Local Differential Privacy: a tutorial

Björn Bebensee

Seoul National University
`bebensee@snu.ac.kr`

Abstract. In the past decade analysis of big data has proven to be extremely valuable in many contexts. Local Differential Privacy (LDP) is a state-of-the-art approach which allows statistical computations while protecting each individual user's privacy. Unlike Differential Privacy no trust in a central authority is necessary as noise is added to user inputs locally. In this paper we give an overview over different LDP algorithms for problems such as locally private heavy hitter identification and spatial data collection. Finally, we will give an outlook on open problems in LDP.

Keywords: Local differential privacy, privacy, differential privacy

1 Introduction

With the advance of big data analytics and its value for businesses there has been an ever-growing interest in collecting statistics from user data to improve products and to gain valuable insights. However, valuable information is often highly sensitive to the users and can not be collected without giving strong privacy guarantees to users. In the past decade an approach called Differential Privacy [13] which gives strong privacy guarantees for users has risen to popularity. They key idea behind differential privacy is that a user is given plausible deniability by adding random noise to their input. In the centralized Differential Privacy setting noise is added to the database. As the type of noise added is known statistical queries can still be computed by filtering out the noise while maintaining each user's individual privacy. However, this approach to Differential Privacy requires users to have trust the database maintainer to keep their privacy.

A stronger privacy guarantee for each individual users can be given in the local setting as there is no need to trust a centralized authority. Local Differential Privacy (LDP) was first formalized in [17] although an equivalent definition under the name "amplification" had already been introduced by [15]. In the local setting each user encodes and perturbs their own inputs before transmitting them to the untrusted server. This server can then compute statistical queries on the input data. The local settings poses its own challenges however. As each user perturbs their input individually the total variance is higher and in fact depends on the number of participants n [3]. LDP has been a quickly developing field with many new state-of-the-art approaches to important problems in LDP coming out in recent years [3,5,7,26,27]. Although the key idea behind LDP is relatively old [28] it has only recently risen to popularity and seen practical deployments by major technology organizations such as Apple [1], who use it to collect usage statistics and find commonly used emojis, new words that are not part of the dictionary yet and to improve user behaviour, Google [14], who use it in Chrome to collect commonly chosen homepages, settings, and other web browsing behaviour, and Microsoft [10], who use it for their collection of telemetry data.

In this paper we give an introduction to Local Differential Privacy and present some of its applications. We then provide an overview of current state-of-the-art solutions to problems in Local Differential Privacy such as general frequency oracles, heavy hitter identification, itemset mining and locally private spatial data collection. Finally, we give a short outlook on the future of Local Differential Privacy and present some of its open problems.

2 Local Differential Privacy

In differential privacy the introduction of noise by randomization is crucial to ensure privacy through plausible deniability. We will first introduce the centralized notion of differential privacy called ϵ-differential privacy. [12] defines ϵ-differential privacy (ϵ-DP) as follows:

Definition 1. *A randomized function $\mathcal{K}$ gives ϵ-differential privacy if for all data sets D_1 and D_2 differing on at most one element, and all $S \subseteq Range(\mathcal{K})$,*

$$\frac{Pr[\mathcal{K}(D_1) \in S]}{Pr[\mathcal{K}(D_2) \in S]} \leq e^{\epsilon}$$

One major problem with this centralized notion of Differential Privacy is that users still have to trust a central authority, namely the database maintainer, to keep their privacy. In order to be able to give users stronger privacy guarantees the concept of Local Differential Privacy (LDP) was introduced. The to LDP definition equivalent notion of "amplification" was first introduced by [15], and while the idea of local privacy was first formalized in [17] it rose to prominence through the work by Duchi et al. [11]. LDP algorithms have been implemented and used in practice and we will introduce two such deployments by Apple and Google. We will use the definition of LDP given by [14].

Definition 2. *We say that an algorithm π satisfies ϵ-Local Differential Privacy where $\epsilon > 0$ if and only if for any input v and v'*

$$\forall y \in Range(\pi) : \frac{Pr[\pi(v) = y]}{Pr[\pi(v') = y]} \leq e^{\epsilon}$$

where $Range(\pi)$ denotes every possible output of the algorithm π.

For ϵ-LDP the *privacy loss* is captured by ϵ. For $\epsilon = 0$ perfect privacy is ensured as $exp(0) = 1$ while $\epsilon = \infty$ gives no privacy guarantee. It is important to note however that the choice of ϵ is crucial in practice as the increase in privacy risks is proportional to $exp(\epsilon)$. Thus a privacy loss of $exp(1)$ and $exp(50)$ have very different implications [24].

Randomized response Although Local Differential Privacy has only recently been gaining traction and popularity, the ideas behind it are much older. The key idea of Local Differential Privacy is *randomized response*, a surveying technique first introduced by Warner in [28]. This survey technique can be used to get more accurate statistics on topics where survey respondents would like to remain confidentiality. To answer the question the respondent first toss a coin in secret and answers truthfully only if the coin comes up heads. In case of tails the respondent tosses a second coin in secret and answers "Yes" if the coin comes up tails and answers "No" if the coin comes up heads. This preserves respondents' privacy by giving them strong plausible deniability while allowing for computation of accurate population statistics.

Given that respondents comply with the protocol, respondents will answer the question truthfully 75% of the time. An accurate estimate of the true number of "Yes" answers can therefore be computed by $2(X - 0.25)$ where X refers to the fraction of respondents who answered in the positive. The survey participants are given a privacy guarantee of $\epsilon = ln(\frac{0.75}{1-0.75}) = ln(3)$ [14].

Although randomized response is a very basic LDP mechanism it is used in many more sophisticated LDP algorithms. One such implementation utilizing randomized response is *RAPPOR* from Google [14] which uses a variation of randomized response and combines it with bloom filters. We will describe this implementation in more detail in section 2.2.

2.1 Challenges in the local model

As the local model does not use any centralized database and instead enables collection of statistics from a distributed set of inputs while maintaining privacy it has gained a lot of popularity in recent years and even been adopted by Apple [1] and Google [14]. However the local model comes with its own challenges. Specifically it is more difficult to construct efficient LDP protocols and maintaining a low error-bound. Bassily et al. [3] give a good example of noise in the local model: when trying to estimate the number of HIV positives, it suffices to add Laplace noise of magnitude $O(1/\epsilon)$ in a centralized database which is independent of the number of n participants. In the local model however a very high number of participants is needed because the lower-bound of $\Omega(\sqrt{n}/\epsilon)$ is dependent on n [6]. As [3] recognizes this makes low time, space, and communication complexity on the user side in a practical implementation invaluable.

2.2 Practical implementations

RAPPOR One real world deployment of LDP has been made by Google in the form of RAPPOR [14] in Google Chrome. The source code of its implementation has also been published online and can be adopted and implemented by others[1]. Google has been interested in using LDP to collect statistics of its Google Chrome browser to gain better insights in its usage without compromising the user's

[1] https://github.com/google/rappor

privacy. Erlingsson et al. [14] present an algorithm that enables such a privacy-preserving collection of usage statistics.

Responses to the server can be assumed to be a bit vector of length k where each bit represents some property. For collection of statistics on categorical properties such as whether a user belongs to a group or uses a certain one bit is needed for each category. Therefore a bit vector of length k can provide information on k different categorical properties. In order to collect numerical values the response bits can for instance be mapped to ranges of values which are then reported. However it is also possible to collect statistics on non-categorical properties by use of Bloom filters in combination with randomized response.

In particular to report a value v it is added to the bloom filter B of size k using h hash functions. Then, in order to guarantee the user's privacy, a variation of randomized response called permanent randomized response is applied first and then a second variation called instantaneous randomized response is applied second. Permanent randomized response only determines a single bit vector B' for each value v which is then *memoized* and reused every time in place of the real answer. This is done to ensure that no inference of the user's real answer is possible even if a value is reported multiple times. To compute the permanent randomized response given the user's longitudinal privacy guarantee parameter f, for each bit i in B a reporting value B'_i is determined:

$$B'_i = \begin{cases} 1, & \text{with probability } \frac{1}{2}f \\ 0, & \text{with probability } \frac{1}{2}f \\ B_i, & \text{with probability } 1-f \end{cases}$$

Each time the value v is reported the bit vector B' is reused to compute the actual response bit vector S of length k using instantaneous randomized response. Instantaneous randomized response works by perturbing each bit of B' with randomized response parameters p, q. First a bit vector S is initialized and all values are set to 0. Then each bit i of S is set with probabilities

$$P(S_i = 1) = \begin{cases} q, & \text{if } B'_i = 1 \\ p, & \text{if } B'_i = 0 \end{cases}$$

Finally the vector S is sent to the server.

Additionally Erlingsson et al. present a few modifications of the RAPPOR algorithm. *One-time RAPPOR* is a modification that does not require protection against longitudinal attacks, i.e. inference attacks on observations of multiple transmissions, as it is only a one-time report of a value. In this case the instantaneous randomized response step can be skipped and the result of direct randomization B' transmitted instead. *Basic RAPPOR* is a variation which is equivalent to the previously mentioned collection of statistics on categorical properties. Given a well-defined and small set of strings each of the strings can be mapped directly to one of the bits in the bit vector instead of using a bloom filter. *Basic one-time RAPPOR* is the combination of the two variations. Permanent randomized response satisfies ϵ_1-LDP where

$$\epsilon_1 = 2h\ ln\left(\frac{1-\frac{1}{2}f}{\frac{1}{2}f}\right)$$

As the probability of observing a 1 in the instantaneous randomized response step is a function of both p, q and f we calculate probabilities of observing a 1 given that the underlying Bloom filter bit was set or was not set as follows:

$$q' = P(S_i = 1|b_i = 1) = \frac{1}{2}f(p+q) + (1-f)q \quad (1)$$

$$p' = P(S_i = 1|b_i = 0) = \frac{1}{2}f(p+1) + (1-f)p \quad (2)$$

Then, given (1) and (2), instantaneous randomized response satisfies ϵ_2-LDP with

$$\epsilon_2 = h\ ln\left(\frac{q'(1-p')}{p'(1-q')}\right)$$

Apple's implementation of LDP Another real-world deployment of LDP has been made by Apple [1] who use it to collect usage statistics to better understand user behavior and improve the user experience. Apple has been most interested in the problem of frequency estimation. One such example is the estimation of frequently used emojis while maintaining users' privacy by use of LDP. To give its users ϵ-LDP guarantees, Apple uses *count-mean sketch* (CMS), which is a variation of *count-min sketch* [8], a probabilistic sublinear space data structure that allows for efficient operations on data streams.

On the server-side this implementation uses a sketch matrix M of dimensions $k \times m$ with k hash functions. The client then maps domain elements $d \in \mathcal{D}$ to size m allowing for a reasonable transmission size that does not impact the user. Similarly to the count operation in *count-min sketch*, a frequency estimate for a domain element $d \in \mathcal{D}$ can be computed by averaging the counts corresponding to the according k hash functions in M. Instead of taking the minimum count in M like in *count-min sketch*, the average count is computed in order to provide better accuracy for randomly perturbed sketch matrices.

On the client-side, in order to transmit a data entry $d \in \mathcal{D}$ while maintaining ϵ-LDP for a given privacy parameter $\epsilon > 0$, a hash functions h out of the available k hash functions is chosen uniformly at random. Then, for an encoding vector $v \in \{-1,1\}^m$ the entry at position $h(d)$ is set to 1 and every other entry is set to -1. At least, much like in randomized response, each bit of the encoding vector v is flipped with probability $\frac{1}{1+e^{\epsilon/2}}$. This vector is then transmitted to the server where the sketch matrix M is updated accordingly.

Apple also presents another variation of CMS in [1] called *Hadamard count-mean sketch* (HCMS) which makes use of the Hadamard transform to achieve a similar variance to CMS while only transmitting a single bit to the server. However both CMS and HCMS assume that there is some known dictionary of domain elements which the server can query to find counts for all its data entries and to create a histogram. Apple presents an algorithm called Sequence Fragment Puzzle (SFP) which allows for calculation of histograms on unknown dictionaries. One example application of this is the discovery of new popular words that are not yet included in a phone's dictionary while maintaining ϵ-LDP.

There has been some criticism of the practical deployment of LDP by Apple however. Although the algorithms presented can ensure users' privacy the choice of the privacy loss parameter ϵ is important. Tang et al. [24] criticize the lack of transparency in the choice of privacy loss permitted by the system. Upon closer examination of Apple's actual implementation they found that although a privacy loss of $\epsilon = 1$ or $\epsilon = 2$ was guaranteed for single transmissions, Apple allowed up to 16 such transmissions per day meaning the upper bound for privacy loss is as high as $\epsilon = 16$ per day. This shows that although first steps to adopt ϵ-LDP in practice have been taken, not all implementations provide equal privacy guarantees or even transparency of what the privacy guarantees given are.

2.3 Frequency Oracles

One core problem of LDP is locally private frequency estimation. Given a domain $\mathcal{D}$ a *frequency oracle* (FO) is a protocol which estimates the frequency of an element $d \in \mathcal{D}$. A basic FO protocol has first been proposed in [28].

Wang et al. [26] introduce an abstract framework which several other FO protocols can be placed in. They divide FO protocols into three main steps: for each question, the user encodes their answer, perturbs the encoded value and sends it to the aggregator who obtains a frequency estimate for each answer of the question by decoding the reported values. The generalized framework provides several improvements in accuracy over previous algorithms such as the basic RAPPOR protocol [14]. First Wang et al. define pure LDP protocols given PE, the composition of the encoding and perturbation algorithms, and Support, which maps each output y to a set of inputs that support that are supported by the output value y, as follows:

Definition 3. *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,*

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

We call $\{y \mid v_1 \in \mathrm{Support}(y)\}$ the support set of v_1. Then, p^* represents the probability that any value v_1 is mapped to its own support set and q^* denotes the probability that any other value is mapped to the support set of v_1. In order to satisfy ϵ-LDP we must have $q^* > 0$ as it must be possible for a different value to be mapped to the support set of v_1 and specifically it is required that $\frac{p^*}{q^*} \leq e^\epsilon$. Pure LDP requires that p^* be the same for all values and q^* be the same for all pairs of values.

In a pure LDP setting the true frequency $c(i)$ of a value i can be estimated by counting reported outputs that support i. However, because of the perturbation step we expect to see at least $n \cdot q^*$ and at most $n \cdot p^*$ outputs that support i. This unbiased estimate is given by

$$\hat{c}(i) = \frac{\sum_j \mathbb{1}_{\text{Support}(y^j)}(i) - nq^*}{p^* - q^*} \tag{3}$$

where y^j is the value submitted by user j.

Next Wang et al. introduce their framework of pure LDP FO protocols which generalize many previously proposed protocols. They introduce four distinct types of protocols that utilize different encoding techniques. For two protocols they propose their own variants with optimal parameters (Optimized Unary Encoding and Optimized Local Hashing). As all of the protocols are pure LDP protocols the frequency of any value $d \in \mathcal{D}$ can be estimated by (3).

Direct Encoding (DE) Direct Encoding is the generalization of the basic FO protocol by Warner [28] with $\text{Encode}(v) = v$. The values are then perturbed with

$$\Pr[\text{Perturb}(x) = i] = \begin{cases} p = \frac{e^\epsilon}{e^\epsilon + |\mathcal{D}| - 1}, & \text{if } i = x \\ q = \frac{1-p}{|\mathcal{D}|-1} = \frac{1}{e^\epsilon + |\mathcal{D}| - 1}, & \text{if } i \neq x \end{cases}$$

The support function is $\text{Support}_{\text{DE}}(i) = \{i\}$. The aggregator can estimate the frequency of any value $d \in \mathcal{D}$ using (3). However, as the variance is linear in $|\mathcal{D}|$ the accuracy suffers for bigger domains.

Histogram Encoding (HE) Histogram Encoding uses a histogram of length $|\mathcal{D}|$ to encode each input $d \in \mathcal{D}$. Specifically, for a value v its encoding is a vector of length $|\mathcal{D}|$ in which each entry is 0.0 and only the v-th component equals 1.0. This vector is then perturbed using the Laplace distribution such that $B'[i] = B[i] + \text{Lap}(\beta)$ with $\Pr[\text{Lap}(\beta) = x] = \frac{1}{2\beta} e^{-|x|/\beta}$. For aggregation, Wang et al. present two techniques: Summation with Histogram Encoding (SHE) which sums up the reported noisy histograms from all users and is not a pure LDP protocol, and Thresholding with Histogram Encoding (THE) which is pure and interprets each noisy count above a threshold θ as a 1 and each count below θ as a 0. Its support function is given by $\text{Support}_{\text{THE}}(B) = \{v \mid B[v] > \theta\}$. As noise of large magnitude does not sum up when using a threshold the variance of THE is lower than that of SHE.

Unary Encoding (UE) In Unary Encoding a value is encoded as a bit vector similarly to the approach used in the basic RAPPOR protocol [14]. Specifically a value v is encoded as a bit vector where only the v-th position equals 1 and all other positions equal 0. Given probabilities p, q the perturbed output B' is computed as follows:

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

Depending on the choice of p and q Wang et al. differentiate between Symmetric Unary Encoding (SUE) which uses $p + q = 1$ and is equivalent to basic RAPPOR [14] and their own variant Optimized Unary Encoding (OUE) with $p = \frac{1}{2}$ and $q = \frac{1}{e^\epsilon + 1}$ which are optimal parameters that minimize the error. UE's support function is $\text{Support}_{\text{UE}}(B) = \{i \mid B[i] = 1\}$.

Binary Local Hashing (BLH) They key idea behind Binary Local Hashing is that communication cost can be lowered by hashing the input values to a domain of size $k < |\mathcal{D}|$. BLH is logically equivalent to the random matrix-base protocol proposed by Bassily et al. [4]. Consider a universal family of hash functions $\mathbb{H}$, such that each hash function $h \in \mathbb{H}$ hashes an input $d \in \mathcal{D}$ into one bit, and such that the family's universal property is given by

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

A user's input v can now be encoded as $\text{Encode}(v) = \langle H, b \rangle$ with $H \in \mathbb{H}$ chosen uniformly at random and $b = H(v)$. Given ϵ and the input bit b the perturbed input $\langle H, b' \rangle$ is given by

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

Its support function is $\text{Support}_{\text{BLH}}(\langle H, b \rangle) = \{v \mid H(v) = b\}$. It is important to note that each reported value $\langle H, b \rangle$ supports half of the input values $d \in \mathcal{D}$ as the output is only a single bit of information.

Optimal Local Hashing (OLH) As Wang et al. observe that information is lost in BLH as the output is only a single bit they generalize BLH and propose Optimal Local Hashing (OLH) which instead hashes each input value into a value in [g] where $g \geq 2$. The choice of g is crucial as a larger value allows for more information to be preserved in the encoding step but leads to more information being lost in the random response step. They analytically determine the optimal parameter to be $g = e^\epsilon + 1$.

Given a universal family of hash functions $\mathbb{H}$ such that for every $H \in \mathbb{H}$ any input $d \in \mathcal{D}$ is mapped to a value in $[g]$. Then, a user input v is again encoded as $\text{Encode}(v) = \langle H, b \rangle$ with $b = H(v)$ for $H \in \mathbb{H}$ chosen uniformly at random. The perturbed output is $\langle H, b' \rangle$ where

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

Its support function is given by $\text{Support}_{\text{OLH}}(\langle H, b' \rangle) = \{i \mid H(i) = b'\}$.

Wang et al. observe that OLH and OUE have the exact same variance and values for p^* and q^* although they use different approaches to encoding. OLH has a lower communication cost of $O(\log |\mathcal{D}|)$ compared to OUE's cost of $O(|\mathcal{D}|)$. However, as these two protocols use different optimized encoding approaches but yield the exact same variance Wang et al. suggest that it is possible that these protocols may be optimal for large domains $\mathcal{D}$.

Finally, they analyzed which LDP protocol is best depending on the size of the domain $\mathcal{D}$. They came to the conclusion that DE is best for small domains with $|\mathcal{D}| < 3e^{\epsilon} + 2$, and that OUE and OLH are better for domains $|\mathcal{D}| > 3e^{\epsilon} + 2$. However, as OUE has communication cost $\Theta(|\mathcal{D}|)$ we should use OLH for domains where the communication cost becomes too big as it offers the same accuracy at lower communication cost of $O(\log |\mathcal{D}|)$.

2.4 Heavy Hitter Identification

In the heavy hitter identification problem in the local model we wish to estimate the frequency of common domain elements (*heavy hitters*) among a set of users. For small domains this can be done by using a simple frequency oracle protocol and estimating the frequency of all domain elements. This approach is computationally infeasible for larger domains however. The problem of heavy hitter identification is very well studied [3,4,5,16,19,22]. As the error bound for efficient heavy hitter protocols presented by Mishra et al. [19] and Hsu et al. [16] is higher, we will focus on more recent work by Bassily and Smith [4] Bassily et al. [3] and Bun et al. [5].

Formally we consider a set of n users each holding an input $x_i \in \mathcal{D}$. Then S is a "distributed database" with $S = (x_1, \ldots, x_n)$ consisting of all users' inputs. Bun et al. [5] define a domain element $x \in \mathcal{D}$ as Δ-heavy if its multiplicity in S is at least Δ. This condition is satisfied if there are at least Δ users who hold the input x. For Δ as small as possible we want to find all Δ-heavy elements (i.e. *heavy hitters*). As all domain elements with multiplicities smaller than Δ are not excluded, the parameter Δ is also referred to as the protocol's *error*.

An efficient protocol for heavy hitter identification has been presented by Bassily and Smith [4]. First Bassily and Smith construct a protocol for computation of succinct histograms (S-Hist) that satisfies ϵ-LDP. A succinct histogram is a data structure which provides a list of heavy hitters $(v_1, \ldots, v_n)$ and their estimated frequencies $\hat{f}(v_i) : i \in [n]$. The error of a succinct histogram is given by the Chebyshev distance ℓ_∞ between the estimated and actual frequency as $\max_{v \in \mathcal{D}} |\hat{f}(v) - f(v)|$. Then, in order to keep communication cost low Bassily and Smith present a transformation that can transform any $(\epsilon, 0)$-DP local protocol into a 1-bit protocol given that it uses a public coin model where both user and server have access to a common random string. Such a protocol only requires each user to send a single bit to the server for the statistical query to be computed. Using this transformation they finally transform the succinct histogram protocol into a 1-bit protocol.

Bassily et al. revisit the problem in [3] and present new efficient heavy hitter algorithms TreeHist and Bitstogram which achieve near-optimal worst-case error while improving server time complexity to $\tilde{O}(n)$ and user time complexity to $\tilde{O}(1)$. We will give a short overview over the TreeHist and Bitstogram algorithms below.

TreeHist For TreeHist a binary prefix tree whose leaves correspond to domain elements is constructed. The algorithm then scans the levels of the tree starting at the top level and pruning all nodes and their children which cannot be prefixes of a heavy hitters by making queries to a given frequency oracle for that prefix. Finally, when the algorithm reaches the bottom level of the tree, the heavy hitters and a more accurate estimate of their frequencies can be determined using the frequency oracle. The number of surviving nodes does not exceed $O\big(\sqrt{n/(\log(d) \cdot \log(n))}\big)$ with high probability and therefore at most $O\big(\sqrt{n \log(d)/\log(n)}\big)$ nodes are queried with the frequency oracle.

Bitstogram In the Bitstogram protocol every user $j \in [n]$ has access to a matrix $Z \in \{-1, 1\}^{d \times n}$ chosen uniformly at random for a domain $\mathcal{D}$ of size $|\mathcal{D}| = d$. Every user j can then send a randomized response of their input v_j using the corresponding bit $Z[v_j, j]$ from the public matrix Z: they send $y_j = Z[v_j, j]$ with probability $\frac{1}{2} + \frac{\epsilon}{2}$ and $y_j = -Z[v_j, j]$ with probability $\frac{1}{2} - \frac{\epsilon}{2}$. A frequency oracle that estimates the frequency $a(v)$ of a domain element $v \in \mathcal{D}$ can then be constructed as follows:

$$a(v) = \frac{1}{\epsilon} \cdot \sum_{j \in [n]} y_j \cdot Z[v, j]$$

To identify heavy hitters the input domain is hashed into a domain of size $T = \sqrt{n}$ where n is the number of users. Given a database $S_\ell = (h(v_j), v_j[\ell])_{j \in [n])}$ for a random hash function $h : \mathcal{D} \to [T]$ and an input $v_j \in \mathcal{D}$ where $v_j[\ell]$ is the ℓ-th bit of v_j, the entry $(h(x), x[i])$ will have a significantly higher count than $(h(x), \neg x[i])$ for a heavy hitter x. This then allows for recovery of all bits of x using the frequency oracle.

Most recent research on frequency oracles and heavy hitter algorithms has focused on minimizing the error. However as Bassily and Smith [4] have shown the lower bound for the worst-case error of these tasks in LDP is at least $\Omega(\frac{1}{\epsilon}\sqrt{n \cdot log|X|})$. Although the above presented previous work by Bassily and Smith [4] and Bassily et al. [3] was efficient and had a near-optimal worst-case error, it had sub-optimal dependency of the error on the failure probability β (see table 1) [5].

Bun et al. [5] present PrivateExpanderSketch which is a heavy hitter algorithm with optimal theoretical performance. This heavy hitter algorithm is based on the frequency oracle Hashtogram in [3]. An overview over the performance of different heavy hitter algorithms is given in table 1.

Table 1. Table from [5] comparing different heavy hitter algorithms [3,4,5]. $\tilde{O}$ notation is used to hide logarithmic factors. Parameters are the number of users n, size of the domain $|X|$, privacy parameter ϵ, and the failure probability β.

Performance metric	Bun et al. [5]	Bassily et al. [3]	Bassily and Smith. [4]
Server Time	$\tilde{O}(n)$	$\tilde{O}(n)$	$\tilde{O}(n^{2.5})$
User Time	$\tilde{O}(1)$	$\tilde{O}(1)$	$\tilde{O}(n^{1.5})$
Server Memory	$\tilde{O}(\sqrt{n})$	$\tilde{O}(\sqrt{n})$	$\tilde{O}(n^2)$
User Memory	$\tilde{O}(1)$	$\tilde{O}(1)$	$\tilde{O}(n^{1.5})$
Communication/user	$\tilde{O}(1)$	$\tilde{O}(1)$	$\tilde{O}(1)$
Public randomness/user	$\tilde{O}(1)$	$\tilde{O}(1)$	$\tilde{O}(n^{1.5})$
Worst-case error	$O\left(\frac{1}{\epsilon} \cdot \sqrt{n \log(\frac{\lvert X\rvert}{\beta})}\right)$	$O\left(\frac{1}{\epsilon} \cdot \sqrt{n \log(\frac{\lvert X\rvert}{\beta})\log(\frac{1}{\beta})}\right)$	$O\left(\frac{\log^{1.5}(\frac{1}{\beta})}{\epsilon} \cdot \sqrt{n \log\lvert X\rvert}\right)$

2.5 Itemset mining

The itemset mining problem deals with the collection of statistics on set-valued inputs rather than single-valued inputs in a locally differential private setting. An example for this problem of frequency estimation on set-valued inputs is given by Thakurta et al. [25]. Here, Apple wants to estimate the frequency of emojis typed while the user submits a set of emojis that they typed during a given time period. As frequency oracle and heavy hitter protocols function by filtering out the added noise through a big enough population size an application of them to the itemset setting is not possible as transactions may appear only infrequently while their items and itemsets might appear frequently. Consider for example transactions $\{a, b, z\}$, $\{a, c, d, z\}$, $\{a, e, f, z\}$. Then we have frequent itemsets $\{a\}$, $\{z\}$ and $\{a, z\}$ that all occurred three times even though each transaction only occurred once.

Most work on this problem is very recent. A first approach to solving heavy hitter estimation over set-valued data was given by Qin et al. [22] who present a mechanism called LDPMiner. This mechanism works in a two-phase framework: in the first phase the items are filtered and potential heavy hitters are identified, in the second phase the frequency estimates are refined. ϵ-LDP is guaranteed by splitting the privacy budget ϵ into ϵ_1 and ϵ_2 which are then allocated to the two phases. Phase I then works the same way previous heavy hitter algorithms did: each user reports their set of inputs under ϵ_1-LDP and a list of k_{max} items with the highest frequency is estimated. To achieve higher accuracy the data collector then broadcasts this set of k_{max} items to all users. In Phase II each user then reports their inputs with ϵ_2-LDP – however this time the domain is reduced to the k_{max} candidate items and therefore allowing for computation of a higher accuracy estimate.

Qin et al. use a sampling randomizer algorithm to deal with the itemset setting. Here, instead of reporting their entire itemset, each user randomly samples an item which they subsequently report. Assuming that every user would normally submit a set of ℓ items, the frequency oracle should multiply the estimated frequency by ℓ to achieve an unbiased estimate. However, it is important that users with less than ℓ items pad their itemset before sampling while users with more than ℓ items simply generate a new set of exactly ℓ items by random sample without replacement. Without padding the probability of any item being chosen during the sampling process is hard to assess which makes it hard to compute an unbiased frequency estimate.

Based on this sampling randomizer algorithm they present sampling succinct histogram (sampling SH), which is a variant of succinct histogram (S-Hist) by Bassily et al. [4], as well as a variant of RAPPOR [14] called sampling RAPPOR. These sampling-randomizer-based methods can then be used in Phase I and II of the LDPMiner mechanism.

Work by Wang et al. [27] proposes a new Set-Value Item Mining (SVIM) protocol to find frequent items in the set-valued setting which provides significantly higher accuracy than LDPMiner as well as the Set-Value itemSet Mining (SVSM) protocol which can be used to identify frequent itemsets rather than single items. Previous to this work, identifying frequent itemsets had still been an open problem as LDPMiner focused on identifying only frequent items (i.e. singleton itemsets) rather than itemsets.

Wang et al. investigate padding-and-sample-based frequency oracles (PSFO) in regards to the effect of privacy amplification which has previously been studied in the standard DP setting [18]. Consider the sampling randomizer algorithm from [22]. Since the sampling step randomly selects an item, it provides an amplification effect in terms of privacy. As each item is selected with a probability $\beta = \frac{1}{\ell}$ the frequency oracle can be invoked with an $\epsilon' = \ln(\ell \cdot (e^{\ell} - 1) + 1) \geq \epsilon)$ while still guaranteeing ϵ-LDP.

They found that whether such a privacy amplification effect applies depends on the FO protocol used. Generalized Random Response and Optimized Local Hash were found to be the best performing FO protocols by [26], however while the former benefits from the privacy amplification effect the latter does not. Wang et al. therefore propose to adaptively select the best FO protocol based on the size of the item domain $|\mathcal{I}|$, ϵ and ℓ. Due to the amplification effect Generalized Random Response should be used for any

$$|\mathcal{I}| > (4\ell^2 - \ell) \cdot e^{\epsilon} + 1$$

They further note that the choice of ℓ is crucial a choice too small can lead to estimation errors while a choice too big may magnify noise in the estimation of the frequency oracle.

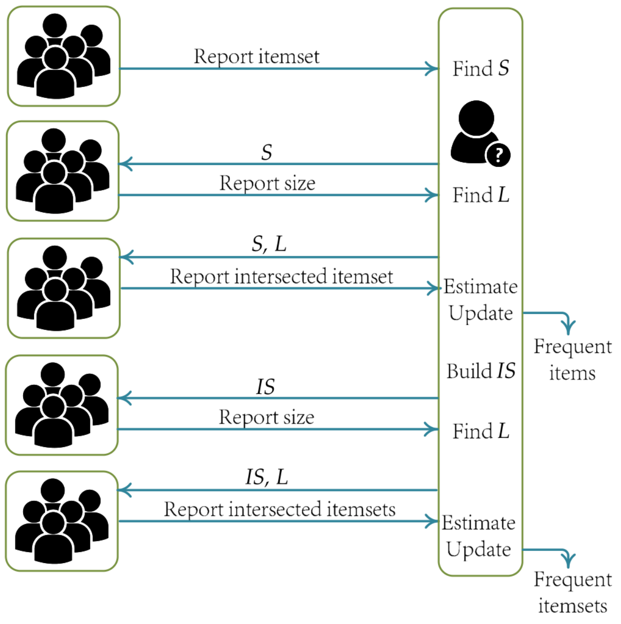

Fig. 1. Illustration of the SVIM and SVSM protocols from [27]. Users (left) are partitioned into five groups. Aggregator (right) runs SVIM to identify frequent items with first three groups, then SVSM to identify frequent itemsets with last two groups.

The SVIM protocol works similarly to LDPMiner but makes use of the privacy amplification effect to achieve a higher accuracy. The protocol works in four steps. The users are partitioned in three groups, each participating in one tasks. It is shown in [26] that this provides improved accuracy. In the first step users report inputs with small ℓ and the aggregator identifies heavy hitter candidates which are subsequently broadcasted back to the users. Second, using a standard FO protocol, users report back to the aggregator the number of candidate items they have. The aggregator can then choose an appropriate ℓ which is broadcasted to users. Third, users can report their candidate items using PSFO with the given ℓ. This allows the aggregator to create frequency estimates for these items. Finally, the aggregator identifies the k most frequent items. The known size distribution from step two can be used to further correct undercounts of these items.

The task of mining itemsets is much more challenging as there are exponentially more candidates to consider. To address this Wang et al. introduce SVSM which can find frequent itemsets effectively. The protocol first applies SVIM to find frequent items and to reduce the range of possible itemsets to a manageable size. Using the list of k most frequent items a candidate set of itemsets IS can be constructed. One can then map this problem to the regular frequent item mining problem and apply SVIM to find the most frequent itemsets.

SVIM significantly outperforms LDPMiner while SVSM solves a previously open problem. Both are the state-of-the-art in frequency estimation on set-valued inputs.

2.6 Private Spatial Data Collection

Many services such as *Google Maps* or *Waze* benefit from collection of user data to identify popular locations and to create traffic congestion maps. Given a large number of users' and their location data, we would like to learn their distribution over a spatial domain while maintaining user privacy. This problem has previously been studied in the centralized differential privacy setting [9,21]. Chen et al. [7] introduce the notion of *personalized local differential privacy* (PLDP) and propose a framework that can learn the user distribution over a spatial domain (location universe) while guaranteeing PLDP for each user.

As the location universe is a large domain in most real-world settings, ϵ-LDP can not be maintained when one wants to obtain reasonably accurate results. Chen et al. therefore introduce the notion of *personalized local differential privacy* (PLDP) which enables users to control their own privacy settings individually by setting their own privacy parameters:

Definition 4. *Given the personalized privacy specification (τ, ϵ) of a user u, a randomized algorithm π satisfies (τ, ϵ)-personalized local differential privacy for u, if for two locations $l, l' \in \tau$ and any $O \subseteq Range(A)$,*

$$\frac{Pr[\pi(l) \in O]}{Pr[\pi(l') \in O]} \leq e^{\epsilon}$$

where the probability space is over the coin flips of π.

Here, τ determines a user's *safe region* which they feel do not mind to reveal but don't want others learning about any more fine-grained location data than that. PLDP is a generalized version of ϵ-LDP as $\tau = \mathcal{L}$ for a location universe $\mathcal{L}$ implies regular ϵ-LDP.

Chen et al. then introduce the private spatial data aggregation (PSDA) framework. To estimate the number of users in a given region they define the personalized count estimation protocol (PCEP). Given n users' locations, their privacy specifications and a confidence parameter β which determines the accuracy, this protocol estimates user counts for each region. The reported locations for each user are perturbed using a local randomizer which guarantees (τ, ϵ)-PLDP. Next, they define user groups as sets of users with the same safe region and partition these user groups into clusters to minimize the maximum absolute error $\max_{l \in \mathcal{L}} |\hat{s}_l - s_l|$ for true and estimated user counts $\hat{s}_l$ and s_l in the location universe $\mathcal{L}$. The untrusted server can then apply PCEP for each cluster $C_i \in \mathcal{C}$ with a confidence parameter $\frac{\beta}{|\mathcal{C}|}$ so that the overall confidence level is guaranteed to be β. Finally the server calculates counts for all locations by combining the estimates from all clusters accordingly.

3 Open problems in LDP

As Local Differential Privacy is still a relatively new field some questions still remain open. Recent work by Avent et al. [2] has proposed a new hybrid model which allows users to choose between the local and the centralized model. Since accuracy can suffer in the pure LDP setting, as LDP gives users very strong privacy guarantees, this hybrid approach can provide improvements in accuracy. The approach by Avent et al. provides a *blended* algorithm for local search that can combine information from both the central and the local model. As their work focuses on local search, which is an application of heavy hitter identification, it remains open how this hybrid model can be applied to other problems in LDP and how much of a benefit it can provide.

It has been previously demonstrated that deep learning models can be trained while preserving differential privacy. It has also been demonstrated that some machine learning algorithms like linear regression, logistic regression and SVM classification can be successfully applied in the LDP setting [20]. It is still open however whether deep learning models can be trained in the LDP setting.

Recent research by Qin et al. [23] has presented a new approach to generate representative synthetic social graphs in an LDP setting using multiple rounds of interactions between the server and the user where based new queries are made based on previous responses. Qin et al. were able to improve the protocol's accuracy using this multi-phase interaction. It is still open how other LDP protocols can benefit from multiple interactions.

4 Conclusion

In recent years LDP has risen to popularity and seen its first real-world deployments. We have presented and discussed some of these deployments such as [14] and [1] that are used by hundreds of millions of users every day. Today LDP is the state-of-the-art approach to give strong privacy guarantees to users while enabling organizations to collect usage statistics on their products using locally private protocols. We have given an overview over the guarantees ϵ-LDP can provide as well as the pure LDP setting by [26]. As frequency estimation is at the core of many LDP protocols we have given an in-depth overview over different and optimized frequency oracle protocols in the pure LDP setting introduced by [26]. Furthermore, we have presented the core problems in LDP such as locally private heavy hitter identification, itemset mining and spatial data collection. Finally, we have given a short outlook on the future of LDP and recent research directions. Further research is necessary to determine how hybrid models [2], that give the user the freedom to choose between the central and local setting, and multi-phase interaction protocols [23] can be applied to other LDP problems and what their theoretical boundaries are.

References

1. Apple Inc., Differential Privacy Team: Learning with privacy at scale (2017)
2. Avent, B., Korolova, A., Zeber, D., Hovden, T., Livshits, B.: {BLENDER}: Enabling local search with a hybrid differential privacy model. In: 26th USENIX Security Symposium (USENIX Security 17). pp. 747–764 (2017)
3. Bassily, R., Nissim, K., Stemmer, U., Thakurta, A.: Practical locally private heavy hitters. CoRR **abs/1707.04982** (2017), `http://arxiv.org/abs/1707.04982`
4. Bassily, R., Smith, A.: Local, private, efficient protocols for succinct histograms. In: Proceedings of the forty-seventh annual ACM symposium on Theory of computing. pp. 127–135. ACM (2015)
5. Bun, M., Nelson, J., Stemmer, U.: Heavy hitters and the structure of local privacy. In: Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems. pp. 435–447. ACM (2018)
6. Chan, T.H., Shi, E., Song, D.: Optimal lower bound for differentially private multiparty aggregation. In: European Symposium on Algorithms. pp. 277–288. Springer (2012)
7. Chen, R., Li, H., Qin, A., Kasiviswanathan, S.P., Jin, H.: Private spatial data aggregation in the local setting. In: 2016 IEEE 32nd International Conference on Data Engineering (ICDE). pp. 289–300. IEEE (2016)
8. Cormode, G., Muthukrishnan, S.: An improved data stream summary: the count-min sketch and its applications. Journal of Algorithms **55**(1), 58–75 (2005)
9. Cormode, G., Procopiuc, C., Srivastava, D., Shen, E., Yu, T.: Differentially private spatial decompositions. In: 2012 IEEE 28th International Conference on Data Engineering. pp. 20–31. IEEE (2012)
10. Ding, B., Kulkarni, J., Yekhanin, S.: Collecting telemetry data privately. In: Advances in Neural Information Processing Systems. pp. 3571–3580 (2017)
11. Duchi, J.C., Jordan, M.I., Wainwright, M.J.: Local privacy and statistical minimax rates. In: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science. pp. 429–438. IEEE (2013)
12. Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) Automata, Languages and Programming. pp. 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
13. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Theory of cryptography conference. pp. 265–284. Springer (2006)
14. Erlingsson, Ú., Pihur, V., Korolova, A.: Rappor: Randomized aggregatable privacy-preserving ordinal response. In: Proceedings of the 2014 ACM SIGSAC conference on computer and communications security. pp. 1054–1067. ACM (2014)
15. Evfimievski, A., Gehrke, J., Srikant, R.: Limiting privacy breaches in privacy preserving data mining. In: Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. pp. 211–222. ACM (2003)
16. Hsu, J., Khanna, S., Roth, A.: Distributed private heavy hitters. In: International Colloquium on Automata, Languages, and Programming. pp. 461–472. Springer (2012)
17. Kasiviswanathan, S.P., Lee, H.K., Nissim, K., Raskhodnikova, S., Smith, A.: What can we learn privately? SIAM Journal on Computing **40**(3), 793–826 (2011)
18. Li, N., Qardaji, W., Su, D.: On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy. In: Proceedings of the 7th ACM

Symposium on Information, Computer and Communications Security. pp. 32–33. ACM (2012)

19. Mishra, N., Sandler, M.: Privacy via pseudorandom sketches. In: Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. pp. 143–152. ACM (2006)
20. Nguyên, T.T., Xiao, X., Yang, Y., Hui, S.C., Shin, H., Shin, J.: Collecting and analyzing data from smart device users with local differential privacy. arXiv preprint arXiv:1606.05053 (2016)
21. Qardaji, W., Yang, W., Li, N.: Differentially private grids for geospatial data. In: 2013 IEEE 29th international conference on data engineering (ICDE). pp. 757–768. IEEE (2013)
22. Qin, Z., Yang, Y., Yu, T., Khalil, I., Xiao, X., Ren, K.: Heavy hitter estimation over set-valued data with local differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 192–203. ACM (2016)
23. Qin, Z., Yu, T., Yang, Y., Khalil, I., Xiao, X., Ren, K.: Generating synthetic decentralized social graphs with local differential privacy. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 425–438. ACM (2017)
24. Tang, J., Korolova, A., Bai, X., Wang, X., Wang, X.: Privacy loss in apple's implementation of differential privacy on macos 10.12. arXiv preprint arXiv:1709.02753 (2017)
25. Thakurta, A.G., Vyrros, A.H., Vaishampayan, U.S., Kapoor, G., Freudinger, J., Prakash, V.V., Legendre, A., Duplinsky, S.: Emoji frequency detection and deep link frequency (Jul 11 2017), US Patent 9,705,908
26. Wang, T., Blocki, J., Li, N., Jha, S.: Locally differentially private protocols for frequency estimation. In: 26th USENIX Security Symposium (USENIX Security 17). pp. 729–745 (2017)
27. Wang, T., Li, N., Jha, S.: Locally differentially private frequent itemset mining. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 127–143. IEEE (2018)
28. Warner, S.L.: Randomized response: A survey technique for eliminating evasive answer bias. Journal of the American Statistical Association **60**(309), 63–69 (1965)