if $count \geq c$ then abort
        else
            output $\perp$
```

It's clear to see how this template makes it easier to prove differential privacy for settings like sparse vector, as one can focus in on how the randomness aligns and then verify properties of the alignment. If we also want to ensure that the proposed mechanism can be implemented exactly, there is a simple preface to this proof template:

(A) Specify the mechanism using discrete (but not necessarily finite) random variables that are post-processed to bounded magnitude, e.g. DiscLap post-processed with Clamp.

(B) Demonstrate that the mechanism can be simulated using finite precision, e.g., substituting ClampedDiscLap for DiscLap post-processed with Clamp.

(C) Proceed with specifying an alignment and the rest of the proof template based on the original discrete mechanism from step (A).

The primary observation in the template preface is that proving privacy with respect to infinite, discrete random variables that are post-processed is straightforward. The main technical question in applying the preface is choosing the appropriate post-processing of the infinite discrete random variables to enable a simple alignment and exact simulation.

Sparse vector with gap. To demonstrate this technique, we will extend Algorithm 2 to return the gap information between the noisy estimate for any query that returns $\top$ and the noisy threshold ρ. To do this, we will take a slightly different view of how the gap is released. Instead of releasing a value directly, we will instead release a set of gaps $G = \{g_1, g_2, g_3, \dots\}$ which the noisy query exceeds. To determine the gap for any non-$\perp$, query, we simply take the maximum g_j reported as $\top$. This is identical to releasing the gap itself for some pre-specified granularity (for example, if $G = \{\gamma, 2\gamma, \dots, g_{max}\gamma\}$), and as we'll see below, it makes the proof extension very simple. By using a set of gaps or a maximum possible gap, we sidestep the problem of the potential for an infinitely large gap which could occur with infinitely large ν_i or ρ. Algorithm 4 presents the discrete version with access to infinite random variables, and 5 presents the finite simulation version.

To complete the preface, we need to show that Algorithm 5 exactly simulates Algorithm 4.

- Simulating ρ. The argument for the simulation of ρ is identical to the basic discrete sparse vector mechanism. ρ is post-processed into ρ_i using $\textsc{Clamp}(\rho, q_i(D), w)$, and as such, sampling $\hat{\rho}$ from ClampedDiscLap on the range $[Q_{min} - w, Q_{max} + w]$ is sufficient to simulate any post-processed value of ρ.

- Simulating ν_i. We use LazyThreshold and CondLazyThreshold to simulate whether a draw of ν_i would have exceeded each gap, and the argument is analogous to the basic discrete sparse vector mechanism.

We now propose a randomness alignment for Algorithm 4 and apply the template. As before, given an output $\omega = a_1, a_2, \dots$ where each $a_i \in \{\perp, \top\}^{|G|}$, denote the set of indices with at least one $\top$ as $\mathbf{I}_\omega^\top$ and the set of indices with no instances of $\top$ as $\mathbf{I}_\omega^\perp$. Consider the alignment

$$\begin{aligned}
&\phi_{D,D',\omega}(\rho, \nu_1, \dots, \nu_n): \\
&\quad \rho' = \rho + \Delta \\
&\quad \nu_i' = \nu_i \text{ if } i \in \mathbf{I}_\omega^\perp \\
&\quad \nu_i' = \nu_i + q_i(D) - q_i(D') + \textsc{Clamp}(\rho + \Delta, q_i(D'), w) \\
&\qquad\qquad - \textsc{Clamp}(\rho, q_i(D), w) \text{ if } i \in \mathbf{I}_\omega^\top
\end{aligned}$$

That is, map ρ to $\rho + \Delta$, and map ν_i to itself if the index is a $\perp$ output, and otherwise map ν_i to ν_i' such that $\nu_i' + q_i(D') - \textsc{Clamp}(\rho+\Delta, q_i(D'), w) = \nu_i + q_i(D) - \textsc{Clamp}(\rho, q_i(D), w)$.

We now verify each part of the template.

1. *Alignment correctness.* Show that if $\mathcal{M}(D, H) = \omega$ then $\mathcal{M}(D', \phi_{D,D',\omega}(H)) = \omega$. We'll proceed by showing that $a_i = a_i'$ and $a_i^j = a_i'^j$ if $a_i = \top$.
    (a) $i \in \mathbf{I}_\omega^\perp$. If $a_i = \perp$, then $0 < \textsc{Clamp}(\rho, q_i(D), w) - q_i(D) - \nu_i$. If $a_i' = \perp$ then $0 < \textsc{Clamp}(\rho', q_i(D'), w) - q_i(D') - \nu_i'$. Recall from Lemma 3.1 that $\textsc{Clamp}(\rho+\Delta, q_i(D'), w) - q_i(D') \geq \textsc{Clamp}(\rho, q_i(D), w) - q_i(D)$ if $|q_i(D) - q_i(D')| \leq \Delta$, thus

    $$\begin{aligned}
    \textsc{Clamp}(\rho', q_i(D'), w) - q_i(D') - \nu_i' &= \\
    \textsc{Clamp}(\rho + \Delta, q_i(D'), w) - q_i(D') - \nu_i' &\geq \\
    \textsc{Clamp}(\rho, q_i(D), w) - q_i(D) - \nu_i' &= \\
    \textsc{Clamp}(\rho, q_i(D), w) - q_i(D) - \nu_i &> 0
    \end{aligned}$$

    where the second equality follows from the alignment $\nu_i = \nu_i'$ for $i \in \mathbf{I}_\omega^\perp$. Therefore, if $a_i = \perp$, then $a_i' = \perp$.

    (b) $i \in \mathbf{I}_\omega^\top$. First consider $a_i = \top$. If $a_i = \top$, then $0 \geq \textsc{Clamp}(\rho, q_i(D), w) - q_i(D) - \nu_i$. $a_i' = \top$ if $0 \geq \textsc{Clamp}(\rho', q_i(D'), w) - q_i(D') - \nu_i'$. Using the alignment substitution for ν_i', we have

    $$\begin{aligned}
    \textsc{Clamp}(\rho', q_i(D'), w) - q_i(D') - \nu_i' &= \\
    \textsc{Clamp}(\rho, q_i(D), w) - q_i(D) - \nu_i &\leq 0
    \end{aligned}$$

    Thus if $a_i = \top$ then $a_i' = \top$. Consider a_i^j versus $a_i'^j$. Given the equivalence above, if $\textsc{Clamp}(\rho, q_i(D), w) - q_i(D) - \nu_i \geq g_j$, then $\textsc{Clamp}(\rho', q_i(D'), w) - q_i(D') - \nu_i' \geq g_j$, and the comparisons for $a_i'^j$ are equivalent to those for a_i^j.

2. *Alignment cost.* As ρ and each ν_i are drawn from the discrete Laplace distribution (with appropriate parameters), we have that $2^{-\eta_1} \leq \frac{f(\rho)}{f(\rho')} \leq 2^{\eta_1}$ and $2^{-\eta_2/c} \leq \frac{f(\nu_i)}{f(\nu_i')} \leq 2^{\eta_2/c}$, as $|\nu_i - \nu_i'| = |\textsc{Clamp}(\rho + \Delta, q_i(D'), w) - \textsc{Clamp}(\rho, q_i(D), w) + q_i(D) - q_i(D')| \leq 2\Delta$. Notice that in any alignment, there are at most c indices included in $\mathbf{I}_\omega^\top$, and thus the total cost of the alignment is given by $2^{-(\eta_1+\eta_2)} \leq \frac{f(H)}{f(H')} \leq 2^{\eta_1+\eta_2}$.

3. *One-to-one.* From Lemma 1 of [1], we have that if a local alignment is acyclic, then it is one-to-one. The proposed alignment is acyclic as any order which place ρ before the ν_i satisfies the requirements. (It's also straightforward to verify that the alignment is one-to-one by construction.)

With the template verified, we can conclude that Algorithm 4 is $\eta = (\eta_1 + \eta_2)|_2$-DP. Notice that by using a fixed set of positive gaps, rather than releasing $\nu_i + q_i(D) - \textsc{Clamp}(\rho, q_i(D), w)$, we avoid the potential for an infinitely large gap. Another way of achieving this is to release $\min\{g_{max}, \nu_i + q_i(D) - \textsc{Clamp}(\rho, q_i(D), w)\}$ for some fixed g_{max}.

6 Private selection from private candidates

Algorithm 6 Private selection from private candidates [5]

***Inputs**: a threshold τ, budgets $\gamma \leq 1$ and $\varepsilon_0 \leq 1$, number of steps $T \geq \max\{\frac{1}{\gamma}\ln(\frac{2}{\varepsilon_0}), 1 + \frac{1}{e\gamma}\}$ and sampling access to $Q(D)$. **Outputs**: $\perp$ or a pair (x, q) in the output space of $Q(D)$.*

```
1: procedure PSPC($Q(D), \tau, \gamma, \varepsilon_0$)
2:     for $j = 1, 2, \dots T$ do
3:         draw $(x, q) \sim Q(D)$
4:         if $q \geq \tau$ then output $(x, q)$ and halt
5:         with probability $\gamma$ output $\perp$ and halt
6:     output $\perp$ and halt
```

Recent work of Liu and Talwar proposes a set of procedures for "private selection from private candidates," i.e. choosing a differentially private mechanism $\mathcal{M}_i$ from a set of differentially private mechanisms $\mathcal{M}$ based on the *quality* of the output of the mechanism [5]. This type of procedure is particularly relevant for hyperparameter tuning and other settings where the optimal differentially private mechanism is unknown, but a set of reasonable candidates is available.

We remark that the way in which the primary procedure of [5] is written is particularly amenable to exact implementation, as the proof of privacy follows from the privacy of the mechanism and quality scoring procedures themselves, rather than direct analysis of the probability of exceeding the given threshold. We briefly recap the terminology for their primary procedure (Algorithm 1: "Thresholding with a known threshold τ") outlined in Algorithm 6.

Given a database D, a set of mechanisms $\mathcal{M} = \{\mathcal{M}_1, \dots \mathcal{M}_K\} : \mathcal{D} \to \Omega$ and a quality score function $q : \Omega \times D \to \mathbb{R}$, define the randomized mechanism $Q : \mathcal{D} \to$

$([K], \Omega) \times \mathbb{R}$ as

$$
\begin{aligned}
i &\sim_{\textbf{Unif}} [K] \\
m &\sim \mathcal{M}_i(D) \\
Q(D) &:= ((i, m), q(m))
\end{aligned}
$$

The privacy of Q follows from the privacy of each $\mathcal{M}_i$ and the quality function q.[9] For example, q might indicate computing an accuracy score with low sensitivity, and adding sensitivity scaled Laplace noise. Given that each of these components of Q is differentially private, it's straightforward to argue the privacy of Q itself.

For ε_1-DP Q, Algorithm 6 is $(2\varepsilon_0 + \varepsilon_1)-$DP. The privacy of the selection procedure doesn't exactly follow the usual sparse vector template, as the threshold itself is not randomized. However, in this procedure, the number of "failed" samples is *not* released, unlike the $\perp$ outputs for below threshold in the standard SVT setup. Loosely speaking, privacy follows from observing that the probability of outputting any non-$\perp$ outcome is close between adjacent databases and that conditioned on outputting a non-$\perp$ outcome, the probability of any particular outcome is also close between adjacent databases. Note, that although γ does not directly appear in privacy guarantee of the procedure, it is critical to the first step of the argument, and is implicitly included through the size of T.

Why is it particularly convenient to implement this procedure? Once we have purely differentially private implementations of each $\mathcal{M}_i$, notice that using clamped discrete Laplace for the quality score (so long as the boundaries are always set the same regardless of the underlying database) is sufficient to ensure that Q is purely differentially private given a low sensitivity quality function. Furthermore, the biased coin flip for outputting $\perp$ with probability γ can either be implemented exactly using NormalizedSample or any other appropriate method. Thus, given the ingredients we already have, it's possible to implement private selection from private candidates exactly, and achieve pure differential privacy. To more fully illustrate this technique, Algorithm 7 outlines the "hard-stopping" variant of the procedure given in Algorithm 2 of [5].[10]

To prove privacy of the exact implementation of Algorithm 7 assuming access to pure-DP implementations of each of the $\mathcal{M}_i$, we first observe that in Line 7, $y = \perp$ with probability $\gamma_\perp$ given an appropriate choice of $\gamma_\perp$ (i.e., it can be written with finite precision). Second we prove a simple lemma stating that GetSample is $(\eta_{\mathcal{M}} + \eta)|_2-$DP, which suffices to show that sampling from $Q(D)$ implemented by GetSample is $\ln(2)(\varepsilon_{\mathcal{M}} + \eta)-$DP. With these two ingredients and an appropriate choice of T parameterized by $\gamma_\perp$ and ε_0 (as given in Theorem 3.6 of [5]), HardStop is $(3\ln(2)(\eta + \eta_{\mathcal{M}}) + 3\varepsilon_0)-$DP.[11]

Algorithm 7 Maximum score with hard stopping

```
Inputs: The database $D$, sampling access to pure-DP implementations of each $\mathcal{M}_i \in \mathcal{M}$ with privacy budget $\eta_{\mathcal{M}}$, number of steps $T$ and budgets $\gamma_\perp \le 1$ and $\eta_q$, a granularity parameter $\gamma$, and clamping parameter $w$. Outputs: a pair $(x, q)$ in the output space of $Q(D)$.
1:  procedure HardStop($D, \mathcal{M}, T, \gamma_\perp, \eta_q, \gamma_g, w$)
2:      $S \leftarrow \{\}$
3:      for $j = 1, 2, \ldots, T$ do
4:          $(x, q) \leftarrow$ GetSample($\mathcal{M}, w_{min}, w_{max}, \gamma_g$)
5:          $S \leftarrow S \cup (x, q)$
6:          $(x^*, q^*) \leftarrow \arg\overset{\text{CMP}}{\max}_{(x,q)\in S}\{q\}$
7:          $y \leftarrow$ NormalizedSample($\{\perp, \top\}, \{\gamma_\perp, 1 - \gamma_\perp\}$)

8:          if $y = \perp$ then output $(x^*, q^*)$ and halt
9:      output $(x^*, q^*)$ and halt
    Returns the pair with the larger quality score, breaking ties based on the mechanism number.
10: function CMP($(x_1, q_1), (x_2, q_2)$)
11:     if $q_1 > q_2$ then return $(x_1, q_1)$
12:     else if $q_2 > q_1$ then return $(x_2, q_2)$
13:     else                              ▷ Break ties uniformly at random
14:         $W \leftarrow \{1, 1\}$
15:         $O \leftarrow \{(x_1, q_1), (x_2, q_2)\}$
16:         $(x_i, q_i) \leftarrow$ NormalizedSample($W, O$)
17:         return $(x_i, q_i)$
    Returns a sample from $Q(D)$.
18: function GetSample($D, \mathcal{M}, q, \eta, w, \gamma_g$)
19:     $K \leftarrow |\mathcal{M}|$
20:     $W \leftarrow \{1, 1^2, \ldots, 1^K\}$
21:     $i \leftarrow$ NormalizedSample($[K], W$)   ▷ Draw an index uniformly from $[K]$
22:     $m \leftarrow \mathcal{M}_i(D)$              ▷ Run the mechanism selected
23:     $\nu \leftarrow$ BoundedLap($\eta, \gamma_g, -2w, 2w$)
24:     $\hat{q} \leftarrow$ Clamp($q(m, D), 0, w$)
25:     $\tilde{q} \leftarrow$ Clamp($\hat{q} + \nu, 0, w$) ▷ Compute the quality score and add discrete Laplace noise; clamp to $[-w, w]$.
26:     return $((m, i), \tilde{q})$
```

[9] Strictly speaking, each mechanism $\mathcal{M}_i$ may have a distinct quality function, we use a single quality function for simplicity and clarity.

[10] We explicitly define the tie-breaking functionality of Cmp for clarity.

[11] For the proof of privacy of the original version, please see Appendix A.2 of the full version of [5] on arXiv.

Lemma 6.1. *GetSample is* $(\eta + \eta_{\mathcal{M}})|_2$*−DP.*

Proof. The probability of selecting an index i is identical regardless of the underlying database, thus $2^{-\eta_{\mathcal{M}}} \leq \frac{\Pr[i \sim [K]] \Pr[\mathcal{M}_i(D)=m]}{\Pr[i \sim [K]] \Pr[\mathcal{M}_i(D')=m]} \leq 2^{\eta_{\mathcal{M}}}$. Notice that the sensitivity of clamped q is no greater than the sensitivity of q, thus

$$2^{-\eta} \leq \frac{\Pr[\textsc{Clamp}(\hat{q} + \nu \sim \mathsf{DiscLap}^{\gamma_g}(\frac{1}{\eta}), 0, w) = \tilde{q}]}{\Pr[\textsc{Clamp}(\hat{q}' + \nu \sim \mathsf{DiscLap}^{\gamma_g}(\frac{1}{\eta}), 0, w) = \tilde{q}]} \leq 2^{\eta}$$

Notice that for any value of ν outside of $[-2w, 2w]$, the value of Clamp is the same as if ν were $\pm 2w$. Thus, sampling ν clamped to the range $[-2w, 2w]$ is sufficient to exactly simulate the behavior of sampling from $\mathsf{DiscLap}^{\gamma_g}(\frac{1}{\eta})$ and then clamping.

Thus, taking $\mathcal{G}$ as the function GetSample,

$$\begin{aligned}
&\Pr[\mathcal{G}(D, \mathcal{M}, q, \eta, w, \gamma_g) = ((i, m), \tilde{q})] \\
&\quad = \Pr[i \sim [K]] \Pr[\mathcal{M}_i(D) = m] * \\
&\qquad \Pr[\textsc{Clamp}(\hat{q} + \nu \sim \mathsf{DiscLap}^{\gamma_g}(\frac{1}{\eta}), 0, w) = \tilde{q}] \\
&\quad \leq 2^{\eta + \eta_{\mathcal{M}}} \Pr[i \sim [K]] \Pr[\mathcal{M}_i(D') = m] * \\
&\qquad \Pr[\textsc{Clamp}(\hat{q}' + \nu \sim \mathsf{DiscLap}^{\gamma_g}(\frac{1}{\eta}), 0, w) = \tilde{q}] \\
&\quad = 2^{\eta + \eta_{\mathcal{M}}} \Pr[\mathcal{G}(D', \mathcal{M}, q, \eta, w, \gamma_g) = ((i, m), \tilde{q})]
\end{aligned}$$

which completes the proof. □

Another nice property of proving privacy based on the privacy of sampling from $Q(D)$ is that the combination of correct sampling of the halt condition and Lemma 6.1 are sufficient to construct pure-DP implementations of the relevant variants of private selection from private candidates given in [5]. (Note that the percentile-finding variants do not solely rely on privacy of $Q(D)$, so the extensions to those cases are more nuanced.)

7 Discussion

The trouble with ties. The simplicity of extension of the discrete sparse vector implementation to release gap information is an encouraging sign for the existence of simple, exact implementation strategies of many related techniques. However, we explicitly caveat that randomness alignment proofs which rely on probability zero of ties aren't as straightforward, and furthermore, naive implementations are particularly vulnerable to adversarial query choices.

Consider the problem of releasing the top k queries from a query set Q along with the gaps between the i^{th} and $i + 1^{st}$ largest queries. The randomness alignment proposed in [1] for this problem relies on the fact that the noise added to each query cannot result in a tie, which implies that the top k and the ordering of the top k are uniquely identified once the noise values have been specified. This allows them to leave the noise added to the non-top-k values unmodified between D and D', i.e. $\xi'_i = \xi_i$. This keeps the cost of the alignment dependent only on the modifications to the noise values for the top k. They reason that an implementation which could not guarantee no ties would therefore be approximately differentially private with a very small δ corresponding to the probability ties. Unfortunately, in discrete implementations the probability of ties is non-zero *and can be manipulated by an adversary.* For example, if clamped Laplace noise is used, choosing query values close to the edge of the range is more likely to result in ties. Fortunately, the exponential mechanism is a good choice for this setting, and can be implemented exactly.

Timing channels: not just for adversaries. As noted in Section 4, any timing channels in individual steps of the sparse vector technique can be amplified if there are a large number of $\perp$ outputs. Although it may seem that timing channels are only problematic in the case of adversarial analysts, we argue that this issue is more complicated.

Suppose a benign analyst starts a run of the sparse vector technique with a set of queries Q. If the query completes by the end of an hour, the analyst considers the outputs and draws a conclusion C. Alternatively, suppose that the query does not complete by the end of the hour. In this case, the analyst terminates the queries *without looking at the results* and issues a new set of queries Q'. If the new set of queries completes within an hour, the analyst draws a conclusion C'. (Otherwise, they give up).

Although in the second case the privacy budget expended is twice that of the first, the conclusions drawn may be *completely distinct.* That is, the probability that the analyst would have concluded C' given the result of Q on either D or D' is zero. For example, if the database suffers from Simpson's paradox, i.e., trends hold for subgroups but not for aggregations, and Q considers subgroups but Q' considers aggregations, the analyst may draw very different conclusions simply because D' results in a longer run-time for Q. This problem can also arise if the analyst considers only a subset of the results on D versus D' given the slow running time for one database versus the other. In some sense, such problems are a result of "user error", but it is important to note that the presence of timing channels can break privacy and validity guarantees without explicitly malicious behavior by the analyst.

Is exact implementation required? Throughout this work, our goal has been exact implementation. However, this is not strictly necessary to achieve pure differential privacy. For example, instead of an exact implementation of discrete Laplace, one could use the snapping mechanism [7] with appropriate clamping and boundary condition management, as it does have the appropriate guarantees on the difference of probabilities of sampling ξ versus $\xi + \Delta$ within a given range. We omit the details of this implementation, and we stress that it's not immediately obvious that the snapping mechanism would be a trivial replacement of ClampedDiscLap. However, as inexact implementations may have desirable efficiency or timing channel properties, versions of sparse vector based on inexact implementation are an interesting area for future work.

8 Summary

In this work, we have shown discrete constructions for sparse vector as well as noisy threshold and discrete clamped Laplace that can be implemented exactly with base-2 DP. We have also given a reference implementation in Rust, which demonstrates that the code to exactly implement the mechanisms is straightforward, with limited complexity and performance overhead. Future directions include: integrating the implementation into a more complete DP library, better convenience methods for constructing η in the required form, an implementation of Discrete Sparse Vector with Gap (Algorithm 4) and additional protections against timing attacks (e.g., implementing a constant time version of GetSum that doesn't depend on recursive calls).

We have also demonstrated how to combine our technique for exact implementation with the proof template of [1], which should make proving privacy and exact implementation simpler for variants of sparse vector and other mechanisms which are good candidates for randomness alignment style proofs. The primary takeaway from this work is that proving privacy for mechanisms intended to be implemented exactly is not significantly more difficult than for mechanisms assuming access to continuous random variables. Although there may be some subtleties, e.g., choosing how to clamp, this work demonstrates that it's possible to achieve (for the most part) logically equivalent mechanisms which can be implemented exactly, and thus suffer no degradation or loss of privacy guarantee between the original specification and the mechanism implementation.

References

[1] Zeyu Ding, Yuxin Wang, Danfeng Zhang, and Daniel Kifer. 2019. Free Gap Information from the Differentially Private Sparse Vector and Noisy Max Mechanisms. *CoRR* abs/1904.12773 (2019). arXiv:1904.12773 `http://arxiv.org/abs/1904.12773`

[2] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography*, Shai Halevi and Tal Rabin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 265–284.

[3] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[4] Christina Ilvento. 2019. Implementing the Exponential Mechanism with Base-2 Differential Privacy. arXiv:cs.CR/1912.04222

[5] Jingcheng Liu and Kunal Talwar. 2019. Private selection from private candidates. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 298–309.

[6] Min Lyu, Dong Su, and Ninghui Li. 2016. Understanding the Sparse Vector Technique for Differential Privacy. *CoRR* abs/1603.01699 (2016). arXiv:1603.01699 `http://arxiv.org/abs/1603.01699`

[7] Ilya Mironov. 2012. On significance of the least significant bits for differential privacy. In *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 650–661.

[8] Yuxin Wang, Zeyu Ding, Guanhong Wang, Daniel Kifer, and Danfeng Zhang. 2019. Proving Differential Privacy with Shadow Execution. *CoRR* abs/1903.12254 (2019). arXiv:1903.12254 `http://arxiv.org/abs/1903.12254`

[9] Danfeng Zhang and Daniel Kifer. 2016. AutoPriv: Automating Differential Privacy Proofs. *CoRR* abs/1607.08228 (2016). arXiv:1607.08228 `http://arxiv.org/abs/1607.08228`

A Additional Proofs

Proof of Lemma 3.1.

Proof. The proof consists of the following steps. Suppose that $|q_i(D) - q_i(D')| = \alpha$. We show that $\textsc{Clamp}(z, q_i(D), w) - q_i(D) \leq \textsc{Clamp}(z + \alpha, q_i(D'), w) - q_i(D') \leq \textsc{Clamp}(z + \Delta, q_i(D'), w) - q_i(D')$, where the final inequality follows from the fact that Clamp is increasing in z. Recall that

$$\textsc{Clamp}(z, q_i(D), w) - q_i(D) = \begin{cases} -w \text{ if } z < q_i(D) - w \\ w \text{ if } z > q_i(D) + w \\ z - q_i(D) \text{ otherwise} \end{cases}$$

$$\textsc{Clamp}(z + \alpha, q_i(D'), w) - q_i(D') = \begin{cases} -w \text{ if } z + \alpha < q_i(D') - w \\ w \text{ if } z + \alpha > q_i(D') + w \\ z + \alpha - q_i(D') \text{ otherwise} \end{cases}$$

Suppose $q_i(D) + \alpha = q_i(D')$:

$$\begin{aligned} \textsc{Clamp}(z + \alpha, q_i(D'), w) - q_i(D') &= \begin{cases} -w \text{ if } z < q_i(D) - w \\ w \text{ if } z > q_i(D) + w \\ z - q_i(D) \text{ otherwise} \end{cases} \\ &= \textsc{Clamp}(z, q_i(D), w) - q_i(D) \end{aligned}$$

Alternatively, suppose $q_i(D) - \alpha = q_i(D')$:

$$\textsc{Clamp}(z + \alpha, q_i(D'), w) - q_i(D') = \begin{cases} -w \text{ if } z < q_i(D) - 2\alpha - w \\ w \text{ if } z > q_i(D) - 2\alpha + w \\ z + 2\alpha - q_i(D) \text{ otherwise} \end{cases}$$

We verify the inequality by cases. If $z < q_i(D) - w - 2\alpha$ or $z > q_i(D) + w$, then $\textsc{Clamp}(z + \alpha, q_i(D'), w) - q_i(D') = \textsc{Clamp}(z, q_i(D), w) - q_i(D)$. If $z \in [q_i(D) - 2 - 2\alpha, q_i(D) - w)$, then $\textsc{Clamp}(z + \alpha, q_i(D'), w) - q_i(D') = z + 2\alpha - q_i(D) \geq -w$ and $\textsc{Clamp}(z, q_i(D), w) - q_i(D) = -w$. If $z \in [q_i(D) - w, q_i(D) - 2\alpha + w)$, then $\textsc{Clamp}(z + \alpha, q_i(D'), w) - q_i(D') = z + 2\alpha - q_i(D) \geq z - q_i(D) = \textsc{Clamp}(z, q_i(D), w) - q_i(D)$. If $z \in [q_i(D) - 2\alpha + w, q_i(D) + w)$, then $\textsc{Clamp}(z + \alpha, q_i(D'), w) - q_i(D') = w \geq z - q_i(D) = \textsc{Clamp}(z, q_i(D), w) - q_i(D)$. □

Closed-form sums

Proposition A.1. *GetSum correctly computes the value of* $(1 - B)\sum_{k=\ell}^{u} B^{|k|}$ *for* $|B| < 1$.

Proof. Recall that for $|r| < 1$, $\sum_{k=0}^{\infty} Ar^k = \frac{A}{1-r}$, $\sum_{k=1}^{\infty} Ar^k = \frac{Ar}{1-r}$ and $\sum_{k=0}^{n-1} A\frac{(1-r^n)}{1-r} = \frac{A}{1-r}$. For completeness, we will demonstrate:

1. $(1 - B)\sum_{-\infty}^{\infty} B^{|k|} = 1 + B$. This follows from taking $A = (1 - B)$ and observing that

$$\begin{aligned}\sum_{-\infty}^{\infty} Ar^{|k|} &= \sum_{-\infty}^{0} Ar^{|k|} + \sum_{1}^{\infty} Ar^k \\ &= \sum_{0}^{\infty} Ar^k + \sum_{1}^{\infty} Ar^k \\ &= \frac{A}{1-r} + \frac{Ar}{1-r} \\ &= \frac{A(1+r)}{1-r}\end{aligned}$$

Substituting $B = r$, $A = (1 - B)$,

$$\sum_{-\infty}^{\infty}(1-B)B^{|k|} = \frac{(1-B)(1+B)}{1-B} = 1 + B$$

2. $(1 - B)\sum_{\ell}^{\infty} B^{|k|} = \begin{cases} B^{|\ell|} \text{ if } \ell \geq 0 \\ 1 + B - B^{|\ell|+1} \text{ if } \ell < 0 \end{cases}$ for finite ℓ.

This follows from observing that if $\ell \leq 0$,

$$\begin{aligned}\sum_{k=\ell\leq 0}^{\infty} Ar^{|k|} &= \sum_{k=0}^{\infty} Ar^{|k|} + \sum_{k=\ell}^{-1} Ar^{|k|} \\ &= \sum_{k=0}^{\infty} Ar^k + \sum_{k=1}^{|\ell|} Ar^k \\ &= \sum_{k=0}^{\infty} Ar^k + \sum_{j=0}^{|\ell|-1} rAr^k \\ &= \frac{A}{1-r} + \frac{Ar(1-r^{|\ell|})}{1-r}\end{aligned}$$

Substituting $B = r$, $A = (1 - B)$,

$$\begin{aligned}\sum_{k=\ell\leq 0}^{\infty}(1-B)B^{|k|} &= 1 + B(1 - B^{|\ell|}) \\ &= 1 + B - B^{|\ell|+1}\end{aligned}$$

Similarly for $\ell \geq 0$,

$$\begin{aligned}\sum_{k=\ell\geq 0}^{\infty} Ar^{|k|} &= \sum_{k=0}^{\infty} Ar^{|k|} - \sum_{k=0}^{\ell-1} Ar^{|k|} \\ &= \sum_{k=0}^{\infty} Ar^k - \sum_{k=0}^{\ell-1} Ar^k \\ &= \frac{A}{1-r} - \frac{A(1-r^{|\ell|})}{1-r}\end{aligned}$$

Substituting $B = r$, $A = (1 - B)$,

$$\begin{aligned}\sum_{k=\ell\geq 0}^{\infty}(1-B)B^{|k|} &= 1 - (1 - B^{|\ell|}) \\ &= B^{|\ell|}\end{aligned}$$

3. $(1 - B)\sum_{-\infty}^{u} B^{|k|} = \begin{cases} B^{|u|} \text{ if } u \leq 0 \\ 1 + B - B^{|u|+1} \text{ if } u > 0 \end{cases}$ for finite u. This follows from observing that $(1 - B)\sum_{-\infty}^{u} B^{|k|} = (1 - B)\sum_{-u}^{\infty} B^{|k|}$ and applying the result of the previous part.

□