Locally Differentially Private Protocols for Frequency Estimation

Tianhao Wang, Jeremiah Blocki, and Ninghui Li, ***Purdue University;***
Somesh Jha, ***University of Wisconsin Madison***

https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-tianhao

This paper is included in the Proceedings of the 26th USENIX Security Symposium

August 16–18, 2017 • Vancouver, BC, Canada

ISBN 978-1-931971-40-9

Open access to the Proceedings of the 26th USENIX Security Symposium is sponsored by USENIX

Locally Differentially Private Protocols for Frequency Estimation

Tianhao Wang, Jeremiah Blocki, Ninghui Li
Purdue University

Somesh Jha
University of Wisconsin-Madison

Abstract

Protocols satisfying Local Differential Privacy (LDP) enable parties to collect aggregate information about a population while protecting each user's privacy, without relying on a trusted third party. LDP protocols (such as Google's RAPPOR) have been deployed in real-world scenarios. In these protocols, a user encodes his private information and perturbs the encoded value locally before sending it to an aggregator, who combines values that users contribute to infer statistics about the population. In this paper, we introduce a framework that generalizes several LDP protocols proposed in the literature. Our framework yields a simple and fast aggregation algorithm, whose accuracy can be precisely analyzed. Our in-depth analysis enables us to choose optimal parameters, resulting in two new protocols (i.e., Optimized Unary Encoding and Optimized Local Hashing) that provide better utility than protocols previously proposed. We present precise conditions for when each proposed protocol should be used, and perform experiments that demonstrate the advantage of our proposed protocols.

1 Introduction

Differential privacy [10, 11] has been increasingly accepted as the *de facto* standard for data privacy in the research community. While many differentially private algorithms have been developed for data publishing and analysis [12, 19], there have been few deployments of such techniques. Recently, techniques for satisfying differential privacy (DP) in the local setting, which we call LDP, have been deployed. Such techniques enable gathering of statistics while preserving privacy of every user, without relying on trust in a single data curator. For example, researchers from Google developed RAPPOR [13, 16], which is included as part of Chrome. It enables Google to collect users' answers to questions such as the default homepage of the browser, the default search engine, and so on, to understand the unwanted or malicious hijacking of user settings. Apple [1] also uses similar methods to help with predictions of spelling and other things, but the details of the algorithm are not public yet. Samsung proposed a similar system [21] which enables collection of not only categorical answers (e.g., screen resolution) but also numerical answers (e.g., time of usage, battery volume), although it is not clear whether this has been deployed by Samsung.

A basic goal in the LDP setting is frequency estimation. A protocol for doing this can be broken down into following steps: For each question, each user **encodes** his or her answer (called input) into a specific format, **randomizes** the encoded value to get an output, and then sends the output to the aggregator, who then **aggregates** and decodes the reported values to obtain, for each value of interest, an estimate of how many users have that value. With improvement on the basic task of frequency estimation, solutions to more complex problems that rely on it, such as heavy hitter identification, frequent itemset mining, can also be improved.

We introduce a framework for what we call "pure" LDP protocols, which has a nice symmetric property. We introduce a simple, generic aggregation and decoding technique that works for all pure LDP protocols, and prove that this technique results in an unbiased estimate. We also present a formula for the variance of the estimate. Most existing protocols fit our proposed framework. The framework also enables us to precisely analyze and compare the accuracy of different protocols, and generalize and optimize them. For example, we show that the Basic RAPPOR protocol [13], which essentially uses unary encoding of input, chooses sub-optimal parameters for the randomization step. Optimizing the parameters results in what we call the Optimized Unary Encoding (OUE) protocol, which has significantly better accuracy.

Protocols based on unary encoding require $\Theta(d)$ com-

munication cost, where d is the number of possible input values, and can be very large (or even unbounded) for some applications. The RAPPOR protocol uses a Bloom filter encoding to reduce the communication cost; however, this comes with a cost of decreasing accuracy as well as increasing computation cost for aggregation and decoding. The random matrix projection-based approach introduced in [6] has $\Theta(\log n)$ communication cost (where n is the number of users); however, its accuracy is unsatisfactory. We observe that in our framework this protocol can be interpreted as binary local hashing. Generalizing this and optimizing the parameters results in a new Optimized Local Hashing (OLH) protocol, which provides much better accuracy while still requiring $\Theta(\log n)$ communication cost. The variance of OLH is orders of magnitude smaller than the previous methods, for ε values used in RAPPOR's implementation. Interestingly, OLH has the same error variance as OUE; thus it reduces communication cost at no cost of utility.

With LDP, it is possible to collect data that was inaccessible because of privacy issues. Moreover, the increased amount of data will significantly improve the performance of some learning tasks. Understanding customer statistics help cloud server and software platform operators to better understand the needs of populations and offer more effective and reliable services. Such privacy-preserving crowd-sourced statistics are also useful for providing better security while maintaining a level of privacy. For example, in [13], it is demonstrated that such techniques can be applied to collecting windows process names and Chrome homepages to discover malware processes and unexpected default homepages (which could be malicious).

Our paper makes the following contributions:

- We introduce a framework for "pure" LDP protocols, and develop a simple, generic aggregation and decoding technique that works for all such protocols. This framework enables us to analyze, generalize, and optimize different LDP protocols.
- We introduce the Optimized Local Hashing (OLH) protocol, which has low communication cost and provides much better accuracy than existing protocols. For $\varepsilon \approx 4$, which was used in the RAPPOR implementation, the variance of OLH's estimation is $1/2$ that of RAPPOR, and close to $1/14$ that of Random Matrix Projection [6]. Systems using LDP as a primitive could benefit significantly by adopting improved LDP protocols like OLH.

Roadmap. In Section 2, we describe existing protocols from [13, 6]. We then present our framework for pure LDP protocols in Section 3, apply it to study LDP protocols in Section 4, and compare different LDP protocols in Section 5. We show experimental results in Section 6. We review related work in Section 7, discuss in Section 8, and conclude in Section 9.

2 Background and Existing Protocols

The notion of differential privacy was originally introduced for the setting where there is a **trusted data curator**, who gathers data from individual users, processes the data in a way that satisfies DP, and then publishes the results. Intuitively, the DP notion requires that any single element in a dataset has only a limited impact on the output.

Definition 1 (Differential Privacy) *An algorithm* $\mathbf{A}$ *satisfies* ε*-differential privacy (*ε*-DP), where* $\varepsilon \geq 0$*, if and only if for any datasets* D *and* D' *that* differ in one element*, we have*

$$\forall t \in Range(\mathbf{A}) : \Pr[\mathbf{A}(D) = t] \leq e^{\varepsilon} \Pr\left[\mathbf{A}(D') = t\right],$$

where $Range(\mathbf{A})$ *denotes the set of all possible outputs of the algorithm* $\mathbf{A}$.

2.1 Local Differential Privacy Protocols

In the local setting, there is no trusted third party. An **aggregator** wants to gather information from **users**. Users are willing to help the aggregator, but do not fully trust the aggregator for privacy. For the sake of privacy, each user perturbs her own data before sending it to the aggregator (via a secure channel). For this paper, we consider that each user has a single value v, which can be viewed as the user's answer to a given question. The aggregator aims to find out the frequencies of values among the population. Such a data collection protocol consists of the following algorithms:

- Encode is executed by each user. The algorithm takes an input value v and outputs an encoded value x.
- Perturb, which takes an encoded value x and outputs y. Each user with value v reports $y = \mathsf{Perturb}(\mathsf{Encode}(v))$. For compactness, we use $\mathsf{PE}(\cdot)$ to denote the composition of the encoding and perturbation algorithms, i.e., $\mathsf{PE}(\cdot) = \mathsf{Perturb}(\mathsf{Encode}(\cdot))$. $\mathsf{PE}(\cdot)$ should satisfy ε-local differential privacy, as defined below.
- Aggregate is executed by the aggregator; it takes all the reported values, and outputs aggregated information.

Definition 2 (Local Differential Privacy) *An algorithm* $\mathbf{A}$ *satisfies* ε*-local differential privacy (*ε*-LDP),*

where $\varepsilon \geq 0$, if and only if for any input v_1 and v_2, we have

$$\forall y \in Range(\mathbf{A}): \Pr[\mathbf{A}(v_1) = y] \leq e^{\varepsilon} \Pr[\mathbf{A}(v_2) = y],$$

where $Range(\mathbf{A})$ denotes the set of all possible outputs of the algorithm $\mathbf{A}$.

This notion is related to *randomized response* [24], which is a decades-old technique in social science to collect statistical information about embarrassing or illegal behavior. To report a single bit by random response, one reports the true value with probability p and the flip of the true value with probability $1-p$. This satisfies $\left(\ln \frac{p}{1-p}\right)$-LDP.

Comparing to the setting that requires a trusted data curator, the local setting offers a stronger level of protection, because the aggregator sees only perturbed data. Even if the aggregator is malicious and colludes with all other participants, one individual's private data is still protected according to the guarantee of LDP.

Problem Definition and Notations. There are n users. Each user j has one value v^j and reports once. We use d to denote the size of the domain of the values the users have, and $[d]$ to denote the set $\{1,2,\ldots,d\}$. Without loss of generality, we assume the input domain is $[d]$. The most basic goal of Aggregate is **frequency estimation**, i.e., estimate, for a given value $i \in [d]$, how many users have the value i. Other goals have also been considered in the literature. One goal is, when d is very large, identify values in $[d]$ that are frequent, without going through every value in $[d]$ [16, 6]. In this paper, we focus on frequency estimation. This is the most basic primitive and is a necessary building block for all other goals. Improving this will improve effectiveness of other protocols.

2.2 Basic RAPPOR

RAPPOR [13] is designed to enable longitudinal collections, where the collection happens multiple times. Indeed, Chrome's implementation of RAPPOR [3] collects answers to some questions once every 30 minutes. Two protocols, Basic RAPPOR and RAPPOR, are proposed in [13]. We first describe Basic RAPPOR.

Encoding. $\mathsf{Encode}(v) = B_0$, where B_0 is a length-d binary vector such that $B_0[v] = 1$ and $B_0[i] = 0$ for $i \neq v$. We call this **Unary Encoding**.

Perturbation. $\mathsf{Perturb}(B_0)$ consists of two steps:
Step 1: Permanent randomized response: Generate B_1 such that:

$$\Pr[B_1[i] = 1] = \begin{cases} 1 - \frac{1}{2}f, & \text{if } B_0[i] = 1, \\ \frac{1}{2}f, & \text{if } B_0[i] = 0. \end{cases}$$

RAPPOR's implementation uses $f = 1/2$ and $f = 1/4$. Note that this randomization is **symmetric** in the sense that $\Pr[B_1[i] = 1 | B_0[i] = 1] = \Pr[B_1[i] = 0 | B_0[i] = 0] = 1 - \frac{1}{2}f$; that is, the probability that a bit of 1 is preserved equals the probability that a bit of 0 is preserved. This step is carried out only once for each value v that the user has.
Step 2: Instantaneous randomized response: Report B_2 such that:

$$\Pr[B_2[i] = 1] = \begin{cases} p, & \text{if } B_1[i] = 1, \\ q, & \text{if } B_1[i] = 0. \end{cases}$$

This step is carried out each time a user reports the value. That is, B_1 will be perturbed to generate different B_2's for each reporting. RAPPOR's implementation [5] uses $p = 0.75$ and $q = 0.25$, and is hence also symmetric because $p + q = 1$.

We note that as both steps are symmetric, their combined effect can also be modeled by a symmetric randomization. Moreover, we study the problem where each user only reports once. Thus without loss of generality, we ignore the instantaneous randomized response step and consider only the permanent randomized response when trying to identify effective protocols.

Aggregation. Let B^j be the reported vector of the j-th user. Ignoring the Instantaneous randomized response step, to estimate the number of times i occurs, the aggregator computes:

$$\tilde{c}(i) = \frac{\sum_j \mathbb{1}_{\{i|B^j[i]=1\}}(i) - \frac{1}{2}fn}{1-f}$$

That is, the aggregator first counts how many time i is reported by computing $\sum_j \mathbb{1}_{\{i|B^j[i]=1\}}(i)$, which counts how many reported vectors have the i'th bit being 1, and then corrects for the effect of randomization. We use $\mathbb{1}_X(i)$ to denote the indicator function such that:

$$\mathbb{1}_X(i) = \begin{cases} 1, & \text{if } i \in X, \\ 0, & \text{if } i \notin X. \end{cases}$$

Cost. The communication and computing cost is $\Theta(d)$ for each user, and $\Theta(nd)$ for the aggregator.

Privacy. Against an adversary who may observe multiple transmissions, this achieves ε-LDP for $\varepsilon = \ln\left(\left(\frac{1-\frac{1}{2}f}{\frac{1}{2}f}\right)^2\right)$, which is $\ln 9$ for $f = 1/2$ and $\ln 49$ for $f = 1/4$.

2.3 RAPPOR

Basic RAPPOR uses unary encoding, and does not scale when d is large. To address this problem, RAPPOR uses Bloom filters [7]. While Bloom filters are typically used

to encode a set for membership testing, in RAPPOR it is used to encode a single element.

Encoding. Encoding uses a set of m hash functions $\mathbb{H} = \{H_1, H_2, \ldots, H_m\}$, each of which outputs an integer in $[k] = \{0, 1, \ldots, k-1\}$. $\mathsf{Encode}(v) = B_0$, which is k-bit binary vector such that

$$B_0[i] = \begin{cases} 1, & \text{if } \exists H \in \mathbb{H}, s.t., H(v) = i, \\ 0, & \text{otherwise.} \end{cases}$$

Perturbation. The perturbation process is identical to that of Basic RAPPOR.

Aggregation. The use of shared hashing creates challenges due to potential collisions. If two values happen to be hashed to the same set of indices, it becomes impossible to distinguish them. To deal with this problem, RAPPOR introduces the concept of cohorts. The users are divided into a number of cohorts. Each cohort uses a different set of hash functions, so that the effect of collisions is limited to within one cohort. However, partial collisions, i.e., two values are hashed to overlapping (though not identical) sets of indices, can still occur and interfere with estimation. These complexities make the aggregation algorithm more complicated. RAPPOR uses LASSO and linear regression to estimate frequencies of values.

Cost. The communication and computing cost is $\Theta(k)$ for each user. The aggregator's computation cost is higher than Basic RAPPOR due to the usage of LASSO and regression.

Privacy. RAPPOR achieves ε-LDP for $\varepsilon = \ln\left(\left(\frac{1-\frac{1}{2}f}{\frac{1}{2}f}\right)^{2m}\right)$. The RAPPOR implementation uses $m = 2$; thus this is $\ln 81 \approx 4.39$ for $f = 1/2$ and $\ln 7^4 \approx 7.78$ for $f = 1/4$.

2.4 Random Matrix Projection

Bassily and Smith [6] proposed a protocol that uses random matrix projection. This protocol has an additional Setup step.

Setup. The aggregator generates a public matrix $\Phi \in \{-\frac{1}{\sqrt{m}}, \frac{1}{\sqrt{m}}\}^{m \times d}$ uniformly at random. Here m is a parameter determined by the error bound, where the "error" is defined as the maximal distance between the estimation and true frequency of any domain.

Encoding. $\mathsf{Encode}(v) = \langle r, x \rangle$, where r is selected uniformly at random from $[m]$, and x is the v's element of the r's row of Φ, i.e., $x = \Phi[r, v]$.

Perturbation. $\mathsf{Perturb}(\langle r, x \rangle) = \langle r,\ b \cdot c \cdot m \cdot x \rangle$, where

$$b = \begin{cases} 1 & \text{with probability } p = \frac{e^\varepsilon}{e^\varepsilon + 1}, \\ -1 & \text{with probability } q = \frac{1}{e^\varepsilon + 1}, \end{cases}$$
$$c = (e^\varepsilon + 1)/(e^\varepsilon - 1).$$

Aggregation. Given reports $\langle r^j, y^j \rangle$'s, the estimate for $i \in [d]$ is given by

$$\tilde{c}(i) = \sum_j y^j \cdot \Phi[r^j, i].$$

The effect is that each user with input value i contributes c to $\tilde{c}(i)$ with probability p, and $-c$ with probability q; thus the expected contribution is

$$(p-q) \cdot c = \left(\frac{e^\varepsilon}{e^\varepsilon + 1} - \frac{1}{e^\varepsilon + 1}\right) \cdot \frac{e^\varepsilon + 1}{e^\varepsilon - 1} = 1.$$

Because of the randomness in Φ, each user with value $\neq i$ contributes to $\tilde{c}(i)$ either c or $-c$, each with probability $1/2$; thus the expected contribution from all such users is 0. Note that each row in the matrix is essentially a random hashing function mapping each value in $[d]$ to a single bit. Each user selects such a hash function, uses it to hash her value into one bit, and then perturbs this bit using random response.

Cost. A straightforward implementation of the protocol is expensive. However, the public random matrix Φ does not need to be explicitly computed. For example, using a common pseudo-random number generator, each user can randomly choose a seed to generate a row in the matrix and send the seed in her report. With this technique, the communication cost is $\Theta(\log m)$ for each user, and the computation cost is $O(d)$ for computing one row of the Φ. The aggregator needs $\Theta(dm)$ to generate Φ, and $\Theta(md)$ to compute the estimations.

3 A Framework for LDP Protocols

Multiple protocols have been proposed for estimating frequencies under LDP, and one can envision other protocols. A natural research question is how do they compare with each other? Under the same level of privacy, which protocol provides better accuracy in aggregation, with lower cost? Can we come up with even better ones? To answer these questions, we define a class of LDP protocols that we call "pure".

For a protocol to be pure, we require the specification of an additional function $\mathsf{Support}$, which maps each possible output y to a set of input values that y "supports". For example, in the basic RAPPOR protocol, an output binary vector B is interpreted as supporting each

input whose corresponding bit is 1, i.e., $\mathsf{Support}(B) = \{i \mid B[i] = 1\}$.

Definition 3 (Pure LDP Protocols) *A protocol given by* PE *and* Support *is pure if and only if there exist two probability values $p^* > q^*$ such that for all v_1,*

$$\Pr[\mathsf{PE}(v_1) \in \{y \mid v_1 \in \mathsf{Support}(y)\}] = p^*,$$
$$\forall_{v_2 \neq v_1} \Pr[\mathsf{PE}(v_2) \in \{y \mid v_1 \in \mathsf{Support}(y)\}] = q^*.$$

A pure protocol is in some sense "pure and simple". For each input v_1, the set $\{y \mid v_1 \in \mathsf{Support}(y)\}$ identifies all outputs y that "support" v_1, and can be called the support set of v_1. A pure protocol requires the probability that any value v_1 is mapped to its own support set be the same for all values. We use p^* to denote this probability. In order to satisfy LDP, it must be possible for a value $v_2 \neq v_1$ to be mapped to v_1's support set. It is required that this probability, which we use q^* to denote, must be the same for all pairs of v_1 and v_2. Intuitively, we want p^* to be as large as possible, and q^* to be as small as possible. However, satisfying ε-LDP requires that $\frac{p^*}{q^*} \leq e^{\varepsilon}$.

Basic RAPPOR is pure with $p^* = 1 - \frac{f}{2}$ and $q^* = \frac{f}{2}$. RAPPOR is not pure because there does not exist a suitable q^* due to collisions in mapping values to bit vectors. Assuming the use of two hash functions, if v_1 is mapped to $[1,1,0,0]$, v_2 is mapped to $[1,0,1,0]$, and v_3 is mapped to $[0,0,1,1]$, then because $[1,1,0,0]$ differs from $[1,0,1,0]$ by only two bits, and from $[0,0,1,1]$ by four bits, the probability that v_2 is mapped to v_1's support set is higher than that of v_3 being mapped to v_1's support set.

For a pure protocol, let y^j denote the submitted value by user j, a simple aggregation technique to estimate the number of times that i occurs is as follows:

$$\tilde{c}(i) = \frac{\sum_j \mathbb{1}_{\mathsf{Support}(y^j)}(i) - nq^*}{p^* - q^*} \tag{1}$$

The intuition is that each output that supports i gives an count of 1 for i. However, this needs to be normalized, because even if every input is i, we only expect to see $n \cdot p^*$ outputs that support i, and even if input i never occurs, we expect to see $n \cdot q^*$ supports for it. Thus the original range of 0 to n is "compressed" into an expected range of nq^* to np^*. The linear transformation in (1) corrects this effect.

Theorem 1 *For a pure LDP protocol* PE *and* Support*, (1) is unbiased, i.e., $\forall_i \mathrm{E}[\tilde{c}(i)] = nf_i$, where f_i is the fraction of times that the value i occurs.*

Proof 1

$$\begin{aligned}
\mathrm{E}[\tilde{c}(i)] =& \mathrm{E}\left[\frac{\left(\sum_j \mathbb{1}_{\mathsf{Support}(y^j)}(i)\right) - nq^*}{p^* - q^*}\right] \\
=& \frac{nf_i p^* + n(1-f_i)q^* - nq^*}{p^* - q^*} \\
=& n \cdot \frac{f_i p^* + q^* - f_i q^* - q^*}{p^* - q^*} \\
=& nf_i
\end{aligned}$$

The variance of the estimator in 1 is a valuable indicator of an LDP protocol's accuracy:

Theorem 2 *For a pure LDP protocol* PE *and* Support*, the variance of the estimation $\tilde{c}(i)$ in* (1) *is:*

$$\mathrm{Var}[\tilde{c}(i)] = \frac{nq^*(1-q^*)}{(p^*-q^*)^2} + \frac{nf_i(1-p^*-q^*)}{p^*-q^*} \tag{2}$$

Proof 2 *The random variable $\tilde{c}(i)$ is the (scaled) summation of n independent random variables drawn from the Bernoulli distribution. More specifically, nf_i (resp. $(1-f_i)n$) of these random variables are drawn from the Bernoulli distribution with parameter p^* (resp. q^*). Thus,*

$$\begin{aligned}
\mathrm{Var}[\tilde{c}(i)] &= \mathrm{Var}\left[\frac{\left(\sum_j \mathbb{1}_{\mathsf{Support}(y^j)}(i)\right) - nq^*}{p^* - q^*}\right] \\
&= \frac{\sum_j \mathrm{Var}[\mathbb{1}_{\mathsf{Support}(y^j)}(i)]}{(p^*-q^*)^2} \\
&= \frac{nf_i p^*(1-p^*) + n(1-f_i)q^*(1-q^*)}{(p^*-q^*)^2} \\
&= \frac{nq^*(1-q^*)}{(p^*-q^*)^2} + \frac{nf_i(1-p^*-q^*)}{p^*-q^*}
\end{aligned} \tag{3}$$

In many application domains, the vast majority of values appear very infrequently, and one wants to identify the more frequent ones. The key to avoid having lots of false positives is to have low estimation variances for the infrequent values. When f_i is small, the variance in (2) is dominated by the first term. We use Var^* to denote this approximation of the variance, that is:

$$\mathrm{Var}^*[\tilde{c}(i)] = \frac{nq^*(1-q^*)}{(p^*-q^*)^2} \tag{4}$$

We also note that some protocols have the property that $p^* + q^* = 1$, in which case $\mathrm{Var}^* = \mathrm{Var}$.

As the estimation $\tilde{c}(i)$ is the sum of many independent random variables, its distribution is very close to a normal distribution. Thus, the mean and variance of $\tilde{c}(i)$ fully characterizes the distribution of $\tilde{c}(i)$ for all practical purposes. When comparing different methods, we observe that fixing ε, the differences are reflected in the constants for the variance, which is where we focus our attention.

4 Optimizing LDP Protocols

We now cast many protocols that have been proposed into our framework of "pure" LDP protocols. Casting these protocols into the framework of pure protocols enables us to derive their variances and understand how each method's accuracy is affected by parameters such as domain size, ε, etc. This also enables us to generalize and optimize these protocols and propose two new protocols that improve upon existing ones. More specifically, we will consider the following protocols, which we organize by their encoding methods.

- **Direct Encoding (DE).** There is no encoding. It is a generalization of the Random Response technique.
- **Histogram Encoding (HE).** An input v is encoded as a histogram for the d possible values. The perturbation step adds noise from the Laplace distribution to each number in the histogram. We consider two aggregation techniques, SHE and THE.
  - **Summation with Histogram Encoding** (SHE) simply sums up the reported noisy histograms from all users.
  - **Thresholding with Histogram Encoding** (THE) is parameterized by a value θ; it interprets each noisy count above a threshold θ as a 1, and each count below θ as a 0.
- **Unary Encoding (UE).** An input v is encoded as a length-d bit vector, with only the bit corresponding to v set to 1. Here two key parameters in perturbation are p, the probability that 1 remains 1 after perturbation, and q, the probability that 0 is perturbed into 1. Depending on their choices, we have two protocols, SUE and OUE.
  - **Symmetric Unary Encoding** (SUE) uses $p+q=1$; this is the Basic RAPPOR protocol [13].
  - **Optimized Unary Encoding** (OUE) uses optimized choices of p and q; this is newly proposed in this paper.
- **Local Hashing (LH).** An input v is encoded by choosing at random H from a universal hash function family $\mathbb{H}$, and then outputting $(H, H(v))$. This is called Local Hashing because each user chooses independently the hash function to use. Here a key parameter is the range of these hash functions. Depending on this range, we have two protocols, BLH and OLH.
  - **Binary Local Hashing** (BLH) uses hash functions that outputs a single bit. This is equivalent to the random matrix projection technique in [6].
  - **Optimized Local Hashing** (OLH) uses optimized choices for the range of hash functions; this is newly proposed in this paper.

4.1 Direct Encoding (DE)

One natural method is to extend the binary response method to the case where the number of input values is more than 2. This is used in [23].

Encoding and Perturbing. $\mathsf{Encode}_{\mathrm{DE}}(v) = v$, and $\mathsf{Perturb}$ is defined as follows.

$$\Pr\left[\mathsf{Perturb}_{\mathrm{DE}}(x) = i\right] = \begin{cases} p = \frac{e^{\varepsilon}}{e^{\varepsilon}+d-1}, & \text{if } i = x \\ q = \frac{1-p}{d-1} = \frac{1}{e^{\varepsilon}+d-1}, & \text{if } i \neq x \end{cases}$$

Theorem 3 (Privacy of DE) *The Direct Encoding (DE) Protocol satisfies ε-LDP.*

Proof 3 *For any inputs v_1, v_2 and output y, we have:*

$$\frac{\Pr\left[\mathsf{PE}_{\mathrm{DE}}(v_1) = y\right]}{\Pr\left[\mathsf{PE}_{\mathrm{DE}}(v_2) = y\right]} \leq \frac{p}{q} = \frac{e^{\varepsilon}/(e^{\varepsilon}+d-1)}{1/(e^{\varepsilon}+d-1)} = e^{\varepsilon}$$

Aggregation. Let the $\mathsf{Support}$ function for DE be $\mathsf{Support}_{\mathrm{DE}}(i) = \{i\}$, i.e., each output value i supports the input i. Then this protocol is pure, with $p^* = p$ and $q^* = q$. Plugging these values into (4), we have

$$\mathrm{Var}^*[\tilde{c}_{\mathrm{DE}}(i)] = n \cdot \frac{d-2+e^{\varepsilon}}{(e^{\varepsilon}-1)^2}$$

Note that the variance given above is linear in nd. As d increases, the accuracy of DE suffers. This is because, as d increases, $p = \frac{e^{\varepsilon}}{e^{\varepsilon}+d-1}$, the probability that a value is transmitted correctly, becomes smaller. For example, when $e^{\varepsilon} = 49$ and $d = 2^{16}$, we have $p = \frac{49}{65584} \approx 0.00075$.

4.2 Histogram Encoding (HE)

In Histogram Encoding (HE), an input $x \in [d]$ is encoded using a length-d histogram.

Encoding. $\mathsf{Encode}_{\mathrm{HE}}(v) = [0.0, 0.0, \cdots, 1.0, \cdots, 0.0]$, where only the v-th component is 1.0. Two different input v values will result in two vectors that have L1 distance of 2.0.

Perturbing. $\mathsf{Perturb}_{\mathrm{HE}}(B)$ outputs B' such that $B'[i] = B[i] + \mathsf{Lap}\left(\frac{2}{\varepsilon}\right)$, where $\mathsf{Lap}(\beta)$ is the Laplace distribution where $\Pr\left[\mathsf{Lap}(\beta) = x\right] = \frac{1}{2\beta} e^{-|x|/\beta}$.

Theorem 4 (Privacy of HE) *The Histogram Encoding protocol satisfies ε-LDP.*

Proof 4 *For any inputs v_1, v_2, and output B, we have*

$$\frac{\Pr[B|v_1]}{\Pr[B|v_2]} = \frac{\prod_{i\in[d]} \Pr[B[i]|v_1]}{\prod_{i\in[d]} \Pr[B[i]|v_2]} = \frac{\Pr[B[v_1]|v_1]\Pr[B[v_2]|v_1]}{\Pr[B[v_1]|v_2]\Pr[B[v_2]|v_2]} \leq e^{\varepsilon/2} \cdot e^{\varepsilon/2} = e^{\varepsilon}$$

Aggregation: Summation with Histogram Encoding (SHE) works as follows: For each value, sum the noisy counts for that value reported by all users. That is, $\tilde{c}_{\text{SHE}}(i) = \sum_j B^j[i]$, where B^j is the noisy histogram received from user j. This aggregation method does not provide a Support function and is not pure. We prove its property as follows.

Theorem 5 *In* SHE*, the estimation $\tilde{c}_{\text{SHE}}$ is unbiased. Furthermore, the variance is*

$$\text{Var}[\tilde{c}_{\text{SHE}}(i)] = n\frac{8}{\varepsilon^2}$$

Proof 5 *Since the added noise is* 0*-mean; the expected value of the sum of all noisy counts is the true count.*

The $\text{Lap}(\beta)$ *distribution has variance $\frac{2}{\beta^2}$, since $\beta = \frac{\varepsilon}{2}$ for each $B^j[i]$, then the variance of each such variable is $\frac{8}{\varepsilon^2}$, and the sum of n such independent variables have variance $n\frac{8}{\varepsilon^2}$.*

Aggregation: Thresholding with Histogram Encoding (THE) interprets a vector of noisy counts discretely by defining

$$\text{Support}_{\text{THE}}(B) = \{v \mid B[v] > \theta\}$$

That is, each noise count that is $> \theta$ supports the corresponding value. This thresholding step can be performed either by the user or by the aggregator. It does not access the original value, and thus does not affect the privacy guarantee. Using thresholding to provide a Support function makes the protocol pure. The probability p^* and q^* are given by

$$p^* = 1 - F(\theta - 1);\ q^* = 1 - F(\theta),$$
$$\text{where } F(x) = \begin{cases} \frac{1}{2}e^{\frac{\varepsilon}{2}x}, & \text{if } x < 0 \\ 1 - \frac{1}{2}e^{-\frac{\varepsilon}{2}x}, & \text{if } x \geq 0 \end{cases}$$

Here, $F(\cdot)$ is the cumulative distribution function of Laplace distribution. If $0 \leq \theta \leq 1$, then we have

$$p^* = 1 - \frac{1}{2}e^{\frac{\varepsilon}{2}(\theta-1)};\ q^* = \frac{1}{2}e^{-\frac{\varepsilon}{2}\theta}.$$

Plugging these values into (4), we have

$$\text{Var}^*[\tilde{c}_{\text{HET}}(i)] = n \cdot \frac{2e^{\varepsilon\theta/2} - 1}{(1 + e^{\varepsilon(\theta-1/2)} - 2e^{\varepsilon\theta/2})^2}$$

Comparing SHE **and** THE**.** Fixing ε, one can choose a θ value to minimize the variance. Numerical analysis shows that the optimal θ is in $(\frac{1}{2}, 1)$, and depends on ε. When ε is large, $\theta \to 1$. Furthermore, $\text{Var}[\tilde{c}_{\text{THE}}] < \text{Var}[\tilde{c}_{\text{SHE}}]$ is always true. This means that by thresholding, one improves upon directly summing up noisy counts, likely because thresholding limits the impact of noises of large magnitude. In Section 5, we illustrate the differences between them using actual numbers.

4.3 Unary Encoding (UE)

Basic RAPPOR, which we described in Section 2.2, takes the approach of directly perturbing a bit vector. We now explore this method further.

Encoding. $\text{Encode}(v) = [0, \cdots, 0, 1, 0, \cdots, 0]$, a length-$d$ binary vector where only the v-th position is 1.

Perturbing. $\text{Perturb}(B)$ outputs B' as follows:

$$\Pr[B'[i] = 1] = \begin{cases} p, & \text{if } B[i] = 1 \\ q, & \text{if } B[i] = 0 \end{cases}$$

Theorem 6 (Privacy of UE) *The Unary Encoding protocol satisfies ε-LDP for*

$$\varepsilon = \ln\left(\frac{p(1-q)}{(1-p)q}\right) \tag{5}$$

Proof 6 *For any inputs v_1, v_2, and output B, we have*

$$\frac{\Pr[B|v_1]}{\Pr[B|v_2]} = \frac{\prod_{i\in[d]}\Pr[B[i]|v_1]}{\prod_{i\in[d]}\Pr[B[i]|v_2]} \tag{6}$$
$$\leq \frac{\Pr[B[v_1]=1|v_1]\Pr[B[v_2]=0|v_1]}{\Pr[B[v_1]=1|v_2]\Pr[B[v_2]=0|v_2]} \tag{7}$$
$$= \frac{p}{q} \cdot \frac{1-q}{1-p} = e^{\varepsilon}$$

(6) *is because each bit is flipped independently, and* (7) *is because v_1 and v_2 result in bit vectors that differ only in locations v_1 and v_2, and a vector with position v_1 being* 1 *and position v_2 being* 0 *maximizes the ratio.*

Aggregation. A reported bit vector is viewed as supporting an input i if $B[i] = 1$, i.e., $\text{Support}_{\text{UE}}(B) = \{i \mid B[i] = 1\}$. This yields $p^* = p$ and $q^* = q$. Interestingly, (5) does not fully determine the values of p and q for a fixed ε. Plugging (5) into (4), we have

$$\text{Var}^*[\tilde{c}_{\text{UE}}(i)] = \frac{nq(1-q)}{(p-q)^2} = \frac{nq(1-q)}{(\frac{e^{\varepsilon}q}{1-q+e^{\varepsilon}q} - q)^2}$$
$$= n \cdot \frac{((e^{\varepsilon}-1)q+1)^2}{(e^{\varepsilon}-1)^2(1-q)q}. \tag{8}$$

Symmetric UE (SUE**).** RAPPOR's implementation chooses p and q such that $p + q = 1$; making the treatment of 1 and 0 symmetric. Combining this with (5), we have

$$p = \frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1},\ q = \frac{1}{e^{\varepsilon/2}+1}$$

Plugging these into (8), we have

$$\text{Var}^*[\tilde{c}_{\text{SUE}}(i)] = n \cdot \frac{e^{\varepsilon/2}}{(e^{\varepsilon/2}-1)^2}$$

Optimized UE (OUE)**.** Instead of making p and q symmetric, we can choose them to minimize (8). Take the partial derivative of (8) with respect to q, and solving q to make the result 0, we get:

$$\begin{aligned}\frac{\partial\left[\frac{((e^\varepsilon-1)q+1)^2}{(e^\varepsilon-1)^2(1-q)q}\right]}{\partial q} &= \frac{\partial\left[\frac{1}{(e^\varepsilon-1)^2}\cdot\left(\frac{(e^\varepsilon-1)^2 q}{1-q}+\frac{2(e^\varepsilon-1)}{1-q}+\frac{1}{q(1-q)}\right)\right]}{\partial q}\\ &= \frac{\partial\left[\frac{1}{(e^\varepsilon-1)^2}\cdot\left(-(e^\varepsilon-1)^2+\frac{e^{2\varepsilon}}{1-q}+\frac{1}{q}\right)\right]}{\partial q}\\ &= \frac{1}{(e^\varepsilon-1)^2}\left(\frac{e^{2\varepsilon}}{(1-q)^2}-\frac{1}{q^2}\right)=0\\ \Longrightarrow \frac{1-q}{q} &= e^\varepsilon, \text{i.e.,} q=\frac{1}{e^\varepsilon+1} \text{ and } p=\frac{1}{2}\end{aligned}$$

Plugging $p=\frac{1}{2}$ and $q=\frac{1}{e^\varepsilon+1}$ into (8), we get

$$\mathrm{Var}^*[\tilde{c}_{\mathrm{OUE}}(i)] = n\frac{4e^\varepsilon}{(e^\varepsilon-1)^2} \tag{9}$$

The reason why setting $p=\frac{1}{2}$ and $q=\frac{1}{e^\varepsilon+1}$ is optimal when the true frequencies are small may be unclear at first glance; however, there is an intuition behind it. When the true frequencies are small, d is large. Recall that $e^\varepsilon=\frac{p}{1-p}\frac{1-q}{q}$. Setting p and q can be viewed as splitting ε into $\varepsilon_1+\varepsilon_2$ such that $\frac{p}{1-p}=e^{\varepsilon_1}$ and $\frac{1-q}{q}=e^{\varepsilon_2}$. That is, ε_1 is the privacy budget for transmitting the 1 bit, and ε_2 is the privacy budget for transmitting each 0 bit. Since there are many 0 bits and a single 1 bit, it is better to allocate as much privacy budget for transmitting the 0 bits as possible. In the extreme, setting $\varepsilon_1=0$ and $\varepsilon_2=\varepsilon$ means that setting $p=\frac{1}{2}$.

4.4 Binary Local Hashing (BLH)

Both HE and UE use unary encoding and have $\Theta(d)$ communication cost, which is too large for some applications. To reduce the communication cost, a natural idea is to first hash the input value into a domain of size $k<d$, and then apply the UE method to the hashed value. This is the basic idea underlying the RAPPOR method. However, a problem with this approach is that two values may be hashed to the same output, making them indistinguishable from each other during decoding. RAPPOR tries to address this in several ways. One is to use more than one hash functions; this reduces the chance of a collision. The other is to use cohorts, so that different cohorts use different sets of hash functions. These remedies, however, do not fully eliminate the potential effect of collisions. Using more than one hash functions also means that every individual bit needs to be perturbed more to satisfy ε-LDP for the same ε.

A better approach is to make each user belong to a cohort by herself. We call this the **local hashing** approach. The random matrix-base protocol in [6] (described in Section 2.4), in its very essence, uses a local hashing encoding that maps an input value to a single bit, which is then transmitted using randomized response. Below is the Binary Local Hashing (BLH) protocol, which is logically equivalent to the one in Section 2.4, but is simpler and, we hope, better illustrates the essence of the idea.

Let $\mathbb{H}$ be a universal hash function family, such that each hash function $H\in\mathbb{H}$ hashes an input in $[d]$ into one bit. The universal property requires that

$$\forall x,y\in[d], x\neq y: \Pr_{H\in\mathbb{H}}[H(x)=H(y)]\leq\frac{1}{2}.$$

Encoding. $\mathsf{Encode}_{\mathrm{BLH}}(v)=\langle H,b\rangle$, where $H\leftarrow_R\mathbb{H}$ is chosen uniformly at random from $\mathbb{H}$, and $b=H(v)$. Note that the hash function H can be encoded using an index for the family $\mathbb{H}$ and takes only $O(\log n)$ bits.

Perturbing. $\mathsf{Perturb}_{\mathrm{BLH}}(\langle H,b\rangle)=\langle H,b'\rangle$ such that

$$\Pr\left[b'=1\right]=\begin{cases} p=\frac{e^\varepsilon}{e^\varepsilon+1}, & \text{if } b=1\\ q=\frac{1}{e^\varepsilon+1}, & \text{if } b=0\end{cases}$$

Aggregation. $\mathsf{Support}_{\mathrm{BLH}}(\langle H,b\rangle)=\{v\mid H(v)=b\}$, that is, each reported $\langle H,b\rangle$ supports all values that are hashed by H to b, which are half of the input values. Using this Support function makes the protocol pure, with $p^*=p$ and $q^*=\frac{1}{2}p+\frac{1}{2}q=\frac{1}{2}$. Plugging the values of p^* and q^* into (4), we have

$$\mathrm{Var}^*[\tilde{c}_{\mathrm{BLH}}(i)]=n\cdot\frac{(e^\varepsilon+1)^2}{(e^\varepsilon-1)^2}.$$

4.5 Optimal Local Hashing (OLH)

Once the random matrix projection protocol is cast as binary local hashing, we can clearly see that the encoding step loses information because the output is just one bit. Even if that bit is transmitted correctly, we can get only one bit of information about the input, i.e., to which half of the input domain does the value belong. When ε is large, the amount of information loss in the encoding step dominates that of the random response step. Based on this insight, we generalize Binary Local Hashing so that each input value is hashed into a value in $[g]$, where $g\geq 2$. A larger g value means that more information is being preserved in the encoding step. This is done, however, at a cost of more information loss in the random response step. As in our analysis of the Direct Encoding method, a large domain results in more information loss.

Let $\mathbb{H}$ be a universal hash function family such that each $H\in\mathbb{H}$ outputs a value in $[g]$.

Encoding. $\mathsf{Encode}(v)=\langle H,x\rangle$, where $H\in\mathbb{H}$ is chosen

uniformly at random, and $x = H(v)$.

Perturbing. $\mathsf{Perturb}(\langle H,x\rangle) = (\langle H,y\rangle)$, where

$$\forall_{i\in[g]}\ \Pr[y=i] = \begin{cases} p = \frac{e^\varepsilon}{e^\varepsilon+g-1}, & \text{if } x=i \\ q = \frac{1}{e^\varepsilon+g-1}, & \text{if } x\neq i \end{cases}$$

Theorem 7 (Privacy of LH**)** *The Local Hashing (*LH*) Protocol satisfies ε-LDP*

Proof 7 *For any two possible input values v_1, v_2 and any output $\langle H,y\rangle$, we have,*

$$\frac{\Pr[\langle H,y\rangle|v_1]}{\Pr[\langle H,y\rangle|v_2]} = \frac{\Pr[\mathsf{Perturb}(H(v_1))=y]}{\Pr[\mathsf{Perturb}(H(v_2))=y]} \leq \frac{p}{q} = e^\varepsilon$$

Aggregation. Let $\mathsf{Support}_{\mathsf{LH}}(\langle H,y\rangle) = \{i \mid H(i)=y\}$, i.e., the set of values that are hashed into the reported value. This gives rise to a pure protocol with

$$p^* = p \text{ and } q^* = \frac{1}{g}p + \frac{g-1}{g}q = \frac{1}{g}.$$

Plugging these values into (4), we have the

$$\mathrm{Var}^*[\tilde{c}_{\mathsf{LP}}(i)] = n\cdot\frac{(e^\varepsilon-1+g)^2}{(e^\varepsilon-1)^2(g-1)}. \tag{10}$$

Optimized LH **(OLH)** Now we find the optimal g value, by taking the partial derivative of (10) with respect to g.

$$\frac{\partial\left[\frac{(e^\varepsilon-1+g)^2}{(e^\varepsilon-1)^2(g-1)}\right]}{\partial g} = \frac{\partial\left[\frac{g-1}{(e^\varepsilon-1)^2} + \frac{1}{g-1}\cdot\frac{e^{2\varepsilon}}{(e^\varepsilon-1)^2} + \frac{2e^\varepsilon}{(e^\varepsilon-1)^2}\right]}{\partial g}$$

$$= \frac{1}{(e^\varepsilon-1)^2} - \frac{1}{(g-1)^2}\cdot\frac{e^{2\varepsilon}}{(e^\varepsilon-1)^2} = 0$$

$$\Longrightarrow\ g = e^\varepsilon+1$$

When $g = e^\varepsilon+1$, we have $p^* = \frac{e^\varepsilon}{e^\varepsilon+g-1} = \frac{1}{2}$, $q^* = \frac{1}{g} = \frac{1}{e^\varepsilon+1}$ into (8), and

$$\mathrm{Var}^*[\tilde{c}_{\mathsf{OLH}}(i)] = n\cdot\frac{4e^\varepsilon}{(e^\varepsilon-1)^2}. \tag{11}$$

Comparing OLH **with** OUE**.** It is interesting to observe that the variance we derived for optimized local hashing (OLH), i.e., (11) is exactly that we have for optimized unary encoding (OUE), i.e., (9). Furthermore, the probability values p^* and q^* are also exactly the same. This illustrates that OLH and OUE are in fact deeply connected. OLH can be viewed as a compact way of implementing OUE. Compared with OUE, OLH has communication cost $O(\log n)$ instead of $O(d)$.

The fact that optimizing two apparently different encoding approaches, namely, unary encoding and local hashing, results in conceptually equivalent protocol, seems to suggest that this may be optimal (at least when d is large). However, whether this is the best possible protocol remains an interesting open question.

5 Which Protocol to Use

We have cast most of the LDP protocols proposed in the literature into our framework of pure LDP protocols. Doing so also enables us to generalize and optimize existing protocols. Now we are able to answer the question: Which LDP protocol should one use in a given setting?

Guideline. Table 1 lists the major parameters for the different protocols. Histogram encoding and unary encoding requires $\Theta(d)$ communication cost, and is expensive when d is large. Direct encoding and local hashing require $\Theta(\log d)$ or $\Theta(\log n)$ communication cost, which amounts to a constant in practice. All protocols other than DE have $O(n\cdot d)$ computation cost to estimate frequency of all values.

Numerical values of the approximate variances using (4) for all protocols are given in Table 2 and Figure 1 ($n = 10,000$). Our analysis gives the following guidelines for choosing protocols.

- When d is small, more precisely, when $d < 3e^\varepsilon+2$, DE is the best among all approaches.
- When $d > 3e^\varepsilon+2$, and the communication cost $\Theta(d)$ is acceptable, one should use OUE. (OUE has the same variance as OLH, but is easier to implement and faster because no hash functions is used.)
- When d is so large that the communication cost $\Theta(d)$ is too large, we should use OLH. It offers the same accuracy as OUE, but has communication cost $O(\log d)$ instead of $O(d)$.

Discussion. In addition to the guidelines, we make the following observations. Adding Laplacian noises to a histogram is typically used in a setting with a trusted data curator, who first computes the histogram from all users' data and then adds the noise. SHE applies it to each user's data. Intuitively, this should perform poorly relative to other protocols specifically designed for the local setting. However, SHE performs very similarly to BLH, which was specifically designed for the local setting. In fact, when $\varepsilon > 2.5$, SHE performs better than BLH.

While all protocols' variances depend on ε, the relationships are different. BLH is least sensitive to change in ε because binary hashing loses too much information. Indeed, while all other protocols have variance goes to 0 when ε goes to infinity, BLH has variance goes to n. SHE is slightly more sensitive to change in ε. DE is most sensitive to change in ε; however, when d is large, its variance is very high. OLH and OUE are able to better benefit from an increase in ε, without suffering the poor performance for small ε values.

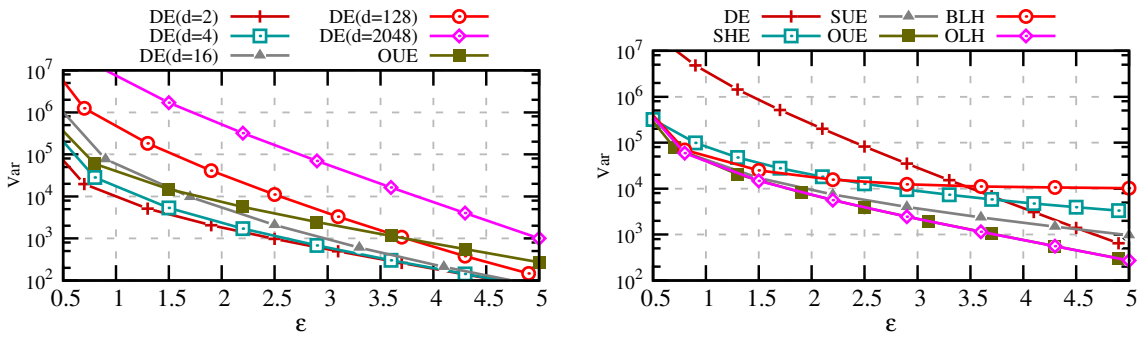
Another interesting finding is that when $d = 2$, the variance of DE is $\frac{e^\varepsilon}{(e^\varepsilon-1)^2}$, which is exactly $\frac{1}{4}$ of that of

	DE	SHE	THE ($\theta = 1$)	SUE	OUE	BLH	OLH
Communication Cost	$O(\log d)$	$O(d)$	$O(d)$	$O(d)$	$O(d)$	$O(\log n)$	$O(\log n)$
$\mathrm{Var}[\tilde{c}(i)]/n$	$\frac{d-2+e^{\varepsilon}}{(e^{\varepsilon}-1)^2}$	$\frac{8}{\varepsilon^2}$	$\frac{2e^{\varepsilon/2}-1}{(e^{\varepsilon/2}-1)^2}$	$\frac{e^{\varepsilon/2}}{(e^{\varepsilon/2}-1)^2}$	$\frac{4e^{\varepsilon}}{(e^{\varepsilon}-1)^2}$	$\frac{(e^{\varepsilon}+1)^2}{(e^{\varepsilon}-1)^2}$	$\frac{4e^{\varepsilon}}{(e^{\varepsilon}-1)^2}$

Table 1: Comparison of communication cost and variances for different methods.

	DE ($d = 2$)	DE ($d = 32$)	DE ($d = 2^{10}$)	SHE	THE ($\theta = 1$)	SUE	OUE	BLH	OLH
$\varepsilon = 0.5$	3.92	75.20	2432.40	32.00	19.44	15.92	15.67	16.67	15.67
$\varepsilon = 1.0$	0.92	11.08	347.07	8.00	5.46	3.92	3.68	4.68	3.68
$\varepsilon = 2.0$	0.18	0.92	25.22	2.00	1.50	0.92	0.72	1.72	0.72
$\varepsilon = 4.0$	0.02	0.03	0.37	0.50	0.34	0.18	0.08	1.08	0.08

Table 2: Numerical values of $\mathrm{Var}[\tilde{c}(i)]/n$ for different methods.



(a) Vary ε

(b) Vary ε (fixing $d = 2^{10}$)

Figure 1: Numerical values of $\mathrm{Var}[\tilde{c}(i)]$ for different methods.

OUE and OLH, whose variances do not depend on d. Intuitively, it is easier to transmit a piece of information when it is binary, i.e., $d = 2$. As d increases, one needs to "pay" for this increase in source entropy by having higher variance. However, it seems that there is a cap on the "price" one must pay no matter how large d is, i.e., OLH's variance does not depend on d and is always 4 times that of DE with $d = 2$. There may exist a deeper reason for this rooted in information theory. Exploring these questions is beyond the scope of this paper.

6 Experimental Evaluation

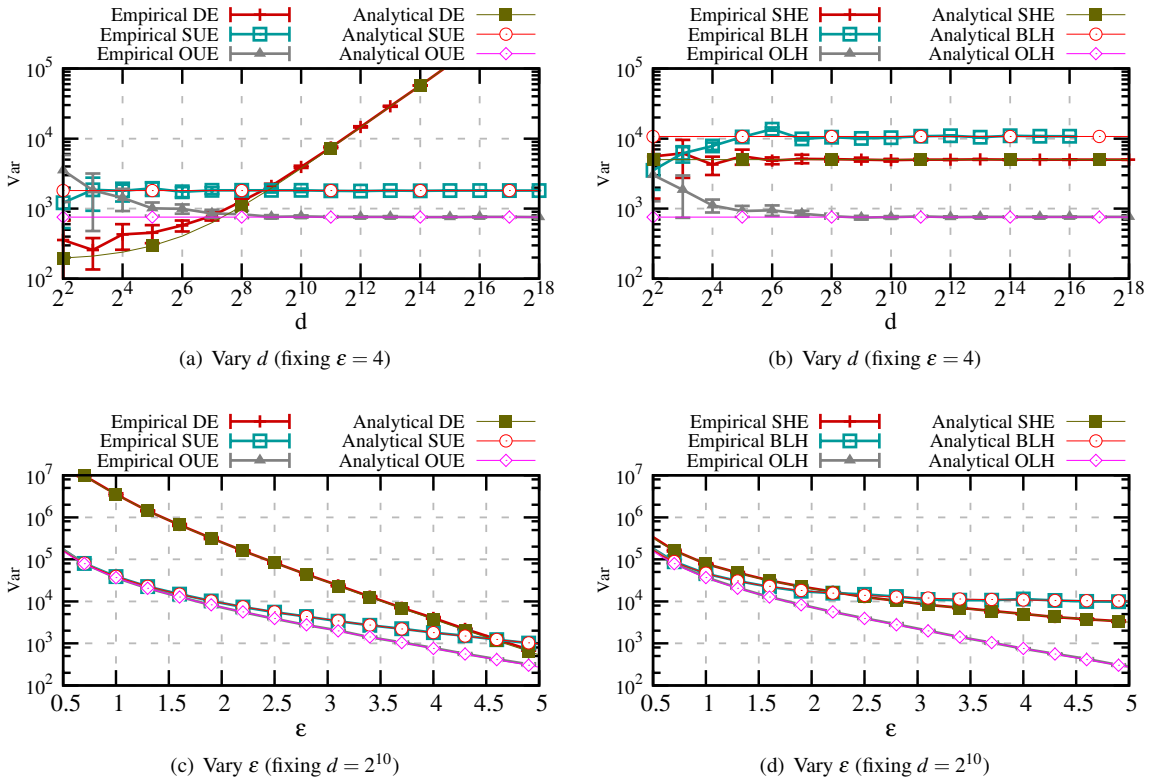
We empirically evaluate these protocols on both synthetic and real-world datasets. All experiments are performed ten times and we plot the mean and standard deviation.

6.1 Verifying Correctness of Analysis

The conclusions we drew above are based on analytical variances. We now show that our analytical results of variances match the empirically measured squared errors. For the empirical data, we issue queries using the protocols and measure the average of the squared errors, namely, $\frac{1}{d}\sum_{i\in[d]}[\tilde{c}(i) - nf_i]^2$, where f_i is the fraction of users taking value i. We run queries for all i values and repeat for ten times. We then plot the average and standard deviation of the squared error. We use synthetic data generated by following the Zipf's distribution (with distribution parameter $s = 1.1$ and $n = 10,000$ users), similar to experiments in [13].

Figure 2 gives the empirical and analytical results for all methods. In Figures 2(a) and 2(b), we fix $\varepsilon = 4$ and vary the domain size. For sufficiently large d (e.g., $d \geq 2^6$), the empirical results match very well with the analytical results. When $d < 2^6$, the analytical variance tends to underestimate the variance, because in (4) we ignore the f_i terms. Standard deviation of the measured squared error from different runs also decreases when the domain size increases. In Figures 2(c) and 2(d), we fix the domain size to $d = 2^{10}$ and vary the privacy budget. We can see that the analytical results match the empirical results for all ε values and all methods.

In practice, since the group size g of OLH can only be



(a) Vary d (fixing $\varepsilon = 4$)

(b) Vary d (fixing $\varepsilon = 4$)

(c) Vary ε (fixing $d = 2^{10}$)

(d) Vary ε (fixing $d = 2^{10}$)

Figure 2: Comparing empirical and analytical variance.

integers, we round $g = e^{\varepsilon} + 1$ to the nearest integer.

6.2 Towards Real-world Estimation

We run OLH, BLH, together with RAPPOR, on real datasets. The goal is to understand how does each protocol perform in real world scenarios and how to interpret the result. Note that RAPPOR does not fall into the pure framework of LDP protocols so we cannot use Theorem 2 to obtain the variance analytically. Instead, we run experiments to examine its performance empirically. Following the setting of Erlingsson et al. [13], we use a 128-bit Bloom filter, 2 hash functions and 8/16 cohorts in RAPPOR. In order to vary ε, we tweak the f value. The instantaneous randomization process is omitted. We implement RAPPOR in Python. The regression part, which RAPPOR introduces to handle the collisions in the Bloom filter, is implemented using Scikit-learn library [4].

Datasets. We use the **Kosarak** dataset [2], which contains the click stream of a Hungarian news website. There are around 8 million click events for 41,270 different pages. The goal is to estimate the popularity of each page, assuming all events are reported.

6.2.1 Accuracy on Frequent Values

One goal of estimating a distribution is to find out the frequent values and accurately estimate them. We run different methods to estimate the distribution of the Kosarak dataset. After the estimation, we issue queries for the 30 most frequent values in the original dataset. We then calculate the average squared error of the 30 estimations produced by different methods. Figure 3 shows the result. We try RAPPOR with both 8 cohorts (RAP(8)) and 16 cohorts (RAP(16)). It can be seen that when $\varepsilon > 1$, OLH starts to show its advantage. Moreover, variance of OLH decreases fastest among the four. Due to the internal collision caused by Bloom filters, the accuracy of RAPPOR does not benefit from larger ε. We also perform this experiment on different datasets, and the results are similar.

6.2.2 Distinguish True Counts from Noise

Although there are noises, infrequent values are still unlikely to be estimated to be frequent. Statistically, the frequent estimates are more reliable, because the probability it is generated from an infrequent value is quite low. However, for the infrequent estimates, we don't know whether it comes from an originally infrequent value or

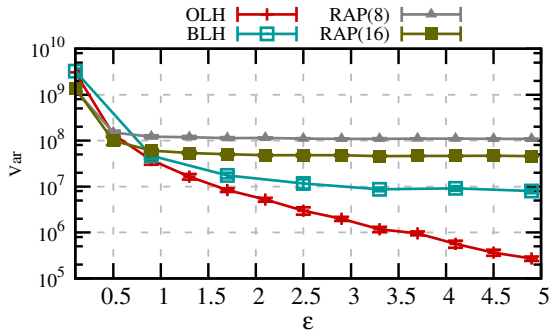


Figure 3: Average squared error, varying ε.

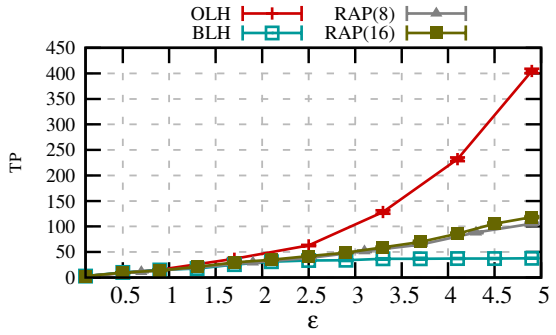


Figure 4: Number of true positives, varying ε, using significance threshold. The dashed line corresponds to the average number of items identified.

a zero-count value. Therefore, after getting the estimation, we need to choose which estimate to use, and which to discard.

Significance Threshold. In [13], the authors propose to use the significance threshold. After the estimation, all estimations above the threshold are kept, and those below the threshold T_s are discarded.

$$T_s = \Phi^{-1}\left(1 - \frac{\alpha}{d}\right)\sqrt{\text{Var}^*},$$

where d is the domain size, Φ^{-1} is the inverse of the cumulative density function of standard normal distribution, and the term inside the square root is the variance of the protocol. Roughly speaking, the parameter α controls the number of values that originally have low frequencies but estimated to have frequencies above the threshold (also known as false positives). We use $\alpha = 0.05$ in our experiment.

For the values whose estimations are discarded, we don't know for sure whether they have low or zero frequencies. Thus, a common approach is to assign the remaining probability to each of them uniformly.

Recall Var^* is the term we are trying to minimize. So a protocol with a smaller variance will have a lower threshold; thus more values can be detected reliably.

Number of Reliable Estimation. We run different protocols using the significance threshold T_s on the Kosarak dataset. Note that T_s will change as ε changes. We define a true (false) positive as a value that has frequency above (below) the threshold, and is estimated to have frequency above the threshold. In Figure 4, we show the number of true positives versus ε. As ε increases, the number of true positives increases. When $\varepsilon = 4$, RAPPOR can output 75 true positives, BLH can only output 36 true positives, but OLH can output nearly 200 true positives. We also notice that the output sizes are similar for RAPPOR and OLH, which indicates that OLH gives out very few false positives compared to RAPPOR. The cohort size does not affect much in this setting.

6.2.3 On Information Quality

Now we test both the number of true positives and false positives, varying the threshold. We run OLH, BLH and RAPPOR on the Kosarak dataset.

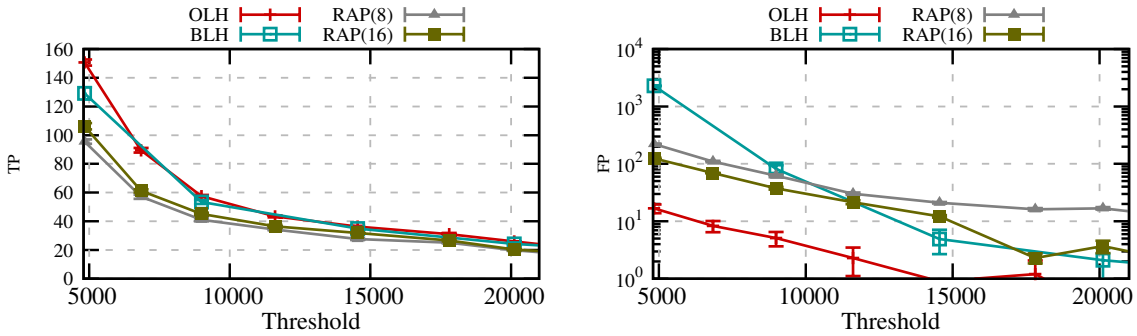
As we can see in Figure 5(a), fixing a threshold, OLH and BLH performs similarly in identifying true positives, which is as expected, because frequent values are rare, and variance does not change much the probability it is identified. RAPPOR performs slightly worse because of the Bloom filter collision.

As for the false positives, as shown in Figure 5(b), different protocols perform quite differently in eliminating false positives. When fixing T_s to be $5,000$, OLH produces tens of false positives, but BLH will produce thousands of false positives. The reason behind this is that, for the majority of infrequent values, their estimations are directly related to the variance of the protocol. A protocol with a high variance means that more infrequent values will become frequent during estimation. As a result, because of its smallest Var^*, OLH produces the least false positives while generating the most true positives.

7 Related Work

The notion of differential privacy and the technique of adding noises sampled from the Laplace distribution were introduced in [11]. Many algorithms for the centralized setting have been proposed. See [12] for a theoretical treatment of these techniques, and [19] for a treatment from a more practical perspective. It appears that only algorithms for the LDP settings have seen real world deployment. Google deployed RAPPOR [13] in Chrome, and Apple [1] also uses similar methods to help with predictions of spelling and other things.

State of the art protocols for frequency estimation under LDP are RAPPOR by Erlingsson et al. [13] and Random Matrix Projection (BLH) by Bassily and Smith [6],



(a) Number of True Positives

(b) Number of False Positives

Figure 5: Results on Kosarak dataset. The y axes are the number of identified hash values that is true/false positive. The x axes are the threshold. We assume $\varepsilon = 4$.

which we have presented in Section 2 and compared with in detail in the paper. These protocols use ideas from earlier work [20, 9]. Our proposed Optimized Unary Encoding (OUE) protocol builds upon the Basic RAPPOR protocol in [13]; and our proposed Optimized Local Hashing (OLH) protocol is inspired by BLH in [6]. Wang et al. [23] uses both generalized random response (Section 4.1) and Basic RAPPOR for learning weighted histogram. Some researchers use existing frequency estimation protocols as primitives to solve other problems in LDP setting. For example, Chen et al. [8] uses BLH [6] to learn location information about users. Qin et al. [22] use RAPPOR [13] and BLH [6] to estimate frequent items where each user has a set of items to report. These can benefit from the introduction of OUE and OLH in this paper.

There are other interesting problems in the LDP setting beyond frequency estimation. In this paper we do not study them. One problem is to identify frequent values when the domain of possible input values is very large or even unbounded, so that one cannot simply obtain estimations for all values to identify which ones are frequent. This problem is studied in [17, 6, 16]. Another problem is estimating frequencies of itemsets [14, 15]. Nguyên et al. [21] studied how to report numerical answers (e.g., time of usage, battery volume) under LDP. When these protocols use frequency estimation as a building block (such as in [16]), they can directly benefit from results in this paper. Applying insights gained in our paper to better solve these problems is interesting future work.

Kairouz et al. [18] study the problem of finding the optimal LDP protocol for two goals: (1) hypothesis testing, i.e., telling whether the users' inputs are drawn from distribution P_0 or P_1, and (2) maximize mutual information between input and output. We note that these goals are different from ours. Hypothesis testing does not reflect dependency on d. Mutual information considers only a single user's encoding, and not aggregation accuracy. For example, both global and local hashing have exactly the same mutual information characteristics, but they have very different accuracy for frequency estimation, because of collisions in global hashing. Nevertheless, it is found that for very large ε's, Direct Encoding is optimal, and for very small ε's, BLH is optimal. This is consistent with our findings. However, analysis in [18] did not lead to generalization and optimization of binary local hashing, nor does it provide concrete suggestion on which method to use for a given ε and d value.

8 Discussion

On answering multiple questions. In the setting of traditional DP, the privacy budget is split when answering multiple queries. In the local setting, previous work follow this tradition and let the users split privacy budget evenly and report on multiple questions. Instead, we suggest partitioning the users randomly into groups, and letting each group of users answer a separate question. Now we compare the utilities by these approaches.

Suppose there are $Q \geq 2$ questions. We calculate variances on one question. Since there are different number of users in the two cases (n versus n/Q), we normalize the estimations into the range from 0 to 1. In OLH, the variance is $\sigma^2 = \mathrm{Var}^*[\tilde{c}_{\mathsf{OLH}}(i)/n] = \frac{4e^{\varepsilon}}{(e^{\varepsilon}-1)^2 \cdot n}$.

When partitioning the users, n/Q users answer one question, rendering $\sigma_1^2 = \frac{4Qe^{\varepsilon}}{(e^{\varepsilon}-1)^2 \cdot n}$; when privacy budget is split, ε/Q is used for one question, we have $\sigma_2^2 = \frac{4e^{\varepsilon/Q}}{\left(e^{\varepsilon/Q}-1\right)^2 \cdot n}$. We want to show $\sigma_1^2 < \sigma_2^2$:

$$\sigma_2^2 - \sigma_1^2$$

$$
\begin{aligned}
&= \frac{4}{n}\left(\frac{e^{\varepsilon/Q}}{\left(e^{\varepsilon/Q}-1\right)^2} - \frac{Qe^{\varepsilon}}{\left(e^{\varepsilon}-1\right)^2}\right) \\
&= \frac{4e^{\varepsilon/Q}}{n\left(e^{\varepsilon/Q}-1\right)^2\left(e^{\varepsilon}-1\right)^2} \\
&\quad \cdot \left[\left(e^{\varepsilon}-1\right)^2 - Qe^{\varepsilon-\varepsilon/Q}\left(e^{\varepsilon/Q}-1\right)^2\right]
\end{aligned}
$$

The first term is always greater than zero since $\varepsilon > 0$. For the second term, we define $e^{\varepsilon/Q} = z$, and write it as:

$$
\begin{aligned}
&(z^Q-1)^2 - Qz^{Q-1}(z-1)^2 \\
=&(z-1)^2 \cdot \left[(z^{Q-1}+z^{Q-2}+\ldots+1)^2 - Qz^{Q-1}\right] > 0
\end{aligned}
$$

Therefore, σ_1^2 is always smaller than σ_2^2. Thus utility of partitioning users is better than splitting privacy budget.

Limitations. The current work only considers the framework of pure LDP protocols. It is not known whether a protocol that is not pure will produce more accurate result or not. Moreover, current protocols can only handle the case where the domain is limited, or a dictionary is available. Other techniques are needed when the domain size is very big.

9 Conclusion

In this paper, we study frequency estimation in the Local Differential Privacy (LDP) setting. We have introduced a framework of pure LDP protocols together with a simple and generic aggregation and decoding technique. This framework enables us to analyze, compare, generalize, and optimize different protocols, significantly improving our understanding of LDP protocols. More concretely, we have introduced the Optimized Local Hashing (OLH) protocol, which has much better accuracy than previous frequency estimation protocols satisfying LDP. We provide a guideline as to which protocol to choose in different scenarios. Finally we demonstrate the advantage of the OLH in both synthetic and real-world datasets.

10 Acknowledgment

This paper is based on work supported by the United States National Science Foundation under Grant No. 1640374.

References

[1] Apples differential privacy is about collecting your databut not your data. https://www.wired.com/2016/06/apples-differential-privacy-collecting-data/.

[2] Kosarak. http://fimi.ua.ac.be/data/.

[3] Rappor online description. http://www.chromium.org/developers/design-documents/rappor.

[4] Scikit-learn. http://scikit-learn.org/.

[5] Source code of rappor in chromium. https://cs.chromium.org/chromium/src/components/rappor/public/rappor_parameters.h.

[6] BASSILY, R., AND SMITH, A. Local, private, efficient protocols for succinct histograms. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing* (2015), ACM, pp. 127–135.

[7] BLOOM, B. H. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM 13*, 7 (July 1970), 422–426.

[8] CHEN, R., LI, H., QIN, A. K., KASIVISWANATHAN, S. P., AND JIN, H. Private spatial data aggregation in the local setting. In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016* (2016), pp. 289–300.

[9] DUCHI, J. C., JORDAN, M. I., AND WAINWRIGHT, M. J. Local privacy and statistical minimax rates. In *FOCS* (2013), pp. 429–438.

[10] DWORK, C. Differential privacy. In *ICALP* (2006), pp. 1–12.

[11] DWORK, C., MCSHERRY, F., NISSIM, K., AND SMITH, A. Calibrating noise to sensitivity in private data analysis. In *TCC* (2006), pp. 265–284.

[12] DWORK, C., AND ROTH, A. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science 9*, 34 (2014), 211–407.

[13] ERLINGSSON, Ú., PIHUR, V., AND KOROLOVA, A. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security* (2014), ACM, pp. 1054–1067.

[14] EVFIMIEVSKI, A., GEHRKE, J., AND SRIKANT, R. Limiting privacy breaches in privacy preserving data mining. In *PODS* (2003), pp. 211–222.

[15] EVFIMIEVSKI, A., SRIKANT, R., AGRAWAL, R., AND GEHRKE, J. Privacy preserving mining of association rules. In *KDD* (2002), pp. 217–228.

[16] FANTI, G., PIHUR, V., AND ERLINGSSON, Ú. Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proceedings on Privacy Enhancing Technologies (PoPETS) issue 3, 2016* (2016).

[17] HSU, J., KHANNA, S., AND ROTH, A. Distributed private heavy hitters. In *International Colloquium on Automata, Languages, and Programming* (2012), Springer, pp. 461–472.

[18] KAIROUZ, P., OH, S., AND VISWANATH, P. Extremal mechanisms for local differential privacy. In *Advances in neural information processing systems* (2014), pp. 2879–2887.

[19] LI, N., LYU, M., SU, D., AND YANG, W. *Differential Privacy: From Theory to Practice*. Synthesis Lectures on Information Security, Privacy, and Trust. Morgan Claypool, 2016.

[20] MISHRA, N., AND SANDLER, M. Privacy via pseudorandom sketches. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (2006), ACM, pp. 143–152.

[21] NGUYÊN, T. T., XIAO, X., YANG, Y., HUI, S. C., SHIN, H., AND SHIN, J. Collecting and analyzing data from smart device users with local differential privacy. *arXiv preprint arXiv:1606.05053* (2016).

[22] QIN, Z., YANG, Y., YU, T., KHALIL, I., XIAO, X., AND REN, K. Heavy hitter estimation over set-valued data with local differential privacy. In *CCS* (2016).

[23] WANG, S., HUANG, L., WANG, P., DENG, H., XU, H., AND YANG, W. Private weighted histogram aggregation in crowdsourcing. In *International Conference on Wireless Algorithms, Systems, and Applications* (2016), Springer, pp. 250–261.

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

A Additional Evaluation

This section provides additional experimental evaluation results. We first try to measure average squared variance on other datasets. Although RAPPOR did not specify a particular optimal setting, we vary the number of cohorts and find differences. In the end, we evaluate different methods on the Rockyou dataset.

A.1 Effect of Cohort Size

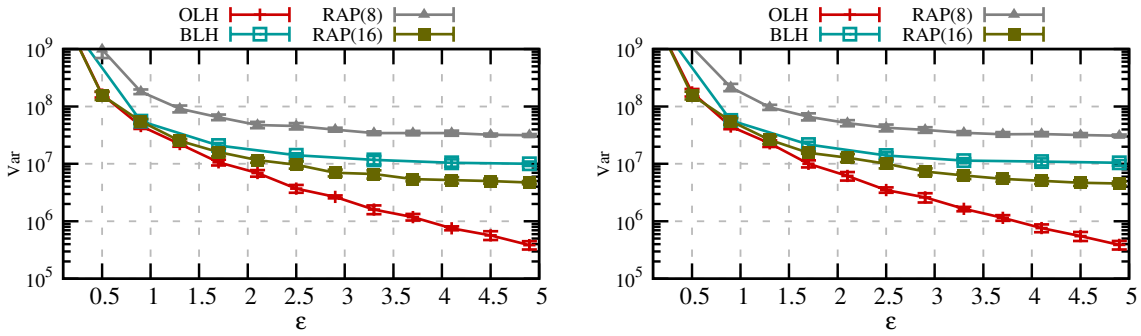
In [13], the authors did not identify the best cohort size to use. Intuitively, if there are too few cohorts, many values will be hashed to be the same in the Bloom filter, making it difficult to distinguish these values. If there are more cohorts, each cohort cannot convey enough useful information. Here we try to test what cohort size we should use. We generate 10 million values following the Zipf's distribution (with parameter 1.5), but only use the first 128 most frequent values because of memory limitation caused by regression part of RAPPOR. We then run RAPPOR using 8, 16, 32, and 64, and 128 cohorts. We measure the average squared errors of queries about the top 10 values, and the results are shown in Figure 7. As we can see, more cohorts does not necessarily help lower the squared error because the reduced probability of collision within each cohort. But it also has the disadvantage that each cohort may have insufficient information. It can be seen OLH still performs best.

A.2 Performance on Synthetic Datasets

In Figure 6, we test performance of different methods on synthetic datasets. We generate 10 million points following a normal distribution (rounded to integers, with mean 500 and standard deviation 10) and a Zipf's distribution (with parameter 1.5). The values range from 0 to 1000. We then test the average squared errors on the most frequent 100 values. It can be seen that different methods perform similarly in different distributions. RAPPOR using 16 cohorts performs better than BLH. This is because when the number of cohort is enough, each user in a sense has his own hash functions. This can be viewed as a kind of local hashing function. When we only test the top 10 values instead of top 50, RAP(16) and BLH perform similarly. Note that OLH performs best among all distributions.

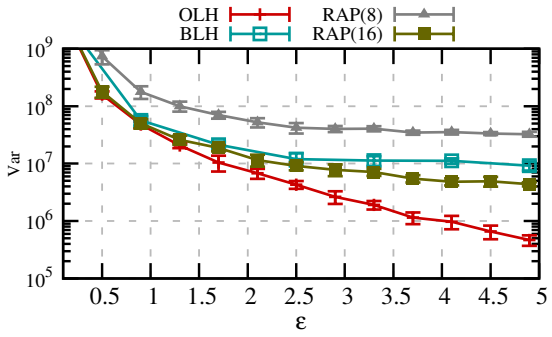
A.3 Performance on Rockyou Dataset

We run experiments on the **Rockyou** dataset, which contains 21 million users' password in plaintext. We first hash the plaintext into 20 bits, and use OLH, BLH, and Basic RAPPOR (also known as SUE in our framework) to test all hashed values. It can be seen that OLH performs best in all settings, and basic RAPPOR outperforms BLH consistently. When $\varepsilon = 4$, and threshold is 6000, OLH can recover around 50 true frequent hashes and 10 of false positives, which is 4 and 2 magnitudes smaller than BLH and basic RAPPOR, respectively. The advantage is not significant when ε is small, since the variance difference is small.

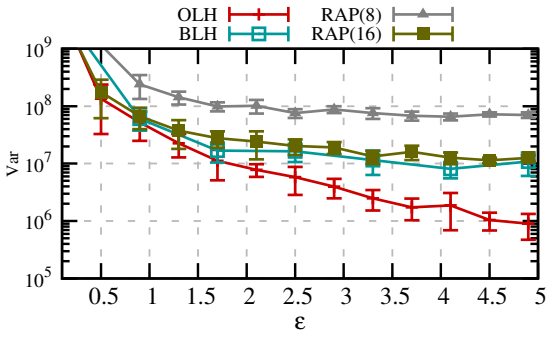


(a) Zipf's Top 100 Values

(b) Normal Top 100 Values



(c) Zipf's Top 50 Values



(d) Zipf's Top 10 Values

Figure 6: Average squared errors on estimating a distribution of 10 million points. RAPPOR is used with 128-bit long Bloom filter and 2 hash functions.

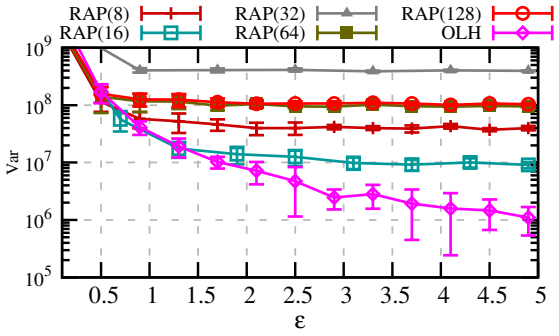
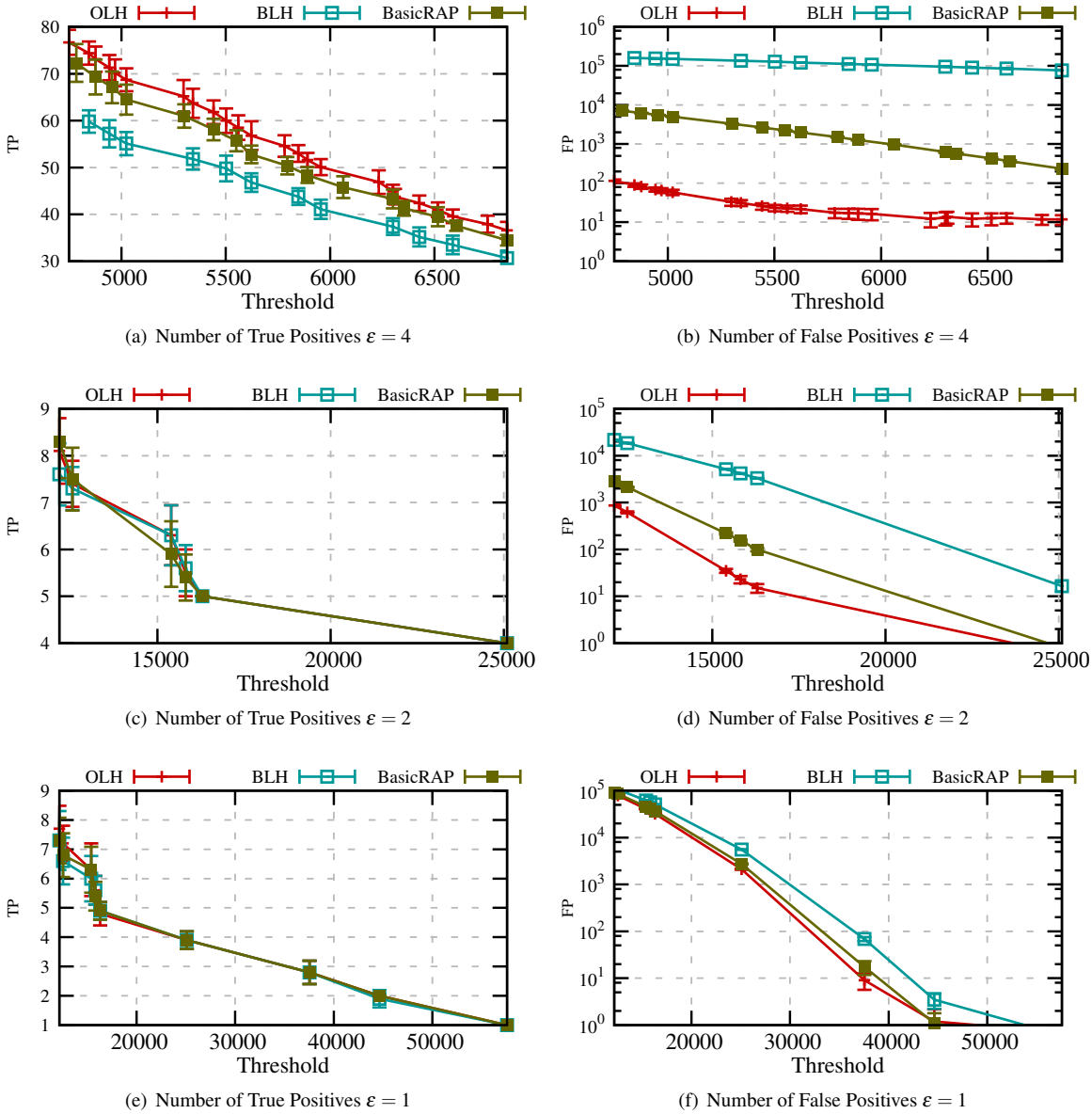


Figure 7: Average squared error on estimating a normal distribution of 1 million points. RAPPOR is used with 128-bit long Bloom filter and 2 hash functions.



(a) Number of True Positives $\varepsilon = 4$

(b) Number of False Positives $\varepsilon = 4$

(c) Number of True Positives $\varepsilon = 2$

(d) Number of False Positives $\varepsilon = 2$

(e) Number of True Positives $\varepsilon = 1$

(f) Number of False Positives $\varepsilon = 1$

Figure 8: Results on Rockyou dataset for $\varepsilon = 4, 2$ and 1. The y axes are the number of identified hash values that is true/false positive. The x axes are the threshold.