PRIVATE QUANTILE ESTIMATION IN THE TWO-SERVER MODEL

Jacob Imola
University of Copenhagen
Denmark
jaim@di.ku.dk

Fabrizio Boninsegna
University of Padova
Italy
fabrizio.boninsegna@phd.unipd.it

Hannah Keller
Aarhus University
Denmark
hkeller@cs.au.dk

Anders Aamand
University of Copenhagen
Denmark
aa@di.ku.dk

Rasmus Pagh
BARC, University of Copenhagen
Denmark
pagh@di.ku.dk

Amrita Roy Chowdhury
University of Michigan, Ann Arbor
United States of America
aroyc@umich.edu

ABSTRACT

Estimating quantiles is a fundamental statistics in distributed private learning. However, there is an accuracy gap between what can be achieved in the local model vs the central model of privacy. Specifically, prior work shows an error bound of $O(\frac{\log(B)^2}{\varepsilon\sqrt{n}})$ in the local model where B is the domain size, while the estimates in the central DP model can achieve an error bound of $O(\frac{\log(B)}{\varepsilon n})$. In this paper, we bridge this gap with the help of cryptographic primitives while working in the two-server model.

1 Introduction

Distributed learning refers to a scenario where data is distributed across multiple clients, and a data analyst seeks to compute aggregate statistics of the joint dataset. A particularly important and fundamental statistic of interest is the estimation of quantiles. However, when dealing with sensitive data, publishing quantiles can inadvertently expose information about individuals within the dataset. For example, if an organization wants to publish the median of its users' ages, it could reveal the birth date of a particular user, thereby compromising their privacy. Local differential privacy (LDP) offers a solution to this issue. However, there is a significant gap in accuracy compared to what can be achieved in the central model. Specifically, prior work shows an error bound of $O(\frac{\log(B)^2}{\varepsilon\sqrt{n}})$ in the local model where B is the domain size, while the estimates in the central DP model can achieve an error bound of $O(\frac{\log(B)}{\varepsilon n})$.

In this work, we focus on bridging this gap by using cryptographic primitives, particularly secure multiparty computation (MPC). MPC allows two or more parties to collaboratively compute a function based on their private inputs, without revealing anything other than the function's output during the process. Thus, high-accuracy quantile estimates with rigorous privacy guarantees can be achieved in our setting by using MPC to implement state-of-the-art central DP mechanisms. While this is theoretically feasible using off-the-shelf MPC tools, practical implementations pose a number of challenges. First, directly applying this approach to compute aggregate user statistics would require running a multi-round protocol across the devices of all clients which is impractical for most real-world settings. Second, generic solutions introduce significant computational overhead, which becomes prohibitively costly with large client populations and/or data domains. We tackle these challenges as follows. First, we consider an intermediate trust model called *outsourced MPC model* where the analyst's role is distributed across a small group of non-colluding parties.

These parties receive secret-shared inputs from the clients and collaboratively compute the required aggregate statistics using an MPC protocol. As long as at least one party remains honest, the privacy of the clients' inputs is preserved, and only the desired aggregate result is disclosed. A particular instance of this model, known as the *two-server* model, has been successfully implemented in several large-scale MPC applications [1, 2]. Second, we introduce a novel MPC protocol for estimating quantiles that is highly efficient. Our key innovation is to allocate a portion of the privacy budget to release carefully chosen intermediate statistics from the joint dataset, which helps to reduce the complexity of the MPC operations required. In terms of asymptotic cost analysis, our protocol needs to secure sort only $O(\frac{\log^2(B)}{\varepsilon})$ elements, compared to the naive implementation which would require sorting the entire dataset of length n which becomes a bottleneck as n increases.

2 Background

We consider the discrete data domain $[B]$, where B is a large integer. We let $X = \{x_1, \ldots, x_n\}$ be a private dataset, where n is the dataset size, assumed to be public. We let $x_{(i)}$ denote the ith element of X when X is in sorted order. For $q \in [0, 1]$, the qth quantile of X is $x_{(r)}$ where $r = \lfloor qn \rfloor$. For $z \in [B]$, we will consider the quantile error, defined by

$$\mathsf{err}_{X,q}(z) = \left| q - \frac{1}{n} \sum_{x \in X} \mathbf{1}[x \leq z] \right|,$$

that is, the difference between the desired quantile q and the actual quantile q' of z. To avoid edge cases with the above definition, we assume for the rest of the paper that all points in X are distinct, so $x_{(1)} < x_{(2)} < \cdots < x_{(n)}$. However, our results can easily generalize to this edge case with no major changes.

We are interested in computing approximations to m given quantiles $q_1, \ldots, q_m \in [0, 1]$. We will need a mild assumption that the quantiles are not too close to each other, formally that $|q_i - q_j| \geq \Omega(\frac{1}{n\varepsilon} \log(B)^2 \log\log(\frac{B}{\delta}))$. This assumption is often true, such as when the quantiles are equally spaced from 0 to 1.

2.1 Cryptographic Primitives

Linear Secret Shares (LSS). Linear secret shares (LSS) is a secure multi-party computation (MPC) technique that allows mutually distrusting parties to securely compute over secret inputs. We use the notation $[x]$ to denote a linear secret sharing of an input $x \in \mathcal{F}$ that is shared between k parties. Each party P_i holds a share $[x]_i$ such that $\sum_{i=1}^{k} [x]_i = x$. The secret x can be constructed iff all the parties reveal their shares and then sum them up, which means this scheme preserves perfect security against $k - 1$ corrupted parties. LSS supports the following local linear operations.

- $[z] \leftarrow [x] + [y]$: each parties can locally compute $[z]_i \leftarrow [x]_i + [y]_i$
- $[z] \leftarrow c \cdot [x]$: each parties can locally compute $[z]_i \leftarrow c \cdot [x]_i$
- $[z] \leftarrow c + [x]$: P_1 computes $[z]_i \leftarrow c + [x]_i$, P_i computes $[z]_i \leftarrow [x]_i$ for all $i \in [k] \backslash \{1\}$.

Multiplication requires interaction between the parties and is done with the help of Beaver's triples [3].

2.2 Differential Privacy

In this work we consider (ε, δ)-differential privacy (DP), where the neighboring relation is defined for datasets that differ by the data of a single individual. Additionally, we will make use of privacy amplification by shuffling [4], which states that a uniform random perturbation of n private messages, each with $1 \leq \varepsilon_0 \leq O(\log(n)/\log(1/\delta))$ privacy budget, satisfies (ε, δ)-DP with $\varepsilon = O\left(\sqrt{e^{\varepsilon_0} \log(1/\delta)/n}\right)$.

3 Proposed Protocol

Setting. We consider n clients $C_i, i \in [n]$, each with a private input x_i. Additionally, we have two untrusted and non-colluding servers S_0 and S_1 Each client splits their input into two shares $[x]_b$ which is shared with the server S_b. The two servers then engage in an MPC protocol to compute m quantiles Q of the joint dataset $X = \{x_1, \cdots, x_n\}$. We consider the semi-honest threat model where all the parties follow the protocol honestly, but their contents and computations can be observed by an adversary.

Algorithm 1 $\Pi_{\texttt{Quant}}$, protocol for computing quantiles under MPC.

Input. $[X]$, $n = |X|$, a set Q of m quantiles, parameters ε, δ, domain size B

$I_1, \ldots, I_{2m+1} = \mathsf{FormIntervals}([X], Q, B, \mathcal{M}_{\varepsilon,\delta})$

$[X_1], \ldots, [X_m] = \mathsf{FormBuckets}([X], n, I_1, \ldots, I_{2m+1}, \varepsilon, \delta)$

for $1 \leq j \leq k$ **do**

  Let $\tilde{q}_j$ be renormalized q_j, defined in (1). ▷ Goal is to find quantile $\tilde{q}_j$ in $[X_j]$

  Securely shuffle $[X_j]$

  Securely sort $[X_j]$.

  $z_j = \mathsf{EMSingle}([X_j], \tilde{q}_j)$

end for

Output $(z_1, z_2, \cdots z_m)$

3.1 Naive Approach under MPC

For a data domain $[B]$, the exponential mechanism is a reasonable choice to compute a quantile privately. For a candidate $b \in [B]$ and quantile $q \in [0, 1]$, the utility function $u_q(b, X) = -\mathsf{err}_{X,q}(b)$ measures the *quantile* error. This utility has sensitivity just $\frac{1}{n}$, and has been used successfully in the central model [5, 6]. In our case, the exponential mechanism simply returns a candidate b with probability proportional to $\exp(\varepsilon n u_q(b, X)/2)$. As B is typically very large (such as 2^{32}), practical implementations of this mechanism use the observation that $u_q(b, X)$ is constant for any b that lies between consecutive (sorted) points in X, and thus implicitly represent the probability distribution on $[B]$ as n intervals. However, there are several challenges to this approach under MPC.

1. Sorting is a particularly computationally expensive as it uses a super-linear number of comparison operations, which is costly under secret shares, and has $O(\log n)$ span. For concrete practical efficiency, these $\log n$ factors can make a large impact.
2. The utility scores $u_q(b, X)$ are themselves sensitive, and thus computing the exponential $\exp(\varepsilon n u_q(b, X)/2)$ requires exponentiating a secure value. Secure exponentiation is an especially costly operation, and using it $\Omega(n)$ times should be avoided.

Our algorithm addresses each of these bottlenecks, which we describe in the next section.

3.2 Our Approach

First, we outline the high-level ideas, followed by a more detailed explanation of the individual steps. The full protocol is presented in Alg. 1. Our approach is based on the following key ideas:

- For each quantile to be estimated, we first allocate a portion of the privacy budget to compute an interval containing the quantile. This serves as an initial data filtering step, effectively reducing the size of the dataset to be sorted.
- Next, we need to bucket the data elements into these intervals. However, a naive bucketing strategy under MPC would involve padding the length of each bucket to be n, which would undermine the data filtering step mentioned earlier. We address this issue by leaking a differentially private count of the number of elements in each bucket.
- Finally, we provide an efficient implementation of the secure exponential mechanism to compute the individual quantiles. Our key insight is that, by releasing a differentially private count of the buckets, we effectively release a noisy ranking of the data elements. This allows us to compute the weights for the exponential mechanism without the need for the costly secure exponentiation operation.

We will elaborate on each of these steps in the next section.

FormIntervals - Reduce Effective Data Size. The naive approach requires sorting the entire dataset which is computationally expensive. Hence, our first idea is to reduce the size of the effective dataset to be considered by constructing for each q_ith quantile an interval I_i containing it. This approach would limit the sorting operation to only the elements within these intervals. One way to construct these intervals is to use an LDP protocol $\mathcal{M}$ for CDF estimation with maximum absolute error α. If such a protocol exists, then any $c \in [B]$ such that $\mathcal{M}(c) \in [q - \alpha, q + \alpha]$ would return a 2α approximate qth-quantile. Thus, for each q_j we can construct an interval $I_j = [c_1, c_2)$, with c_1, c_2 s.t. $\mathcal{M}(c_1) \in [q_j - 2\alpha, q_j - \alpha]$ and $\mathcal{M}(c_2) \in [q_j + \alpha, q_j + 2\alpha]$ so that any q_jth quantile would lie in I_j with constant

probability, and $|\{x \in X : x \in I_j\}| = O(\alpha n)$. In Appendix A we show that there exists a (ε, δ)-shuffle DP[1] protocol for CDF estimation with expected maximum absolute error $\tilde{O}(\log^2 B/\varepsilon n)$ if ε is sufficiently small and there are enough users. Thus, our assumption $|q_i - q_j| \geq \tilde{\Omega}(\log^2 B/\varepsilon n)$ assures that $I_i \cap I_j \neq \emptyset$. The full algorithm is presented in Alg. 3.

FormBuckets - Leak Differentially Private Counts. Based on the aforementioned intervals, we construct a partition of $[B]$ into $2m+1$ buckets $\mathcal{B}$ (by filling the holes between I_{j-1} and I_j). Thus $B_j = (B_1, \ldots, B_{2m+1})$ contains q_j-th quantile for j even. However, we cannot directly populate these buckets since it reveal the bucket sizes. The naive strategy under MPC is to pad the size of each bucket to n by adding dummy records. However, doing this we lose our advantange of forming the intervals. We mitigate this by leaking a differentially private count of the bucket sizes, i.e., instead of exhaustive padding we only need enough dummy records to satisfy DP (which is a much smaller quantity). Specifically, for each bucket B_j, each server S_b adds η_j^b dummy records where η_j^b is drawn from a modification of the binary mechanism [7] that releases only positive noise. Let X_j be the subset of the data X contained in bucket B_{2j}. We can now search for each quantile q_j over the subset X_j of size $O(\alpha n)$, rather than over the entire dataset. However, these quantiles must be normalized to account for both the smaller dataset size and the continual counting noise. This is done as follows:

$$\tilde{q}_j = \frac{q_j n - \sum_{i=1}^{j-1} |X_j| + j\frac{\log^3(2m)\log(1/\delta)}{\varepsilon}}{|X_j|}. \tag{1}$$

The algorithm is formalized in Alg. 4.

EMSingle **- Efficient MPC Implementation of Exponential Mechanism.** We have effectively reduced the general quantiles problem to outputting one quantile for each of the disjoint intervals, each containing $O(\alpha n)$ elements, via the exponential mechanism. At first glance, it appears that computing the exponent $\exp(\varepsilon n u_{\tilde{q_j}}(b, X)/2)$ requires exponentiating a secure value, since $u_{q_j}(b, X)$ depends on the user's sensitive data. However, we can observe that the utility function is a *public* constant now for all elements $b \in [x_{(i)}, x_{(i+1)})$, and this constant is in fact $u_{q_j}(b, X) = |\tilde{q_j} - \frac{i}{n}|$ because (1)$\sum_{x \in X_j} \mathbf{1}[x \leq b] = i$, (2) the quantile $\tilde{q_j}$ is normalized using the noisy rank information as computed before. Thus, the exponentiation can now be done in the clear since it involves public information, and the implementation may proceed by sampling an index i with probability proportional to $\exp(\varepsilon n u_{q_j}(b, X)/2)|x_{(i+1)} - x_{(i)}|$, and then sampling uniformly from $[x_{(i)}, x_{(i+1)})$. Importantly, this can be achieved without any costly secure exponentiation operation. The full algorithm is presented in Alg. 5.

4 Formal Analysis

In this section, we formalize the privacy and utility guarantees of our protocols.

Theorem 4.1. *Our proposed protocol $\Pi_{\texttt{Quant}}$ (Alg. 1) securely implements $\mathcal{F}_{\texttt{Quant}}$ (Figure 1) with leakage $\mathcal{L}_{\texttt{Quant}}$ (Figure 1).*

The concept of DP leakage has been previously studied in many MPC based works [8, 9, 10, 11, 12] . This relaxes the standard MPC guarantee, where no participant can learn anything beyond the output, by permitting participants to access additional information, with the condition that this information remains differentially private. Formally, this is represented by introducing a leakage term that captures the extra information revealed during the protocol execution, which is then provided to the simulator in the security proof. This framework allows for comparing different protocols performing the same functionality based on the extent of their leakage, which can vary significantly. It also provides a more granular level of control over the leaked information, extending beyond the standard DP definition.

We adopt this approach for our security definition, requiring protocols to explicitly specify their leakage term, denoted by $\mathcal{L}$, that is revealed alongside the output in the ideal-world functionality. Specifically, we follow the formalism outlined in [8]. A protocol that implements functionality $\mathcal{F}$ is considered secure with leakage $\mathcal{L}$ if it computes $\mathcal{F}$ and its view can be simulated using $(\mathcal{F}, \mathcal{L})$. We require that $\mathcal{F}$ and $\mathcal{L}$ be jointly defined to characterize their joint distribution by a function $\hat{\mathcal{F}}$. For further details, see Appendix E.

Privacy Analysis. We now state that the outputs to both servers combined with either of the leakages is DP.

Lemma 4.2. *(Corollary of Theorem D.1). With respect to adding or removing an element from X, $(\mathcal{F}^{S_0}_{\texttt{Quant}}, \mathcal{L}^{S_0}_{\texttt{Quant}}, \mathcal{F}^{S_1}_{\texttt{Quant}})$ satisfies $(\varepsilon_1 + \varepsilon_2 + \varepsilon_3, \delta_1 + \delta_2)$-DP.*

[1]The shuffler here is instantiated by our two servers.

Public Parameters. $(\varepsilon_1, \delta_1), (\varepsilon_2, \delta_2), \varepsilon_3$ - Privacy parameters for the three stages; Q - Set of m quantiles
Initialize. $c_0 = \frac{4}{\varepsilon_2}\log^3(m)\log(\frac{1}{\delta})$, Frequency oracle $\texttt{FO} : [B] \times \mathcal{R}_{>0} \mapsto \mathcal{Y}$
Client Input. $x_i \in [B]$ - Each client's private input
Functionality.

- Compute $y_i = \texttt{FO}(x_i, \varepsilon_1), i \in [p+1, n]$. Compute a c.d.f from the responses $\{y_1, \cdots, y_n\}$ to learn intervals $I_1, I_2, \ldots, I_{2m+1}$ partitioning $[B]$, where each I_{2j} has size $|I_{2j}| \leq O(\frac{\log(B)^2 \log\log(B/\delta)}{\varepsilon_1})$ and consists of the elements with rank near nq_j (by assumption, the intervals do not intersect). (Details in Appendix A).
- Partition the dataset X into $X_j = \{[x_i] : i \in I_j\}$ for $1 \leq j \leq 2m+1$. For every partition X_j, add $n_j + n'_j$ dummy records with data $\min(I_j)$ where $n_j \sim \mathsf{ComputeDummyCounts}(2m+1, \varepsilon_2, \delta_2)$. Let $\hat{X}_j$ denote the new set of records. Normalize the quantile as $\tilde{q}_j = \frac{q_j n - \sum_{i=1}^{j-1}|\hat{X}_i| - 2jc_0}{|\hat{X}_j|}$. (Details in Appendix C)
- For $j = 1, 2, \ldots, m$:
  - Run the exponential mechanism with privacy budget ε_3 and utility function $u_{\tilde{q}_j}(\cdot, X_{2j})$; that is, sample an element $z_j \in [B]$ with probability proportional to $\exp(-n\varepsilon_3 \mathsf{err}_{X_{2j},\tilde{q}_j}(z_j)/2)$. (Details in Appendix C)
- Define $\mathcal{F}_{\texttt{Quant}} \leftarrow (z_1, \ldots, z_m)$
- Define $\mathcal{L}^{S_0}_{\texttt{Quant}} \leftarrow (y_{\pi(1)}, \ldots, y_{\pi(n)}, \sum_{i=1}^{2m+1} n'_i, |\hat{X}_1|, \ldots, |\hat{X}_{2m+1}|)$,
- Define $\mathcal{L}^{S_1}_{\texttt{Quant}} \leftarrow (y_{\pi(1)}, \ldots, y_{\pi(n)}, \sum_{i=1}^{2m+1} n_i, |\hat{X}_1|, \ldots, |\hat{X}_{2m+1}|)$
- The *functionality with leakage* $\hat{\mathcal{F}}_{\texttt{Quant}}$ is defined to be the joint distribution $(\hat{\mathcal{F}}^{S_0}_{\texttt{Quant}}, \hat{\mathcal{F}}^{S_1}_{\texttt{Quant}})$ with $\hat{\mathcal{F}}^{S_i}_{\texttt{Quant}} = (\mathcal{F}^{S_i}_{\texttt{Quant}}, \mathcal{L}^{S_i}_{\texttt{Quant}})$

Figure 1: Functionality $\hat{\mathcal{F}}_{\texttt{Quant}} = (\mathcal{F}_{\texttt{Quant}}, \mathcal{L}_{\texttt{Quant}})$. We show that $\Pi_{\texttt{Quant}}$ securely implements $\mathcal{F}_{\texttt{Quant}}$ with leakage $\mathcal{L}_{\texttt{Quant}}$

Lemma 4.3. *(Corollary of Theorem D.1). With respect to adding or removing an element from X, $(\mathcal{F}^{S_1}_{\texttt{Quant}}, \mathcal{L}^{S_1}_{\texttt{Quant}}, \mathcal{F}^{S_0}_{\texttt{Quant}})$ satisfies $(\varepsilon_1 + \varepsilon_2 + \varepsilon_3, \delta_1 + \delta_2)$-DP.*

Utility Analysis. Next, we present the formal utility analysis of $\Pi_{\texttt{Quant}}$:

Theorem 4.4. *(Proven as Theorem D.2) For any input $X \in [B]^n$, with probability at least 0.8, all estimated quantile $z_1, \ldots, z_m$ satisfy*

$$\mathsf{err}_{X,q}(z) \leq O\left(\frac{\log(mB)}{\varepsilon_3 n} + \frac{\log^3(m)\log(\frac{1}{\delta})}{\varepsilon_2 n}\right).$$

Cost Analysis. The following theorem formalizes the computational cost of our protocol.

Theorem 4.5. $\Pi_{\texttt{Quant}}$ *takes $O(m\frac{\log^2(B)}{\varepsilon}\log\log(B))$ server-side secure comparison operations.*

5 Related Work

The closest to us is [13]. We have three major differences with their work as follows. First, we consider the case of answering multiple quantiles while [13] only computes a single quantile (median). Second, their algorithm does not partition the dataset into smaller buckets, and thus involves $O(n \log n)$ secure comparisons. Third, their most efficient construction only considers $\varepsilon = \ln 2$ to avoid costly secure exponentiations the whereas we can support arbitrary values of ε without performing any secure exponentiation.

References

[1] Apple and Google. Exposure notifications private analytics. `https://github.com/google/exposure-notifications-android/blob/master/doc/ENPA.pdf`.

[2] Henry Corrigan-Gibbs, Dan Boneh, Gary Chen, Steven Englehardt, Robert Helmer, Chris Hutten-Czapski, Anthony Miyaguchi, Eric Rescorla, and Peter Saint-Andre. Privacy-preserving firefox telemetry with prio, 2020.

[3] Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 420–432, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[4] Vitaly Feldman, Audra McMillan, and Kunal Talwar. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 954–964, 2022.

[5] Haim Kaplan, Shachar Schnapp, and Uri Stemmer. Differentially private approximate quantiles. In *International Conference on Machine Learning*, pages 10751–10761. PMLR, 2022.

[6] Adam Smith. Privacy-preserving statistical estimation with optimal convergence rates. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 813–822, 2011.

[7] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, page 715–724, New York, NY, USA, 2010. Association for Computing Machinery.

[8] James Bell, Adria Gascon, Badih Ghazi, Ravi Kumar, Pasin Manurangsi, Mariana Raykova, and Phillipp Schoppmann. Distributed, private, sparse histograms in the two-server model. Cryptology ePrint Archive, Paper 2022/920, 2022.

[9] Amrita Roy Chowdhury, Chenghong Wang, Xi He, Ashwin Machanavajjhala, and Somesh Jha. Crypte: Crypto-assisted differential privacy on untrusted servers. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 603–619, New York, NY, USA, 2020. Association for Computing Machinery.

[10] Amrita Roy Chowdhury, Bolin Ding, Somesh Jha, Weiran Liu, and Jingren Zhou. Strengthening order preserving encryption with differential privacy. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 2519–2533, New York, NY, USA, 2022. Association for Computing Machinery.

[11] Phillipp Schoppmann, Lennart Vogelsang, Adrià Gascón, and Borja Balle. Secure and scalable document similarity on distributed databases: Differential privacy to the rescue. Cryptology ePrint Archive, Paper 2018/289, 2018.

[12] Adam Groce, Peter Rindal, and Mike Rosulek. Cheaper private set intersection via differentially private leakage. Cryptology ePrint Archive, Paper 2019/239, 2019.

[13] Jonas Böhler and Florian Kerschbaum. Secure multi-party computation of differentially private median. In *Proceedings of the 29th USENIX Conference on Security Symposium*, SEC'20, USA, 2020. USENIX Association.

[14] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. Answering range queries under local differential privacy. *Proc. VLDB Endow.*, 12(10):1126–1138, June 2019.

[15] Graham Cormode, Samuel Maddock, and Carsten Maple. Frequency estimation under local differential privacy. *Proc. VLDB Endow.*, 14(11):2046–2058, July 2021.

[16] Vitaly Feldman, Jelani Nelson, Huy Nguyen, and Kunal Talwar. Private frequency estimation via projective geometry. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 6418–6433. PMLR, 17–23 Jul 2022.

[17] Clément L Canonne and Abigail Gentle. Locally private histograms in all privacy regimes. In *16th Innovations in Theoretical Computer Science Conference (ITCS 2025)*, pages 25–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2025.

[18] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103. IEEE, 2007.

[19] Goldreich Oded. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

A Quantile Estimation under LDP

In this section, we show a protocol for qth quantile estimation under local differential privacy, based on [14]. The mechanism is based on CDF estimation under shuffled differential privacy.

Theorem A.1. *Given a dataset $X \in [B]^n$, for any $\delta \in (0,1)$ and $\varepsilon \in (0, C\sqrt{\log(\log B/\delta)}]$ for some $C > 0$, and $n = \Omega\left(\frac{\log B \log^2(\log B/\delta)}{\varepsilon^2}\right)$ there exists a (ε,δ)-shuffle DP mechanism that returns the CDF of X with expected maximum absolute error $O\left(\frac{\log^2 B}{\varepsilon n}\log\left(\frac{\log B}{\delta}\right)\right)$.*

Proof. Consider a dataset $X \in [B]^n$ and let $w_I = \frac{1}{n}\sum_{x\in X} \mathbb{1}[x \in I]$ be a range query. The idea, developed in [14], is to arrange the data domain $[B]$ into a b-ary tree, so that any range query can be answered by summing at most $O(\log B)$ estimates. Then, perform a b-ary search on the tree to release the q-th quantile estimate.

A b-ary partitioning of B corresponds to sets $B_0, B_1, \dots, B_\ell$ intervals with $\ell = \lceil \log_b B \rceil + 1$. Each B_i contains the intervals $\{[b^{\ell-i}(j-1), b^{\ell-i}j) : 1 \le j \le b^i\}$. We let $B_{i,j} = [b^{\ell-i}(j-1), b^{\ell-i}j)$ for convenience. For all $i \in [1,\dots,\ell]$ a frequency LDP oracle $\texttt{FO}_i$ is instantiated with X so as to provide estimates of the frequency of $B_{i,j}$ for each $j \in [b^i]$, that we indicate as $\texttt{FO}_i(j)$. To increase utility, all the n private messages, used to populate each of the frequency oracles, are randomly shuffled, so to apply amplification by shuffling [4].

Regarding frequency LDP oracles, there has been extensive research in this field in the recent year [14, 15], such that there have been found protocols returning unbiased and uncorrelated estimates with low communication $O(\log b^i)$; the utility depends on the particular mechanism. Here, we choose the Private Geometric Response [16] which achieves on each level of the tree an expected maximum absolute error for frequency estimation of $O(\frac{\sqrt{\log(b^i)\log(1/\tilde\delta)}}{n\tilde\varepsilon})$ with $n = \Omega(\frac{\log(1/\delta)}{\tilde\varepsilon^2})$ under $(\tilde\varepsilon,\tilde\delta)$-shuffle DP for $\tilde\varepsilon \in (0,1]$ (Theorem 12 in [17]). By applying advanced composition on all the ℓ frequency oracles, we obtain $(\Theta(\tilde\varepsilon\sqrt{\ell\log(1/\tilde\delta)}), (\ell+1)\tilde\delta)$-shuffle DP for $\tilde\varepsilon \le 1/\sqrt{\ell}$. Thus, by setting $\delta = (\ell+1)\tilde\delta$, there exists $C > 0$ such that $\tilde\varepsilon = \frac{\varepsilon}{C\sqrt{\ell\log(1/\tilde\delta)}} \le \frac{1}{\sqrt{\ell}}$, where the last inequality holds for $\varepsilon \le C\sqrt{\log(1/\tilde\delta)} = C\sqrt{\log((\ell+1)/\delta)}$. Therefore, we have that any i-th frequency protocol returns estimates with expected maximum absolute error $O(\frac{\sqrt{\ell\log(b^i)}\log(\ell/\delta)}{n\varepsilon})$ for $n = \Omega(\frac{\ell\log^2(\ell/\delta)}{\varepsilon^2})$.

Using the b-ary tree structure, the fact that the estimates are unbiased, and the linearity of expectation over at most $(b-1)\ell$ estimates, we get that

$$\mathbb{E}\left[\max_I |\tilde w_I - w_I|\right] \le O\left(\frac{\ell^2\log(\ell/\delta)}{\varepsilon n}\right) = O\left(\frac{\log^2 B\log(\log B/\delta)}{\varepsilon n}\right) \quad \text{for} \quad n = \Omega\left(\frac{\log B\log^2(\log B/\delta)}{\varepsilon^2}\right),$$

as $\log(b^i) \le \log(b^\ell) = O(\ell)$ and we assume $b = \Theta(1)$. □

Thus, there exists an algorithm that releases α approximate qth quantiles for $\alpha = \tilde O(\log^2 B/\varepsilon n)$. It is sufficient to release any number $r \in [B]$ such that $\tilde w_{[0,r)} \in [q-\alpha, q+\alpha]$.

B Omitted Details on Continual Observation

We write our algorithm for padding with dummy elements in Algorithm B. It generates noise which satisfies the following properties:

Lemma B.1. *With probability at least $1-\delta$, Algorithm B does not fail.*

Proof. With probability at least $1-\delta$, all $\eta_{i,j}$ satisfy $|\eta_{i,j}| \le \frac{T^2\log(2/\delta)}{\varepsilon}$. Indeed, the tail probability of the laplace distribution is $\Pr_{Y\sim\mathrm{Lap}(T/\varepsilon)}[Y \ge \log(1/\delta)\frac{T}{\varepsilon}] = \delta$, then by applying a union bound over 2^T samples we get the result. Each u_i involves at most T of these variables, and thus it satisfies $\max_i u_i \le \frac{T^3}{\varepsilon}\log(2/\delta)$. Thus, $|n_i| \ge 0$ with probability $1-\delta$ for any $i \in [k]$. □

Theorem B.2. *With respect to a single user being added or removed, the sizes of the padded bins $|X'_1| = x_1 + n_1, \dots, |X'_k| = x_k + n_k$ satisfy (ε,δ)-DP when $n_1,\dots,n_k$ are drawn from* ComputeDummyCounts.

Algorithm 2 ComputeDummyCounts, Algorithm for adding dummy users to bins with non-negative continual observation.

Input: k number of bins, privacy parameters ε, δ
Output: Bins with dummy users added for privacy.
$T \leftarrow \lfloor \log_2 k \rfloor + 1$ ▷ depth of the binary tree
for $i = 1, \ldots, T$ **do**
  for $j = 1, \ldots, 2^{i-1}$ **do**
    Sample $\eta_{j,i} \sim \text{Lap}(\frac{T}{\varepsilon})$
  end for
end for
for $i = 1, \ldots, k$ **do** ▷ For each bin, compute dummy users
  Let $(b_1, \ldots, b_T)$ be binary representation of i
  $u_i \leftarrow \sum_{j:b_j=1} \eta_{\lfloor i/2^{T-j} \rfloor, j}$
  $n_i \leftarrow \frac{1}{\varepsilon} T^3 \log(2/\delta) + u_i - u_{i-1}$
  if $n_i < 0$ **then**
    return "Fail"
  end if
end for
Return $n_1, \ldots, n_k$

Algorithm 3 FormIntervals

Input. $X, Q, B, \varepsilon, \delta$
for $j = 1$ to m **do**
  $\mathcal{M} \leftarrow$ protocol for CDF estimation with max absolute error α under (ε, δ)-shuffle DP (Theorem A.1)
  $c_{2j-1} \leftarrow c$ s.t. $\mathcal{M}([0, c)) \in [q_j - 2\alpha, q_j - \alpha]$
  $c_{2j} \leftarrow c$ s.t. $\mathcal{M}([0, c)) \in [q_j + \alpha, q_j + 2\alpha]$
end for
Define $I_j = [c_{i-1}, c_i)$ for $j = 1$ to $2m + 1$ with $c_0 = 0$ and $c_{2m+1} = B$.
Output. $I_1, \ldots, I_{2m+1}$ ▷ This is a partitioning of $[B]$

Proof. Conditioned on no fails occurring, the sizes are the set of consecutive differences of continual observation applied to $x_i + n_i$. Thus, as the addition of $\frac{T^3}{\varepsilon} \log(2/\delta)$ is a post processing step, the sizes satisfy $(\varepsilon, 0)$-DP by standard analysis of the continual counting mechanism [7]. The final privacy guarantee holds from the probability δ of a fail. □

Theorem B.3. *With probability at least* $1 - \delta$*,*

$$\left| \sum_{j=1}^{i} n_i - \frac{i}{\varepsilon} T^3 \log(2/\delta) \right| \leq \frac{1}{\varepsilon} T^3 \log(2/\delta)).$$

for all $1 \leq i \leq k$ *and* $T = \lfloor \log_2 k \rfloor + 1$.

Proof. For any $j \in [k]$, by construction, the left hand side is equal to $|u_j|$ which is the sum of at most T Laplace random variables with maximum error $\frac{T^2}{\varepsilon} \log(2/\delta)$. Thus the claim follows. □

C Implementation Details of Main Algorithm

In this section, we describe the secure implementation of the subroutines of Π_{Quant} (Algorithm 1). The first subroutine, FormIntervals, first uses local differential privacy and a secure shuffle to estimate the cumulative distribution function of X, which can be used to estimate $\mathsf{rank}_X(z) = \sum_{i=1}^{n} \mathbf{1}[x_i \leq z]$ within error $\tilde{O}(\frac{\log^2(B)}{\varepsilon})$ by Theorem A.1. We can then query the CDF to form intervals $I_1, \ldots, I_{2m+1}$ which partition the data domain $[B]$, such that I_{2j} corresponds to quantile q_j and is guaranteed, by the assumption that $|q_i - q_j| \geq \tilde{\Omega}(\frac{\log^2 B}{\varepsilon n})$, to have at most $O(\frac{\log^2(B)}{\varepsilon})$ elements of $[X]$ in it. This process is shown in Algorithm 3.

Once the intervals are computed, the servers interact with the users again to form the buckets $X_j = \{x \in X : x \in I_j\}$. We denote this subroutine by FormBuckets. To do so, the servers query the users' data x_i together with an index b_i

Algorithm 4 FormBuckets

Input. $[X]$ of length n, intervals $I_1, \dots, I_{2\ell+1}$, parameters ε, δ, B
$S = \emptyset$
for $i = 1$ to n **do**
  Collect $([x_i], [b_i])$ from user i, where b_i is the index where $x_i \in I_{b_i}$.
  $S = S \cup \{([x_i], [b_i])\}$
end for
Servers $a \in \{0, 1\}$ sample $n_0^a, n_1^a, \dots, n_{2\ell+1}^a \sim \mathsf{ComputeDummyCounts}(2\ell + 1, \varepsilon)$
Servers $a \in \{0, 1\}$ initializes S_a to be n_0^a users with data $([\min I_1], 1)$
for $b = 1$ to $2\ell + 1$ **do**
  Server 0 computes shares for n_b^0 users with data $([\min I_b], [b])$
  Add shares to S_a
end for
Server 1 adds to S_1 similarly
Servers send shares for S_0, S_1 to each other.
Securely shuffle $\mathcal{S} = [S] \cup [S_0] \cup [S_1]$
Reveal b_s for $1 \le s \le |\mathcal{S}|$
Output $[X'_1], \dots, [X'_\ell]$ where $[X_j] = \{[x_s] : b_s \in I_{2j}\}$

such that $x_i \in I_{b_i}$. The users send these data with secret sharing. The servers could form each X_j by shuffling the responses, then revealing the b_i. However, while this would leak nothing about the x_i, it would leak the sizes of X_j, which are sensitive values.

To fix this, the servers first add a random number n_j of dummy users in each interval I_j. The n_j are drawn using continual counting so that they are non-negative and so that the partial sums $\sum_{j=1}^{i} n_j$ are concentrated—see Theorem B.3. This allows quantiles to be accurately renormalized later. Each server independently samples and adds dummy users, which ensures that sufficient noise for differential privacy is present even when one server behaves maliciously.

This process appears in Algorithm 4.

Having formed data buckets $[X_j]$ such that $[X_{2j}]$ corresponds to quantile q_j, the final step is to run the exponential mechanism on $[X_{2j}]$ to produce a quantile. Within bucket X_{2j}, the quantile $\tilde{q}_j$ corresponding to quantile q_j in all of X is $\tilde{q}_j = \frac{nq_j - \sum_{i=1}^{2j-1} |X_i|}{|X_{2j}|}$. Accounting for the dummy users added to the system, it is

$$\tilde{q}_j = \frac{q_j n - \sum_{i=1}^{j-1} |X_j| + j\frac{\log^3(2m)\log(1/\delta)}{\varepsilon}}{|X_j|}.$$

Thus, the exponential mechanism is run on the data $[X_j]$ with quantile $\tilde{q}_j$. It returns an element $z \in [B]$ with probability $\exp(-\varepsilon n \mathsf{err}_{X_j, \tilde{q}_j}(z)/2)$, which satisfies ε-DP because the sensitivity of $\mathsf{err}_{X_j, \tilde{q}_j}(z)$ is $\frac{1}{n}$. Our implementation uses the observation that $\mathsf{err}_{X,\tilde{q}}(z) = |\frac{1}{n}i - \tilde{q}|$ for all $z \in x_{(i)}, x_{(i+1)}$. Thus, if we securely sort $[X_j]$, then the exponential mechanism can be run on the intervals $[x_{(0)}, x_{(1)}), \dots, [x_{(t)}, x_{(t+1)})$ where $t = |X_j|$ and $x_{(0)} = 0$ and $x_{(t+1)} = B$. If each interval is chosen with probability proportional to $(x_{(i+1)} - x_{(i)})|\frac{1}{n}i - \tilde{q}|$ and then a uniformly random integer from the interval is returned, this is an equivalent sampling procedure as the exponential mechanism. We show our implementation of this, EMSingle, in Algorithm 5. Crucially, the algorithm keeps all its data secret shared except for the final revealed sample, and only requires $O(|X_j|)$ secret-shared operations to do this.

Having put all the pieces together, we may then analyze the running time of AlgMPCQuantile

Theorem C.1. AlgMPCQuantile *takes* $O(k + \log(B)^2)$ *user-side operations,* $O(nk\log(B)^3)$ *plaintext server-side operations, and* $O(k\frac{\log^2(B)}{\varepsilon}\log\log(B))$ *server-side secret share operations.*

Proof. The user-side computations follow from the fact that each user runs local DP on their data $\log B$ times, each taking $O(\log(B))$ time [16]. Additionally, they must compute which interval they belong to, requiring an additional $O(k)$ time. Server-side, computing the c.d.f. of the data in FormIntervals requires $O(nk\log(B)^3)$ server-side plaintext computations [16]. The next step, FormBuckets, uses $O(n)$ plaintext computations. Then, EMSingle takes $O(|X_{2j}|\log(|X_{2j}|))$ secret-shared computations accounting for initially sorting the data. This is at most $O(\frac{\log^2(B)}{\varepsilon}\log\log(B))$ accounting for the upper bound on the size of $|X_{2j}|$. Thus, the total number of operations is $O(n + k\frac{\log^2(B)}{\varepsilon}\log\log(B))$. □

Algorithm 5 EMSingle

Input. $[X]$ (in sorted order), quantile q, ε, δ, B

Servers insert $[x_0] = 0$ and $[x_{n+1}] = B$.
for $i = 1$ to $n + 1$ **do**
    $[l_i] = [x_i - x_{i-1}]$ (Interval lengths)
    $[p_i] = \exp(-\frac{\varepsilon}{2}|i - qn|) * [l_i]$
end for
$[P] = \sum_{i=1}^{n}[p_i]$
Generate $[u] \sim [0, 1]$
Compute $[r] = [u * P]$
Use linear scan to find $[i]$ such that $[p_{i-1}] \leq [r] \leq [p_i]$
Generate $[u'] \sim [0, 1]$
Compute $[ans] = [x_{i-1}] + \mathsf{truncate}(u' * [x_i - x_{i-1}])$ ▷ Indexing at $[i]$ can be implemented with linear scan.
return ans in the clear

D Privacy and Utility Analysis for $\Pi_{\texttt{Quant}}$

Theorem D.1. *With respect to adding or removing an element from X, $\mathcal{M}_{Quant}$ satisfies $(\varepsilon_1 + \varepsilon_2 + \varepsilon_3, \delta_1 + \delta_2)$-DP.*

Proof. The output $z_1, \ldots, z_m$ satisfies ε_3-DP from the exponential mechanism and parallel composition. The output $\sum_{i=1}^{j} |\hat{X}_i|, |\tilde{X}_1|, |\tilde{X}_2|, \ldots, |\tilde{X}_k|$ satisfies $(\varepsilon_2, \delta_2)$-DP from Theorem B.2, where the initial sum can be made independent of the rest of the counts by adding an extra $X_0 \sim \frac{\ln(1/\delta)}{\varepsilon} + Lap(\frac{1}{\varepsilon})$ users to the first bin (creating a negligible change in utility). Finally, the privacy of the release $(y_{\pi(1)}, \ldots, y_{\pi(n)}$ satisfies $(\varepsilon_1, \delta_1)$-DP by Theorem A.1. The total privacy guarantee follows from composition. □

For utility, we have the following guarantee.

Theorem D.2. *For any input $X \in [B]^n$, if both servers follow the protocol $\mathcal{M}_{Quant}$, with probability at least 0.8, all estimated quantile $z_1, \ldots, z_m$ satisfy*

$$\mathsf{err}_{X,q}(z) \leq O\left(\frac{\log(mB)}{\varepsilon_3 n} + \frac{\log^3(m)\log(\frac{1}{\delta})}{\varepsilon_2 n}\right).$$

Proof. Our proof consists of arguing that each $\tilde{q}_j$ quantile in $\tilde{X}_j$ is close to the q_jth quantile in X_j, combined with the utility of the exponential mechanism. The second part is straightforward; by a standard argument such as that in [18], since the output size is bounded by B, the exponential mechanism is guaranteed to return a z_j such that

$$\mathsf{err}_{\tilde{X}_j,\tilde{q}_j}(z_j) \leq O\left(\frac{\log(mB)}{|\tilde{X}_j|\varepsilon_3}\right)$$

for each $1 \leq j \leq m$, with probability at least $\frac{1}{10}$. Let $\mathrm{rank}_X(z) = \sum_{x \in X} \mathbf{1}[x \leq z]$. We have

$$|\tilde{X}_j|\mathsf{err}_{\tilde{X}_j,\tilde{q}_j}(z_j) = |\tilde{q}_j|\tilde{X}_j| - \mathsf{rank}_{\tilde{X}_j}(z_j)|$$
$$|X|\mathsf{err}_{X,q_j}(z_j) = |q_j|X| - \mathsf{rank}_X(z_j)|$$

Therefore, we have

$$\begin{aligned}
|X|\mathsf{err}_{X,q_j}(z_j) - |\tilde{X}_j|\mathsf{err}_{\tilde{X}_j,\tilde{q}_j}(z_j) &\leq \left|\tilde{q}_j|\tilde{X}_j| - q_j|X| - \mathsf{rank}_{\tilde{X}_j}(z_j) + \mathsf{rank}_X(z_j)\right| \\
&= \left|\frac{q_j n - \sum_{i=1}^{j-1}|\tilde{X}_i| + 2jc_0}{|\tilde{X}_j|}|\tilde{X}_j| - q_j|X| - \mathsf{rank}_{X_j}(z_j) - n_j + \mathsf{rank}_X(z_j)\right| \\
&= \left|-\sum_{i=1}^{j-1}|\tilde{X}_i| + 2jc_0 - \mathsf{rank}_{X_j}(z_j) - n_j + \mathsf{rank}_X(z_j)\right|.
\end{aligned}$$

Suppose that $\min(X_j) \le z_j \le \max(X_j)$. We have that $\mathsf{rank}_{X_j}(z_j) + \sum_{i=1}^{j-1} |X_j| = \mathsf{rank}_X(z_j)$, and the above sum reduces to

$$\left| -\sum_{i=1}^{j-1}(|X_i| + n_i) + 2jc_0 - n_j + \sum_{i=1}^{j-1} |X_j| \right| = \left| 2jc_0 - \sum_{i=1}^{j} n_i \right|.$$

By Theorem B.3, we know the last sum is at most $\frac{4\log(m)^3 \log(\frac{1}{\delta})}{\varepsilon_2}$ with probability at least $1-\delta$, and thus we have

$$\begin{aligned}\mathsf{err}_{X,q_j}(z_j) &\le \frac{1}{|X|}|\tilde{X}_j|\mathsf{err}_{\tilde{X}_j,\tilde{q}_j}(z_j) + \frac{1}{|X|}O\left(\frac{\log(m)^3}{\varepsilon_2}\right) \\ &\le O\left(\frac{\log(mB)}{n\varepsilon_3}\right) + O\left(\frac{\log(m)^3}{n\varepsilon_2}\right).\end{aligned}$$

Finally, we show that $\min(X_j) \le z_j \le \max(X_j)$. It is sufficient to show $\tilde{q}_j \in [\beta, 1-\beta]$, with $\beta = \frac{\log(mB)}{|\tilde{X}_j|\varepsilon_2}$. This follows by construction, as

$$\begin{aligned}&q_j n - \sum_{i=1}^{j-1} |X_i| \in \left[\frac{\log(m)^3}{\varepsilon_2} + \frac{\log(mB)}{\varepsilon_3}, |X_j| - \frac{\log^3(m)}{\varepsilon_2} - \frac{\log(mB)}{\varepsilon_3}\right] \\ &\Rightarrow q_j n - \sum_{i=1}^{j-1} |\tilde{X}_i| + 2jc_0 \in \left[\frac{\log(mB)}{\varepsilon_3}, |X_j| - \frac{\log(mB)}{\varepsilon_3}\right] \\ &\Rightarrow \tilde{q}_j|\tilde{X}_j| \in \left[\frac{\log(mB)}{\varepsilon_3}, |\tilde{X}_j| - \frac{\log(mB)}{\varepsilon_3}\right],\end{aligned}$$

and the claim follows since $\left| z_j - \tilde{q}_j|\tilde{X}_j| \right| \le \frac{\log(mB)}{\varepsilon_2}$ from the error guarantee. □

E Functionality with Leakage

Definition E.1. *(Functionality with leakage). Let $\hat{\mathcal{F}} = (\hat{\mathcal{F}}_1, \hat{\mathcal{F}}_2) = ((\mathcal{F}_1, \mathcal{L}_1), (\mathcal{F}_2, \mathcal{L}_2))$ be a two-party functionality. Let $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2)$ and $\mathcal{L} = (\mathcal{L}_1, \mathcal{L}_2)$. We say that a two-party protocol Π securely implements $\mathcal{F}$ with leakage $\mathcal{L}$, if for each $b \in \{1,2\}$ there exists a probabilistic polynomial-time algorithm Sim_b such that for all $x_1 \in X_1, x_2 \in X_2$ the output of $Sim_b((x_b, \hat{\mathcal{F}}_b(x_1,x_2)), \mathcal{F}(x_1,x_2))$ is computationally indistinguishable from $(View^b_\Pi(x_1,x_2), \Pi(x_1,x_2))$. We call $\hat{F}$ the functionality with leakage.*

Proof. The proof of Thm. 4.1 follows from standard arguments in the $(\mathcal{F}_{\texttt{Shuffle}}, \mathcal{F}_{\texttt{Sort}})$ hybrid model [19] where $\mathcal{F}_{\texttt{Shuffle}}(\mathcal{F}_{\texttt{Shuffle}})$ represents the ideal functionality for secure shuffle (sort). □