An Adversarial Approach to Protocol Analysis and Selection in Local Differential Privacy

M. Emre Gursoy, Ling Liu, Ka-Ho Chow, *Graduate Student Member, IEEE*, Stacey Truex, *Member, IEEE*, and Wenqi Wei

***Abstract*— Local Differential Privacy (LDP) is a popular standard for privacy-preserving data collection. Numerous LDP protocols have been proposed in the literature which differ in how they provide higher utility in different settings. Yet, few have engaged in analyzing the privacy relationships of these protocols under varying settings, and consequently, it is non-trivial to select which LDP protocol is best to use in a newly emerging application. In this paper, we present an adversarial approach to protocol analysis and selection and make three original contributions. First, we introduce a Bayesian adversary to analyze the privacy relationships of LDP protocols under varying settings. We show that different protocols have substantially different responses to the attack effectiveness of the Bayesian adversary, measured in terms of Adversarial Success Rate (ASR). Second, we provide a formal and empirical analysis on a set of privacy and utility-critical factors, including encoding parameters, privacy budget, data domain, adversarial knowledge, and statistical distribution. We show that different settings of these factors have significant effects on the ASRs of LDP protocols, and no protocol provides consistently low ASR across all settings. Third, we design and develop LDPLens, a prototype implementation of our proposed framework. Given a data collection scenario with various factors and constraints, LDPLens enables optimized selection of a desirable LDP protocol for the given scenario. We evaluate the effectiveness of LDPLens using three case studies with real-world datasets. Results show that LDPLens recommends a different protocol in each case study, and the protocol recommended by LDPLens can yield up to 1.5-2 fold reduction in utility loss, ASR or privacy budget compared to a randomly selected protocol.**

***Index Terms*— Data privacy, differential privacy, privacy-preserving data collection, adversarial analysis.**

I. Introduction

In recent years, Local Differential Privacy (LDP) has emerged as a popular notion for privacy-preserving data collection in client-server settings [1]–[6]. In LDP, each client locally perturbs their sensitive data on their device before sending the perturbed output to the data collector. Since privacy is achieved on the client device via randomized perturbation, LDP enables privacy-preserving analytics in scenarios where clients do not trust the data collector. Due to its desirable properties, LDP has received significant attention from the research community as well as the industry, including Google's RAPPOR for analyzing Chrome browser settings [4], Apple's implementation in iOS to collect popular emojis and trending words for typing recommendation [2], [7], and Microsoft's implementation in Windows 10 for collecting application telemetry [3].

The popularity of the LDP notion has led to the development of several LDP protocols such as GRR, BLH, OLH, RAPPOR, OUE and SS [4], [6], [8]–[10]. New applications often use existing LDP protocols as building blocks for building complex systems with richer capabilities. For example, among recent applications, [11] uses GRR, RAPPOR and OUE protocols as building blocks, [12] uses GRR and OLH protocols as building blocks, [13] uses the OUE protocol, and [14] uses GRR and OLH protocols. Yet, each protocol has a different way of encoding and perturbing client data, and each different application has a different setting in terms of privacy budget ε, data domain, encoding parameters, statistical data distribution, and so forth. While existing research has recognized the utility impact of such factors [1], [6], [12], [13], [15], their privacy impacts have been relatively less studied. Consequently, it is also difficult for a non-expert to select which LDP protocol and ε to use for a newly emerging application.

In this paper, we present an adversarial approach to LDP protocol analysis and selection, and make three contributions. First, we introduce a Bayesian adversary model to formally analyze the privacy relationships of LDP protocols under varying settings. We show that different protocols may have substantially different responses to the attack effectiveness of the Bayesian adversary, measured in terms of Adversarial Success Rate (ASR). Second, we provide a formal and empirical analysis of six popular LDP protocols (GRR, BLH, OLH, RAPPOR, OUE, SS) using a set of privacy and utility-critical factors including encoding parameters, privacy budget ε, data domain $\mathcal{U}$, adversarial background knowledge, and statistical data distribution. We mathematically derive expected ASR under each protocol using expected value analysis; furthermore, we perform experimental analysis using four datasets, including real-world URL page visit datasets. Our derivations and experimental results show that: (i) Changing the values of protocols' internal encoding and construction-related parameters has substantial impact on ASR (3-4 fold or more). (ii) ASR tends to differ from protocol to protocol even under the same ε, due to changes in data domain, encoding,

Manuscript received August 9, 2021; revised December 20, 2021 and March 26, 2022; accepted April 4, 2022. Date of publication April 25, 2022; date of current version May 17, 2022. This work was supported in part by the National Science Foundation under Grant NSF 2038029 and Grant NSF 1564097, and in part by the IBM Faculty Award. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. George Theodorakopoulos. *(Corresponding author: M. Emre Gursoy.)*

M. Emre Gursoy is with the Department of Computer Engineering, Koç University, 34450 Istanbul, Turkey (e-mail: emregursoy@ku.edu.tr).

Ling Liu, Ka-Ho Chow, and Wenqi Wei are with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: ling.liu@cc.gatech.edu; khchow@gatech.edu; wenqiwei@gatech.edu).

Stacey Truex is with the Department of Computer Science, Denison University, Granville, OH 43023 USA (e-mail: truexs@denison.edu).

Digital Object Identifier 10.1109/TIFS.2022.3170242

perturbation strategy, and statistical data distribution. (iii) One protocol may yield higher ASR than another protocol under a certain set of factors, but the opposite may be true under a different set of factors. These findings show that no single protocol yields optimal utility and lowest ASR consistently across different combinations of factors.

Motivated by the above findings, the third contribution of this paper is the design and development of LDPLens, a prototype implementation of our proposed framework for LDP protocol selection. Through its customizable modules, LDPLens enables its user to specify the aforementioned factors together with the adversary model (including but not limited to our Bayesian adversary) and the utility measurement method. Then, given a utility loss constraint (resp. ASR constraint), LDPLens formulates the protocol selection problem as an optimization problem with the goal of finding the protocol that minimizes ASR (resp. utility loss) under the given constraint. This process enables finding the most desirable protocol and suitable ε budget to be used in the given scenario.

We demonstrate the effectiveness of LDPLens through three case studies on real datasets. We use our Bayesian adversary's ASR as the adversary model and L_1-norm error in frequency estimation as the utility measurement method. Results of the case studies show that: (i) The most desirable protocol in each case study is different; thus, finding the most desirable protocol in a new scenario is not trivial without LDPLens. (ii) There is clear benefit in using the protocol selected by LDPLens rather than a random protocol, since it enables substantial utility loss reduction, ASR reduction or ε budget savings.

The rest of this paper is organized as follows. In Section II, we review LDP background and briefly describe popular LDP protocols. In Section III, we formally define our Bayesian adversary $\mathcal{A}$ and mathematically derive the expected ASR of $\mathcal{A}$ under each protocol. In Section IV, we perform experimental analysis with our adversary model. In Section V, we describe the design and development of LDPLens. We summarize related work in Section VI and conclude in Section VII.

II. Background and Preliminaries

A. Local Differential Privacy

Local Differential Privacy (LDP) is a popular notion for privacy-preserving data collection. In a typical LDP scenario, there exist several clients (*users*) and a data collector (*server*). We denote by $\mathcal{L}$ the client population and by $\mathcal{U}$ the universe (domain) of clients' possible true values. For client $\ell \in \mathcal{L}$ we say that $v_\ell \in \mathcal{U}$ is ℓ's true value. To ensure LDP, v_ℓ is encoded and perturbed by a randomized algorithm Ψ on the client's side, and the perturbed output is sent to the server.

Definition 1 (ε-LDP): A randomized algorithm Ψ satisfies ε-local differential privacy (ε-LDP), where $\varepsilon > 0$, if and only if for any two inputs v_1, v_2 in universe $\mathcal{U}$, it holds that:

$$\forall y \in Range(\Psi): \quad \frac{\Pr[\Psi(v_1) = y]}{\Pr[\Psi(v_2) = y]} \leq e^{\varepsilon} \tag{1}$$

where $Range(\Psi)$ denotes the set of all possible outputs of Ψ.

ε-LDP ensures that given the perturbed output y, an adversary will not be able to distinguish whether the original value was v_1 or v_2 with probability odds-ratio higher than e^{ε}. Parameter ε is often called the privacy budget. Lower ε yields stronger privacy.

Upon collecting perturbed outputs from many clients, the server performs estimation to recover statistics pertaining to the general client population. One fundamental building block for many downstream tasks is *frequency estimation* [2], [6], [13], [15]. For $v \in \mathcal{U}$, let $C(v)$ denote the true count of v (i.e., number of true observations of v among $\mathcal{L}$) and let $\bar{C}(v)$ denote the estimated count of v (i.e., the count estimated by the server after the LDP protocol). The difference between $C(v)$ and $\bar{C}(v)$ is called the *estimation error*. Several protocols were developed in the literature with goals such as minimizing estimation error, client-side computation cost or client-server communication cost for various settings of $\mathcal{U}$, $\mathcal{L}$, ε, etc. In the next section, we briefly characterize six representative LDP protocols, showing that these protocols by design differ in: (1) the mechanisms for encoding and perturbing client data on client devices, and (2) the mechanisms for aggregating and estimating from perturbed data on the server-side.

B. LDP Protocols

Here, we give the technical characterizations of six LDP protocols: GRR, BLH, OLH, RAPPOR, OUE, SS. For each protocol, we first explain the client-side encoding and perturbation procedures, then the server-side estimation procedure.

Generalized Randomized Response (GRR) is a generalization of the randomized response survey technique introduced in [16] to support non-binary $\mathcal{U}$ and arbitrary ε. GRR uses direct encoding such that $\text{ENCODE}_{\text{GRR}}(v_\ell) = v_\ell$. The perturbation algorithm Ψ_{GRR} perturbs the encoded value and outputs $y_\ell \in \mathcal{U}$ with probability:

$$\Pr[\Psi_{\text{GRR}}(v_\ell) = y_\ell] = \begin{cases} p = \frac{e^{\varepsilon}}{e^{\varepsilon}+|\mathcal{U}|-1} & \text{if } y_\ell = v_\ell \\ q = \frac{1}{e^{\varepsilon}+|\mathcal{U}|-1} & \text{if } y_\ell \neq v_\ell \end{cases} \tag{2}$$

where $|\mathcal{U}|$ denotes the size of the universe. This satisfies ε-LDP since $\frac{p}{q} = e^{\varepsilon}$. The client sends y_ℓ to the server.

On the server side, upon receiving perturbed responses from all clients, to perform estimation for some value $v \in \mathcal{U}$ the server first finds $\widehat{C}(v)$: total number of clients who reported v as their perturbed output. Then, the estimate $\bar{C}(v)$ is computed as:

$$\bar{C}(v) = \frac{\widehat{C}(v) - |\mathcal{L}| \cdot q}{p - q} \tag{3}$$

Binary Local Hashing (BLH) is inspired by [8], which uses random matrix projection for building Succinct Histograms (SH). Instead of a random matrix projection that can be expensive to construct and perform matrix multiplication, [6] proposed that using a random hash function H from a universal hash function family $\mathcal{H}$ is logically equivalent. It was noted in [15] that Hadamard transform is also similar in essence to BLH. It can be used to speed up BLH in cases where evaluating a Hadamard entry is faster than evaluating hash functions.

Let $\mathcal{H}$ be a universal hash function family such that each hash function $H \in \mathcal{H}$ maps a value from $\mathcal{U}$ into one bit, i.e.,

$H : \mathcal{U} \to \{0, 1\}$. Each client ℓ first draws a hash function uniformly randomly from $\mathcal{H}$, i.e., $H_\ell \leftarrow_\$ \mathcal{H}$. Then, bit b is computed as $b_\ell = H_\ell(v_\ell)$. Encoding result is the tuple: $\text{ENCODE}_{\text{BLH}}(v_\ell) = \langle H_\ell, b_\ell \rangle$. Perturbation algorithm Ψ_{BLH} perturbs b_ℓ to b'_ℓ such that:

$$\Pr[b'_\ell = 1] = \begin{cases} \frac{e^\varepsilon}{e^\varepsilon+1} & \text{if } b_\ell = 1 \\ \frac{1}{e^\varepsilon+1} & \text{if } b_\ell = 0 \end{cases} \tag{4}$$

The client sends tuple $\langle H_\ell, b'_\ell \rangle$ to the server.

The server receives tuples of the form $\langle H_\ell, b'_\ell \rangle$ from all clients $\ell \in \mathcal{L}$. To perform estimation for value v, the server computes $Sup(v)$: total number of clients whose reported tuples satisfy the constraint: $b'_\ell = H_\ell(v)$. Then, the estimate $\bar{C}(v)$ is computed as:

$$\bar{C}(v) = \frac{(e^\varepsilon + 1) \cdot (2 \cdot Sup(v) - |\mathcal{L}|)}{e^\varepsilon - 1} \tag{5}$$

Optimized Local Hashing (OLH) differs from BLH in that the output space of the hash functions in family $\mathcal{H}$ is no longer binary. Instead, they are g-ary. That is, OLH allows the client to encode v_ℓ into an integer in range $[1, g]$ instead of a single bit, where $g \geq 2$ is an adjustable parameter of the protocol. The rationale is to address BLH's excessive utility loss in cases where binary encoding is suboptimal. OLH's approach was shown to yield substantial utility improvement compared to BLH when ε and $\mathcal{U}$ are large [6]. The default value of g is $g = e^\varepsilon + 1$ as derived and used in [6], [12].

Let $\mathcal{H}$ be a universal hash function family where each $H \in \mathcal{H}$ maps a value from $\mathcal{U}$ into an integer in $[1, g]$, i.e., $H : \mathcal{U} \to [1, g]$. Each client ℓ randomly draws $H_\ell \leftarrow_\$ \mathcal{H}$ and computes integer x_ℓ as: $x_\ell = H_\ell(v_\ell)$. Encoding result is the tuple: $\text{ENCODE}_{\text{OLH}}(v_\ell) = \langle H_\ell, x_\ell \rangle$. Perturbation algorithm Ψ_{OLH} perturbs x_ℓ to x'_ℓ such that:

$$\forall_{i \in [1,g]}: \quad \Pr[x'_\ell = i] = \begin{cases} \frac{e^\varepsilon}{e^\varepsilon+g-1} & \text{if } x_\ell = i \\ \frac{1}{e^\varepsilon+g-1} & \text{if } x_\ell \neq i \end{cases} \tag{6}$$

The client sends tuple $\langle H_\ell, x'_\ell \rangle$ to the server.

The server receives tuples of the form $\langle H_\ell, x'_\ell \rangle$ from all clients $\ell \in \mathcal{L}$. To perform estimation for value v, the server computes $Sup(v)$: total number of clients whose reported tuples satisfy the constraint: $x'_\ell = H_\ell(v)$. Then, the estimate $\bar{C}(v)$ is computed as:

$$\bar{C}(v) = \frac{(e^\varepsilon + g - 1) \cdot (g \cdot Sup(v) - |\mathcal{L}|)}{(e^\varepsilon - 1) \cdot (g - 1)} \tag{7}$$

RAPPOR was originally developed by Google and implemented in Chrome [4], [9]. While the original version of RAPPOR relies on Bloom filters for encoding string data, variants of RAPPOR were deployed in follow-up works for data that can be encoded via binary, unary or complex encoding depending on the application scenario [5], [6], [17], [18]. Here, without loss of generality, we give the version with unary encoding.

Client ℓ initializes a bitvector B_ℓ with length $|\mathcal{U}|$. The client sets $B_\ell[v_\ell] = 1$, and for all remaining positions $j \neq v_\ell$, $B_\ell[j] = 0$. Then, the perturbation step of RAPPOR takes as input B_ℓ and outputs a perturbed bitvector B'_ℓ. Perturbation algorithm Ψ_{RAP} considers each bit in B_ℓ one-by-one, and either keeps or flips it with probability:

$$\forall_{i \in [1,|\mathcal{U}|]}: \quad \Pr[B'_\ell[i] = 1] = \begin{cases} \frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1} & \text{if } B_\ell[i] = 1 \\ \frac{1}{e^{\varepsilon/2}+1} & \text{if } B_\ell[i] = 0 \end{cases} \tag{8}$$

The client sends B'_ℓ to the server.

The server receives perturbed bitvectors B'_ℓ from all clients $\ell \in \mathcal{L}$. To perform estimation for value v, $Sup(v)$ is computed as the total number of received bitvectors that satisfy: $B'_\ell[v] = 1$. Then, the estimate $\bar{C}(v)$ is computed as:

$$\bar{C}(v) = \frac{Sup(v) + |\mathcal{L}| \cdot (\alpha - 1)}{2\alpha - 1} \tag{9}$$

where α is the bit keeping probability: $\alpha = \frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1}$.

Optimized Unary Encoding (OUE) has the same encoding phase as RAPPOR with unary encoding, but its bit keeping and flipping probabilities are different. It treats the 0 and 1 bits asymmetrically to improve accuracy of server-side estimation [6], [13].

Client ℓ initializes bitvector B_ℓ with length $|\mathcal{U}|$ such that $B_\ell[v_\ell] = 1$, and for all remaining positions $j \neq v_\ell$, $B_\ell[j] = 0$. Perturbation algorithm Ψ_{OUE} takes as input B_ℓ and produces perturbed bitvector B'_ℓ such that:

$$\forall_{i \in [1,|\mathcal{U}|]}: \quad \Pr[B'_\ell[i] = 1] = \begin{cases} \frac{1}{2} & \text{if } B_\ell[i] = 1 \\ \frac{1}{e^\varepsilon+1} & \text{if } B_\ell[i] = 0 \end{cases} \tag{10}$$

The client sends B'_ℓ to the server.

The server receives perturbed bitvectors B'_ℓ from all clients $\ell \in \mathcal{L}$. To perform estimation for value v, $Sup(v)$ is computed as the total number of received bitvectors that satisfy: $B'_\ell[v] = 1$. Then, the estimate $\bar{C}(v)$ is computed as:

$$\bar{C}(v) = \frac{2 \cdot \left((e^\varepsilon + 1) \cdot Sup(v) - |\mathcal{L}|\right)}{e^\varepsilon - 1} \tag{11}$$

Subset Selection (SS) operates by having each client ℓ report a randomly selected subset Z_ℓ of $\mathcal{U}$ to the server [10], [15]. Client's true value v_ℓ has higher probability of being included in their reported Z_ℓ, compared to other values in $\mathcal{U} \setminus \{v_\ell\}$ which are sampled uniformly randomly without replacement. The subset size $k = |Z_\ell|$ is a key parameter of the protocol. In [10], [15], it was noted that the default value of k is $k = \frac{|\mathcal{U}|}{e^\varepsilon+1}$.

The execution of the SS protocol starts by initializing an empty subset Z_ℓ. Algorithm Ψ_{SS} adds v_ℓ to Z_ℓ with probability $\frac{k \cdot e^\varepsilon}{k \cdot e^\varepsilon + |\mathcal{U}| - k}$. It constructs the remainder of Z_ℓ as follows:

- If v_ℓ was added to Z_ℓ in the previous step, then $k-1$ items are sampled from $\mathcal{U} \setminus \{v_\ell\}$ uniformly randomly without replacement, and they are added to Z_ℓ.
- If v_ℓ was not added to Z_ℓ in the previous step, then k items are sampled from $\mathcal{U} \setminus \{v_\ell\}$ uniformly randomly without replacement, and they are added to Z_ℓ.

The client sends resulting Z_ℓ to the server.

The server receives randomized subsets Z_ℓ from all clients $\ell \in \mathcal{L}$. The server defines σ_k and θ_k as:

$$\sigma_k = \frac{ke^\varepsilon}{ke^\varepsilon + |\mathcal{U}| - k} \qquad \theta_k = \frac{(k-1)(ke^\varepsilon) + (|\mathcal{U}| - k)k}{(|\mathcal{U}| - 1)(ke^\varepsilon + |\mathcal{U}| - k)}$$

To perform estimation for value v, $Sup(v)$ is computed as the total number of clients in $\mathcal{L}$ whose reported subset Z_ℓ contains v. Then, the estimate $\bar{C}(v)$ is computed as:

$$\bar{C}(v) = \frac{Sup(v) - |\mathcal{L}| \cdot \theta_k}{\sigma_k - \theta_k} \tag{12}$$

III. Bayesian Adversary and Analysis

In this section, we introduce our Bayesian adversary model and analyze LDP protocols under this common adversary. In Section III-A, we formalize the Bayesian adversary, its prediction strategy, and adversarial success measurement method. In Sections III-B to III-G, we apply the adversary to each protocol and mathematically derive the adversary's expected success rate under each protocol. In Section III-H we summarize the findings of our formal analysis; and in Section III-I we explain the intuition behind the results and provide practical examples.

A. Adversary Model $\mathcal{A}$

1) Adversary Setting: In accordance with the LDP assumption that each client's data is perturbed locally on their device and only the perturbed version is visible to the outside world, we assume that the adversary only observes the perturbed report sent from a client to the server. The adversary can be the server itself, a man-in-the-middle who observes the communication between the client and the server, or a third party analyst with whom the collected data is shared. Furthermore, we consider two adversary flavors:

- *Without Background Knowledge (w/o BK):* Adversary has no prior knowledge of client data distribution, and performs Bayesian inference using only the client's perturbed report.
- *With Background Knowledge (w/ BK):* Adversary has prior knowledge of statistical data distribution, but does not know the individual true value v_ℓ of any ℓ. Prior knowledge could be obtained from auxiliary resources, e.g., node degrees in social networks and video viewings on Youtube follow power-law distributions [13], [19], [20], browser, homepage, and password-related statistics are published for research purposes [21], [22], and the likes of geolocation density and city traffic statistics can be retrieved from the Web [23]–[25]. The adversary may use such information sources to form his/her BK.

By default, our mathematical analyses in this section assume an adversary w/o BK. The impact of BK is explored in the next sections, for varying levels of BK (Section IV-D) and for varying statistical distributions (Section IV-E).

2) Adversary Formalization: Let $\mathcal{A}$ denote the adversary. For client ℓ, let $\mathcal{O}_\ell$ denote $\mathcal{A}$'s observations of LDP protocol outputs sent from ℓ to the server, e.g., in case of RAPPOR, $\mathcal{O}_\ell = \{B'_\ell\}$; in case of OLH, $\mathcal{O}_\ell = \{\langle H_\ell, x'_\ell\rangle\}$. The goal of $\mathcal{A}$ is to correctly predict v_ℓ given $\mathcal{O}_\ell$. Denoting by v_ℓ^p the adversary's prediction, the optimal Bayesian prediction strategy is:

$$v_\ell^p = \underset{\hat{v}\in\mathcal{U}}{\text{argmax}}\ \Pr[\hat{v}|\mathcal{O}_\ell] \tag{13}$$

$$= \underset{\hat{v}\in\mathcal{U}}{\text{argmax}}\ \frac{\Pr[\mathcal{O}_\ell|\hat{v}]\cdot\Pr[\hat{v}]}{\Pr[\mathcal{O}_\ell]} \tag{14}$$

$$= \underset{\hat{v}\in\mathcal{U}}{\text{argmax}}\ \Pr[\mathcal{O}_\ell|\hat{v}]\cdot\Pr[\hat{v}] \tag{15}$$

The first step follows from Bayes' theorem and the second step is because $\Pr[\mathcal{O}_\ell]$ is constant as the adversary maximizes over variable $\hat{v}$. We say that $\mathcal{A}$ made a correct prediction if and only if $v_\ell = v_\ell^p$ and a false prediction otherwise. For the adversary w/ BK, we insert the background knowledge into Equation 15 via $\Pr[\hat{v}]$, i.e., by specifying an informed prior.

3) Measurement Methods: We use the Adversarial Success Rate (ASR) metric for measuring $\mathcal{A}$'s prediction success. ASR can be defined as the probability that the adversary's prediction is correct, i.e., $\Pr[v_\ell = v_\ell^p]$. It can be measured both empirically and mathematically. Empirically, for a client population $\mathcal{L}$, ASR is measured as the ratio of clients whose true value is correctly predicted by $\mathcal{A}$:

$$ASR = \Pr[v_\ell = v_\ell^p] = \frac{\#\text{ of clients } \ell \in \mathcal{L} \text{ such that } v_\ell = v_\ell^p}{|\mathcal{L}|} \tag{16}$$

Mathematically, it is also possible to derive the *expected* ASR of $\mathcal{A}$ through formal expected value analysis:

$$\mathbb{E}[ASR] = \mathbb{E}[\Pr[v_\ell = v_\ell^p]] \tag{17}$$

We perform expected ASR analysis in the remainder of this section. We perform empirical ASR analysis in Section IV.

4) Relationship Between ASR and the Ability to Protect Privacy: Intuitively, given the LDP protocol's outcome $\mathcal{O}_\ell$, adversary $\mathcal{A}$ predicts the user's true value as the most likely value that would have led to the observed $\mathcal{O}_\ell$. Then, ASR measures the ratio of cases in which $\mathcal{A}$'s predictions are correct. Clearly, a protocol that better "protects" the user's true value will yield lower ASR. Therefore, higher ASR can be interpreted as lower ability to protect the user's true value, hence less privacy. On the other hand, we also note the goal for ASR is not to become the "most perfect" privacy measurement method. Rather, it serves as a tool to successfully demonstrate that under a practical Bayesian adversary model, different protocols yield different adversarial resilience.

B. Applying $\mathcal{A}$ to GRR

Analysis of GRR is most straightforward among all protocols. Observe from Ψ_{GRR} in Equation 2 that $\Pr[y_\ell = v_\ell] > \Pr[y_\ell = v']$ for all $v' \in \mathcal{U} \setminus \{v_\ell\}$. Thus, the Bayes-optimal prediction strategy for $\mathcal{A}$ is to predict v_ℓ^p as: $v_\ell^p = y_\ell$. Then:

$$\mathbb{E}[ASR] = \mathbb{E}[\Pr[v_\ell = v_\ell^p]] = \Pr[v_\ell = y_\ell] = \frac{e^\varepsilon}{e^\varepsilon + |\mathcal{U}| - 1} \tag{18}$$

Since GRR uses direct encoding, expected ASR under GRR need not be conditioned on the expected behavior of a hash function (as in BLH/OLH) or item sampling (as in SS). Hence, GRR's expected ASR behavior is closest to its empirical performance.

C. Applying $\mathcal{A}$ to BLH

BLH uses client-side hashing, hence our analysis must take into account the expected behavior of the hash function H_ℓ. Since H_ℓ is picked from a universal family, its average behavior is to hash half of $\mathcal{U}$ into 0 bit whereas the remaining half of $\mathcal{U}$ is hashed to 1 bit. Observe from Ψ_{BLH} in Equation 4 that $\Pr[b'_\ell = b_\ell] > \Pr[b'_\ell = \neg b_\ell]$. Thus, the Bayes-optimal prediction strategy for $\mathcal{A}$ is to predict v^p_ℓ by random choice from subset of items which hash to b'_ℓ given H_ℓ. That is: $v^p_\ell \leftarrow_\$ \mathcal{U}_{H_\ell,b'_\ell}$ where $\mathcal{U}_{H_\ell,b'_\ell}$ is the subset of $\mathcal{U}$ satisfying the condition $\mathcal{U}_{H_\ell,b'_\ell} = \{v | v \in \mathcal{U}, H_\ell(v) = b'_\ell\}$. From this, we derive expected ASR under BLH as:

$$\mathbb{E}[ASR] = \mathbb{E}[\Pr[v_\ell = v^p_\ell]] = 1 - \mathbb{E}[\Pr[v_\ell \neq v^p_\ell]] \tag{19}$$

$$= 1 - \frac{e^\varepsilon}{e^\varepsilon + 1} \cdot \frac{\mathbb{E}[|\mathcal{U}_{H_\ell,b'_\ell}|] - 1}{\mathbb{E}[|\mathcal{U}_{H_\ell,b'_\ell}|]} - \frac{1}{e^\varepsilon + 1} \tag{20}$$

$$= 1 - \frac{e^\varepsilon}{e^\varepsilon + 1} \cdot \frac{\frac{|\mathcal{U}|}{2} - 1}{\frac{|\mathcal{U}|}{2}} - \frac{1}{e^\varepsilon + 1} \tag{21}$$

$$= 1 - \frac{e^\varepsilon \cdot (|\mathcal{U}| - 2)}{(e^\varepsilon + 1) \cdot |\mathcal{U}|} - \frac{1}{e^\varepsilon + 1} = \frac{2e^\varepsilon}{(e^\varepsilon + 1) \cdot |\mathcal{U}|} \tag{22}$$

In BLH's expected ASR, $|\mathcal{U}|$ is a multiplicative factor in the denominator as opposed to GRR's expected ASR in which $|\mathcal{U}|$ is additive. Consequently, $|\mathcal{U}|$ plays a large role in BLH's expected ASR often being lower than GRR's expected ASR.

D. Applying $\mathcal{A}$ to OLH

Recall that OLH uses g-ary hash encoding. Thus, in OLH, the average behavior of hash function H_ℓ is to hash $|\mathcal{U}|/g$ values to each hash bucket $x \in [1, g]$. By Equation 6, we know that $\Pr[x'_\ell = x_\ell] > \Pr[x'_\ell = i]$ for all $i \neq x_\ell$ and $1 \leq i \leq g$. Hence, the Bayes-optimal prediction strategy for $\mathcal{A}$ is to predict v^p_ℓ by random choice from subset of items which hash to x'_ℓ, i.e.: $v^p_\ell \leftarrow_\$ \mathcal{U}_{H_\ell,x'_\ell}$ where $\mathcal{U}_{H_\ell,x'_\ell}$ is the subset of $\mathcal{U}$ defined as: $\mathcal{U}_{H_\ell,x'_\ell} = \{v | v \in \mathcal{U}, H_\ell(v) = x'_\ell\}$. Expected ASR is derived as:

$$\mathbb{E}[ASR] = \mathbb{E}[\Pr[v_\ell = v^p_\ell]] = 1 - \mathbb{E}[\Pr[v_\ell \neq v^p_\ell]] \tag{23}$$

$$= 1 - \frac{e^\varepsilon}{e^\varepsilon + g - 1} \cdot \frac{\mathbb{E}[|\mathcal{U}_{H_\ell,x'_\ell}|] - 1}{\mathbb{E}[|\mathcal{U}_{H_\ell,x'_\ell}|]} - \frac{g-1}{e^\varepsilon + g - 1} \tag{24}$$

To solve further, we must consider two cases.
Case 1: $(|\mathcal{U}|/g) \leq 1$: In this case, $\mathbb{E}[|\mathcal{U}_{H_\ell,x'_\ell}|] = 1$ since x'_ℓ is reported. Then, continuing from Equation 24, $\mathbb{E}[ASR]$ becomes:

$$\mathbb{E}[ASR] = 1 - \frac{e^\varepsilon}{e^\varepsilon + g - 1} \cdot \frac{1-1}{1} - \frac{g-1}{e^\varepsilon + g - 1} = \frac{e^\varepsilon}{e^\varepsilon + g - 1} \tag{25}$$

Case 2: $(|\mathcal{U}|/g) > 1$: In this case, $\mathbb{E}[|\mathcal{U}_{H_\ell,x'_\ell}|] = (|\mathcal{U}|/g)$. Then, continuing from Equation 24, $\mathbb{E}[ASR]$ becomes:

$$\mathbb{E}[ASR] = 1 - \frac{e^\varepsilon}{e^\varepsilon + g - 1} \cdot \frac{\frac{|\mathcal{U}|}{g} - 1}{\frac{|\mathcal{U}|}{g}} - \frac{g-1}{e^\varepsilon + g - 1} \tag{26}$$

$$= 1 - \frac{e^\varepsilon(|\mathcal{U}| - g)}{|\mathcal{U}|(e^\varepsilon + g - 1)} - \frac{g-1}{e^\varepsilon + g - 1} \tag{27}$$

$$= \frac{e^\varepsilon g}{|\mathcal{U}|(e^\varepsilon + g - 1)} \tag{28}$$

We finish the derivation by combining the final steps of Case 1 and Case 2. We find that expected ASR under OLH overall is:

$$\mathbb{E}[ASR] = \frac{e^\varepsilon}{(e^\varepsilon + g - 1) \cdot \max\left\{\frac{|\mathcal{U}|}{g}, 1\right\}} \tag{29}$$

Note that expected ASR under OLH found in Equation 29 is also dependent on the encoding parameter g. If the default value of g, which is $g = e^\varepsilon + 1$ [6], [12], is plugged into Equation 29 we obtain:

$$\mathbb{E}[ASR] = \frac{1}{2} \cdot \frac{1}{\max\left\{\frac{|\mathcal{U}|}{e^\varepsilon + 1}, 1\right\}} \tag{30}$$

From Equation 30, we observe that when ε is large enough that $e^\varepsilon + 1 > |\mathcal{U}|$, then OLH's expected ASR will be upper bounded by $\frac{1}{2}$. For smaller ε, expected ASR under OLH will be smaller than $\frac{1}{2}$.

E. Applying $\mathcal{A}$ to RAPPOR

RAPPOR uses bitvector encoding with client ℓ sending perturbed bitvector B'_ℓ to the server. We define a set of pairwise disjoint events E_0, E_1, ..., $E_{|\mathcal{U}|}$ as follows. Event E_0 is the case where the original 1 bit in the client's bitvector is flipped to 0 in B'_ℓ. For all remaining events E_i where $i \in [1, |\mathcal{U}|]$, event E_i is the case where the original 1 bit is kept and $i - 1$ other bits that were 0 in the client's original bitvector became 1 in B'_ℓ due to bit flipping. Then, we have:

$$\mathbb{E}[ASR] = \sum_{i=0}^{|\mathcal{U}|} \Pr[v_\ell = v^p_\ell \wedge E_i] \tag{31}$$

Below, we calculate each entry in the right hand side summation in Equation 31 separately. Finally, we sum them up.
$\mathbb{E}[ASR]$ with Event E_0:

$$\Pr[v_\ell = v^p_\ell \wedge E_0] = \Pr[v_\ell = v^p_\ell | E_0] \cdot \Pr[E_0] \tag{32}$$

$$= \frac{1}{|\mathcal{U}|} \cdot \left(\frac{e^{\varepsilon/2}}{e^{\varepsilon/2} + 1}\right)^{|\mathcal{U}|-1} \cdot \frac{1}{e^{\varepsilon/2} + 1} \tag{33}$$

The rationale is as follows. By definition of Ψ_{RAP}, we have: $\Pr[E_0] = \frac{1}{e^{\varepsilon/2}+1}$. To compute $\Pr[v_\ell = v^p_\ell | E_0]$, we observe that given the original bit was flipped to 0, if any other bit is flipped to 1, $\mathcal{A}$'s optimal prediction strategy is to pick among the indexes with 1 bit. Thus, the only way for $\mathcal{A}$ to predict $v^p_\ell = v_\ell$ is if no original 0 bits are flipped to 1, which happens with probability $\left(\frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1}\right)^{|\mathcal{U}|-1}$, and under that scenario, the correct prediction probability is $\frac{1}{|B|} = \frac{1}{|\mathcal{U}|}$. We combine them to arrive at Equation 33.
$\mathbb{E}[ASR]$ with Event E_i where $1 \leq i \leq |\mathcal{U}|$:

$$\Pr[v_\ell = v^p_\ell \wedge E_i] = \Pr[v_\ell = v^p_\ell | E_i] \cdot \Pr[E_i] \tag{34}$$

$$= \frac{1}{i} \cdot \frac{e^{\varepsilon/2}}{e^{\varepsilon/2} + 1} \cdot \text{Bin}\left(i - 1; |\mathcal{U}| - 1, \frac{1}{e^{\varepsilon/2} + 1}\right) \tag{35}$$

where $\text{Bin}\left(i-1; |\mathcal{U}|-1, \frac{1}{e^{\varepsilon/2}+1}\right)$ denotes a Binomial distribution with $|\mathcal{U}|-1$ trials, success probability $\frac{1}{e^{\varepsilon/2}+1}$, and exactly $i-1$ successes. The rationale is as follows. To compute $\Pr[E_i]$, we observe that the original 1 bit is kept with probability $\frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1}$ by Ψ_{RAP}. Among the $|\mathcal{U}|-1$ bits that were initially 0, the probability of $i-1$ of them being flipped to 1 by Ψ_{RAP} can be modeled by a Binomial distribution that is reflected in the rightmost Binomial entry of Equation 35. To compute $\Pr[v_\ell = v_\ell^p | E_i]$, we observe that the Bayes-optimal prediction strategy for $\mathcal{A}$ is random guess among the indexes in B that contain the 1 bit. By definition of E_i, the original 1 bit is kept, and additionally $i-1$ bits that were originally 0 were flipped to 1; thus there are a total of i 1 bits. The success probability of $\mathcal{A}$'s guess is therefore $\frac{1}{i}$. Combining all, we arrive at Equation 35.

Total $\mathbb{E}[ASR]$ under RAPPOR: To arrive at RAPPOR's expected ASR, we plug Equation 33 and 35 into Equation 31 to sum up individual ASRs from all individual events. The final result is:

$$\mathbb{E}[ASR] = \frac{1}{|\mathcal{U}|(e^{\varepsilon/2}+1)} \cdot \left(\frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1}\right)^{|\mathcal{U}|-1} + \sum_{i=1}^{|\mathcal{U}|} \frac{e^{\varepsilon/2}}{(e^{\varepsilon/2}+1)i} \cdot \text{Bin}\left(i-1; |\mathcal{U}|-1, \frac{1}{e^{\varepsilon/2}+1}\right) \quad (36)$$

F. Applying $\mathcal{A}$ to OUE

While OUE uses the same bitvector encoding as unary RAPPOR, its perturbation algorithm Ψ_{OUE} behaves differently. In particular, for any bit that is originally 1, Ψ_{OUE} keeps or flips the bit with equal probability $\frac{1}{2}$. In contrast, for any bit that is originally 0, it is kept with high probability $\frac{e^{\varepsilon}}{e^{\varepsilon}+1}$ and flipped with smaller probability $\frac{1}{e^{\varepsilon}+1}$. For this Ψ_{OUE}, we establish that for any index j, if $B'_\ell[j] = 0$ is observed by $\mathcal{A}$, then in the original bitvector B_ℓ, the probability that $B_\ell[j] = 0$ is higher than the probability that $B_\ell[j] = 1$. Consequently, the Bayes-optimal prediction strategy for $\mathcal{A}$ is picking among the indexes j in B'_ℓ that have: $B'_\ell[j] = 1$.

Similar to RAPPOR, we define a set of pairwise disjoint events $E_0, E_1, \ldots, E_{|\mathcal{U}|}$ as follows. Event E_0 is the case where the original 1 bit in the client's bitvector is flipped to 0 in B'_ℓ. For all remaining events E_i where $i \in [1, |\mathcal{U}|]$, event E_i is the case where the original 1 bit is kept and $i-1$ other bits that were 0 in the client's original bitvector became 1 in B'_ℓ. Then:

$$\mathbb{E}[ASR] = \sum_{i=0}^{|\mathcal{U}|} \Pr[v_\ell = v_\ell^p \wedge E_i] \quad (37)$$

$\mathbb{E}[ASR]$ with Event E_0:

$$\Pr[v_\ell = v_\ell^p \wedge E_0] = \Pr[v_\ell = v_\ell^p | E_0] \cdot \Pr[E_0] \quad (38)$$

$$= \frac{1}{|\mathcal{U}|} \cdot \frac{1}{2} \cdot \left(\frac{e^{\varepsilon}}{e^{\varepsilon}+1}\right)^{|\mathcal{U}|-1} \quad (39)$$

The rationale is as follows. By definition of Ψ_{OUE}, we have: $\Pr[E_0] = \frac{1}{2}$. To compute $\Pr[v_\ell = v_\ell^p | E_0]$, we observe that given the original bit was flipped to 0, if any other bit is flipped to 1, $\mathcal{A}$'s optimal strategy is to pick among the indexes with 1 bit. Thus, the only way for $\mathcal{A}$ to predict $v_\ell^p = v_\ell$ is if no original 0 bits are flipped to 1, which happens with probability $\left(\frac{e^{\varepsilon}}{e^{\varepsilon}+1}\right)^{|\mathcal{U}|-1}$, and under that scenario, the probability that $\mathcal{A}$'s prediction is successful is $\frac{1}{|\mathcal{U}|}$.

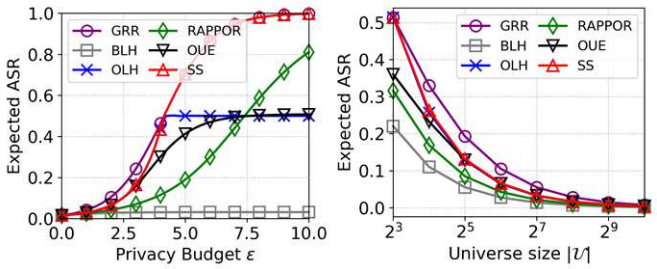


Fig. 1. Expected ASR of $\mathcal{A}$ under all six protocols. On the left, we fix $|\mathcal{U}| = 64$ and vary ε. On the right, we fix $\varepsilon = 2$ and vary $|\mathcal{U}|$.

$\mathbb{E}[ASR]$ with Event E_i where $1 \le i \le |\mathcal{U}|$:

$$\Pr[v_\ell = v_\ell^p \wedge E_i] = \Pr[v_\ell = v_\ell^p | E_i] \cdot \Pr[E_i] \quad (40)$$

$$= \frac{1}{i} \cdot \frac{1}{2} \cdot \text{Bin}\left(i-1; |\mathcal{U}|-1, \frac{1}{e^{\varepsilon}+1}\right) \quad (41)$$

The rationale is as follows. To compute $\Pr[E_i]$, we observe that the original 1 bit is kept with probability $\frac{1}{2}$ by Ψ_{OUE}. Among the $|\mathcal{U}|-1$ bits that were initially 0, the probability of $i-1$ of them being flipped to 1 by Ψ_{OUE} can be modeled by a Binomial distribution with $|\mathcal{U}|-1$ trials, success probability $\frac{1}{e^{\varepsilon}+1}$, and exactly $i-1$ successes. Having established that $\mathcal{A}$'s Bayes-optimal prediction strategy is picking among the indexes that contain the 1 bit, to compute $\Pr[v_\ell = v_\ell^p | E_i]$, we observe that the original 1 bit is kept, and additionally $i-1$ bits that were originally 0 were flipped to 1; thus there are a total of i 1 bits. The success probability of $\mathcal{A}$'s guess is therefore $\frac{1}{i}$. Combining all, we arrive at Equation 41.

Total $\mathbb{E}[ASR]$ under OUE: To arrive at OUE's expected ASR, we plug Equation 39 and 41 into Equation 37 to sum up individual ASRs from all individual events. The final result is:

$$\mathbb{E}[ASR] = \frac{1}{2|\mathcal{U}|} \cdot \left(\frac{e^{\varepsilon}}{e^{\varepsilon}+1}\right)^{|\mathcal{U}|-1} + \sum_{i=1}^{|\mathcal{U}|} \frac{1}{2i} \cdot \text{Bin}\left(i-1; |\mathcal{U}|-1, \frac{1}{e^{\varepsilon}+1}\right) \quad (42)$$

While the analysis of OUE may seem similar to RAPPOR, the final result is semantically different. A key difference between OUE's final result in Equation 42 and RAPPOR's final result in Equation 36 is that the latter contains the term $\frac{e^{\varepsilon/2}}{(e^{\varepsilon/2}+1)i}$ whereas the prior contains the term $\frac{1}{2i}$ in its place. Ignoring i in both terms, as ε increases and approaches $+\infty$, RAPPOR's term increases and approaches 1. However, OUE's term is always upper bounded by the constant $\frac{1}{2}$ regardless of how large ε is. We will observe in our analysis and empirical evaluation that this difference causes ASR under RAPPOR to increase and approach 1 as ε becomes larger, but ASR under OUE increases very slowly after a certain ε.

G. Applying $\mathcal{A}$ to SS

In SS, each client ℓ builds and sends a subset Z_ℓ with size $k = |Z_\ell|$ to the server. First, recall that Ψ_{SS} adds v_ℓ to Z_ℓ with probability $\frac{k \cdot e^\varepsilon}{k \cdot e^\varepsilon + |\mathcal{U}| - k}$. Second, observe that a fake value $v \in \mathcal{U} \setminus \{v_\ell\}$ is added to Z_ℓ by Ψ_{SS} with probability:

$$\frac{ke^\varepsilon}{ke^\varepsilon + |\mathcal{U}| - k} \cdot \frac{k-1}{|\mathcal{U}| - 1} + \frac{|\mathcal{U}| - k}{ke^\varepsilon + |\mathcal{U}| - k} \cdot \frac{k}{|\mathcal{U}| - 1} \tag{43}$$

For this to be larger than the probability that v_ℓ is added to Z_ℓ, the following must be true:

$$\frac{k^2 e^\varepsilon - ke^\varepsilon + k|\mathcal{U}| - k^2}{(ke^\varepsilon + |\mathcal{U}| - k) \cdot (|\mathcal{U}| - 1)} > \frac{ke^\varepsilon}{ke^\varepsilon + |\mathcal{U}| - k} \tag{44}$$

However, if we solve Equation 44, we arrive at: $k > |\mathcal{U}|$ which yields a contradiction because k is the size of a *subset* of $\mathcal{U}$, thus k cannot exceed $|\mathcal{U}|$. Therefore, we find that it is not possible for any fake value's probability of being added to Z_ℓ to be higher than the probability that v_ℓ is added to Z_ℓ. Then, the prediction strategy of $\mathcal{A}$ has to be to predict v_ℓ^p from reported Z_ℓ, since the probability of v_ℓ being included in Z_ℓ is highest. $\mathcal{A}$'s prediction strategy is therefore: $v_\ell^p \leftarrow_{\$} Z_\ell$. Expected ASR under SS can be derived as:

$$\mathbb{E}[ASR] = \Pr[v_\ell = v_\ell^p] = \Pr[v_\ell = v_\ell^p | v_\ell \in Z_\ell] \cdot \Pr[v_\ell \in Z_\ell] \tag{45}$$

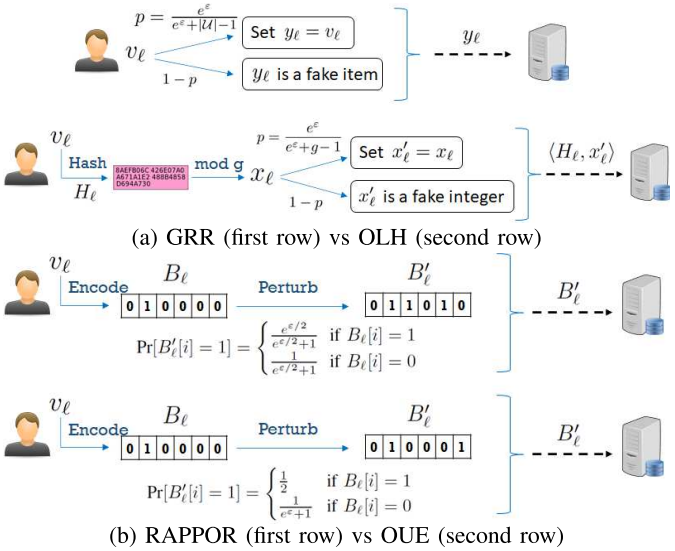
$$= \frac{ke^\varepsilon}{ke^\varepsilon + |\mathcal{U}| - k} \cdot \frac{1}{k} = \frac{e^\varepsilon}{ke^\varepsilon + |\mathcal{U}| - k} \tag{46}$$

which completes the derivation. Note that expected ASR under SS found in Equation 46 is also dependent on the protocol subset size parameter k. If the default value of $k = \frac{|\mathcal{U}|}{e^\varepsilon + 1}$ is plugged in to Equation 46, we get: $\frac{e^\varepsilon + 1}{2|\mathcal{U}|}$ as the expected ASR. This confirms the intuitive expectations that: (i) when ε increases, ASR will increase since privacy becomes more relaxed; and (ii) when $|\mathcal{U}|$ increases, ASR will decrease since it is more difficult for $\mathcal{A}$ to predict v_ℓ correctly from a larger domain.

H. Summary and Analysis of Expected ASR

Having derived the expected ASR of $\mathcal{A}$ under each protocol, we now perform a side-by-side comparison of protocols' expected ASR to analyze their behavior with varying ε and $\mathcal{U}$. Note that ε and $\mathcal{U}$ are the two common factors that appear in the expected ASR equations of all protocols. In Figure 1, we plot the expected ASRs of each protocol: on the left, we fix $|\mathcal{U}| = 64$ and vary ε; on the right, we fix $\varepsilon = 2$ and vary $|\mathcal{U}|$. The graphs show some general trends that apply to all LDP protocols, which agree with intuitive expectations. In particular, with increasing ε, the privacy achieved by randomized perturbation decreases, therefore ASR values increase. With a larger domain $\mathcal{U}$, ASR values decrease, which is intuitive given that a larger domain makes $\mathcal{A}$'s predictions more difficult (e.g., the ASR of a completely random guess is $(1/|\mathcal{U}|)$, which becomes smaller as $|\mathcal{U}|$ becomes larger). The results across all six protocols agree on these general trends.

We observe that different protocols may yield substantially different ASR values under the same ε and $\mathcal{U}$. For example, expected ASR under GRR can be several times higher than



(a) GRR (first row) vs OLH (second row)

(b) RAPPOR (first row) vs OUE (second row)

Fig. 2. Examples for highlighting similarities and differences in client-side encoding and perturbation of different LDP protocols.

BLH with the same ε and $\mathcal{U}$. The intuition behind this is further explained with practical examples in Section III-I. Another observation is that one protocol may have higher ASR over another protocol under certain ε and $\mathcal{U}$ values, whereas the opposite may be true under different ε and $\mathcal{U}$ values. For example, comparing OLH and RAPPOR in the first figure, OLH's ASR is higher than RAPPOR's ASR when $2.5 \leq \varepsilon \leq 7$, but RAPPOR's ASR exceeds OLH when $\varepsilon \geq 8$. Another example is that SS has ASR close to OUE for small ε, but when $\varepsilon \geq 5$, SS becomes very different than OUE and more similar to GRR. These observations are partly caused by OLH and OUE's ASR being heavily impacted by the $\frac{1}{2}$ terms in their expected ASR, as discussed at the end of Section III-D and III-F respectively. Altogether, the observations show that the protocol which yields lower ASR may change across different settings; thus, the protocol recommended to minimize ASR in one setting can be different than the recommended protocol in another setting.

I. Why Do Protocols Have Different ASR?

An important observation from the expected ASR analysis, which will also be confirmed by empirical ASR analysis in the next section, is that different protocols can have different ASR under the same ε. This may seem counter-intuitive at first, considering that the level of privacy is typically controlled by the ε parameter in the context of LDP. However, there is an intuition behind it. While all protocols satisfy ε-LDP, the individual steps they perform to do so (i.e., their encoding and perturbation steps) are different. Consequently, when one considers a practical Bayesian adversary, different protocols can yield different ASR, depending on the combined effect of their encoding and perturbation. Past research has showed that different protocols can have different estimation utility under the same ε [1], [6]; our paper shows that a similar behaviour can be observed from an adversarial perspective. We explain this phenomenon with practical examples below.

1) Example 1: Recall GRR vs OLH protocols from Section II-B. GRR uses direct encoding (i.e., $\text{ENCODE}_{\text{GRR}}(v_\ell) = v_\ell$) and then perturbs v_ℓ probabilistically. In contrast, OLH encodes v_ℓ into an integer x_ℓ using a hash function H_ℓ (i.e., $\text{ENCODE}_{\text{OLH}}(v_\ell) = \langle H_\ell, x_\ell \rangle$) and then perturbs x_ℓ. These two processes are illustrated in Figure 2a. For an adversary receiving OLH and GRR's outputs, even if there is little or no perturbation in both cases, it is more difficult to infer v_ℓ from OLH's output since OLH makes use of hash functions which are many-to-one and non-invertible. This usage of a many-to-one function for encoding yields an additional privacy benefit for OLH which does not exist in GRR. Comparing the expected ASR results of GRR vs OLH in Figure 1, we observe that this is reflected in the fact that GRR's ASR is much higher than OLH's ASR when ε is large (perturbation is small).

2) Example 2: Consider RAPPOR vs OUE protocols. Both protocols use bitvector encoding to construct B_ℓ as shown in Figure 2b. However, the perturbation probabilities are different, as indicated in the figure and in Equations 8 and 10. We showed that the Bayes-optimal strategy for $\mathcal{A}$ is to pick among indexes j in B'_ℓ that have: $B'_\ell[j] = 1$ in both protocols. Yet, using Equations 8 and 10 it can be derived that the total number of indexes that satisfy this condition are different in RAPPOR and OUE – in expectation, it is $\frac{e^{\varepsilon/2}+|\mathcal{U}|-1}{e^{\varepsilon/2}+1}$ in RAPPOR and $\frac{e^{\varepsilon}+2|\mathcal{U}|-1}{2(e^{\varepsilon}+1)}$ in OUE. Since the probability of $\mathcal{A}$'s correct prediction is inversely proportional to these quantities and since they are different, $\mathcal{A}$'s expected ASR is also different for RAPPOR and OUE. This can be verified from Figure 1.

IV. Experimental Analysis

A. Experiment Setup and Datasets

We conduct experiments with the six LDP protocols under consideration (GRR, BLH, OLH, RAPPOR, OUE, SS) and four datasets: MSNBC, Kosarak, Uniform, and Exponential. All code is implemented in Python. Experiments are repeated 10 times and results are averaged.

MSNBC contains logs from msnbc.com for the day of September 28, 1999 where each row in the dataset corresponds to one user's page visit sequence.[1] Visits are recorded at the granularity of page category (news, tech, weather, sports, etc.). There are $|\mathcal{U}| = 17$ categories and $|\mathcal{L}| = 989,818$ users. A large number of users either visit one category of pages, or for users that visit multiple categories, their visits are often dominated by one category. Hence, for each user we determine the most visited category and assume the user's true value is equal to that category.

Kosarak contains click stream data from a Hungarian online news portal.[2] Each user is associated with a set of clicked URL IDs. There are around one million users and 41,270 unique URL IDs. Since many URLs are visited few times (e.g., once or twice), we pre-processed the dataset by identifying the top-128 most visited URLs across the whole dataset and removing the remaining URLs from each user's click stream. We discarded those users whose resulting click stream was empty. For users who had more than one URL in their resulting stream, we randomly picked a URL as their v_ℓ. In the resulting dataset, we had $|\mathcal{U}| = 128$ and $|\mathcal{L}| = 929,669$.

Uniform: We create synthetic client populations consisting of $|\mathcal{L}|$ clients whose true data, when aggregated, corresponds to a uniform distribution across domain $\mathcal{U}$. Unless otherwise noted, default values are $|\mathcal{L}| = 100,000$ and $|\mathcal{U}| = 40$.

Exponential: We use an Exponential distribution to generate client data, with probability density function $f(x;\beta) = \beta^{-1}e^{-x/\beta}$ where β is the scale parameter. Lower β means the distribution is more skewed, i.e., denser in the characteristic peak of the Exponential. The default value of β is $\beta = 3$, but when measuring the impacts of data skewness, we create different client populations with varying skewness by changing the β parameter. Each client's true value is a data point sampled from the corresponding Exponential and rounded to the nearest integer. By default, $|\mathcal{L}| = 100,000$ and $|\mathcal{U}| = 50$.

B. Comparison Between Protocols

Here, we assume that each protocol uses their default parameters (as noted in Section II-B) and compare them to see how ASR changes under different protocols, datasets, and ε. Results are given in Figure 3. We first observe that these results agree with the theoretical derivations from Section III. It is often the case that GRR's ASR is higher than remaining protocols. SS protocol has lower ASR than GRR for small ε, but its ASR becomes very close to GRR when ε is larger. BLH often has lower ASR due to many hash collisions (on average $|\mathcal{U}|/2$ collisions). Among the remaining protocols, RAPPOR has lower ASR than OLH and OUE when ε is small, but RAPPOR's ASR exceeds OLH and OUE when ε is large. The ε value at which RAPPOR's ASR starts exceeding OLH and OUE changes from dataset to dataset due to different $|\mathcal{U}|$ in different datasets. In the MSNBC dataset in which $|\mathcal{U}|$ is small, this ε value is between 4 and 6. In the Uniform and Exponential datasets in which $|\mathcal{U}|$ is relatively larger, this ε value is between 6 and 8. In the Kosarak dataset in which $|\mathcal{U}|$ is largest, this ε value is between 8 and 10. A key factor why RAPPOR's ASR exceeds OLH and OUE when ε is large is that OLH and OUE's ASR increase very slowly after they hit 0.5. This was observed as part of the theoretical expected ASR derivations, as well as Figure 1. The experimental results in Figure 3 validate the theoretical finding.

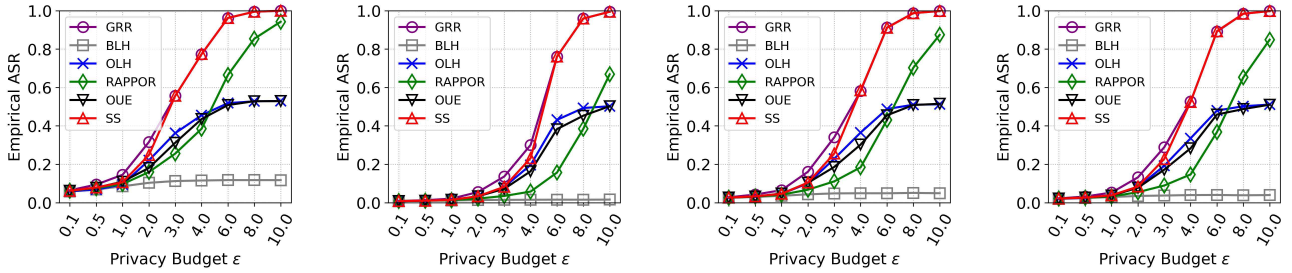
Figure 3 also shows that OLH and OUE's empirical ASR are similar for various values of ε although they differ slightly between $2 \leq \varepsilon \leq 6$. This is also true for their expected ASR, as illustrated in Figure 1. An interesting observation here is that OLH and OUE, which use very different encoding styles (g-ary hash versus unary bitvector encoding), can have similar ASR; as opposed to two protocols which use similar encoding principle having substantially different ASR behavior (such as BLH and OLH that both use hash encoding, or RAPPOR and OUE that both use bitvector encoding).

C. Impact of Encoding Parameters

Recall that some protocols have encoding-related parameters, e.g., OLH has parameter g which determines the

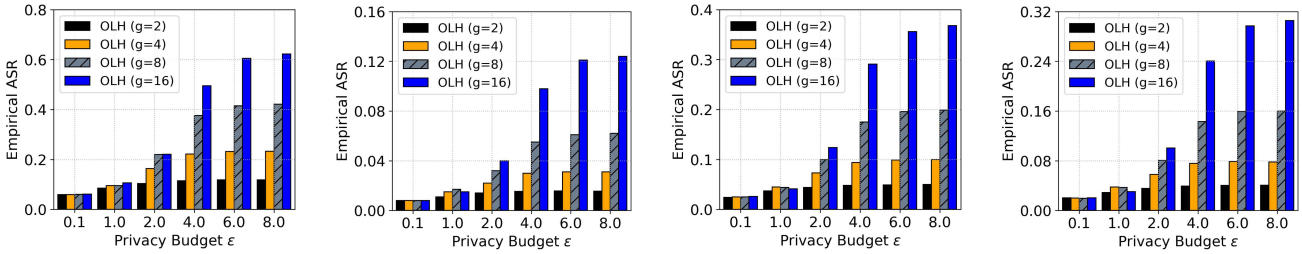
[1] UCI Machine Learning Repository: MSNBC.com Anonymous Web Data Set. http://archive.ics.uci.edu/ml/datasets/msnbc.com+anonymous+web+data

[2] Frequent Itemset Mining Dataset Repository. http://fimi.uantwerpen.be/data/



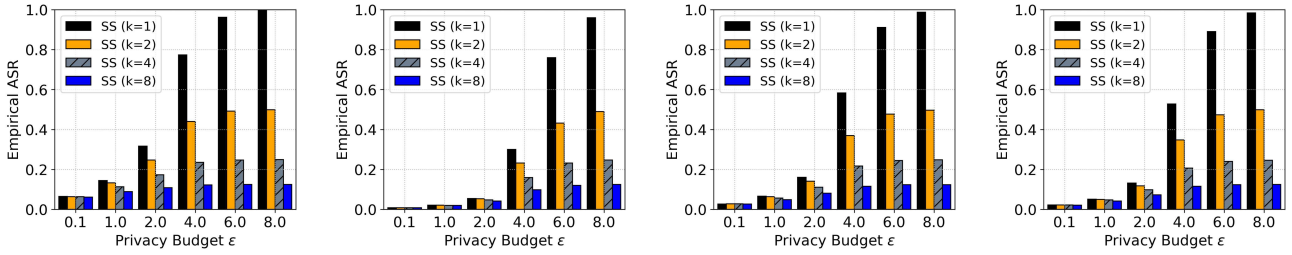
(a) MSNBC dataset (b) Kosarak dataset (c) Uniform dataset (d) Exponential dataset (β=3)

Fig. 3. Empirical ASR of $\mathcal{A}$ under all six protocols. While individual ASR values change from one dataset to another, overall trends agree that: (i) ASR increases as ε is increased, (ii) some protocols have lower ASR than others for low values of ε but the opposite is true for higher values of ε. These empirical results also agree with the trends observed in the theoretical analysis in Sec. III and Fig. 1.



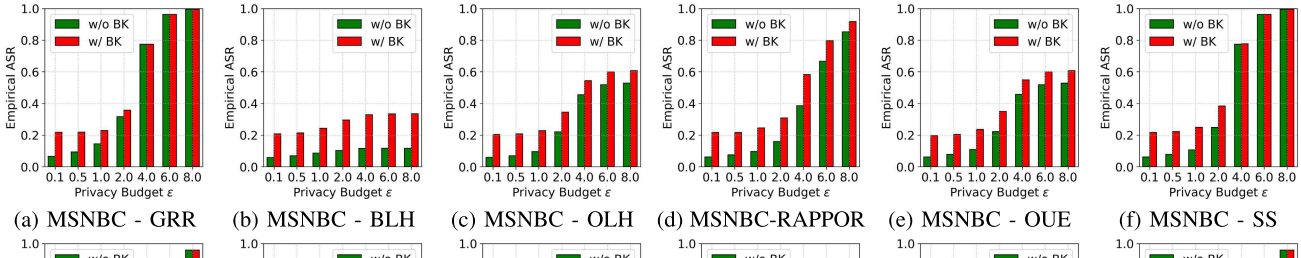
(a) MSNBC dataset (b) Kosarak dataset (c) Uniform dataset (d) Exponential dataset (β=3)

Fig. 4. Adversarial Success Rate (ASR) changes substantially when OLH protocol's encoding parameter g is varied.



(a) MSNBC dataset (b) Kosarak dataset (c) Uniform dataset (d) Exponential dataset (β=3)

Fig. 5. Adversarial Success Rate (ASR) changes substantially when SS protocol's subset size parameter k is varied.



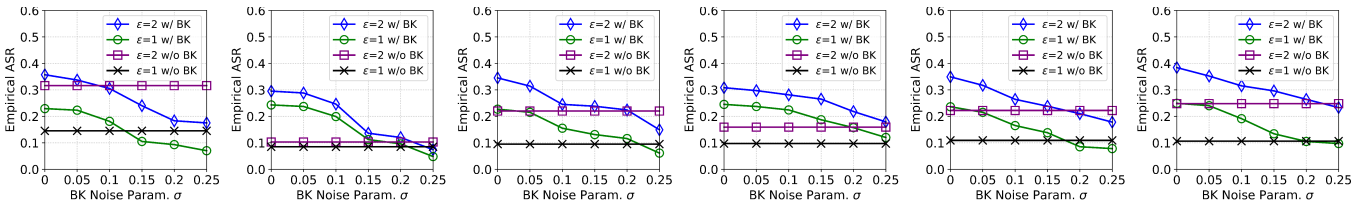
(a) MSNBC - GRR (b) MSNBC - BLH (c) MSNBC - OLH (d) MSNBC-RAPPOR (e) MSNBC - OUE (f) MSNBC - SS

(g) Kosarak - GRR (h) Kosarak - BLH (i) Kosarak - OLH (j) Kosarak - RAPPOR (k) Kosarak - OUE (l) Kosarak - SS

Fig. 6. Comparison of adversary with and without background knowledge (w/ BK versus w/o BK). Experiments using two real datasets and six protocols agree that adversarial knowledge of aggregate statistical data distribution improves ASR.

output space of hash encoding and SS has parameter k which determines the reported subset size. Here, we study the impact of changing g and k parameters on OLH and SS.

In Figure 4, we show how empirical ASR changes under the OLH protocol when its g parameter is varied between 2 and 16. While the differences between varying g values



(a) GRR (b) BLH (c) OLH (d) RAPPOR (e) OUE (f) SS

Fig. 7. Analyzing the impact of noise in adversarial BK on ASR. The noise parameter σ on x axis controls the amount of noise. As expected, adversary w/o BK is not impacted by σ. For adversary w/BK, lower σ leads to higher ASR since adversary's BK remains more accurate (i.e., closer to ground truth).

is small for small ε budgets, for larger ε budgets such as $\varepsilon \geq 2$, there starts to be large discrepancies between the ASR values. For $\varepsilon = 8$, the empirical ASR of OLH with $g = 16$ is often roughly 3-4 fold higher than ASR of OLH with $g = 2$. The rationale behind observing different ASR under the same ε but different g is as follows. In OLH, the expected number of hash collisions due to g-ary hash encoding is $|\mathcal{U}|/g$. When g is smaller, more collisions are expected. Increased number of collisions provide an added amount of privacy (i.e., added on top of the privacy already achieved by ε-LDP perturbation), thanks to the irreversibility of hash collisions. Even in the extreme case where $\varepsilon = +\infty$ in which no privacy is given by ε-LDP perturbation, hash collisions will still cause $\mathcal{A}$'s prediction accuracy and ASR to go down, yielding *some* privacy to clients. This supports the observation that especially for large ε such as 8, ASR remains low when g is small (many collisions) compared to $g = 16$ (fewer collisions).

In Figure 5, we perform a similar experiment for the SS protocol by varying its k parameter between 1 and 8. The results for SS protocol show that when ε is large, varying k values may cause 4-5 fold difference in ASR. In particular, for $\varepsilon \geq 4$, the ASR of SS with $k = 1$ is several folds larger than the ASR of SS with $k = 8$. This can be intuitively explained through a crowd-blending privacy perspective as follows. In SS, v_ℓ blends in a crowd of k items (reported subset). When k is small, v_ℓ does not have a large crowd to blend in, making the adversary's prediction easier. In contrast, when k is large, the crowd is larger therefore ASR can remain low due to the existence of a large number of $k-1$ fake reported items.

D. Impact of Adversary Background Knowledge

Recall from Section III-A that, for the adversary with BK, BK is injected into Bayesian inference via term $\Pr[\hat{v}]$ in Equation 15. In this section, we perform two experiments. In the first experiment, we compare ASR of adversary with BK (w/ BK) and without BK (w/o BK). For adversary w/o BK, we assume that no injection takes place. For adversary w/ BK, we assume that BK is equal to the ground truth exact distribution, i.e., adversary injects $\Pr[\hat{v}]$ as:

$$\Pr[\hat{v}] = \frac{C(\hat{v})}{\sum_{v \in \mathcal{U}} C(v)} \tag{47}$$

Results of this experiment are given in Figure 6 for the two real datasets (Kosarak and MSNBC) and for all six protocols. Comparing the adversary w/ BK and w/o BK under each scenario in which other factors (such as ε, dataset, and protocol) are fixed, we see that while the individual ASR values may differ in different protocols and datasets, all protocols and datasets agree that the adversary w/ BK outperforms the adversary w/o BK in terms of ASR. In GRR and SS, the adversary w/o BK catches up with the adversary w/ BK when ε is large such as 6 or 8. However, in the remaining protocols, there remains a gap between the adversary w/ BK and w/o BK across varying ε. The take-away message is that in many scenarios, auxiliary knowledge and background knowledge may be used by adversaries to increase their prediction success rates under the same ε, dataset and protocol.

In the second experiment, we assume that the adversary's BK is not "perfect", i.e., adversary does not have ground truth knowledge as BK, but rather a noisy version (a "rough distribution"). To simulate such behaviour, we modify adversary's BK by adding a noise term:

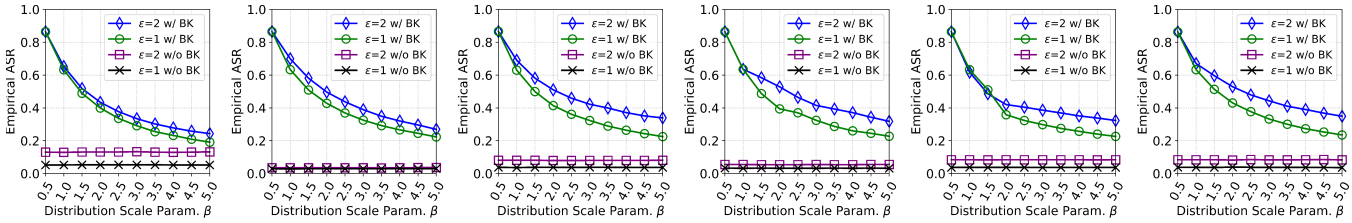
$$\Pr[\hat{v}] = \frac{C(\hat{v})}{\sum_{v \in \mathcal{U}} C(v)} + \mathcal{G}(0, \sigma) \tag{48}$$

where $\mathcal{G}(0, \sigma)$ denotes Gaussian noise with mean 0 and standard deviation σ. When $\sigma = 0$, adversary's BK is perfect, i.e., equal to ground truth. As σ increases, adversary's BK becomes more noisy.

We conduct this experiment with all six protocols and using the MSNBC dataset. Results are given in Figure 7. As expected, adversary w/o BK is not impacted by σ. However, for adversary w/ BK, higher σ causes ASR to decrease. For small σ such as $\sigma = 0.05$ or 0.1, ASR of adversary w/ BK remains substantially higher than ASR of adversary w/o BK. This shows that although the adversary may not have the fully correct BK, even with a roughly correct BK, the adversary can still achieve higher ASR. However, interestingly, when σ is high (such as $\sigma = 0.2$ or 0.25) the ASR of adversary w/ BK can fall below the ASR of adversary w/o BK. It is worth noting that such high values of σ cause BK to be extremely noisy (largely incorrect or almost opposite the ground truth), since the original $\Pr[\hat{v}]$ values are between 0 and 1. This shows that extremely noisy BK may cause more harm than good for the adversary, i.e., adversary would be better off not using BK if his/her BK is extremely noisy.

E. Impact of Data Distribution and Skewness

We now study the impact of data skewness, i.e., statistical distribution of client data. If some values $v \in \mathcal{U}$ are frequently



(a) GRR (b) BLH (c) OLH (d) RAPPOR (e) OUE (f) SS

Fig. 8. Analyzing the impact of data skewness on ASR, for adversary with and without BK. The distribution scale parameter β on x axis controls data skewness, where smaller β indicates skewness is higher. Results show that adversary w/o BK is not impacted by change in skewness, but adversary w/ BK enjoys substantially higher ASR on more skewed datasets.

observed in the client population whereas other values are rarely observed, we say that data distribution is skewed. If the observation frequency of all values in $\mathcal{U}$ is roughly even, we say that the data is uniform (i.e., less skewed or not skewed). We create synthetic client populations with varying data skewness by changing the distribution scale parameter of the Exponential dataset as explained in Section IV-A. We measure ASR under each dataset with two scenarios: adversary w/ BK and adversary w/o BK.

The results are given in Figure 8. We first observe that varying data skewness does not impact ASR when the adversary does not have BK. This is confirmed by roughly constant ASR values for the adversary w/o BK while skewness is varied, in all six protocols and $\varepsilon = 1$ and 2. However, for the adversary w/ BK, ASR values change substantially with varying skewness. In cases where there is large data skewness (leftmost end of each graph in Figure 8), ASR is much higher than cases that are not skewed (rightmost end of each graph in Figure 8). Therefore, the take-away message is that skewness plays a role in ASR when the adversary has background knowledge of the skewed statistics; otherwise, it does not impact ASR.

V. LDPLens

A. Motivation

We established both mathematically and empirically that under the common Bayesian adversary, different LDP protocols yield different ASR results according to factors such as: (i) encoding type and parameters, (ii) data domain, (iii) adversarial background knowledge, and (iv) adversarial knowledge of statistical distribution and skewness of data. Furthermore, when these factors are set in a certain way, one protocol may yield lower ASR than the other; whereas for a different setting of the factors, the other protocol may yield lower ASR and therefore become more desirable. For example, Fig. 1 and 3 agree that when $\varepsilon = 4$, RAPPOR has lower ASR compared to OUE; whereas when $\varepsilon = 8$, OUE has lower ASR compared to RAPPOR. This latter property that protocols have interchanging desirability under different settings was known from the perspective of minimizing estimation error [1], [6], [12], [13]; our results in this paper demonstrate a similar property from the perspective of ASR.

The observation that no single protocol yields lowest ASR and highest estimation utility simultaneously across all possible factors motivates the problems of *protocol selection* and *budget selection*, i.e.: Given the setting of the related factors, a set of LDP protocols and an adversarial resilience constraint [or a utility loss constraint]; which LDP protocol is most desirable and what value of ε should be used?

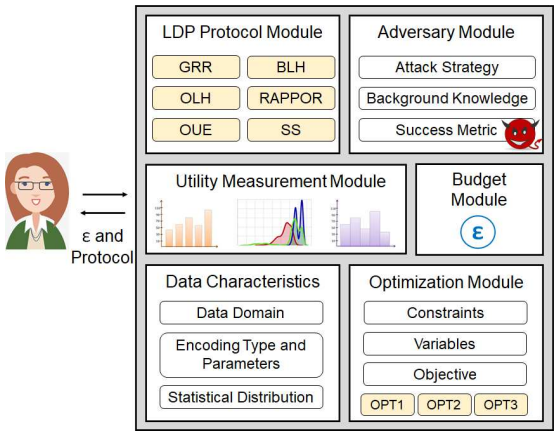


Fig. 9. Overview of LDPLens design.

We design LDPLens to address the protocol and budget selection problems as well as to assist researchers and application designers in analyzing the privacy-utility tradeoffs of LDP protocols, thereby allowing them to make informed decisions regarding protocol and budget selection. Consider the following use of LDPLens. Let Alice be a researcher who wants to design an LDP system for a niche application in her area such as healthcare, IoT or cybersecurity. Alice knows the syntax, domain $\mathcal{U}$, statistical distribution, and suitable encoding types and parameters for her data. Alice also knows the adversary model and/or utility measurement method that are relevant in her application. However, Alice is not an expert in LDP; thus, she wants to learn which protocol and ε value she should use. LDPLens enables Alice to specify a utility loss constraint (or adversarial resilience constraint) that is appropriate for her application, and then, Alice can learn the desirable protocol and suitable ε for her constraint.

B. Design of LDPLens

LDPLens consists of several modules as shown in Figure 9. It is designed to be modular, customizable and extensible. While each module comes with default settings and implementations in place, it is possible to add new LDP protocols, adversary models, utility measurement metrics, optimization

problems and so forth by implementing them under the corresponding modules.

1) LDP Protocol Module: This module contains the implementations of LDP protocols, e.g.: GRR, BLH, OLH, RAPPOR, OUE, SS. A new protocol named PROT can be added to LDPLens by implementing two functions in this module: one for client-side encoding and perturbation Ψ_{PROT}, and one for server-side estimation.

2) Adversary Module: This module consists of 3 parts: (i) the attack strategy, (ii) background knowledge of the adversary, and (iii) the metric or measurement method to evaluate the success of the adversary. Our Bayesian adversary and ASR metric from Section III-A constitute one option that is readily implemented in this module. New adversary models and metrics may be added to LDPLens by implementing them in this module.

3) Utility Measurement Module: Since LDP relies on randomized perturbation, LDP protocols often cause utility loss which manifests itself in ways such as frequency estimation error, accuracy degradation of a machine learning model, etc. Depending on the application area, the relevant metric to measure utility loss can be different. Thus, the Utility Measurement Module of LDPLens enables users to choose their desired metric among available ones or to define a new metric to measure utility loss.

One metric that is readily implemented in LDPLens, which we also use in our case studies in Section V-D, is L_p-norm frequency estimation error. Recall from Section II-A that $|\mathcal{L}|$ denotes client population size, $C(v)$ denotes the true count of v, and $\bar{C}(v)$ denotes the estimated count of v. Then, v's true frequency is $f(v) = C(v)/|\mathcal{L}|$ and its estimated frequency is $\bar{f}(v) = \bar{C}(v)/|\mathcal{L}|$. L_p-norm error is measured as:

$$\frac{\sum_{v \in \mathcal{U}} \left(|f(v) - \bar{f}(v)| \right)^p}{|\mathcal{U}|} \tag{49}$$

In our case studies, we use this metric with $p = 1$. This naturally yields to the problem of *frequency estimation*, which has received significant attention as a fundamental task in LDP and a common utility measurement method for LDP protocols in the literature [6], [11], [17], [26], [27]. Our usage with $p = 1$ measures frequency estimation error in terms of Mean Absolute Error (MAE). Different p can yield different measurements, e.g., $p = 2$ yields Mean Squared Error (MSE). Thus, L_p-norm is quite versatile. Other utility measurement methods, such as domain-specific ones, can also be added to LDPLens by implementing them in the Utility Measurement Module. For example, in the set-valued data collection domain, metrics such as Kendall-Tau (KT) and Normalized Cumulative Rank (NCR) can be applied in addition to L_p-norm [12], [17].

4) Budget Module: This module serves to enable the specification of the allowed range for budget parameter ε. By definition, $\varepsilon > 0$ is a positive float. The current implementation of LDPLens uses the range: $0.01 \leq \varepsilon \leq 16$ by default, to allow for a wide range of possible values.

5) Data Characteristics Module: This module consists of 3 parts: (i) data domain $\mathcal{U}$, (ii) suitable data encoding type and encoding parameters (e.g., bitvector or hash), (iii) statistical distribution of data. Data domain and statistical distribution can be learned either from a data sample provided by the LDPLens user or direct manual specification into LDPLens.

6) Optimization Module: The protocol and budget selection problems aim to find the *most desirable* protocol and ε value to use under a given scenario. To achieve this, the Optimization Module of LDPLens is designed such that the protocol and budget selection problems can be mathematically formulated as optimization problems. An optimization problem consists of 3 parts: constraints, variables, and the optimization objective. There currently exist 3 built-in optimization problems in LDPLens (OPT1, OPT2, OPT3) that differ in terms of their constraints, variables, and objectives. Their details will be given in the next section. The user can choose from one of these problems or formulate a new optimization problem by specifying a new set of constraints, variables, and objective. Then, the optimization problem is solved and the result of the optimization (protocol and ε) are output to the end user.

C. Optimization Problems

The optimization module of LDPLens comes with 3 built-in optimization problems, all of which solve the protocol and budget selection problems simultaneously. The two important factors in the formulation of the optimization problems is how adversarial success is measured (which is specified in LDPLens using the *Adversary Module*) and how utility loss is measured (which is specified using the *Utility Measurement Module*). While these are user-specified factors, without loss of generality, we denote by ASR the metric for adversarial success and by UL the metric for utility loss.

1) Utility-Constrained Adversarial Optimization (OPT1): The user provides a utility loss constraint γ, which is the maximum utility loss that can be tolerated. Objective is to find the protocol and ε value such that the success rate of the adversary is minimized while the provided utility loss constraint is met.

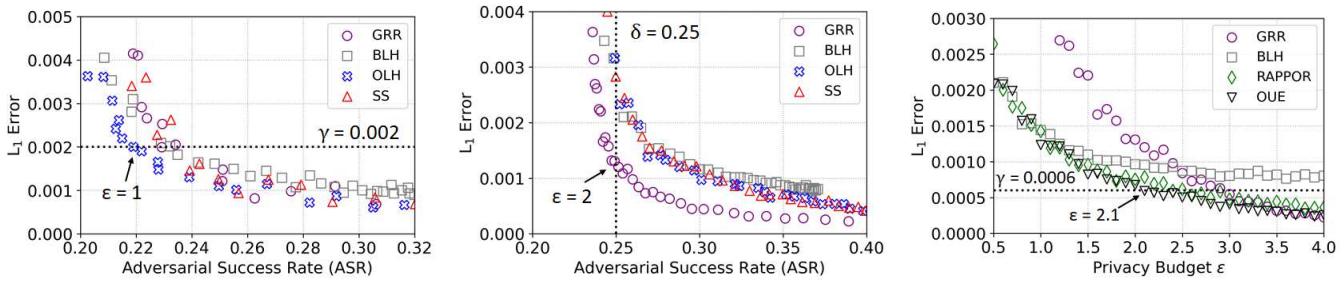
$$\begin{aligned} \underset{(\varepsilon,\ \text{PROT})}{\arg\min} \quad & ASR \\ \text{subject to} \quad & UL \leq \gamma \end{aligned}$$

2) Adversarially-Constrained Utility Optimization (OPT2): The user provides an adversarial success rate constraint δ, which is the maximum success rate that can be tolerated. Objective is to find the protocol and ε value such that utility loss is minimized while the provided adversarial success rate constraint is met.

$$\begin{aligned} \underset{(\varepsilon,\ \text{PROT})}{\arg\min} \quad & UL \\ \text{subject to} \quad & ASR \leq \delta \end{aligned}$$

3) Utility-Constrained Budget Optimization (OPT3): The user provides a utility loss constraint similar to OPT1. Objective is to find the protocol that allows the minimum ε to be used while the provided utility loss constraint is met.

$$\begin{aligned} \underset{\text{PROT}}{\arg\min} \quad & \varepsilon \\ \text{subject to} \quad & UL \leq \gamma \end{aligned}$$



(a) MSNBC, OPT1 (b) Kosarak, OPT2 (c) Kosarak, OPT3

Fig. 10. Demonstration of LDPLens with 3 case studies on real datasets: MSNBC and Kosarak. Results show that: (i) For different datasets and different constraints, the most desirable protocol is also different. (ii) Using the most desirable protocol selected via LDPLens, one can achieve substantial utility loss reduction, ASR reduction or budget savings.

D. Demonstration via Case Studies

We now demonstrate the usefulness of LDPLens via case studies. In our case studies, we use the Bayesian adversary $\mathcal{A}$ and ASR metric from Section III-A for adversarial success measurement, and L_1 norm frequency estimation error from Equation 49 for UL measurement. The same datasets as in Section IV-A are used. Results are given in Figure 10. Each data point in the graphs correspond to a protocol's execution with one value of ε. Different points are plotted for ε ranging between 0.1 and 4.0 in increments of 0.1. Since lower ASR means lower adversarial success and lower L_1 error means higher utility, the bottom-left corner of each graph is most desirable. However, as expected, all protocols show tradeoffs between adversarial success and utility, e.g., as L_1 errors decrease, ASRs increase. Yet, some protocols achieve more favorable tradeoffs than others, which we will discuss.

1) Case Study #1: Consider the comparison between GRR, BLH, OLH and SS protocols in Figure 10a on the MSNBC dataset. Assume that OPT1 is used with constraint $\gamma = 0.002$, which is represented using the horizontal dotted line in the figure. The goal is to select the LDP protocol and ε value that minimizes ASR under this constraint. In this case, LDPLens outputs OLH as the most desirable protocol, since its ASR is lowest compared to other protocols when $\gamma = 0.002$. This can be seen from the figure by observing that OLH's data points are the leftmost ones that intersect with the horizontal dotted line, which means that remaining protocols yield higher ASR than OLH under the same γ constraint. In addition, the recommended value of ε is $\varepsilon \approx 1$ since the data points that intersect with the dotted line satisfy $\varepsilon = 1$. We also observe that the selection of OLH as the most desirable protocol is robust to small changes in the L_1 error constraint, e.g., for all γ values between 0.0015 and 0.003, OLH remains the most desirable protocol.

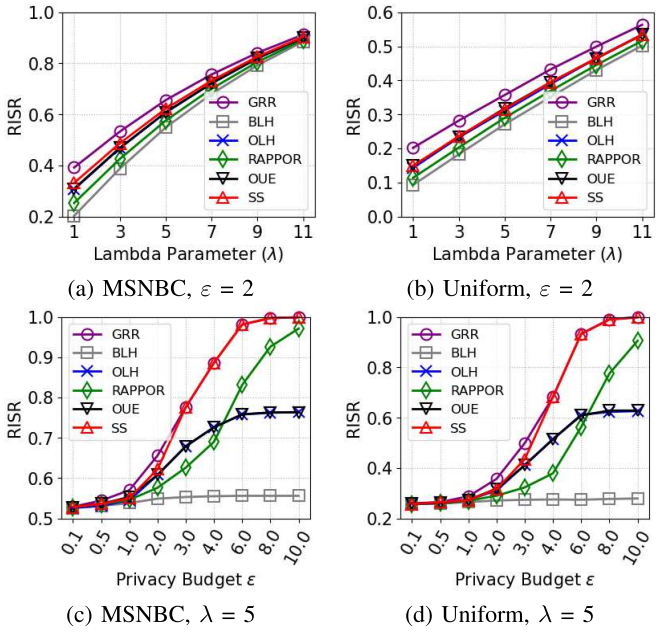
2) Case Study #2: Consider the comparison between GRR, BLH, OLH and SS protocols in Figure 10b on the Kosarak dataset. Assume that OPT2 is used with constraint $\delta = 0.25$, which is represented with the vertical dotted line in the figure. The goal is to select the LDP protocol and ε value that minimizes L_1 error under this constraint. In this case, LDPLens outputs GRR as the most desirable protocol, since its L_1 error is lowest compared to other protocols when $\delta = 0.25$. This can be seen from the figure by observing that GRR's data points are the lowermost ones that intersect with the vertical dotted line. The recommended value of ε here is $\varepsilon \approx 2$. Interestingly, compared to GRR, the L_1 errors of all remaining protocols are 1.5-2x higher. This shows the clear benefit of protocol selection using LDPLens: If one failed to select GRR and used a suboptimal protocol instead, they would have incurred a 1.5-2x higher utility loss.

3) Case Study #3: Consider the comparison between GRR, BLH, RAPPOR and OUE protocols in Figure 10c on the Kosarak dataset. Assume that OPT3 is used with constraint $\gamma = 0.0006$, which is represented with the vertical dotted line in the figure. The goal is to select the LDP protocol that minimizes ε under this constraint. In this case, LDPLens outputs OUE as the most desirable protocol, since it allows the smallest ε to be used while satisfying $\gamma = 0.0006$. For example, OUE can satisfy the γ constraint with $\varepsilon = 2.1$, whereas RAPPOR can do so with $\varepsilon = 2.5$ and GRR with $\varepsilon = 2.9$. By selecting the desirable protocol with LDPLens, we can achieve substantial budget savings.

4) Discussion: First, we observe that the most desirable protocol in each case study is different: in Case Study 1 it is OLH, in Case Study 2 it is GRR, in Case Study 3 it is OUE. This validates the initial motivation behind LDPLens that the most desirable protocol can change from scenario to scenario, and thus finding the most desirable protocol in a new scenario is a non-trivial problem. Second, we observe that the choice of the most desirable protocol can be robust, i.e., even if there are small changes in γ or δ, the most desirable protocol remains the same. Finally, there is substantial benefit in using the protocol selected using LDPLens rather than randomly selecting a protocol. In Case Study 1 it enables %10 ASR reduction, in Case Study 2 it enables 1.5-2 fold utility improvement, in Case Study 3 it enables up to %38 budget savings. These demonstrate the benefit in using LDPLens to identify a protocol that is "good fit" for a given scenario, rather than randomly selecting a protocol.

E. Alternative Adversary Models

We noted earlier that while the Bayesian adversary and ASR metric from Section III-A constitute one option existing in LDPLens, other adversary models can also be supported. Consider an alternative adversary whose goal is not to infer v_ℓ precisely (i.e., $v_\ell^p = v_\ell$), but instead, the adversary is



(a) MSNBC, $\varepsilon = 2$ (b) Uniform, $\varepsilon = 2$

(c) MSNBC, $\lambda = 5$ (d) Uniform, $\lambda = 5$

Fig. 11. RISR comparison of six LDP protocols (GRR, BLH, OLH, RAPPOR, OUE) on MSNBC and Uniform datasets. In the upper row, $\varepsilon = 2$ is fixed, λ parameter is varying. In the bottom row, $\lambda = 5$ is fixed, ε is varying.

successful if he/she infers within a range $v_\ell \pm \lambda$, i.e., $v_\ell - \lambda \leq v_\ell^p \leq v_\ell + \lambda$. We can measure the success rate of this adversary using the *Range Inference Success Rate (RISR)* metric:

$$RISR = \frac{\text{\# of clients } \ell \in \mathcal{L} \text{ such that } v_\ell - \lambda \leq v_\ell^p \leq v_\ell + \lambda}{|\mathcal{L}|}$$

We perform an experimental comparison of the RISR values of six LDP protocols under varying ε and λ. The results are illustrated in Figure 11. As expected, we observe clear positive correlations between RISR and λ, as well as between RISR and ε. Yet, similar to the ASR metric, different protocols can have substantially different RISR values under the same ε and λ. For example, when $\lambda = 5$ and $\varepsilon = 3$ on the MSNBC dataset, RISR of BLH is 0.55 whereas RISR of RAPPOR is 0.63, RISRs of OLH and OUE are nearly 0.68, and RISRs of GRR and SS are nearly 0.78. This shows that protocols' varying responses to attack effectiveness is not limited to ASR, but it also holds for RISR. Furthermore, we observe from Figures 11a and b that protocols' differences in RISR are larger when λ is small, e.g., $\lambda = 1$ and 3. In contrast, as λ increases, RISRs of different protocols tend to converge. The reason is because large λ causes the inference range to also become larger, therefore inference success rate increases. Hence, RISRs of all protocols increase and converge.

Comparison between ASR and RISR: Comparing the ASR results from earlier sections and the RISR results in Figure 11, we observe that protocols' RISRs are generally higher than ASRs under the same ε. This is because ASR requires adversary's prediction to be exactly correct, i.e., $v_\ell^p = v_\ell$. In contrast, RISR looks for correctness within a range, i.e., $v_\ell - \lambda \leq v_\ell^p \leq v_\ell + \lambda$. However, ASR and RISR results agree on two important trends. First, different protocols can have different RISR, which agrees with the findings of ASR. Second, as ε increases, both ASR and RISR increase since privacy becomes more relaxed. Therefore, although the individual results of ASR and RISR may be different (as expected due to their different constructions), they both support this paper's motivation behind LDPLens and adversarial analysis of LDP protools.

VI. Related Work

LDP has recently received significant attention from academia as well as industry [1], [3], [4], [7]. Several LDP protocols were developed, such as GRR, BLH, OLH, RAPPOR, OUE and SS [4], [6], [8]–[10]. These protocols often serve as building blocks for more advanced tasks, e.g., heavy hitter identification [14], [28], [29], frequent item and itemset mining from set-valued data [5], [12], key-value store analysis [11], [27], releasing high-dimensional datasets or marginals [30]–[32], answering multi-dimensional range queries [33], [34], mean estimation [35], [36], graph mining and synthetic graph generation [18], [37], and discovery of frequent terms [9], [38]. Recent studies also started using LDP in the context of machine learning, such as boosting [39], deep learning [40] and federated learning [41].

With growing interest in LDP, several works aimed at analyzing and improving LDP protocols under various environments. It was shown that estimation accuracy in LDP can be improved via post-processing techniques such as non-negativity and consistency requirements [15], [38], prior knowledge [13], and statistical methods, e.g., expectation maximization, maximum likelihood estimation [9], [15], [42]. However, these studies are from utility perspective and not adversarial perspective; thus, they are not comparable to our work. From adversarial perspective, new attacks to LDP protocols were recently proposed, such as data poisoning [43] and manipulation attacks [44]. However, in these attacks, adversaries are capable of compromising a subset of client devices, which is a stronger adversary model than ours. Nevertheless, these attacks can be integrated into the *Adversary Module* of LDPLens, and LDPLens can serve as a unifying platform that integrates multiple adversary models and metrics.

VII. Conclusion

In this paper, we presented an adversarial approach to protocol analysis and selection in LDP, and made three original contributions. First, we used mathematical derivations and empirical measurements to analyze the privacy properties of different LDP protocols under the Bayesian adversary model. Second, we analyzed a set of critical parameters that cause the diverse responses of protocols to the Bayesian adversary in terms of ASR, including privacy budget ε, data domain $\mathcal{U}$, encoding parameters, data distribution, and background knowledge. Third, we designed and developed LDPLens, a prototype system to enable optimized protocol selection in a given data collection scenario. Tested with three case studies using real-world datasets, we showed that the protocol recommended by LDPLens in each scenario is different and offers substantial improvement compared to other protocols. In future work, we plan to extend LDPLens with new adversary and utility metrics to enable holistic benchmarking of LDP protocols.

REFERENCES

[1] G. Cormode, S. Jha, T. Kulkarni, N. Li, D. Srivastava, and T. Wang, "Privacy at scale: Local differential privacy in practice," in *Proc. Int. Conf. Manage. Data*, 2018, pp. 1655–1658.

[2] A. G. Thakurta *et al.*, "Emoji frequency detection and deep link frequency," U.S. Patent App. 15 640 266, Dec. 14, 2017.

[3] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3571–3580.

[4] U. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: Randomized aggregatable privacy-preserving ordinal response," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 1054–1067.

[5] Z. Qin, Y. Yang, T. Yu, I. Khalil, X. Xiao, and K. Ren, "Heavy hitter estimation over set-valued data with local differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 192–203.

[6] T. Wang, J. Blocki, N. Li, and S. Jha, "Locally differentially private protocols for frequency estimation," in *Proc. 26th USENIX Secur. Symp.*, 2017, pp. 729–745.

[7] (2020). *Apple—Learning With Privacy at Scale*. [Online]. Available: https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf

[8] R. Bassily and A. Smith, "Local, private, efficient protocols for succinct histograms," in *Proc. 47th Annu. ACM Symp. Theory Comput.*, Jun. 2015, pp. 127–135.

[9] G. Fanti, V. Pihur, and Ú. Erlingsson, "Building a RAPPOR with the unknown: Privacy-preserving learning of associations and data dictionaries," *Proc. Privacy Enhancing Technol.*, vol. 2016, no. 3, pp. 41–61, 2016.

[10] S. Wang *et al.*, "Mutual information optimally local private discrete distribution estimation," 2016, *arXiv:1607.08025*.

[11] X. Gu, M. Li, Y. Cheng, L. Xiong, and Y. Cao, "PCKV: Locally differentially private correlated key-value data collection with optimized utility," in *Proc. USENIX Secur. Symp.*, 2020, pp. 967–984.

[12] T. Wang, N. Li, and S. Jha, "Locally differentially private frequent itemset mining," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 127–143.

[13] J. Jia and N. Z. Gong, "Calibrate: Frequency estimation and heavy hitter identification with local differential privacy via incorporating prior knowledge," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 2008–2016.

[14] T. Wang, N. Li, and S. Jha, "Locally differentially private heavy hitter identification," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 2, pp. 982–993, Mar. 2021.

[15] T. Wang, M. Lopuhaä-Zwakenberg, Z. Li, B. Skoric, and N. Li, "Locally differentially private frequency estimation with consistency," in *Proc. NDSS*. 2020.

[16] S. L. Warner, "Randomized response: A survey technique for eliminating evasive answer bias," *J. Amer. Statist. Assoc.*, vol. 60, no. 309, pp. 63–69, 1965.

[17] M. E. Gursoy, A. Tamersoy, S. Truex, W. Wei, and L. Liu, "Secure and utility-aware data collection with condensed local differential privacy," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2365–2378, Sep./Oct. 2021.

[18] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, "Generating synthetic decentralized social graphs with local differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 425–438.

[19] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system," in *Proc. 7th ACM SIGCOMM Conf. Internet Meas.*, 2007, pp. 1–14.

[20] A. Clauset, C. R. Shalizi, and M. E. J. Newman, "Power-law distributions in empirical data," *SIAM Rev.*, vol. 51, no. 4, pp. 661–703, 2009.

[21] J. Blocki, A. Datta, and J. Bonneau, "Differentially private password frequency lists," in *Proc. NDSS*, vol. 16, 2016, p. 153.

[22] R. Jansen and A. Johnson, "Safely measuring tor," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 1553–1567.

[23] (2020). *Uber Movement*. [Online]. Available: https://movement.uber.com

[24] L. Yu, L. Liu, and C. Pu, "Dynamic differential location privacy with personalized error bounds," in *Proc. NDSS*, 2017, pp. 1–15.

[25] R. Shokri, G. Theodorakopoulos, C. Troncoso, J.-P. Hubaux, and J.-Y. Le Boudec, "Protecting location privacy: Optimal strategy against localization attacks," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 617–627.

[26] T. Murakami and Y. Kawamoto, "Utility-optimized local differential privacy mechanisms for distribution estimation," in *Proc. 28th USENIX Secur. Symp. (USENIX Secur.)*, 2019, pp. 1877–1894.

[27] Q. Ye, H. Hu, X. Meng, and H. Zheng, "PrivKV: Key-value data collection with local differential privacy," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 317–331.

[28] R. Bassily *et al.*, "Practical locally private heavy hitters," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2288–2296.

[29] M. Bun, J. Nelson, and U. Stemmer, "Heavy hitters and the structure of local privacy," in *Proc. 37th ACM SIGMOD-SIGACT-SIGAI Symp. Princ. Database Syst.*, May 2018, pp. 435–447.

[30] G. Cormode, T. Kulkarni, and D. Srivastava, "Marginal release under local differential privacy," in *Proc. Int. Conf. Manage. Data*, May 2018, pp. 131–146.

[31] X. Ren *et al.*, "Lopub: High-dimensional crowdsourced data publication with local differential privacy," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 9, pp. 2151–2166, Sep. 2018.

[32] Z. Zhang, T. Wang, N. Li, S. He, and J. Chen, "CALM: Consistent adaptive local marginal for marginal release under local differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 212–229.

[33] G. Cormode, T. Kulkarni, and D. Srivastava, "Answering range queries under local differential privacy," *Proc. VLDB Endowment*, vol. 12, no. 10, pp. 1126–1138, Jun. 2019.

[34] T. Wang *et al.*, "Answering multi-dimensional analytical queries under local differential privacy," in *Proc. Int. Conf. Manage. Data*, Jun. 2019, pp. 159–176.

[35] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Minimax optimal procedures for locally private estimation," *J. Amer. Statist. Assoc.*, vol. 113, no. 521, pp. 182–201, 2018.

[36] N. Wang *et al.*, "Collecting and analyzing multidimensional data with local differential privacy," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 638–649.

[37] C. Wei, S. Ji, C. Liu, W. Chen, and T. Wang, "AsgLDP: Collecting and generating decentralized attributed graphs with local differential privacy," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3239–3254, 2020.

[38] N. Wang *et al.*, "PrivTrie: Effective frequent term discovery under local differential privacy," in *Proc. IEEE 34th Int. Conf. Data Eng. (ICDE)*, Apr. 2018, pp. 821–832.

[39] S. Wang and J. M. Chang, "Privacy-preserving boosting in the local setting," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4451–4465, 2021.

[40] P. C. Mahawaga Arachchige, P. Bertok, I. Khalil, D. Liu, S. Camtepe, and M. Atiquzzaman, "Local differential privacy for deep learning," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5827–5842, Jul. 2020.

[41] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei, "LDP-Fed: Federated learning with local differential privacy," in *Proc. 3rd ACM Int. Workshop Edge Syst., Anal. Netw.*, 2020, pp. 61–66.

[42] T. Murakami, H. Hino, and J. Sakuma, "Toward distribution estimation under local differential privacy with small samples," *Proc. Privacy Enhancing Technol.*, vol. 2018, no. 3, pp. 84–104, Jun. 2018.

[43] X. Cao, J. Jia, and N. Z. Gong, "Data poisoning attacks to local differential privacy protocols," in *Proc. 30th USENIX Secur. Symp. (USENIX Secur.)*, 2021, pp. 947–964.

[44] A. Cheu, A. Smith, and J. Ullman, "Manipulation attacks in local differential privacy," 2019, *arXiv:1909.09630*.