Private Frequency Estimation via Projective Geometry

Vitaly Feldman* Jelani Nelson† Huy L. Nguyen‡ Kunal Talwar§

December 12, 2022

Abstract

In this work, we propose a new algorithm ProjectiveGeometryResponse (PGR) for locally differentially private (LDP) frequency estimation. For a universe size of k and with n users, our ε-LDP algorithm has communication cost $\lceil \log_2 k \rceil$ bits in the private coin setting and $\varepsilon \log_2 e + O(1)$ in the public coin setting, and has computation cost $O(n + k \exp(\varepsilon) \log k)$ for the server to approximately reconstruct the frequency histogram, while achieving the state-of-the-art privacy-utility tradeoff. In many parameter settings used in practice this is a significant improvement over the $O(n + k^2)$ computation cost that is achieved by the recent PI-RAPPOR algorithm (Feldman and Talwar; 2021). Our empirical evaluation shows a speedup of over 50x over PI-RAPPOR while using approximately 75x less memory for practically relevant parameter settings. In addition, the running time of our algorithm is within an order of magnitude of HadamardResponse (Acharya, Sun, and Zhang; 2019) and RecursiveHadamardResponse (Chen, Kairouz, and Ozgur; 2020) which have significantly worse reconstruction error. The error of our algorithm essentially matches that of the communication- and time-inefficient but utility-optimal SubsetSelection (SS) algorithm (Ye and Barg; 2017). Our new algorithm is based on using Projective Planes over a finite field to define a small collection of sets that are close to being pairwise independent and a dynamic programming algorithm for approximate histogram reconstruction on the server side. We also give an extension of PGR, which we call HybridProjectiveGeometryResponse, that allows trading off computation time with utility smoothly.

1 Introduction

In the so-called *federated* setting, user data is distributed over many devices which each communicate to some central server, after some local processing, for downstream analytics and/or machine learning tasks. We desire such schemes which (1) minimize communication cost, (2) maintain privacy of the user data while still providing utility to the server, and (3) support efficient algorithms for the server to extract knowledge from messages sent by the devices. Such settings have found applications to training language models for such applications as autocomplete and spellcheck, and other analytics applications in Apple iOS [TVV+17] and analytics on settings in Google Chrome [EPK14].

*Apple Inc.

†UC Berkeley. minilek@berkeley.edu. Supported by NSF grant CCF-1951384, ONR grant N00014-18-1-2562, and ONR DORECG award N00014-17-1-2127.

‡Northeastern University. hu.nguyen@northeastern.edu. Supported in part by NSF CAREER grant CCF-1750716 and NSF grant CCF-1909314.

§Apple Inc. ktalwar@apple.com.

The gold standard for protecting privacy is for a scheme to satisfy *differential privacy*. In the so-called *local model* that is relevant to the federated setting, there are n users with each user i holding some data $d_i \in \mathcal{D}$. Each user then uses its own private randomness r_i and data d_i to run a *local randomizer* algorithm that produces a random message M_i to send to the server. We say the scheme is ε-*differentially private* if for all users i, any possible message m, and any $d \neq d'$,

$$\mathbb{P}(M_i = m | d_i = d) \leqslant e^{\varepsilon} \mathbb{P}(M_i = m | d_i = d').$$

Note a user could simply send an unambiguous encoding of d_i, which allows the server to learn d_i exactly (perfect utility), but privacy is not preserved; such a scheme does not preserve ε-DP for any finite ε. On the opposite extreme, the user could simply send a uniformly random message that is independent of d_i, which provides zero utility but perfect privacy ($\varepsilon = 0$). One can hope to develop schemes that smoothly increase utility by relaxing privacy (i.e., by increasing ε).

This work addresses the problem of designing efficient schemes for locally differentially private frequency estimation. In this problem, one defines a histogram $x \in \mathbb{R}^k$ where x_d is the number of users i with $d_i = d$, and $k = |\mathcal{D}|$. From the n randomized messages it receives, the server would like to approximately reconstruct the histogram, i.e., compute some $\tilde{x}$ such that $\|x - \tilde{x}\|$ is small with good probability over the randomness $r = (r_1, \ldots, r_n)$, for some norm $\|\cdot\|$. Our goal is to design schemes that obtain the best-known privacy-utility trade-offs, while being efficient in terms of communication, computation time, and memory. In this work we measure *utility loss* as the mean squared error (MSE) $\mathbb{E}_r \frac{1}{k}[\|x - \tilde{x}\|_2^2]$, with lower MSE yielding higher utility. Note that such a scheme should specify both the local randomizer employed by users, and the reconstruction algorithm used by the server.

There are several known algorithms for this problem; see Table 1. To summarize, the best known utility in prior work is achieved by SubsetSelection and slightly worse utility is achieved by the RAPPOR algorithm [EPK14] that is based on the classical binary randomized response [War65]. Unfortunately, both RAPPOR and Subset Selection have very high communication cost of $\approx kH(1/(e^{\varepsilon}+1))$, where H is the binary entropy function and server-side running time of $\tilde{O}(nk/\exp(\varepsilon))$. Large k is common in practice, e.g., k may be the size of a lexicon when estimating word frequencies to train language models. This has led to numerous and still ongoing efforts to design low-communication protocols for the problem [HKR12, EPK14, BS15, KBR16, WHN+19, WBLJ17, YB17, ASZ19, BNS19, BNST20, CKÖ20, FT21, SCB+21].

One simple approach to achieve low communication and computational complexity is to use a simple k-ary RandomizedResponse algorithm (e.g. [WBLJ17]). Unfortunately, its utility loss is suboptimal by up to an $\Omega(k/e^{\varepsilon})$ factor; recall k is often large and ε is at most a small constant, and thus this represents a large increase in utility loss. In the $\varepsilon < 1$ regime asymptotically optimal utility bounds are known to be achievable with low communication and computational costs [BS15, BNS19, BNST20]. The first low-communication algorithm that achieves asymptotically optimal bounds in the $\varepsilon > 1$ regime is given in [WBLJ17]. It communicates $O(\varepsilon)$ bits and relies on shared randomness. However, it matches the bounds achieved by RAPPOR only when e^{ε} is an integer and its computational cost is still very high and comparable to that of RAPPOR. Two algorithms, HadamardResponse [ASZ19] and RecursiveHadamardResponse [CKÖ20], show that it is possible to achieve low communication, efficient computation (only $\Theta(\log k)$ slower than RandomizedResponse) and *asymptotically* optimal utility. However, their utility loss in practice is suboptimal by a constant factor (e.g. our experiments show that these algorithms have an MSE that is

scheme name	communication	utility loss	server time
RandomizedResponse	$\lceil \log_2 k \rceil$	$\frac{n(2e^\varepsilon+k)}{(e^\varepsilon-1)^2}$	$n+k$
RAPPOR [EPK14]	k	$\frac{4ne^\varepsilon}{(e^\varepsilon-1)^2}$	nk
SubsetSelection [YB17, WHN+19]	$\frac{k}{e^\varepsilon}(\varepsilon+O(1))$	$\frac{4ne^\varepsilon}{(e^\varepsilon-1)^2}$	$n\frac{k}{e^\varepsilon}$
PI-RAPPOR [FT21]	$\lceil \log_2 k \rceil + O(\varepsilon)$	$\frac{4ne^\varepsilon}{(e^\varepsilon-1)^2}$	$\min(n+k^2, n\frac{k}{e^\varepsilon})$, or $n+ke^{2\varepsilon}\log k$ (*this work*)
HadamardResponse [ASZ19]	$\lceil \log_2 k \rceil$	$\frac{36ne^\varepsilon}{(e^\varepsilon-1)^2}$	$n+k\log k$
RecursiveHadamardResponse [CKÖ20]	$\lceil \log_2 k \rceil$	$\frac{8ne^\varepsilon}{(e^\varepsilon-1)^2}$	$n+k\log k$
ProjectiveGeometryResponse	$\lceil \log_2 k \rceil$	$\frac{4ne^\varepsilon}{(e^\varepsilon-1)^2}$	$n+ke^\varepsilon\log k$
HybridProjectiveGeometryResponse	$\lceil \log_2 k \rceil$	$(1+\frac{1}{q-1})\frac{4ne^\varepsilon}{(e^\varepsilon-1)^2}$	$n+kq\log k$

Table 1: Known local-DP schemes for private frequency estimation compared with ours. Utility bounds are given up to $1+o_k(1)$ multiplicative accuracy for ease of display and running times are asymptotic. For brevity we only state bounds for $\varepsilon \leqslant \log k$. Some of algorithms assume k is either a power of 2 or some other prime power and otherwise potentially worsen in some parameters due to round-up issues; we ignore this issue in the table. The communication and server time for RAPPOR are random variables which are never more than k and nk, respectively, but RAPPOR can be implemented so that in expectation the communication and runtimes are asymptotically equal to SubsetSelection. For HybridProjectiveGeometryResponse, q can be chosen as any prime in $[2, \exp(\varepsilon)+1]$. The utility loss here is the proven upper bound on the variance for PGR, HR and RHR, and the analytic expression for the variance for the others. The communication bounds are in the setting of private coin protocols. As with RHR, PGR and HPGR can also both achieve improved communication in the public coin model; see Appendix B.

over $2\times$ higher for $\varepsilon = 5$ than SubsetSelection; see Fig. 2).

Recent work of Feldman and Talwar [FT21] describes a general technique for reducing communication of a local randomizer without sacrificing utility and, in particular, derives a new low communication algorithm for private frequency estimation via pairwise independent derandomization of RAPPOR. Their algorithm, referred to as PI-RAPPOR, achieves the same utility loss as RAPPOR and has the server-side running time of $\tilde{O}(\min(n+k^2, nk/\exp(\varepsilon)))$. The running time of this algorithm is still prohibitively high when both n and k are large.

We remark that while goals (1)-(3) from the beginning of this section are all important, goal (2) of achieving a good privacy/utility tradeoff is unique in that poor performance cannot be mitigated by devoting more computational resources (more parallelism, better hardware, increased bandwidth, etc.). After deciding upon a required level of privacy ε, there is a fundamental limit as to how much utility can be extracted given that level of privacy; our goal in this work is to understand whether that limit can be attained in a communication- and computation-efficient way.

Our main contributions. We give a new private frequency estimation algorithm ProjectiveGeometryResponse (PGR) that maintains the best known utility and low communication while significantly improving computational efficiency amongst algorithms with similarly good utility. Using

our ideas, we additionally give a new reconstruction algorithm that can be used with the PI-RAPPOR mechanism to speed up its runtime from $O(k^2/\exp(\varepsilon))$ to $O(k\exp(2\varepsilon)\log k)$ (albeit, this runtime is still slower than PGR's reconstruction algorithm by an $\exp(\varepsilon)$ factor). We also show a general approach that can further improve the server-side runtime at the cost of slightly higher reconstruction error, giving a smooth tradeoff: for any prime $2 \leqslant q \leqslant \exp(\varepsilon)+1$, we can get running time $O(n + qk\log k)$ with error only $(1 + 1/(q-1))$ times larger than the best known bound[1]. Note that for $q = 2$ we recover the bounds achieved by HR and RHR. Our mechanisms require $\lceil \log_2 k \rceil$ per device in the private coin model, or $\varepsilon \log_2 e + O(1)$ bits in the public coin model (see Appendix B). As in previous work, our approximate reconstruction algorithm for the server is also parallelizable, supporting linear speedup for any number of processors $P \leqslant \min\{n, k\exp(\varepsilon)\}$. We also perform an empirical evaluation of our algorithms and prior work and show that indeed the error of our algorithm matches the state of the art will still being time-efficient.

As has been observed in previous work [ASZ19], the problem of designing a local randomizer is closely related to the question of existence of set systems consisting of sets of density $\approx \exp(-\varepsilon)$ which are highly symmetric, and do not have positive pairwise dependencies. The size of the set system then determines the communication cost, and its structural properties may allow for efficient decoding. We show that projective planes over finite fields give us set systems with the desired properties, leading to low communication and state-of-the-art utility. We also show a novel dynamic programming algorithm that allows us to achieve server runtime that is not much worse than the fastest known algorithms.

As in a lot of recent work on this problem, we have concentrated on the setting of moderately large values for the local privacy parameter ε. This is a setting of interest due to recent work in privacy amplification by shuffling [BEM+17, CSU+19, EFM+19, BBGN19, FMT21] that shows that local DP responses, when shuffled across a number of users so that the server does not know which user sent which messages, satisfy a much stronger central privacy guarantee. Asymptotically, ε-DP local randomizers aggregated over n users satisfy $(O(\sqrt{\frac{e^\varepsilon \ln \frac{1}{\delta}}{n}}), \delta)$-DP. The hidden constants here are small: as an example with $n = 10,000$ and $\varepsilon = 6$, shuffling gives a central DP guarantee of $(0.3, 10^{-6})$-DP. This motivation from shuffling is also the reason why our work concentrates on the setting of private coin protocols, as shared randomness seems to be incompatible with shuffling of private reports. We note that while constant factors improvement in error may seem small, these algorithm are typically used for discovering frequent items from power law distributions. A constant factor reduction in variance of estimating any particular item frequency then translates to a corresponding smaller noise floor (for a fixed false positive rate, say), which then translates to a constant factor more items being discovered.

1.1 Related Work

A closely related problem is finding "heavy hitters", namely all elements $j \in [k]$ with counts higher than some given threshold; equivalently, one wants to recover an approximate histogram $\tilde{x}$ such that $\|x - \tilde{x}\|_\infty$ is small (the non-heavy hitters i can simply be approximated by $\tilde{x}_i = 0$). In this problem the goal is to avoid linear runtime dependence on k that would result from doing frequency

[1]For both PGR and HPGR we have stated runtime bounds assuming that certain quantities involving $k, \exp(\varepsilon)$ are prime powers. If this is not the case, runtimes may increase by a factor of $\exp(\varepsilon)$ for PGR, or q for HPGR; we note that PI-RAPPOR also has this feature.

estimation and then checking all the estimates. This problem is typically solved using a "frequency oracle" which is an algorithm that for a given $j \in [k]$ returns an estimate of the number of j's held by users (typically without computing the entire histogram) [BS15, BNST20, BNS19]. Frequency estimation is also closely related to the discrete distribution estimation problem in which inputs are sampled from some distribution over $[k]$ and the goal is to estimate the distribution [YB17, ASZ19]. Indeed, bounds for frequency estimation can be translated directly to bounds on distribution estimation by adding the sampling error. We note that even for the problem of implementing a private frequency oracle, our PGR scheme supports answering queries faster than PI-RAPPOR by factor of $\Theta(\exp(\varepsilon))$.

2 Preliminaries

Our mechanisms are based on projective spaces, and below we review some basic definitions and constructions of such spaces from standard vector spaces.

Definition 2.1. For a given vector space V, the projective space $P(V)$ is the set of equivalence classes of $V \setminus \{0\}$, where 0 denotes the zero vector, under the following equivalence relation: $x \sim y$ iff $x = cy$ for some scalar c. Each equivalence class is called a *(projective) "point"* of the projective space. Let $p : V \setminus \{0\} \to P(V)$ be the mapping from each vector $v \in V$ to its equivalence class. If V has dimension t then $P(V)$ has dimension $t - 1$.

We will also use subspaces of the projective space $P(V)$.

Definition 2.2. A projective subspace W of $P(V)$ is a subset of $P(V)$ such that there is a subspace U of V where $p(U \setminus \{0\}) = W$. If U has dimension t then W has dimension $t - 1$.

It should be noted that intersections of projective subspaces are projective subspaces. Let q be a prime power and $\mathbb{F}_q^t$ the t-dimensional vector space over the field $\mathbb{F}_q$. We will work with $P\left(\mathbb{F}_q^t\right)$ and its subspaces.

Definition 2.3. A vector $x \in \mathbb{F}_q^t$ is called canonical if its first non-zero coordinate is 1.

Each equivalence class can be specified by its unique canonical member.

3 ProjectiveGeometryResponse description and analysis

Our PGR scheme is an instantiation of the framework due to [ASZ19]. In their framework, the local randomizer is implemented as follows. There is a universe U of outputs and each input v corresponds to a subset $S(v)$ of outputs. All the subsets $S(v)$ for different values of v have the same size. Given the input v, the local randomizer returns a uniformly random element of $S(v)$ with probability $\frac{e^\varepsilon |S(v)|}{|S(v)|e^\varepsilon + |U| - |S(v)|}$ and a uniformly random element of $U \setminus S(v)$ with probability $\frac{|U| - |S(v)|}{|S(v)|e^\varepsilon + |U| - |S(v)|}$. The crux of the construction is in specifying the universe U and the subsets $S(v)$.

PGR works for $k = \frac{q^t - 1}{q - 1}$ for some integer t (other values of k need to be rounded up to the nearest such value). We identify the k input values with k canonical vectors in $\mathbb{F}_q^t$ and the corresponding projective points in $P\left(\mathbb{F}_q^t\right)$. We also identify the output values with projective points

in $P\left(\mathbb{F}_q^t\right)$. The subsets $S(v)$ are the $(t-2)$-dimensional projective subspaces of $P\left(\mathbb{F}_q^t\right)$. There are $\frac{q^t-1}{q-1}$ $(t-2)$-dimensional projective subspaces, which is the same as the number of projective points. For a canonical vector v, the set $S(v)$ is the $(t-2)$-dimensional projective subspace such that for all $u \in p^{-1}(S(v))$, we have $\langle u, v\rangle = 0$. Each $(t-2)$-dimensional projective subspace contains $\frac{q^{t-1}-1}{q-1}$ projective points. In other words, each set $S(v)$ contains $\frac{q^{t-1}-1}{q-1}$ messages out of the universe of $\frac{q^t-1}{q-1}$ messages.

An important property of the construction is the symmetry among the intersections of any two subsets $S(v)$.

Claim 3.1. Consider a t-dimensional vector space V. The intersection of any two $(t-2)$-dimensional projective subspaces of $P(V)$ is a $(t-3)$-dimensional projective subspace.

Proof. Let I be the intersection of two projective subspaces S_1 and S_2. Recall that I, S_1, S_2 are projective subspaces corresponding to subspaces of V. Assume for contradiction that the dimension $d-1$ of the intersection I is lower than $t-3$. Starting from a basis $v_1, \ldots, v_d$ of $p^{-1}(I) \cup \{0\}$, we can extend it with $u_1, \ldots, u_{t-1-d}$ to form a basis of the subspace $p^{-1}(S_1) \cup \{0\}$. We can also extend $v_1, \ldots, v_d$ with $w_1, \ldots, w_{t-1-d}$ to form a basis of $p^{-1}(S_2) \cup \{0\}$. Because $d + 2(t-1-d) = t + (t-2-d) > t$, the collection of vectors $v_1, \ldots, v_d, u_1, \ldots, u_{t-1-d}, w_1, \ldots, w_{t-1-d}$ must be linearly dependent. There must exist nonzero coefficients so that $\sum_i \alpha_i v_i + \sum_j \beta_j u_j + \sum_k \gamma_k w_k = 0$. This means $\sum_k \gamma_k w_k = -\sum_i \alpha_i v_i - \sum_j \beta_j u_j$ is a non-zero vector in $p^{-1}(S_1) \cap p^{-1}(S_2)$ but it is not in $p^{-1}(I)$, which is a contradiction. □

To ease the presentation we define $c_{int} = \frac{q^{t-2}-1}{q-1}$ to be the size of the intersection of two subsets $S(v)$ and let $c_{set} = \frac{q^{t-1}-1}{q-1}$ denote the size of each subset $S(v)$. Notice that $c_{set}^2 \geqslant k \cdot c_{int}$ i.e. $(c_{set}/c_{int})^2 \geqslant k/c_{int}$.

Each user with input v sends a projective point e with probability $e^\varepsilon p$ if e is in $S(v)$ and probability p otherwise. We have

$$e^\varepsilon p c_{set} + p(k - c_{set}) = 1,$$
$$\text{so that } p = \frac{1}{(e^\varepsilon - 1)\, c_{set} + k}.$$

The server keeps the counts on the received projective points in a vector $y \in \mathbb{Z}^k$. Thus, the total server storage is $O(k)$. We estimate x_v by computing

$$\tilde{x}_v = \alpha \left(\sum_{u \in S_v} y_u \right) + \beta \sum_u y_u$$

where α and β are chosen so that it is an unbiased estimator. Note $\sum_u y_u = n$. We would like $\mathbb{E}\, \tilde{x}_v = x_v$ for all v. Notice that by linearity of expectation, it suffices to focus on the contribution to $\tilde{x}_v$ from a single user.

If that user's input is v, the expectation of the sum $Q := \sum_{u \in S_v} y_u$ is $e^\varepsilon p c_{set}$. On the other hand, if the input is not v, the expectation of the sum $\sum_{u \in S_v} y_u$ is $e^\varepsilon p c_{int} + p\,(c_{set} - c_{int})$. We want $\alpha \cdot \mathbb{E}\,[Q] + \beta = [[input\ is\ v]]$, where $[[T]]$ is defined to be 1 if T is true and 0 if false. Thus,

$$\alpha e^\varepsilon p c_{set} + \beta = 1,$$

$$\text{and } \alpha p\left(\left(e^{\varepsilon}-1\right)c_{int}+c_{set}\right)+\beta=0.$$

Substituting p, we get

$$\alpha e^{\varepsilon}\frac{c_{set}}{\left(e^{\varepsilon}-1\right)c_{set}+k}+\beta=1$$
$$\alpha\frac{\left(e^{\varepsilon}-1\right)c_{int}+c_{set}}{\left(e^{\varepsilon}-1\right)c_{set}+k}+\beta=0$$

Solving for α, β, we get

$$\alpha=\frac{\left(e^{\varepsilon}-1\right)c_{set}+k}{\left(e^{\varepsilon}-1\right)\left(c_{set}-c_{int}\right)};$$
$$\beta=-\frac{\left(e^{\varepsilon}-1\right)c_{set}+k}{\left(e^{\varepsilon}-1\right)\left(c_{set}-c_{int}\right)}\cdot\frac{\left(e^{\varepsilon}-1\right)c_{int}+c_{set}}{\left(e^{\varepsilon}-1\right)c_{set}+k}=-\frac{\left(e^{\varepsilon}-1\right)c_{int}+c_{set}}{\left(e^{\varepsilon}-1\right)\left(c_{set}-c_{int}\right)}.$$

We next analyze the variance, which suggests that q should be chosen close to $\exp(\varepsilon)+1$ for the best utility.

Lemma 3.2. $\mathbb{E}\left[\left\|x-\tilde{x}\right\|_2^2\right]\leqslant\frac{ne^{\varepsilon}c_{set}^2/c_{int}^2+n(k-1)\left(\left(e^{\varepsilon}-1\right)+c_{set}/c_{int}\right)^2}{\left(e^{\varepsilon}-1\right)^2\left(c_{set}/c_{int}-1\right)}$. *In particular, if* $c_{set}/c_{int}=e^{\varepsilon}+1$ *then* $\mathbb{E}\left[\frac{1}{k}\left\|x-\tilde{x}\right\|_2^2\right]\leqslant\frac{n}{k}+\frac{4ne^{\varepsilon}}{\left(e^{\varepsilon}-1\right)^2}$

Proof. By independence, we only need to analyze the variance when there is exactly one user with input v. The lemma then follows from adding up the variances from all users.

$$\begin{aligned}
\mathbb{E}\left[\left(\tilde{x}_v-1\right)^2\right]&=e^{\varepsilon}pc_{set}\left(\alpha+\beta-1\right)^2+p(k-c_{set})\left(\beta-1\right)^2\\
&=\frac{1-\beta}{\alpha}\left(\alpha+\beta-1\right)^2+\frac{\alpha+\beta-1}{\alpha}\left(1-\beta\right)^2\\
&=\left(\alpha+\beta-1\right)\left(1-\beta\right)\\
&=\frac{-c_{set}+k}{\left(e^{\varepsilon}-1\right)\left(c_{set}-c_{int}\right)}\cdot\frac{e^{\varepsilon}c_{set}}{\left(e^{\varepsilon}-1\right)\left(c_{set}-c_{int}\right)}\\
&=\frac{\left(-c_{set}/c_{int}+k/c_{int}\right)e^{\varepsilon}c_{set}/c_{int}}{\left(e^{\varepsilon}-1\right)^2\left(c_{set}/c_{int}-1\right)^2}\\
&\leqslant\frac{\left(-c_{set}/c_{int}+c_{set}^2/c_{int}^2\right)e^{\varepsilon}c_{set}/c_{int}}{\left(e^{\varepsilon}-1\right)^2\left(c_{set}/c_{int}-1\right)^2}\\
&=\frac{e^{\varepsilon}c_{set}^2/c_{int}^2}{\left(e^{\varepsilon}-1\right)^2\left(c_{set}/c_{int}-1\right)}
\end{aligned}$$

Let $z=c_{set}/c_{int}$. Note that $\frac{z^2}{z-1}$ is an increasing function for $z\in[2,+\infty)$ so this part of the variance gets larger as q gets larger.

Next we analyze the contribution to the variance from coordinates $u\neq v$.

$$\begin{aligned}
\mathbb{E}\left[\tilde{x}_u^2\right]&=\left(\left(e^{\varepsilon}-1\right)c_{int}+c_{set}\right)p\left(\alpha+\beta\right)^2+\left(1-e^{\varepsilon}pc_{int}-p\left(c_{set}-c_{int}\right)\right)\beta^2\\
&=-\frac{\beta}{\alpha}\left(\alpha+\beta\right)^2+\left(1+\frac{\beta}{\alpha}\right)\beta^2
\end{aligned}$$

$$
\begin{aligned}
&= \frac{-\beta(\alpha+\beta)^2 + (\alpha+\beta)\beta^2}{\alpha} \\
&= -\beta(\alpha+\beta) \\
&= \frac{(e^\varepsilon-1)c_{int}+c_{set}}{(e^\varepsilon-1)(c_{set}-c_{int})} \cdot \frac{(e^\varepsilon-2)c_{set}+k-(e^\varepsilon-1)c_{int}}{(e^\varepsilon-1)(c_{set}-c_{int})} \\
&= \frac{(e^\varepsilon-1)+c_{set}/c_{int}}{(e^\varepsilon-1)(c_{set}/c_{int}-1)} \cdot \frac{(e^\varepsilon-2)c_{set}/c_{int}+k/c_{int}-(e^\varepsilon-1)}{(e^\varepsilon-1)(c_{set}/c_{int}-1)} \\
&\leqslant \frac{((e^\varepsilon-1)+z)\left((e^\varepsilon-2)z+z^2-(e^\varepsilon-1)\right)}{(e^\varepsilon-1)^2(z-1)^2} \\
&= \frac{((e^\varepsilon-1)+z)^2}{(e^\varepsilon-1)^2(z-1)}
\end{aligned}
$$

Note that the function $\frac{((e^\varepsilon-1)+z)^2}{(e^\varepsilon-1)^2(z-1)}$ is decreasing for $z \in (0, e^\varepsilon+1]$ and it is increasing for $z \in [e^\varepsilon+1, +\infty)$ so this part of the variance is minimized when $z = e^\varepsilon+1$. For $z = e^\varepsilon+1$, we can substitute and get $\frac{4e^\varepsilon}{(e^\varepsilon-1)^2}$. □

Next we discuss the algorithms to compute $\tilde{x}_v$. The naive algorithm takes $O(kc_{set}) = O(k^2/q)$ time and this is the algorithm of choice for $t \leqslant 3$. For $t > 3$, we can use dynamic programming to obtain a faster algorithm. Note in the below that q should be chosen close to $\exp(\varepsilon)+1$.

Theorem 3.3. *In the ProjectiveGeometryResponse scheme, there exists an $O((q^t-1)/(q-1)tq)$ time algorithm for server reconstruction, using $O((q^t-1)/(q-1))$ memory. These bounds are at best $O(ktq)$ time and $O(k)$ memory, and increase by at most a factor of q each if rounding up to the next power of q is needed so that $(q^t-1)/(q-1) \geqslant k$.*

Proof. We use dynamic programming. For $a \in \mathbb{F}_q^j, b \in \mathbb{F}_q^{t-j}, z \in \mathbb{F}_q$, where a is further restricted to have its first nonzero entry be a 1 (it may also be the all-zeroes vector), and b is restricted to be a canonical vector when $j = 0$, define

$$
f(a,b,z) = \sum_{\substack{pref_j(u)=a \\ \langle \mathrm{suff}_{t-j}(u), b\rangle = z}} y_u,
$$

where $\mathrm{pref}_i(u)$ denotes the length-i prefix vector of u, and $\mathrm{suff}_i(u)$ denotes the length-i suffix vector of u. Then, we would like to compute

$$
\tilde{x}_v = \alpha\left(\sum_{u \in S_v} y_u\right) + \beta\sum_u y_u = \alpha \cdot f(\perp, v, 0) + \beta n,
$$

for all projective points v, where $\perp$ denotes the length-0 empty vector. We next observe that f satisfies a recurrence relation, so that we can compute the full array of values $(f(\perp,v,0))_{v \text{ is canonical}}$ efficiently using dynamic programming and then efficiently obtain $\tilde{x} \in \mathbb{R}^k$.

We now describe the recurrence relation. For $w \in \mathbb{F}_q$ and a vector v, let $v \circ w$ denote v with w appended as one extra entry. If j denotes the length of the vector a, then the base case is $j = t$. In

this case, $f(a, \perp, z) = y_a$ iff both $a \neq 0$ and $z = 0$; else, $f(a, \perp, z) = 0$. The recursive step is then when $0 \leqslant j < t$. Essentially, we have to sum over all ways to extend a by one more coordinate. Let $\text{suff}_{-1}(\text{b})$ denote the vector b but with the first entry removed (so it is a vector of length one shorter). There are two cases: a is the all-zeroes vector, versus it is not. In the former case, the recurrence is

$$f(0, b, z) = f(\vec{0} \circ 0, \text{suff}_{-1}(\text{b}), z) + f(\vec{0} \circ 1, \text{suff}_{-1}(\text{b}), z - b_1 \mod q).$$

Note we are not allowed to append $w \in \{2, 3, \ldots, q-1\}$ to a since that would not satisfy the requirement that the first argument to f either be all-zeroes or be canonical. The other case for the recurrence relation is when $a \neq 0$, in which case the recurrence relation becomes

$$f(a, b, z) = \sum_{w=0}^{q-1} f(a \circ w, \text{suff}_{-1}(b), z - d \cdot b_1 \mod q).$$

We now analyze the running time and memory requirements to obtain all $f(a, b, z)$ values via dynamic programming. The runtime is proportional to

$$kq + \sum_{a,b,z,j \neq 0} q.$$

This is because for $j > 0$, for each a, b, z triple we do at most q work. When $j = 0$, there is only one possible value for a (namely $\perp$) and $k = \frac{q^t - 1}{q-1}$ values for v, plus we are only concerned with $z = 0$ in this case. For larger j, the number of possibilities for a is $\frac{q^j - 1}{q-1} + 1$ (the additive 1 is since a can be the all-zeroes vector), whereas the number of possibilities for b is q^{t-j}. Thus the total runtime is proportional to

$$kq + \left(\sum_{j=1}^{t} \left(\frac{q^j - 1}{q-1} + 1 \right) \cdot q^{t-j} \right) \cdot q^2 = O(ktq^2).$$

For the memory requirement, note $f(\cdot)$ values for some fixed j only depend on the values for $j + 1$, and thus using bottom-up dynamic programming we can save a factor of t in the memory, for a total memory requirement of only $O(kq)$ (for any fixed j there are only $O(k)$ a, b pairs, and there are q values for z).

Finally, we add an optimization which improves both the runtime and memory by a factor of q. Specifically, suppose b is not canonical and is not the all-zeroes vector. Let the value of its first nonzero entry be ζ. Then $f(a, b, z)$ is equal to $f(a, b/\zeta, z/\zeta)$, where the division is over $\mathbb{F}_q$. Thus, we only need to compute $f(\cdot)$ for b either canonical or equal to the 0 vector. This reduces the number of b from q^{t-j} to $(q^{t-j} - 1)/(q-1) + 1$, which improves the runtime to $O(ktq)$ and the memory to $O(k)$. Note finite field division over $\mathbb{F}_q$ can be implemented in $O(1)$ time after preprocessing. First, factor $q - 1$ and generate all its divisors in $o(q)$ time, from which we can find a generator g of $\mathbb{F}_q^*$ in $o(q)$ expected time by rejection sampling (it is a generator iff $g^p \not\equiv 1 \mod q$ for every nontrivial divisor p of q, and we can compute $g^p \bmod q$ in $O(\log q)$ time via repeated squaring). Then, in $O(q)$ time create a lookup table $A[0 \ldots q-1]$ with $A[i] := g^i \mod q$. Then create an inverse lookup table by for each $0 \leqslant i < q$, setting the inverse of $A[i]$ to $A[q - 1 - i]$. $\square$

4 HybridProjectiveGeometryResponse: trading off error and time

In this section, we describe a hybrid scheme using an intermediate value for the field size q to trade off between the variance and the running time. Roughly speaking, larger values for q lead to slower running time but also smaller variance. The approach is similar to the way [ASZ19] extended their scheme from the high privacy regime to the general setting. We choose h, q, t such that they satisfy the following conditions:

- $b = \frac{q^t-1}{q-1}$ and $bh \geqslant k > c_{set}h$.
- Let $c_{set} = \frac{q^{t-1}-1}{q-1}$, $c_{int} = \frac{q^{t-2}-1}{q-1}$, and $z = c_{set}/c_{int}$. Note that $c_{set}^2 \geqslant b \cdot c_{int}$ and $q+1 \geqslant z \geqslant q$.
- Choose hz as close as possible to $e^\varepsilon + 1$.

The input coordinates are partitioned into blocks of size at most b each. The algorithm's response consists of two parts: the index of the block and the index inside the block. First, the algorithm uses the randomized response to report the block. Next, if the response has the correct block then the algorithm uses the scheme described in the previous section with field size q to describe the coordinate inside the block. If the first response has the wrong block then the algorithm uses a uniformly random response in the second part.

More precisely, the algorithm works as follows. Each input value is identified with a pair (i, v) where $i \in \mathbb{Z}_h$ and v is a canonical vector in $\mathbb{F}_q^t$. If $k < bh$ then we allocate up to $\lceil k/h \rceil$ input values to each block. The response is a pair (j, u) where $i \in \mathbb{Z}_h$ and u is a canonical vector in $\mathbb{F}_q^t$ chosen as follows. For $j = i$ and $\langle u, v \rangle = 0$, the pair (j, u) is chosen with probability $e^\varepsilon p$. All other choices are chosen with probability p each. Because all probabilities are either p or $e^\varepsilon p$, the scheme is ε-private. We have

$$e^\varepsilon p \cdot c_{set} + p\left(bh - c_{set}\right) = 1$$

$$p = \frac{1}{bh + \left(e^\varepsilon - 1\right) c_{set}}$$

Let $\tilde{x}_{i,v}$ be our estimate for the frequency of input (i, v). The estimates are computed as follows.

$$\tilde{x}_{i,v} = \alpha \left(\sum_{\langle v,u \rangle = 0} y_{i,u} \right) + \beta \left(\sum_u y_{i,u} \right) + \gamma \left(\sum_{j,u} y_{j,u} \right)$$

We need to choose α, β and γ so that $\tilde{x}_{i,v}$ is an unbiased estimator of $x_{i,v}$. By linearity of expectation, we only need to consider the case with exactly one user. If the input is i, v then we have

$$\mathbb{E}\left[\tilde{x}_{i,v}\right] = \alpha e^\varepsilon p c_{set} + \beta p\left(\left(e^\varepsilon - 1\right) c_{set} + b\right) + \gamma = 1$$

If the input is not i, v but in the same block then

$$\mathbb{E}\left[\tilde{x}_{i,v}\right] = \alpha p\left(\left(e^\varepsilon - 1\right) c_{int} + c_{set}\right) + \beta p\left(\left(e^\varepsilon - 1\right) c_{set} + b\right) + \gamma = 0$$

Finally if the input is in a different block then

$$\mathbb{E}\left[\tilde{x}_{i,v}\right] = \alpha p c_{set} + \beta p b + \gamma = 0$$

We solve for α, β, γ and get

$$\begin{aligned}
\alpha &= \frac{1}{p\left(e^{\varepsilon}-1\right)\left(c_{set}-c_{int}\right)} = \frac{bh+\left(e^{\varepsilon}-1\right)c_{set}}{\left(e^{\varepsilon}-1\right)\left(c_{set}-c_{int}\right)} \\
\beta &= -\frac{\alpha c_{int}}{c_{set}} = -\frac{c_{int}/c_{set}}{p\left(e^{\varepsilon}-1\right)\left(c_{set}-c_{int}\right)} \\
&= -\frac{bh+\left(e^{\varepsilon}-1\right)c_{set}}{\left(e^{\varepsilon}-1\right)\left(c_{set}-c_{int}\right)} \cdot \frac{c_{int}}{c_{set}} \\
\gamma &= -\alpha p c_{set} - \beta p b = -\frac{c_{set} - b \cdot \frac{c_{int}}{c_{set}}}{\left(e^{\varepsilon}-1\right)\left(c_{set}-c_{int}\right)} \leqslant 0
\end{aligned}$$

We note that $\alpha+\beta = \left(1-\frac{c_{int}}{c_{set}}\right)\alpha = \frac{bh/c_{set}+\left(e^{\varepsilon}-1\right)}{\left(e^{\varepsilon}-1\right)}$.

Lemma 4.1.

$$\begin{aligned}
\mathbb{E}\left[\|x-\tilde{x}\|_2^2\right] &\leqslant n\left(1+\frac{\left(zh+\left(e^{\varepsilon}-1\right)\right)}{\left(e^{\varepsilon}-1\right)^2\left(z-1\right)}+\frac{2}{\left(e^{\varepsilon}-1\right)}+\frac{e^{\varepsilon}\left(zh-e^{\varepsilon}+1\right)}{\left(e^{\varepsilon}-1\right)^2}\right) \\
&+ n\frac{\left(zh+\left(e^{\varepsilon}-1\right)\right)z}{\left(e^{\varepsilon}-1\right)^2\left(z-1\right)}\left(k-\lceil k/h\rceil+\left(\lceil k/h\rceil-1\right)\frac{\left(z+e^{\varepsilon}-1\right)}{z}\right)
\end{aligned}$$

In particular, if $zh = e^{\varepsilon}+1$ *then* $\mathbb{E}\left[\frac{1}{k}\|x-\tilde{x}\|_2^2\right] \leqslant \frac{n}{k}+\frac{z}{z-1}\cdot n\frac{4e^{\varepsilon}}{\left(e^{\varepsilon}-1\right)^2}$

Proof. By independence, we only need to analyze the variance when there is exactly one user with input (i,v) and response (j,u). The lemma follows from adding up the variances from all users.

$$\begin{aligned}
\mathbb{E}\left[\left(\tilde{x}_{i,v}-1\right)^2\right] \leqslant& \mathbb{E}\left[\left(\tilde{x}_{i,v}-1-\gamma\right)^2\right] \\
=& \mathbb{P}\left[j\neq i\right]\cdot(-1)^2+\mathbb{P}\left[j=i\wedge\langle u,v\rangle\neq 0\right]\left(\beta-1\right)^2+\mathbb{P}\left[j=i\wedge\langle u,v\rangle=0\right]\left(\alpha+\beta-1\right)^2 \\
=& \left(1-\left(e^{\varepsilon}-1\right)pc_{set}-pb\right)+p\left(b-c_{set}\right)\left(\beta-1\right)^2+e^{\varepsilon}pc_{set}\left(\alpha+\beta-1\right)^2 \\
=& 1+p\left(b-c_{set}\right)\left(\beta^2-2\beta\right)+e^{\varepsilon}pc_{set}\left(\alpha+\beta\right)\left(\alpha+\beta-2\right)
\end{aligned}$$

We expand the second and third terms individually:

$$\begin{aligned}
& p\left(b-c_{set}\right)\left(\beta^2-2\beta\right) \\
&= p\left(b-c_{set}\right)\left(\left(\frac{c_{int}/c_{set}}{p\left(e^{\varepsilon}-1\right)\left(c_{set}-c_{int}\right)}\right)^2+\frac{2c_{int}/c_{set}}{p\left(e^{\varepsilon}-1\right)\left(c_{set}-c_{int}\right)}\right) \\
&= \frac{\left(bh+\left(e^{\varepsilon}-1\right)c_{set}\right)\left(b-c_{set}\right)c_{int}^2/c_{set}^2}{\left(e^{\varepsilon}-1\right)^2\left(c_{set}-c_{int}\right)^2}+\frac{2\left(b-c_{set}\right)c_{int}/c_{set}}{\left(e^{\varepsilon}-1\right)\left(c_{set}-c_{int}\right)} \\
&= \frac{\left(bh/c_{set}+\left(e^{\varepsilon}-1\right)\right)\left(b/c_{set}-1\right)}{\left(e^{\varepsilon}-1\right)^2\left(c_{set}/c_{int}-1\right)^2}+\frac{2\left(b/c_{set}-1\right)}{\left(e^{\varepsilon}-1\right)\left(c_{set}/c_{int}-1\right)} \\
&\leqslant \frac{\left(zh+\left(e^{\varepsilon}-1\right)\right)}{\left(e^{\varepsilon}-1\right)^2\left(z-1\right)}+\frac{2}{\left(e^{\varepsilon}-1\right)}
\end{aligned}$$

and

$$
\begin{aligned}
& e^{\varepsilon} p c_{set} (\alpha+\beta)(\alpha+\beta-2) \\
&= \frac{e^{\varepsilon} c_{set}}{bh + (e^{\varepsilon}-1) c_{set}} \cdot \frac{bh/c_{set} + (e^{\varepsilon}-1)}{(e^{\varepsilon}-1)} \cdot \frac{bh/c_{set} - (e^{\varepsilon}-1)}{(e^{\varepsilon}-1)} \\
&= \frac{e^{\varepsilon}(bh/c_{set} - e^{\varepsilon} + 1)}{(e^{\varepsilon}-1)^2} \\
&\leqslant \frac{e^{\varepsilon}(zh - e^{\varepsilon} + 1)}{(e^{\varepsilon}-1)^2}
\end{aligned}
$$

When $zh = e^{\varepsilon} + 1$, we have

$$
\mathbb{E}\left[(\tilde{x}_{i,v} - 1)^2\right] \leqslant 1 + \frac{2e^{\varepsilon}}{(e^{\varepsilon}-1)^2 (z-1)} + \frac{2}{(e^{\varepsilon}-1)} + \frac{2e^{\varepsilon}}{(e^{\varepsilon}-1)^2} < 1 + \frac{4e^{\varepsilon}}{(e^{\varepsilon}-1)^2} \frac{z}{z-1}
$$

Next consider $v' \neq v$.

$$
\begin{aligned}
\mathbb{E}\left[(\tilde{x}_{i,v'} - 0)^2\right] &\leqslant \mathbb{E}\left[(\tilde{x}_{i,v'} - \gamma)^2\right] \\
&= \mathbb{P}\left[j \neq i\right] \cdot 0 + \mathbb{P}\left[j = i \wedge \langle u, v' \rangle \neq 0\right] (\beta)^2 + \mathbb{P}\left[j = i \wedge \langle u, v' \rangle = 0\right] (\alpha+\beta)^2 \\
&= p\left(b + (e^{\varepsilon}-2) c_{set} - (e^{\varepsilon}-1) c_{int}\right) (\beta)^2 + p\left((e^{\varepsilon}-1) c_{int} + c_{set}\right) (\alpha+\beta)^2 \\
&= \left(b + (e^{\varepsilon}-2) c_{set} - (e^{\varepsilon}-1) c_{int}\right) \frac{1}{p(e^{\varepsilon}-1)^2 (c_{set} - c_{int})^2} \frac{c_{int}^2}{c_{set}^2} + \\
&\quad p\left((e^{\varepsilon}-1) c_{int} + c_{set}\right) \frac{(1 - c_{int}/c_{set})^2}{p^2 (e^{\varepsilon}-1)^2 (c_{set} - c_{int})^2} \\
&= \left(b/c_{set} + (e^{\varepsilon}-2) - (e^{\varepsilon}-1) c_{int}/c_{set}\right) \frac{(bh/c_{set} + (e^{\varepsilon}-1))}{(e^{\varepsilon}-1)^2 (1 - c_{int}/c_{set})^2} \frac{c_{int}^2}{c_{set}^2} + \\
&\quad \left((e^{\varepsilon}-1) c_{int}/c_{set} + 1\right) \frac{(bh/c_{set} + (e^{\varepsilon}-1))}{(e^{\varepsilon}-1)^2} \\
&\leqslant \left(z + (e^{\varepsilon}-2) - (e^{\varepsilon}-1)/z\right) \frac{(zh + (e^{\varepsilon}-1))}{(e^{\varepsilon}-1)^2 (z-1)^2} + \left((e^{\varepsilon}-1)/z + 1\right) \frac{(zh + (e^{\varepsilon}-1))}{(e^{\varepsilon}-1)^2} \\
&= \left(z + e^{\varepsilon} - 1\right) \frac{(zh + (e^{\varepsilon}-1))}{(e^{\varepsilon}-1)^2 (z-1)}
\end{aligned}
$$

When $zh = e^{\varepsilon} + 1$, the last expression is bounded by $(z + e^{\varepsilon} - 1) \frac{2e^{\varepsilon}}{(e^{\varepsilon}-1)^2 (z-1) z} + (z + e^{\varepsilon} - 1) \frac{2e^{\varepsilon}}{(e^{\varepsilon}-1)^2 z} = \frac{(z+e^{\varepsilon}-1)}{(z-1)} \cdot \frac{2e^{\varepsilon}}{(e^{\varepsilon}-1)^2} \leqslant \frac{1+h}{z} \frac{z}{(z-1)} \cdot \frac{2e^{\varepsilon}}{(e^{\varepsilon}-1)^2}$

Finally, consider $i' \neq i$ and arbitrary v'.

$$
\begin{aligned}
\mathbb{E}\left[(\tilde{x}_{i',v'} - 0)^2\right] &\leqslant \mathbb{E}\left[(\tilde{x}_{i',v'} - \gamma)^2\right] \\
&= \mathbb{P}\left[j \neq i'\right] \cdot 0^2 + \mathbb{P}\left[j = i' \wedge \langle u, v' \rangle \neq 0\right] (\beta)^2 + \mathbb{P}\left[j = i' \wedge \langle u, v' \rangle = 0\right] (\alpha+\beta)^2 \\
&= p\left(b - c_{set}\right) (\beta)^2 + p c_{set} (\alpha+\beta)^2
\end{aligned}
$$

$$
\begin{aligned}
&= p\left(b - c_{set}\right) \frac{1}{p^2\left(e^\varepsilon - 1\right)^2\left(c_{set} - c_{int}\right)^2} \frac{c_{int}^2}{c_{set}^2} + pc_{set} \frac{\left(1 - c_{int}/c_{set}\right)^2}{p^2\left(e^\varepsilon - 1\right)^2\left(c_{set} - c_{int}\right)^2} \\
&= \left(b/c_{set} - 1\right) \frac{\left(bh/c_{set} + \left(e^\varepsilon - 1\right)\right)}{\left(e^\varepsilon - 1\right)^2\left(1 - c_{int}/c_{set}\right)^2} \frac{c_{int}^2}{c_{set}^2} + \frac{bh/c_{set} + \left(e^\varepsilon - 1\right)}{\left(e^\varepsilon - 1\right)^2} \\
&\leqslant \frac{\left(zh + \left(e^\varepsilon - 1\right)\right)}{\left(e^\varepsilon - 1\right)^2\left(z - 1\right)} + \frac{zh + \left(e^\varepsilon - 1\right)}{\left(e^\varepsilon - 1\right)^2} \\
&= \frac{\left(zh + \left(e^\varepsilon - 1\right)\right) z}{\left(e^\varepsilon - 1\right)^2\left(z - 1\right)}
\end{aligned}
$$

When $zh = e^\varepsilon + 1$, the last expression is bounded by $\frac{2e^\varepsilon}{\left(e^\varepsilon-1\right)^2}\frac{1}{z-1} + \frac{2e^\varepsilon}{\left(e^\varepsilon-1\right)^2} = \frac{2e^\varepsilon}{\left(e^\varepsilon-1\right)^2}\frac{z}{z-1}$

There are $b_i \leqslant \lceil k/h \rceil \leqslant b$ valid coordinates in the same block with the input (i, v). There are $k - b_i$ coordinates in the other blocks. Thus the total variance across all coordinates except for coordinate (i, v) is bounded by

$$
\begin{aligned}
&\frac{\left(zh + \left(e^\varepsilon - 1\right)\right) z}{\left(e^\varepsilon - 1\right)^2\left(z - 1\right)} \left(k - b_i + \left(b_i - 1\right)\frac{\left(z + e^\varepsilon - 1\right)}{z}\right) \\
\leqslant &\frac{\left(zh + \left(e^\varepsilon - 1\right)\right) z}{\left(e^\varepsilon - 1\right)^2\left(z - 1\right)} \left(k - \lceil k/h \rceil + \left(\lceil k/h \rceil - 1\right)\frac{\left(z + e^\varepsilon - 1\right)}{z}\right)
\end{aligned}
$$

For $zh = e^\varepsilon + 1$, we have $\frac{\left(z+e^\varepsilon-1\right)}{z} \leqslant 1 + h$ and $k - b_i + \left(b_i - 1\right)\frac{\left(z+e^\varepsilon-1\right)}{z} \leqslant k - \lceil k/h \rceil + \left(\lceil k/h \rceil - 1\right)(1 + h) = k - \lceil k/h \rceil + \lceil k/h \rceil - 1 + h(\lceil k/h \rceil - 1) < 2k$. □

Regarding the decoding algorithms, notice that the estimates are computed separately by blocks except for an offset γ scaled by the total number of received messages across all blocks. Thus, using the naive algorithm, the time to estimate one count is $O\left(c_{set}\right) = O\left(q^{\lceil \log_q(k/h) \rceil - 1}\right)$. Using the fast algorithm to estimate all counts takes $O\left(bqt\right)$ time per block and in total, $O\left(bqth\right) = O\left(\left\lceil \log_q\left(k/h\right) \right\rceil hq^{1+\lceil \log_q(k/h) \rceil}\right)$ time.

5 Experimental Results

In this section, we compare previously-known algorithms (RAPPOR, PI-RAPPOR, HadamardResponse (HR), RecursiveHadamardResponse (RHR), SubsetSelection (SS)) and our new algorithms ProjectiveGeometryResponse (PGR) and HybridProjectiveGeometryResponse (HPGR). As the variance upper bound of these algorithms do not depend on the underlying data, we perform our experiments on simple synthetic data that realize the worst case for variance. Our experiments show that ProjectiveGeometryOracle matches the best of these algorithms namely SS, RAPPOR, and PI-RAPPOR, and achieves noticeably better MSE than other communication- and computation-efficient approaches. At the same time it is significantly more efficient than those three in terms of server computation time, while also achieving optimal communication.

All experiments were run on a Dell Precision T3600 with six Intel 3.2 GHz Xeon E5-1650 cores running Ubuntu 20.04 LTS, though our implementation did not take advantage of parallelism. We implemented all algorithms and ran experiments in C++, using the GNU C++ compiler version

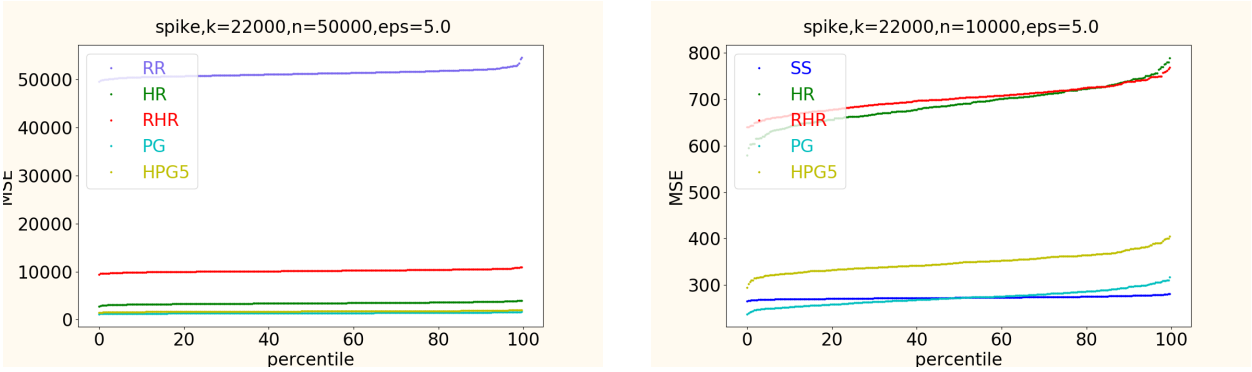
scheme name	runtime (in seconds)
PI-RAPPOR	1,893.82 (approximately 31.5 minutes)
PGR	36.92
HPGR	5.94
RHR	1.20
HR	0.64
RR	0.02

Table 2: Server runtimes for $\varepsilon = 5$, $k =$ 3,307,948. For HPGR, we chose the parameters $h = 50, q = 3, t = 11$, so that the mechanism rounded up the universe size to $h(q^t - 1)/(q - 1)$, which is about 34% larger than k.

9.3.0; code and details on how to run the experiments used to generate data and plots are in our public repository at `https://github.com/minilek/private_frequency_oracles/`.

We first performed one experiment to show the big gap in running times. We took $\varepsilon = 5$, a practically relevant setting, and $n =$ 10,000, $k =$ 3,307,948; this setting of n is smaller than one would see in practice, but the runtimes of the algorithms considered are all linear in n plus additional terms that depend on k, ε, and our aim was to measure the impact of these additive terms, which can be significant even for large n. Furthermore, in practice the server can immediately process messages from each of the n users dynamically as the messages arrive asynchronously, whereas it is the additive terms that must be paid at once at the time of histogram reconstruction. For our settings, the closest prime to $\exp(\varepsilon) + 1 \approx 149.4$ is $q = 151$. Recall that PGR rounds up to universe sizes of the form $(q^t - 1)/(q - 1)$; then $(q^4 - 1)/(q - 1)$ is less than 5% larger than k, so that the negative effect of such rounding on the runtime of PGR is minimal. Meanwhile PI-RAPPOR picks the largest prime q' smaller than $\exp(\varepsilon) + 1$, which in this case is $q' = 149$, and assumes universe sizes of the form $q'^t - 1$; in this case $q'^3 - 1 = k$ exactly, so rounding issues do not negatively impact the running time of PI-RAPPOR (we chose this particular value of k intentionally, to show PI-RAPPOR's performance in the best light for some fairly large universe size). The runtimes of various algorithms with this setting of ε, k, n are shown in Table 2. Note RHR and HR sacrifice a constant factor in utility compared to PI-RAPPOR and PGR, the former of which is four orders of magnitude slower while the latter is only one order of magnitude slower and approximately 51x faster than PI-RAPPOR. Meanwhile, HPGR's runtime is of the same order of magnitude (though roughly 5x slower) than RHR, but as we will see shortly, HPGR can provide significantly improved utility over RHR and HR.

Next we discuss error. Many of our experiments showing reconstruction error with fixed ε take $\varepsilon = 5$, a practically relevant setting, and universe size $k =$ 22,000, for which the closest prime to $\exp(\varepsilon) + 1 \approx 149.4$ is $q = 151$. Recall that PGR rounds up to universe sizes of the form $(q^t - 1)/(q - 1)$; then $(q^3 - 1)/(q - 1) =$ 22,593 is not much larger than k, so that the runtime of PGR is not severely impacted. Also, c_{set}/c_{int} as defined in Section 3 is very close to $\exp(\varepsilon) + 1$, so that the MSE bound in Lemma 3.2 nearly matches that of SS. Furthermore for HPGR for this setting of ε, k, if we choose $q = 5, h = 30, t = 5$, then $h \cdot (q^t - 1)/(q - 1) =$ 23,430, which is not much bigger than k so that the runtime of HPGR is not majorly impacted. Furthermore hz as defined in Section 4 is approximately 150.19, which is very close to $\exp(\varepsilon) + 1$ as recommended



(a) (b)

Figure 1: RandomizedResponse has significantly worse error than other algorithms, even for moderately large universes, followed by HadamardResponse and RecursiveHadamardResponse, which have roughly double the error of state-of-the-art algorithms. HybridProjectiveGeometryResponse trades off having slightly worse error than state-of-the-art for faster runtime.

by Lemma 4.1 to obtain minimal error. We first draw attention to Figs. 2a and 2b. These plots run RAPPOR, PI-RAPPOR, PGR, and SS with k, n, ε as in the figure and show that their error distributions are essentially equivalent. We show the plots for only one some particular parameter settings, but the picture has looked essentially the same to us regardless of which parameters we have tried. In Fig. 2a, we have n users each holding the same item in the universe (item 0); we call this a *spike* distribution as noted in the plot. We have each user apply its local randomizer to send a message to the server, and we ask the server to then reconstruct the histogram (which should be $(n, 0, \ldots, 0)$) and calculate the MSE. We repeat this experiment 300 times, and in this plot we have 300 dots plotted per algorithm, where a dot at point (x, y) signifies that the MSE was at most y for x% of the trial runs; this, it is a plot of the CDF of the empirical error distribution. In Fig. 2b, we plot MSE as a function of increasing ε, where for each value of ε we repeat the above experiment 10 times then plot the average MSE across those 10 trial runs. Because the error performance of RAPPOR, PI-RAPPOR, SS, and PGR are so similar, in all other plots we do not include RAPPOR and PI-RAPPOR since their runtimes are so slow that doing extensive experiments is very time-consuming computationally (note: our implementation of RAPPOR requires $O(nk)$ server time, though $O(n(k/e^{\varepsilon} + 1))$ expected time is possible by having each user transmit only a sparse encoding of the locations of the 1 bits in its message). We finally draw attention to Figs. 2c to 2h. Here we run several algorithms where the distribution over the universe amongst the users is Zipfian (a power law), with power law exponent either 0.1 (an almost flat distribution), or 3.0 (rapid decay). The HPGR algorithm was run with $q = 5$. As can be seen, the qualitative behavior and relative ordering of all the algorithms is essentially unchanged by the Zipf parameter: PGR,SS always have the best error, followed by HPGR, followed by RHR and HR. Figs. 2c and 2d show the CDF of the empirical MSE over 300 independent trials, as discussed above. Figs. 2e and 2f is similar, but the y-axis denotes $\|x - \tilde{x}\|_\infty$ instead of the MSE. Figs. 2g and 2h shows how the MSE varies as ε is increased; in these last plots we do not include HPGR as one essentially one should select a different q for each ε carefully to obtain a good tradeoff between runtime and error (as specified by Lemma 4.1) due to round-up issues in powering q.

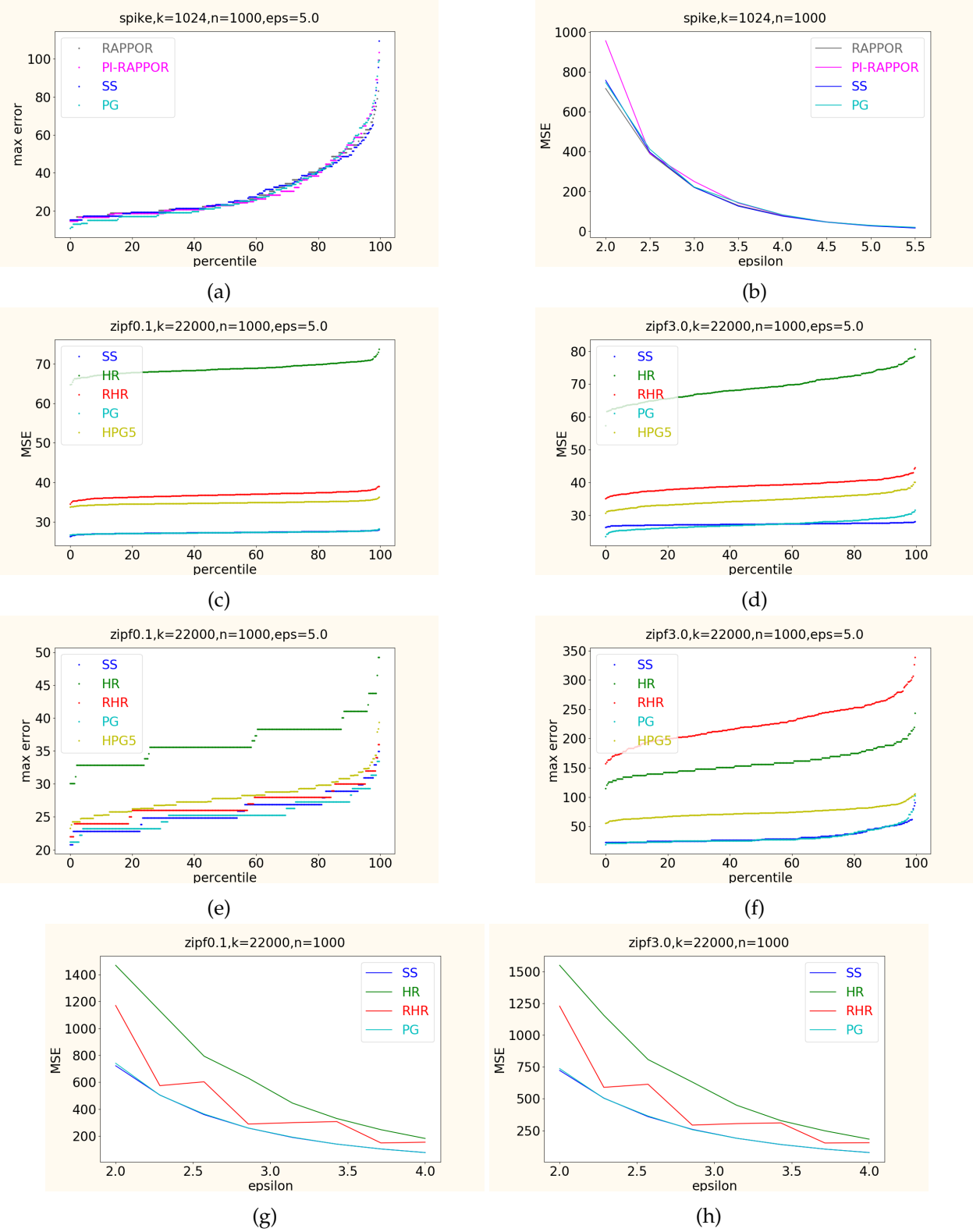


Figure 2: Error distributions from experiments.

Acknowledgments

We thank Noga Alon for pointing out the relevance of projective geometry for constructing the type of set system our mechanism relies on.

References

[ASZ19] Jayadev Acharya, Ziteng Sun, and Huanyu Zhang. Hadamard response: Estimating distributions privately, efficiently, and with little communication. In Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS), pages 1120–1129, 2019.

[BBGN19] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. In Alexandra Boldyreva and Daniele Micciancio, editors, Advances in Cryptology – CRYPTO 2019, pages 638–667, Cham, 2019. Springer International Publishing.

[BEM+17] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In Proceedings of the 26th Symposium on Operating Systems Principles, SOSP '17, pages 441–459, 2017.

[BHO20] Leighton Pate Barnes, Yanjun Han, and Ayfer Özgür. Lower bounds for learning distributions under communication constraints via Fisher information. Journal of Machine Learning Research, 21(236):1–30, 2020.

[BNS19] Mark Bun, Jelani Nelson, and Uri Stemmer. Heavy hitters and the structure of local privacy. ACM Transactions on Algorithms (TALG), 15(4):1–40, 2019.

[BNST20] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Thakurta. Practical locally private heavy hitters. Journal of Machine Learning Research, 21(16):1–42, 2020.

[BS15] Raef Bassily and Adam Smith. Local, private, efficient protocols for succinct histograms. In Proceedings of the forty-seventh annual ACM symposium on Theory of computing, pages 127–135, 2015.

[CKÖ20] Wei-Ning Chen, Peter Kairouz, and Ayfer Özgür. Breaking the communication-privacy-accuracy trilemma. In Proceedings of the 33rd Annual Conference on Advances in Neural Information Processing Systems (NeurIPS), 2020.

[CSU+19] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In Yuval Ishai and Vincent Rijmen, editors, Advances in Cryptology – EUROCRYPT 2019, pages 375–403, Cham, 2019. Springer International Publishing.

[EFM+19] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In Proceedings of the Thirtieth Annual ACM-SIAM

Symposium on Discrete Algorithms, SODA '19, page 2468–2479, USA, 2019. Society for Industrial and Applied Mathematics.

[EPK14] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: randomized aggregatable privacy-preserving ordinal response. In ACM SIGSAC Conference on Computer and Communications Security, pages 1054–1067, 2014.

[FMT21] Vitaly Feldman, Audra McMillan, and Kunal Talwar. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. In Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2021. arXiv:2012.12803 [cs.LG].

[FT21] Vitaly Feldman and Kunal Talwar. Lossless compression of efficient private local randomizers. In Proceedings of the 38th Annual Conference on International Conference on Machine Learning (ICML), pages 3208–3219, 2021.

[HKR12] Justin Hsu, Sanjeev Khanna, and Aaron Roth. Distributed private heavy hitters. In International Colloquium on Automata, Languages, and Programming, pages 461–472. Springer, 2012.

[KBR16] Peter Kairouz, Keith Bonawitz, and Daniel Ramage. Discrete distribution estimation under local privacy. arXiv preprint arXiv:1602.07387, 2016.

[SCB+21] Abhin Shah, Wei-Ning Chen, Johannes Balle, Peter Kairouz, and Lucas Theis. Optimal compression of locally differentially private mechanisms. arXiv preprint arXiv:2111.00092, 2021.

[TVV+17] Abhradeep Guha Thakurta, Andrew H. Vyrros, Umesh S. Vaishampayan, Gaurav Kapoor, Julien Freudiger, Vivek Rangarajan Sridhar, and Doug Davidson. Learning new words, 2017. US Patent 9,594,741.

[War65] Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. Journal of the American Statistical Association, 60(309):63–69, 1965.

[WBLJ17] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. Locally differentially private protocols for frequency estimation. In 26th USENIX Security Symposium (USENIX Security 17), pages 729–745, Vancouver, BC, August 2017. USENIX Association.

[WHN+19] Shaowei Wang, Liusheng Huang, Yiwen Nie, Xinyuan Zhang, Pengzhan Wang, Hongli Xu, and Wei Yang. Local differential private data aggregation for discrete distribution estimation. IEEE Trans. Parallel Distributed Syst., 30(9):2046–2059, 2019.

[YB17] Min Ye and Alexander Barg. Optimal schemes for discrete distribution estimation under local differential privacy. In Proceedings of the 14th Annual IEEE International Symposium on Information Theory (ISIT), pages 759–763, 2017.

A Fast dynamic programming for PI-RAPPOR

In this section, we describe an adaptation of our dynamic programming approach to PI-RAPPOR. First, we briefly review the construction of PI-RAPPOR. We use $\mathbb{F}_q$ with the field size q close to $e^\varepsilon + 1$. Let t be the minimum integer such that $k \leqslant q^t$.

We identify the k input values with vectors in $\mathbb{F}_q^t$. Let $x \in \mathbb{Z}^{q^t}$ denote the input frequency vector i.e. x_v is the number of users with input $v \in \mathbb{F}_q^t$. For each input v, we define a set $S(v) \subset \mathbb{F}_q^t \times \mathbb{F}_q$ where $(a, b) \in S(v)$ if and only if $\langle a, v\rangle + b = 0$.

Each user with input v sends a random element e of $\mathbb{F}_q^t \times \mathbb{F}_q$ with probability $e^\varepsilon p$ if $e \in S(v)$ and probability p if $e \notin S(v)$. Thus, $p = \frac{1}{e^\varepsilon q^t + (q-1)q^t}$. The server keeps the counts on the received elements in a vector y indexed by elements of $\mathbb{F}_q^t \times \mathbb{F}_q$. The total storage is $O\left(q^{t+1}\right)$. We estimate the frequency vector x by computing

$$\tilde{x}_v = \alpha \left(\sum_{u,w:\langle u,v\rangle + w = 0} y_{u,w} \right) + \beta \sum_{u,w} y_{u,w}$$

where α and β are chosen so that this is an unbiased estimator. This condition implies two equations:

$$\alpha \frac{e^\varepsilon q^t}{e^\varepsilon q^t + (q-1)q^t} + \beta = 1$$
$$\alpha \frac{e^\varepsilon q^{t-1} + (q-1)q^{t-1}}{e^\varepsilon q^t + (q-1)q^t} + \beta = 0$$

We obtain

$$\alpha = \frac{e^\varepsilon q + (q-1)q}{(e^\varepsilon - 1)(q-1)}$$
$$\beta = -\frac{e^\varepsilon + (q-1)}{(e^\varepsilon - 1)(q-1)}$$

Next, we describe a fast algorithm to compute $\tilde{x}$ with running time $O\left(tq^{t+2}\right)$. Specifically, for $a \in \mathbb{F}_q^j, b \in \mathbb{F}_q^{t-j}, z \in \mathbb{F}_q$, define

$$f_j(a, b, z) = \sum_{\substack{pref_j(u) = a \\ \langle \mathrm{suff}_{t-j}(u), b\rangle + w = z}} y_{u,w},$$

where $\mathrm{pref}_i(u)$ denotes the length-i prefix vector of u, and $\mathrm{suff}_i(u)$ denotes the length-i suffix vector of u. Then, we would like to compute

$$\tilde{x}_v = \alpha \left(\sum_{u,w:\langle u,v\rangle + w = 0} y_{u,w} \right) + \beta \sum_{u,w} y_{u,w} = \alpha \sum_w f_0(\perp, v, 0) + \beta n,$$

for all $v \in \mathbb{F}_q^t$, where $\perp$ denotes the length-0 empty vector. We next observe that f satisfies a recurrence relation, so that we can compute the full array of values $f_0(\perp, v, w)$ efficiently using dynamic programming and then efficiently obtain $\tilde{x} \in \mathbb{R}^k$. We have

$$
\begin{aligned}
f_j(a,b,z) &= \sum_{\substack{pref_j(u)=a \\ \langle \mathrm{suff}_{t-j}(u),b\rangle + w = z}} y_{u,w} \\
&= \sum_{i=0}^{q-1} \sum_{\substack{pref_{j+1}(u)=a\circ i \\ \langle \mathrm{suff}_{t-j-1}(u),\mathrm{suff}_{t-j-1}(b)\rangle + w = z - i\cdot b_1 \pmod q}} y_{u,w} \\
&= \sum_{i=0}^{q-1} f_{j+1}(a\circ i, \mathrm{suff}_{t-j-1}(b), (z - i\cdot b_1) \bmod q)
\end{aligned}
$$

Note that we have the base cases $f_t(a,\perp,w) = y_{a,w}$. We need to compute the values of $f_j(a,b,z)$ for $j \in \{0,1\ldots,t-1\}, a \in \mathbb{F}_q^j, b \in \mathbb{F}_q^{t-j}, z \in \mathbb{F}_q$ and each value takes $O(q)$ time so the total running time is $O(tq^{t+2})$.

B The public coin setting

We show that versions of PGR and HPGR can be implemented in the public coin setting in a way that the communication is $\lceil \log_2 q \rceil = \varepsilon \log_2 e + O(1)$ bits, which is asymptotically optimal to achieve asymptotically optimal utility loss [BHO20, Corollary 7]. We begin with PGR.

Recall that as described, PGR associates each of the k input values with a canonical vector in $\mathbb{F}_q^t$. In the public coin variant we now describe, we further assume that the canonical vectors have a non-zero last coordinate. This can be ensured by picking q, t such that $k \leqslant 1 + (1-1/q)((q^t - 1)/(q-1) - 1) = q^{t-1}$. We will use $C_{q,t}$ to denote the set of canonical vectors in $\mathbb{F}_q^t$ and $C_{q,t}^*$ to denote those with a non-zero last coordinate.

With this setup, recall that each output in the set S_v can be associated with a vector $u \in C_{q,t}$ such that $\langle u, v\rangle = 0$. Thus a user with input v sends a vector $u \in C_{q,t}$ with probability $e^\varepsilon p$ if $\langle u, v\rangle = 0$ and with probability p otherwise. For a vector u, let $pref_{t-1}(u)$ denote its length $(t-1)$ prefix. Note that for a vector $u \in C_{q,t}$, either $pref_{t-1}(u)$ is itself a canonical vector in $C_{q,t-1}$, or $u = u^* \stackrel{def}{=} (0,\ldots,0,1)$. Also note that for any $v \in C_{q,t}^*$, $u^* \neq S_v$.

This then suggests the following algorithm. We use public randomness to select a vector $w \in \mathbb{F}_q^{t-1}$ such that $w = (0,\ldots,0)$ with probability p, and w is a random vector in $C_{q,t-1}$ otherwise. Thus there are $1 + \frac{q^{t-1}-1}{q-1}$ possible values of w. Given a $w \in C_{q,t-1}$ and a $v \in C_{q,t}^*$, there is a unique $a \in \mathbb{F}_q$ such that $\langle v, w\cdot a\rangle = 0 \mod q$. When $w \neq (0,\ldots,0)$, a user with input $v \in C_{q,t}^*$ sends message a with probability $\frac{e^\varepsilon}{e^\varepsilon+q-1}$ if $\langle v, w\cdot a\rangle = 0 \mod q$, and with probability $\frac{1}{e^\varepsilon+q-1}$ otherwise. If $w = (0,\ldots,0)$, the user always send 1.

The server given w derived from the shared public randomness, and the message $a \in \mathbb{F}_q$, decodes it as

$$Dec(w,a) = w\cdot a.$$

We claim that the distribution of $Dec(w,a)$ is identical to the output in the private coin PGR. First observe that by construction, $Dec(w,a) \in C_{q,t}$. Next notice that for any $u, u' \in S(v)$, we have

$$\mathbb{P}(Dec(w,a) = u) = \mathbb{P}(w = pref_{t-1}(u)) \cdot \mathbb{P}(a = u_t \mid w = pref_{t-1}(u))$$

$$
\begin{aligned}
&= \mathbb{P}(w = pref_{t-1}(u)) \cdot \frac{e^\varepsilon}{e^\varepsilon + q - 1} \\
&= \mathbb{P}(w = pref_{t-1}(u')) \cdot \frac{e^\varepsilon}{e^\varepsilon + q - 1} \qquad \text{(by uniformity of w over canonical vectors)} \\
&= \mathbb{P}(w = pref_{t-1}(u')) \cdot \mathbb{P}(a = u'_t \mid w = pref_{t-1}(u')) \\
&= \mathbb{P}(Dec(w, a) = u').
\end{aligned}
$$

Similarly, for any $u, u' \in C_{q,t} \setminus S_v$ such that $u, u' \neq u^*$, we can write

$$
\begin{aligned}
\mathbb{P}(Dec(w, a) = u) &= \mathbb{P}(w = pref_{t-1}(u)) \cdot \mathbb{P}(a = u_t \mid w = pref_{t-1}(u)) \\
&= \mathbb{P}(w = pref_{t-1}(u)) \cdot \frac{1}{e^\varepsilon + q - 1} \\
&= \mathbb{P}(w = pref_{t-1}(u')) \cdot \frac{1}{e^\varepsilon + q - 1} \qquad \text{(by uniformity of w over canonical vectors)} \\
&= \mathbb{P}(w = pref_{t-1}(u')) \cdot \mathbb{P}(a = u'_t \mid w = pref_{t-1}(u')) \\
&= \mathbb{P}(Dec(w, a) = u').
\end{aligned}
$$

Further, an identical calculation shows that for $u \in S_v, u' \in C_{q,t} \setminus S_v$ with $u' \neq u^*$, $\mathbb{P}[Dec(w, a) = u] = e^\varepsilon \cdot \mathbb{P}(Dec(w, a) = u')$. Moreover, the distribution of w ensures that $\mathbb{P}(Dec(w, a) = u^*) = p$. It follows that for all $u \in C_{q,t}$, $\mathbb{P}(Dec(w, a) = u)$ is $e^\varepsilon p$ if $u \in S_v$ and p if $u \in C_{q,t} \setminus S_v$.

In other words, we have shown how to simulate the output distribution of PGR in the public coin setting while sending only a single element from $\mathbb{F}_q$.

An implementation of HPGR in the public coin model is similar. A message in HPGR is a pair (j, u) where $j \in \{1, \ldots, h\}$ is the index of a block, and $u \in \mathbb{F}_q^t$ is the name of a canonical vector, and as above in the public coin setting we will forbid u from being the all-zeroes vector (so that now we need $hq^{t-1} \geqslant k$). As described in Section 4, h, q are chosen so that $hq \approx e^\varepsilon + 1$. In the public coin model, the user selects j using private randomness and sends it explicitly then uses the PGR public coin protocol described above to determine the first $t - 1$ entries of u with no communication required, then sends the final entry of u to obey the HPGR distribution. The total communication is $\lceil hq \rceil = \varepsilon \log_2 e + O(1)$ bits.