Fang *et al. EURASIP Journal on Wireless Communications and Networking* (2020) 2020:65
https://doi.org/10.1186/s13638-020-01675-8

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Local differential privacy for human-centered computing

Xianjin Fang, Qingkui Zeng and Gaoming Yang*

Abstract

Human-centered computing in cloud, edge, and fog is one of the most concerning issues. Edge and fog nodes generate huge amounts of data continuously, and the analysis of these data provides valuable information. But they also increase privacy risks. The personal sensitive data may be disclosed by untrusted third-party service providers, and the current solutions to privacy protection are inefficient, costly. It is difficult to obtain available statistics. To solve these problems, we propose a local differential privacy sensitive data collection protocol in human-centered computing. Firstly, to maintain high data utility, the selection of the optimal number of hash functions and the mapping length is based on the size of the collected data. Secondly, we hash the sensitive data, add the appropriate Laplace noise to the client side, and send the reports to the server side. Thirdly, we construct the count sketch matrix to obtain privacy statistics on the server side. Finally, the utility of the proposed protocol is verified by synthetic datasets and a real dataset. The experimental results demonstrate that the protocol can achieve a balance between data utility and privacy protection.

Keywords: Human-centered computing, Local differential privacy, Laplace noise, Count sketch

1 Introduction

With the development of Internet of Things, technology [1], edge and fog nodes [2], mobile phones, smart cars, wearable devices, and sensor networks have increasingly become the sources of big data [3]. Human-centered computing in cloud [4], edge, and fog has become necessary tasks for enterprises and governments [5]. On the one hand, big data collection and analysis can be used to train machine learning models and to understand user group characteristics to improve user experienc e[6]; on the other hand, deriving sensitive data, such as user preferences, lifestyle habits, and location information [7], can result in privacy leaks. Researchers have conducted many studies on how to prevent the disclosure of personal sensitive information [8] and have proposed many privacy protocols [9].

Differential privacy (DP) [10] is a widely studied privacy-preserving model; it requires that the addition or deletion of any one record does not affect the query results. The traditional differential privacy model is deployed on the central server, and data collected from different sources are transformed into aggregate response queries for privacy protection, i.e., the central server publishes query information that satisfies differential privacy. Therefore, differential privacy is widely applied in all aspects of big data collection. For example, the US Census Bureau uses differential privacy for demographics [11].

However, in the data collection phase, there is little oversight over third-party service providers; consequently, privacy leaks frequently occur, which happen on Facebook [12] and Snapchat [13]. Such frequent privacy disclosures have attracted the public's attention, but in practice, it is very difficult to find a trusted third-party aggregator. This difficulty limits the application of traditional differential privacy to a certain extent. Therefore, it is necessary to consider how to ensure that private information is not disclosed when there is no trusted third-party service provider.

As a result of extensive research on differential privacy, local differential privacy (LDP) is based on traditional differential privacy protection [14]. LDP can obtain valuable information by aggregating clients' perturbed reports

* Correspondence: gmyang@aust.edu.cn
School of Computer Science and Engineering, Anhui University of Science and Technology, Huainan 232001, China

without obtaining real released data information, and it can prevent untrusted third parties from revealing privacy. LDP can be applied to various data collection scenarios, such as frequency estimation, heavy hitters identification, and frequent itemset mining. Companies in different fields, such as Google [15] and Apple [16], have used LDP protocols to collect users' default browser homepages and search engine settings, which can identify harmful or malicious hijacking user settings [17] and find the most frequently used emojis or words.

However, the LDP model still has shortcomings with respect to big data collection, such as low accuracy, high space-time complexities, and statistical errors. Since different tasks require adopting different LDP protocols in actual applications, to determining the appropriate parameters for each protocol is difficult, which undoubtedly increases the cost of using LDP to protect sensitive data. To solve these problems, we propose using the count sketch [18] and Laplace mechanism [10] to reduce space-time complexity and computational overhead and to obtain high data utility under different distributions. The main contributions of this paper are as follows:

(i) We design the LDP protocol to provide a controllable privacy guarantee level on the client side that does not require trusted third-party servers;

(ii) The proposed protocol solves the problems of large space-time overhead and low data utility and can be applied to different data distributions;

(iii) Experiments show that the proposed protocol can provide available statistical information while protecting user data privacy.

This paper is organized as follows. First, we describe related works in Section 2 and the background knowledge for this paper in Section 3. Next, we introduce the current protocols for big data collection in Section 4 and propose our method in Section 6. Then, we evaluate our method in Section 6 and analyze the results in Section 7. At last, we make a summary of our work in Section 8.

2 Related works

Many scholars and enterprises have studied how to apply LDP in cloud, edge, and fog scenarios and how to improve the performance of LDP protocols. For example, Erlingsson et al. [15] propose the RAPPOR protocol, which uses a Bloom filter and random response to implement an LDP frequency estimated in the Chrome browser. In reference [16], Apple's Differential Privacy team proposes using one-hot encoding technology to encode sensitive data and deploy a CMS algorithm for analyzing the most popular emojis and media playback preferences in Safari. Fanti et al. [19] propose the unknown-RAPPOR protocol to estimate frequency without a data dictionary. Ding et al. [20] propose an algorithm to solve the problem of privacy disclosure when repeatedly collecting telemetry data and apply the algorithm to Microsoft-related products. Wang et al. [14] propose the Harmony protocol to achieve LDP protection. These authors compute the numerical attributes means, and the protocol is deployed in the Samsung's system software. Wang et al. [21] use the LDP protocol to answer private multidimensional queries on Alibaba's e-commerce transaction records. LDP has been deployed in the industry, and it has created practical benefits.

One of LDP's important applications is frequency estimation. Wang et al. [22] propose an OLH algorithm to obtain lower estimation error and to reduce the communication overhead in a larger domain; the algorithm can also be used in heavy hitters identification. The Hadamard response [23] uses the Hadamard transform instead of the hash function; this is because the actual calculation of Hadamard entries is easier, the server-side aggregates report faster. Joseph et al. [24] propose a technique that repeatedly recomputes a statistic with the error which leads to the decays of errors; it happens when the statistic changes significantly rather than the current value of the statistic is recomputed. Wang et al [25] introduce a method that adds postprocessing steps to frequency estimations to make them consistent while achieving high accuracy for a wide range of tasks. Most of the LDP protocols for frequency estimation are implemented by random responses, thereby resulting in low accuracy of the estimation results. Thus, the goal of this paper is to investigate the mechanisms that can achieve LDP with high data utility.

Recent LDP studies have also focused on other applications [26, 27]. Bassily et al. [28] propose an S-HIST algorithm for histogram release and utilize random projection technology to further reduce the communication cost. Wang et al. [22] propose identifying heavy hitters in datasets under LDP protection. Ren et al. [29] apply the LDP model to solve the privacy problem in the case of collecting high-dimensional crowdsourced data. Wang et al. [30] propose privacy amplification by multiparty differential privacy, which introduces an auxiliary server between the client side and the server side. Ye et al .[31] propose PrivKV, which can estimate the mean and frequency of key-value data, and PrivKVM, which can improve estimation accuracy through multiple iterations. Therefore, many LDP protocols have been proposed to solve privacy issues in cloud, edge, and fog computing scenarios.

3 Preliminary

3.1 Differential privacy

Differential privacy requires that any tuple in the dataset be under a limited impact. For example, for two datasets

D and D' that are different by only one tuple, the attacker cannot infer the sensitive information of a specific data tuple from the query results, so it is impossible to know whether the data of a certain user exist in the dataset. The definition of differential privacy is as follows:

Definition 1 ε-differential privacy [10]. Where $\varepsilon > 0$, a randomized mechanism M satisfies ε-differential privacy if for all datasets D and D' that differ at most one tuple, and all S⊆Range(M),;we have

$$\Pr[M(D)\in S] \le e^{\varepsilon} \cdot \Pr[M(D')\in S] \tag{1}$$

3.2 Local differential privacy

Local differential privacy requires any two tuples to be indistinguishable. For example, for any two tuples x and x', the attacker cannot infer the sensitive information of a specific data tuple from the query results, so it is impossible to know the specific tuple. ε- local differential privacy is defined as follows:

Definition 2 ε-local differential privacy [14]. Where $\varepsilon > 0$, a randomized mechanism M satisfies ε-local differential privacy if and only if for any two input tuples x and x' in the domain of M, and for any possible output x^* of M, we have

$$\Pr[M(x) = x^*] \le e^{\varepsilon} \cdot \Pr[M(x') = x^*] \tag{2}$$

It can be concluded from the definition of ε-local differential privacy that the output of a randomized mechanism of any pair of input tuples is similar, and therefore cannot be inferred by the specific input tuple. A smaller privacy budget ε ensures a higher privacy level, but repeating the queries for the same tuple will consume ε, thereby decreasing the level of privacy. Therefore, the choice of ε needs to be determined according to the specific scenario.

3.3 Sensitivity and Laplace mechanism

Differential privacy implements privacy protection by adding noise to the query results, and the amount of noise added should not only protect user privacy but also maintain data utility. Therefore, sensitivity becomes a key parameter of noise control. In local differential privacy, sensitivity is based on a query function of any two tuples; the following definitions are given.

Definition 3 Local sensitivity [32]. For f: $D^n \rightarrow \boldsymbol{R}^d$ and $x\in D^n$, the local sensitivity of f at x (with respect to the $l1$ metric) is:

$$LS_f(x) = \max_{y:d(x,y)=1} \|f(x) - f(y)\|_1 \tag{3}$$

The notion of local sensitivity is a discrete analog of the Laplacian (or maximum magnitude of the partial derivative in different directions). The Laplace mechanism of differential privacy [10] adds noise that satisfies the Laplace distribution to the original dataset to implement differential privacy protection; therefore, we have the following definition.

Definition 4 Laplace mechanism [33]. An algorithm A takes as input a dataset D, and some $\varepsilon > 0$, a query Q with computing function f: $D^n \rightarrow \boldsymbol{R}^d$, and outputs

$$A(D) = f(D) + (Y_1, \ldots, Y_d) \tag{4}$$

where the Y_i is drawn i.i.d from Lap($\mathrm{LS_f}(x)/\varepsilon$), thus obeying the Laplace distribution with the scale parameter ($\mathrm{LS_f}(x)/\varepsilon$). For ease of expression, we denote Δs as the local sensitivity and Lap(λ) as a random variable that obeys the Laplace distribution of scale λ. The corresponding probability density function is $\mathrm{pdf}(y) = 1/2\lambda e^{\left(-\frac{|y|}{\lambda}\right)}$.

3.4 System structure

Local differential privacy can be seen as a special case of differential privacy [34]. Compare with the perturbation process in DP, the perturbation process in LDP shifts from the server side to the client side. The privacy leakage threat from untrusted third-party servers is eliminated because trusted third-party servers are not required. This collection consists of the following main parts:

- The encoding is performed by the client side; each tuple should be encoded into a proper vector to ensure perturbation;
- The perturbation is performed by the client side, and each piece of encoded data generates a perturbed report by the random function, thereby satisfying the definition of ε-local differential privacy. Then, the client side sends these perturbed reports to the server;
- The aggregation process is performed by the server side, which aggregates reports from the client side and generates available statistics, as shown in Fig. 1.

4 Problem setting

To use the LDP model to analyze and protect the collected data, some scholars have proposed many privacy protection schemes when estimating frequency. However, these solutions still have deficiencies, such as high computational overhead and low data utility. Therefore, we propose a modified solution to further improve data utility and algorithm accuracy based on solving existing deficiencies.

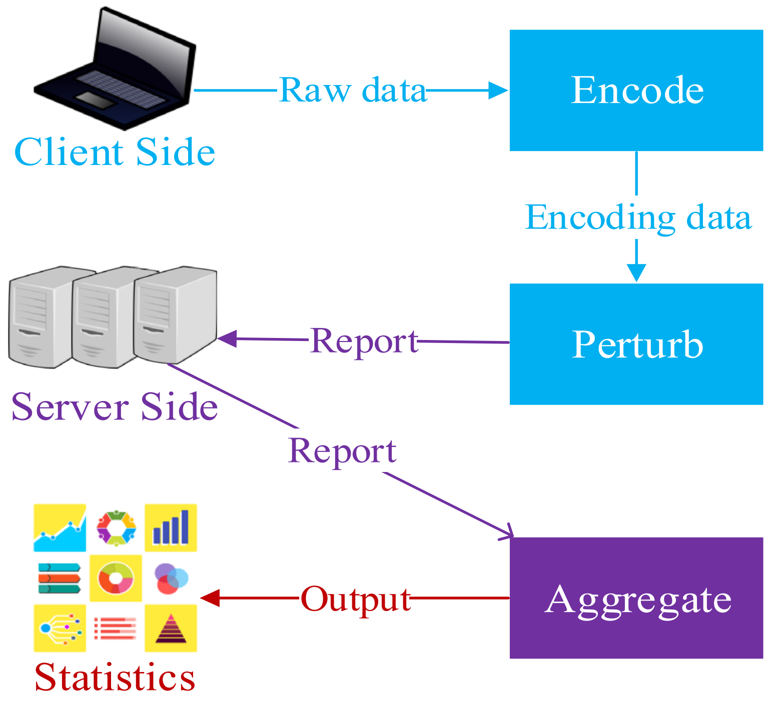

Fig. 1 Local differential privacy system structure

4.1 Analyzing current methods

4.1.1 Generalized random response (GRR)

This random response technique was proposed by Warner et al. [35]. For each piece of collected private data $v \in D$, the user sends the true value of v with probability p and sends randomly selected value v' from $D \backslash \{v\}$ with a probability $1 - p$. Assuming that domain D contains $d = |D|$ values, the perturbation function is as follows:

$$\Pr[\mathrm{GRR}(D) = y] = \begin{cases} p = \dfrac{e^{\varepsilon}}{e^{\varepsilon} + d - 1}, & \text{if } y = v \\ q = \dfrac{1}{e^{\varepsilon} + d - 1}, & \text{if } y \neq v \end{cases} \qquad (5)$$

Since $p/q = e^{\varepsilon}$, the ε-differential privacy definition is satisfied.

4.1.2 Optimal local hash (OLH)

The OLH protocol was proposed in [22] to address the problem of large category attributes. First, the client-side algorithm maps the user's true value v to a smaller hash value domain g by using a hash function. Then, the algorithm performs a random response to the hash value of this smaller domain. The parameter g is a trade-off for the loss of information between the hashing and randomization step; when $g = e^{\varepsilon} + 1$, the trade-off is optimal. The time complexity of the algorithm is $O(\log n)$, and the space complexity is $O(n\log|D|)$.

4.1.3 Randomized aggregatable privacy-preserving ordinal response (RAPPOR)

The RAPPOR protocol [15] is deployed in Google's Chrome browser. In the RAPPOR protocol, the user's real value v is encoded into the bit vector $\boldsymbol{B}$. When there are numerous category attributes, the protocol causes problems, such as a high communication cost and low accuracy. Therefore, RAPPOR uses the Bloom filter for encoding. The value v is mapped to a different position in the bit vector $\boldsymbol{B}$ using k hash functions, i.e., the corresponding position is set to 1, and the remaining positions are set to 0. After encoding, RAPPOR utilizes a perturbation function to obtain the perturbed bit vector $\boldsymbol{B}'$.

4.1.4 Hadamard Count Mean Sketch (HCMS)

The Hadamard Count Mean Sketch protocol was proposed by Apple's Differential Privacy Team [16] in 2016 to complete large-scale data collection with LDP and to obtain accurate counts. By utilizing the Hadamard transform, the sparse vector is transformed to send a single privacy bit, so each user just sends one private bit. A certain tuple x sent by a given user belongs to a set of values D; j is a randomly selected index from k hash functions, and l is a randomly selected index from the m bits of the hash map domain.

Algorithm 1 Hadamard Count Mean Sketch (HCMS) [16] Client-Side Algorithm

```
Input: user's data $d \in D$, privacy budget $\varepsilon$, number of hash functions $k$, hash map length $m$.
Output: perturbed bit $w_l$, hash function index $j$, perturbed bit index $l$.
1. Sample $j$ uniformly at random from $[k]$.
2. Initialize a vector $\boldsymbol{v} \leftarrow \{0\}^m$.
3. Set $\boldsymbol{v}_{hj(d)} \leftarrow 1$.
4. Construct the Hadamard matrix $H_m$, and transform $w \leftarrow H_m\boldsymbol{v}$.
5. Sample $l$ uniformly at random from $[m]$.
6. Randomly flip $w_l$ with probability $(1/e^{\varepsilon}+1)$.
7. Return $s\{w_l, j, l\}$.
```

Algorithm 1 shows the client's perturbation process. First, each user initializes the vector $\boldsymbol{v}$ and sets the mapping value of the attribute value d in $\boldsymbol{v}$ at the j-th hash function to 1, and the vector $\boldsymbol{v}$ forms a one-hot vector. Second, the algorithm randomly flips the l-th bit of the vector, denoted as $w_l \in \{-1,1\}$, with a probability of $(1/e^{\varepsilon} + 1)$. Finally, the client side sends the report $s\{w_l, j, l\}$ to the server. The time complexity of the algorithm is $O(n+k m\log(m)+|D|k)$, and the space complexity is $O(\log(k) + \log(m) + 1)$.

Algorithm 2 Hadamard Count Mean Sketch (HCMS) [16] Server-Side Algorithm

```
Input: Dataset $D=\{(w^{(1)},j^{(1)},l^{(1)}),\ldots,(w^{(n)},j^{(n)},l^{(n)})\}$, privacy budget $\varepsilon$, dimensions $k$ and $m$.
Output: $f(d)$:$(d \in |D|)$
1. Set $c \leftarrow \frac{e^{\varepsilon}+1}{e^{\varepsilon}-1}$.
2. For each $i \in [n]$ set $x^{(i)} \leftarrow k \cdot c \cdot w^{(i)}$.
3. Initialize $M^H \in \{0\}^{k\times m}$.
4. for $i \in [n]$ do
       $M^H_{j^{(i)},l^{(i)}} \leftarrow M^H_{j^{(i)},l^{(i)}} + x^{(i)}$.
5. Transform the rows of sketch back: $M^H \leftarrow M^H H_m^T$.
6. Construct $f: D \rightarrow R$ where
       $f(d) = \left(\frac{m}{m-1}\right)\left(\frac{1}{k}\sum_{l=1}^{k} M_{l,h_l(d)} - \frac{n}{m}\right)$
7. Return $f(d)$:$(d \in |D|)$.
```

Algorithm 2 shows the server aggregation process. First, it takes each report $w^{(i)}$ and transforms it to $x^{(i)}$. Then, the server constructs the sketch matrix M^H and add $x^{(i)}$ to row $j^{(i)}$, column $l^{(i)}$ of M^H. Next, it uses the transpose Hadamard matrix to transform the rows of sketch back. At last, the server estimates the count of entry $d \in |D|$ by debiasing the count and averaging over the corresponding hash entries in M^H.

4.1.5 Deficiencies of current protocols

Many current protocols have been proposed to protect privacy, but they still have deficiencies. First, the LDP protocol has very strict requirements for selecting parameters and concerning the size of the data. For example, the choice of parameters k and m in the HCMS algorithm greatly influences data variance and utility, and different tasks need to identify different suitable parameters. Second, the RAPPOR and CMS algorithms have large space-time complexity and a high communication cost. For data collection in cloud, edge, and fog scenarios, this problem will make computation highly inefficient. Third, due to the use of random response techniques, a data value with low frequency can even be estimated as negative. Finally, when privacy-preserving data are from different distributions, data utility varies greatly, and it is difficult to fit these data to different tasks.

4.2 Design method to address deficiencies

Because of the shortcomings of current LDP protocols, we use the Laplace mechanism to solve the problem that random response technology requires strict data size and construct the count sketch matrix for aggregation to reduce space-time complexity. The protocol consists of the local perturbator and aggregator.

The local perturbator is designed on the client side to perturb the raw data. When a user generates data, the local perturbator selects a random hash function to encode the data as a one-hot encoding and adds the Laplace noises in the mapping location. Then, the local perturbator sends the report containing the selected hash function index and the noised mapping location to the central server. Since the client-side algorithm satisfies the LDP definition, even if the adversary has the relevant background knowledge and acquires another user's data, the adversary cannot infer which data are the user's data. The process is shown in Fig. 2 a.

The aggregator is designed on the server side to aggregate the reports. When the central server receives all the perturbed reports from the client side, the server will aggregate them through an aggregator. The aggregator structures the count sketch matrix and cumulates the number of mapping positions for each attribute value under different hash functions. The server side obtains each data value frequency estimation by matrix count. Estimating the data entry d as an example, the aggregator counts the number of x_j's frequency of the corresponding mapping position under different hash functions and sums up the numbers, as shown in Fig. 2 b.

5 Methods

For human-centered computing in cloud, edge, and fog scenarios, there are generally many users and one data service provider in the LDP model. Therefore, the proposed protocol designs the client and server algorithm for the users and service providers, respectively. The client-side algorithm perturbs the user's raw data and sends an incomplete report; each user sends the perturbed report to the unique service provider. When the service provider receives the user's

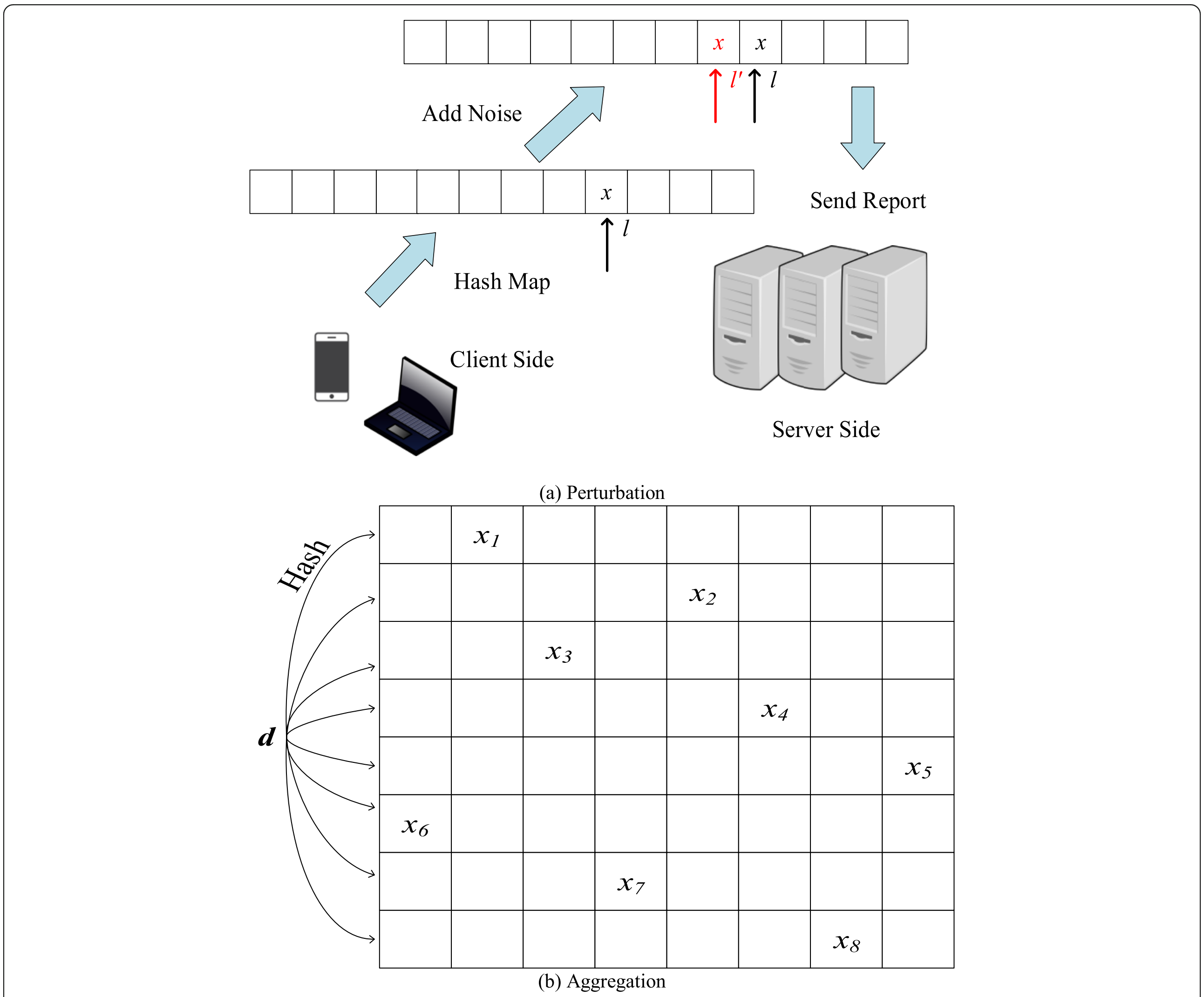

Fig. 2 The algorithm system structure. **a** Perturbation. **b** Aggregation

perturbed reports, the server-side algorithm aggregates the reports to obtain the available statistics.

5.1 Client-side algorithm design

The client-side algorithm is designed to prevent user data leakage by ensuring that the perturbed data obey local differential privacy. The raw data are first encoded by a randomly selected hash function, and then the Laplace mechanism is applied to implement the perturb operation in the hash map location. The parameters passed by the server are used before the algorithm is deployed; these parameters include the privacy budget ε, the number of hash functions k, and the length of hash mapping bit m. According to equation (3), any two different pieces of data have at the most two differences in the one-hot encoding vector $\boldsymbol{v}$, so the local sensitivity of adding Laplace noise is 2. The report sent by the algorithm includes two parts: a randomly selected hash function index j and a hashed map position with noise l'. For convenience, this algorithm is named the Laplace Count Sketch (LCS) client-side algorithm. The specific steps are shown in Algorithm 3.

Algorithm 3 Laplace Count Sketch (LCS) Client-Side Algorithm

```
Input: user's data $d \in D$, privacy budget $\varepsilon$, number
       of hash functions $k$, hash map length $m$.
Output: perturb report $s_i$.
1. Initialize a vector $\boldsymbol{v} \leftarrow \{0\}^m$.
2. Sample $j$ uniformly at random from $[k]$.
3. Set $\boldsymbol{v}_{hj(d)} \leftarrow 1$.
4. Denote the mapping position as $l$.
5. Add Laplace noise $l'=l+Lap(\Delta s/\varepsilon)$.
6. Round($l'$)
7. If $l'<0$:
     $l'=m+l'$
   Else if $l'>=m$:
     $l'=l'-m$
8. Return $s_i\{j, l'\}$
```

Line 1 of Algorithm 2 initializes an all-zero vector of length m, and in lines 2–4, the algorithm randomly selects the hash function and hash mapping on vector $\boldsymbol{v}$, where $h_j(d)$ denotes choosing the function j to hash data d. In addition, the mapping position is added with the Laplace noise in lines 5–7. Since the mapping position is an integer, the noise value should be rounded. As is already known, the length of the mapping vector is m. For the added noise mapping position l', l' will equal l' minus m if the value of l' is greater than or equal to m; else, l' equals l' plus m if the value of l' is less than 0. The algorithm sends the perturbed report s_i in line 8. Each user sends the perturbed report with $O(1)$ time complexity and $O(k+m)$ space complexity; therefore, the time complexity of the client-side algorithm is $O(n)$, and the space complexity of the client-side algorithm is $O(n(k+m))$.

5.2 Server-side algorithm design

The server-side constructs the count sketch matrix using the same parameters as those used by the client side after collecting perturbed reports from different users. First, the server-side algorithm constructs an all-zero matrix of size $k*m$. Second, the algorithm cumulates the index positions for each report position. Third, after constructing the completed count sketch matrix, the algorithm searches the position of each data value corresponding to the row in the matrix in different hash functions and adjusts the sketch counts according to Laplace distribution. Finally, the algorithm computes the counts at these positions to obtain the frequency statistics of each attribute value. The specific steps are shown in Algorithm 4.

Algorithm 4 Laplace Count Sketch (LCS) Server-Side Algorithm

```
Input: report $S = \{(j^{(1)},l^{(1)}),\ldots,(j^{(n)},l^{(n)})\}$
Output: estimate data value frequency $f(d)$
1. Initialize a sketch $M=\{0\}^{k*m}$
2. For $i$ in range(0, $n$)
     $M[j^{(i)}][l^{(i)}]+=1$
3. For $d$ in (0, $|D|$)
       For $j$ in (0, $k$)
           $f(d) = \dfrac{\sum_j^k M[j][h_j(d)] \cdot \int_{-\infty}^{\infty} \frac{\Delta s e^{\left(-\frac{|y|\varepsilon}{\Delta s}\right)}}{2\varepsilon} dy}{\int_{-1.5}^{1.5} \frac{\Delta s e^{\left(-\frac{|y|\varepsilon}{\Delta s}\right)}}{2\varepsilon} dy}$
4. Return $f(d)$
```

Line 1 of Algorithm 3 initializes an all-zero matrix of size $k*m$. In line 2, the algorithm deals with the collected n reports and adds 1 to the index position of the corresponding row and column in the matrix. Then, in line 3, the algorithm uses the count sketch to record the matrix value of each data at the corresponding position of each hash function and estimates the frequency of each attribute value. The time and space complexities are $O(n+|D|*k)$ and $O(k*m)$, respectively, in the server-side algorithm.

5.3 Privacy and utility analysis

This section discusses the privacy and utility of the proposed protocol. We first prove that the LCS protocol satisfies the local differential privacy definition and then theoretical analysis of the algorithm's variance, and a smaller variance ensures the higher data utility.

Theorem 1 The LCS protocol satisfies the definition of ε- local differential privacy.

Proof. Given any pair of input tuples x and x' and any possible output x^*, p_x is labeled as the probability density function of $A(x)$, and $p_{x'}$ is labeled as the probability density function of $A(x')$; compare the probability of these two.

$$\frac{\Pr[A(x)=x^*]}{\Pr[A(x')=x^*]} = \frac{\Pr[x+y]}{\Pr[x'+y]} = \frac{p_x(y)}{p_y(y)} = e^{\varepsilon\left(\frac{|f(x)-f(x')|}{\Delta s}\right)} \le e^{\varepsilon}$$

Since the sensitivity $\Delta s = 2$, the maximum difference between the values of the functions $f(x)$ and $f(x')$ is 2; that is, $|f(x)–f(x')|$ has a value range of [0, 2], so $(|f(x) - f(x')|/\Delta s) \le 1$. Therefore, the definition of local differential privacy is satisfied.

We infer the variance of the LCS algorithm and denote the estimated frequency $f'(d)$, the real frequency $f(d)$, and $\mathbb{E}(f'(d)) = f(d)$.

$$\begin{aligned} Var(f'(d)) &= \mathbb{E}\left(f'(d)^2\right) - \mathbb{E}\left(f'(d)\right)^2 \\ &= \frac{\sum_d^D f(d)^2}{k^2 m^2 n} \end{aligned}$$

The larger the parameters k and m are, the smaller the variance and the higher the utility are. However, space-time complexity increases when k and m are very large.

6 Experimental section

6.1 Experimental datasets

The experiments use three datasets: two synthetic datasets and one real dataset; all datasets are a one-dimensional classification attribute. The real dataset uses the 2017 *Integrated Public Use Microdata Series* [36] (US) and selects the education level *EDU* attribute, which has 25 data categories; we extract 1% from the dataset and take the first million pieces of data as the experimental dataset. The synthetic dataset satisfies the uniform distribution and the Zipf distribution. The parameter a of the Zipf distribution is set to 1.2, and each synthetic dataset contains 100,000 pieces of data.

6.2 Experimental competitors

OLH is a better choice for our experiment as the comparison protocol because it gives near-optimal utility when the communication bandwidth is reasonable. In addition, we choose HCMS as another comparison protocol, which reduces the communication overhead by sending a single private bit at a time.

6.3 Experimental implementation

These protocols were implemented in Python 3.7 with NumPy and xxhash libraries and were performed on a PC with Intel Core i7-7700hq CPU and 16 GB RAM. Each experiment was repeated ten times to reduce the influence of contingency on the experimental results.

6.4 Experimental metrics

To analyze the utility of our protocol for different parameters and scenarios, we compare the error between the true distribution and the estimated distribution for frequency using the mean absolute percentage error (MAPE). For each data value, we calculate the absolute value between the estimated and true frequency, divide the absolute value by the true frequency, then cumulate these values and divide by the size of the data value domain. The definition of MAPE is as follows:

$$\mathrm{MAPE} = \frac{\sum_{i=1}^{|D|} \left| \frac{y_i - x_i}{y_i} \right|}{|D|} \times 100\% \tag{6}$$

where $|D|$ is the category attribute domain size, y_i is the real frequency of the i-th attribute value, and x_i is the estimated frequency of the i-th attribute value. The smaller the MAPE value is, the closer the estimated distribution is to the real distribution, and the better the data utility is.

7 Results and discussion

7.1 Effects of privacy budgets

We validate the effects of privacy budgets parameter ε on the data utility of the LCS protocol through experiments, select the HCMS and OLH protocols as the control group and select the uniform and Zipf synthetic datasets, in which the classification attribute domain value is 50.

For the uniform dataset, we select the number of hash functions k = 128 and the size of the hash map length m = 128 and verify the MAPE values of the three protocols with the variation of the privacy budget ε. As shown in Fig. 3 a, as the privacy budget ε increases, the MAPE value of these three protocols decreases; that is, the data utility increases when the privacy budget increases. The data utility of the LCS protocol is significantly better than that of HCMS and slightly lower than that of OLH

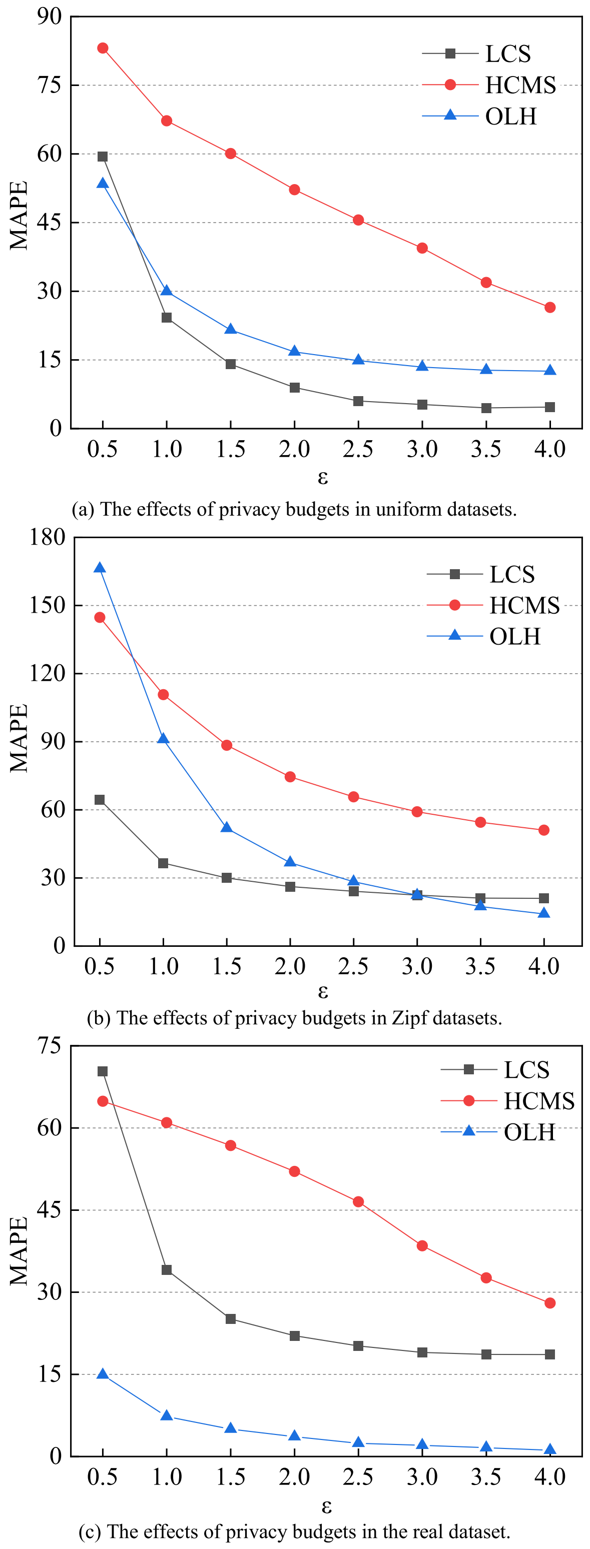

Fig. 3 The effects of privacy budgets on MAPE. **a** The effects of privacy budgets in uniform datasets. **b** The effects of privacy budgets in Zipf datasets. **c** The effects of privacy budgets in the real dataset

when $\varepsilon < 2.5$ and higher than the other protocols when $\varepsilon > 2.5$.

Then, we adjust the parameters of the upper group experiments and verify the effects of the privacy budget ε on the synthetic dataset satisfying the Zipf distribution; we select k = 256 and m = 512, as shown in Fig. 3 b. The data utility of LCS is superior to that of the HCMS protocol throughout the experiments, and the data utility of LCS is marginally lower than that of the OLH protocol when $\varepsilon > 3$.

Next, we validate the effects of privacy budget ε on utility in a real dataset and perform experiments on the IPUMS dataset; we choose parameter k = 256 and m = 128. In Fig. 3 c, the experimental result shows that the utility of the LCS protocol is higher than that of HCMS and lower than that of OLH. It is verified that the

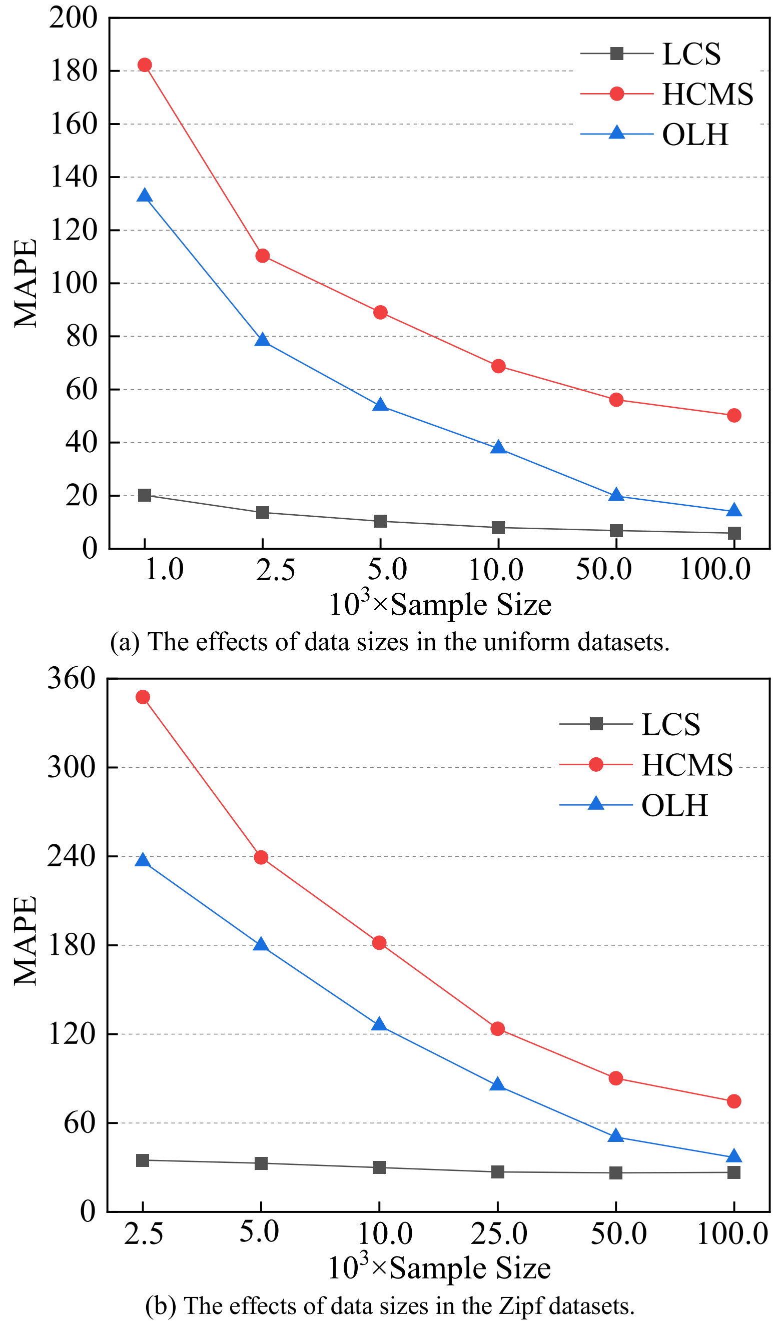

Fig. 4 The effects of different data sizes. **a** The effects of data sizes in the uniform datasets. **b** The effects of data sizes in the Zipf datasets

protocol is also feasible in practical applications, and the utility is better than that of HCMS.

7.2 Effects of data sizes

The current LDP protocols exhibit dramatic changes in data utility when collecting data of different sizes. To verify the utility of the LCS protocol at different sizes of data, we compare the LCS protocol with the HCMS and OLH protocols with uniformly distributed synthetic data while varying data sizes. The parameters are set to $m = 128$, $k = 1024$, and $\varepsilon = 2$. As shown in Fig. 4 a, when the data size n is small, the OLH and HCMS protocols have large errors, and the utility of these protocols is very low, such as when $n = 1000$; thus, the HCMS and OLH protocols are not usable. However, LCS still maintains better data utility at different data sizes. Next, we verify the utility variation under different data sizes in the synthetic datasets satisfying the Zipf distribution, adjust the parameter settings to $k = 256$, $m = 512$, and $\varepsilon = 2$ and select the HCMS and OLH protocols for comparison. In Fig. 4 b, HCMS and OLH are not available when the data size is small, and the utility of LCS is better at varying experimental data sizes.

7.3 Frequency estimation

When the privacy budget parameter ε is small, too much noise is added during the perturbation process, thereby resulting in the estimated frequency being much lower than the original frequency. We set up experiments to observe the frequency estimation under different datasets. To clearly show the frequency distribution trend and facilitate observation, we calculate the estimated frequency of each attribute value and multiply the original data amount to obtain a more accurate frequency estimate. The parameters of LCS and HCMS are both set to $k = 128$, $m = 1024$, and $\varepsilon = 2$; in addition, we choose the Zipf synthetic dataset and the IPUMS dataset as experimental datasets. The domain sizes are 15 and 25, respectively. Figure 5 a shows that the estimated frequencies of the LCS and OLH protocols are close to the true value, and the overall fluctuation of the HCMS protocol is big. Figure 5 b shows that all protocols have fluctuations, but LCS has a small overall fluctuation.

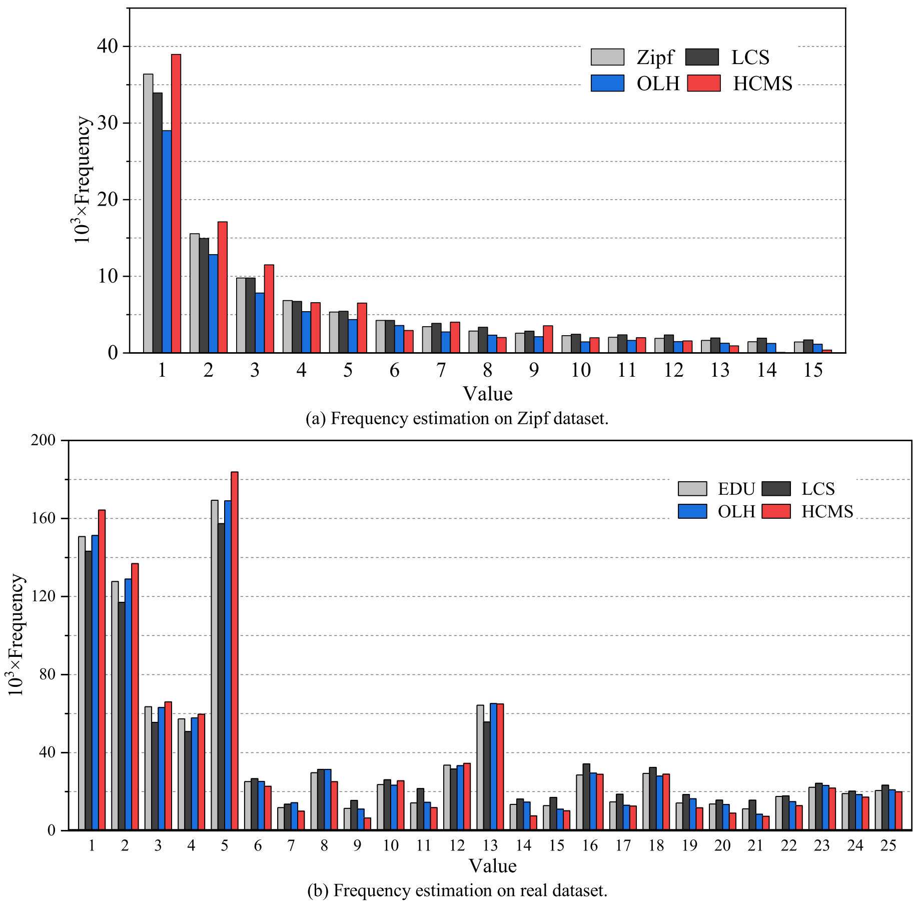

Fig. 5 Frequency estimation observation. **a** Frequency estimation on Zipf dataset. **b** Frequency estimation on real dataset

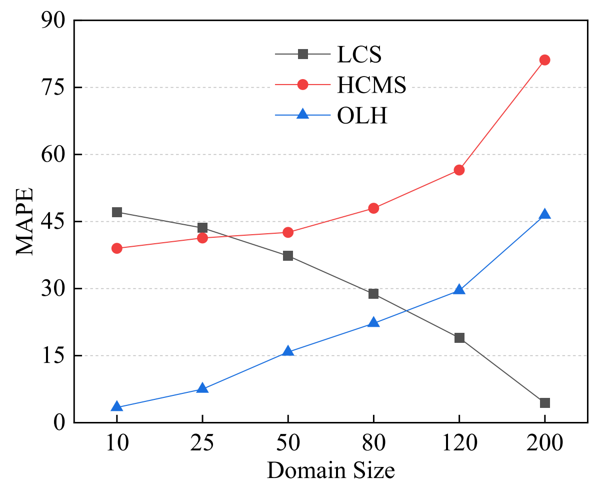

Fig. 6 The effects of different domain size

7.4 Effects of other parameters

The above experiments show that the values of the parameters k and m have a definite impact on the final result. To explore the effect of the parameters on the performance of the protocol, we set up experiments with different k and m values. The LCS protocol uses the hash function to encode data, thereby creating hash collisions. Due to the hash collisions, different values are mapped to the same position under the same hash function, thereby decreasing estimation accuracy. When the length of hash map m is small, and the size of the data domain $|D|$ is large, there will be larger errors. The frequency of hash collisions can be reduced by increasing the length of the hash map domain m; increasing the length of the hash map domain, in turn, increases the computational overhead.

When the LDP protocols process different datasets, changes in the size of the data domain $|D|$ also affect the estimated frequency. We utilize different data domain sizes to generate uniformly distributed datasets, and the protocol utility is tested under different domain sizes; we select $k = 256$, $m = 512$, and $\varepsilon = 2$. As shown in Fig. 6, the utility of the LCS protocol increases when the data domain size increases, and the HCMS and OLH protocols decrease when the data domain size increases.

7.5 Runtime

To verify the time complexity of the proposed protocol, we record its runtime in seconds on the real dataset of different sizes. We choose $k = 256$, $m = 128$, and $\varepsilon = 2$. In Table 1, the experimental result shows that the LCS protocol has a shorter runtime than OLH and HCMS in different sizes of the real dataset. Therefore, we can conclude that the LCS protocol has a much lower time complexity.

Table 1 Runtime in seconds under different dataset sizes

Dataset sizes\protocol	LCS	OLH	HCMS
1.0E+06	44.81	77.04	283.38
3.0E+06	117.50	244.26	925.41
6.0E+06	274.05	513.27	1879.10

8 Conclusion

This paper focuses on human-centered computing in cloud, edge, and fog analyzes the ε-local differential privacy models without a trusted server. However, current LDP protocols have deficiencies in low utility and strict data size requirements. We propose the Laplace Count Sketch protocol, which cannot only protect sensitive data on the client side but also ensure high accuracy and utility, and discuss the reasons for the deficiencies of current LDP protocols. The experimental results show that the proposed protocol has high utility, is suitable for different sizes of datasets, and maintains its utility under different distributions and data domain sizes. The data dictionary for the datasets used in this paper is known; however, the proposed protocol cannot handle datasets with unknown data dictionaries. The next step is to study how to solve these problems, achieve better privacy protection, and protect sensitive data in human-centered computing.

Abbreviations

DP: Differential privacy; LDP: Local differential privacy; GRR: Generalized random response; OLH: Optimal local hash; RAPPOR: Randomized aggregatable privacy-preserving ordinal response; HCMS: Hadamard count mean sketch; LCS: Laplace count sketch; MAPE: Mean absolute percentage error

Acknowledgements

Not applicable.

Authors' contributions

All authors read and approved the final manuscript.

Funding

This work was supported by the National Natural Science Foundation of China (61572034), Major Science and Technology Projects in Anhui Province (18030901025), Anhui Province University Natural Science Fund (KJ2019A0109).

Availability of data and materials

The relevant analysis data used to support the findings of this study are included in the article.

Competing interests

The authors declare that they have no competing interests.

Received: 22 December 2019 Accepted: 18 February 2020
Published online: 17 March 2020

References

1. L. Qi, Q. He, F. Chen, et al., Finding All You Need: Web APIs Recommendation in Web of Things Through Keywords Search. IEEE Trans. Comput. Soc. Syst. **6**(5), 1063–1072 (2019)
2. X. Xu, Y. Li, T. Huang, et al., An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks. J. Net. Comput. Appl. **133**, 75–85 (2019)

3. X. Xu, Q. Liu, Y. Luo, et al., A computation offloading method over big data for IoT-enabled cloud-edge computing. Future Generation Comput. Syst. **95**, 522–533 (2019)
4. L. Qi, Y. Chen, Y. Yuan, et al., A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems. World Wide Web, 1–23 (2019). https://doi.org/10.1007/s11280-019-00684-y
5. Y. Zhang, G. Cui, S. Deng, et al., Efficient Query of Quality Correlation for Service Composition. IEEE Trans.Serv. Comput. (2018). https://doi.org/10.1109/TSC.2018.2830773
6. Y. Zhang, K. Wang, Q. He, et al., Covering-based Web Service Quality Prediction via Neighborhood-aware Matrix Factorization. IEEE Trans. Serv. Comput. (2019). https://doi.org/10.1109/TSC.2019.2891517
7. Y. Zhang, C. Yin, Q. Wu, et al., Location-Aware Deep Collaborative Filtering for Service Recommendation. IEEE Transactions on Systems, Man, and Cybernetics. Systems (2019). https://doi.org/10.1109/TSMC.2019.2931723
8. X. Xu, Q. Liu, X. Zhang, et al., A blockchain-powered crowdsourcing method with privacy preservation in mobile environment. IEEE Trans. Comput. Soc. Syst. **6**(6), 1407–1419 (2019). https://doi.org/10.1109/TCSS.2019.2909137
9. L. Qi, X. Zhang, W. Dou, et al., A two-stage locality-sensitive hashing based approach for privacy-preserving mobile service recommendation in cross-platform edge environment. Future Generation Comp. Syst. **88**, 636–643 (2018)
10. C. Dwork, F. McSherry, K. Nissim, et al., in Theory of Cryptography Conference. Calibrating noise to sensitivity in private data analysis, 265–284 (Springer, 2006)
11. S. Ruggles, C. Fitch, D. Magnuson, et al., Differential privacy and census data: implications for social and economic research. AEA Pap Proc. **109**, 403–408 (2019). https://doi.org/10.1257/pandp.20191107
12. Facebook's privacy problems: a roundup, https://www.theguardian.com/technology/2018/dec/14/facebook-privacy-problems-roundup. Accessed 10 Oct 2019.
13. 'The Snappening' Is Real: 90,000 Private Photos and 9,000 Hacked Snapchat Videos Leak Online, https://www.thedailybeast.com/the-snappening-is-real-90000-private-photos-and-9000-hacked-snapchat-videos-leak-online?ref=scroll. Accessed 10 Oct 2019.
14. N. Wang, X. Xiao, Y. Yang, et al., Collecting and Analyzing Multidimensional Data with Local Differential Privacy. 2019 IEEE 35th Int. Conf. Data Eng (ICDE), 638–649. IEEE (2019)
15. Ú. Erlingsson, V. Pihur, A. Korolova, Rappor: Randomized Aggregatable Privacy-Preserving Ordinal Response. Proc. 2014 ACM SIGSAC Conf. Comput. Commun Secur - CCS '14, 1054–1067 (2014)
16. Differential Privacy Team, Apple, Learning with Privacy at Scale. (2016)
17. M.E. Gursoy, A. Tamersoy, S. Truex, et al., Secure and Utility-Aware Data Collection with Condensed Local Differential Privacy. arXiv preprint arXiv **1905**, 06361 (2019)
18. G. Cormode, S. Muthukrishnan, An improved data stream summary: the count-min sketch and its applications. J. Algorithms **55**(1), 58–75 (2005). https://doi.org/10.1016/j.jalgor.2003.12.001
19. G. Fanti, V. Pihur, Ú. Erlingsson, Building a RAPPOR with the unknown: Privacy-preserving learning of associations and data dictionaries. Proc. Privacy Enhancing Technol. **2016**(3), 41–61 (2016). https://doi.org/10.1515/popets-2016-0015
20. B. Ding, J. Kulkarni, S. Yekhanin, Collecting telemetry data privately. Adv. Neural Inf. Process. Syst., 3571–3580 (2017)
21. T. Wang, B. Ding, J. Zhou, et al., Answering Multi-Dimensional Analytical Queries under Local Differential Privacy. Proc. 2019 Int. Conf. Manage. Data - SIGMOD '19, 159–176 (2019)
22. T. Wang, J. Blocki, N. Li, et al., Locally differentially private protocols for frequency estimation. In, 26th USENIX Secur. Symp., 729–745 (2017)
23. J. Acharya, Z. Sun, H. Zhang, Hadamard response: Estimating distributions privately, efficiently, and with little communication. arXiv preprint arXiv **1802**, 04705 (2018)
24. M. Joseph, A. Roth, J. Ullman, et al., Local differential privacy for evolving data. In, Adv. Neural Inf. Process. Syst., 2375–2384 (2018)
25. T. Wang, Z. Li, N. Li, et al., Consistent and accurate frequency oracles under local differential privacy. arXiv preprint arXiv **1905**, 08320 (2019)
26. L. Qi, X. Zhang, S. Li, et al., Spatial-temporal data-driven service recommendation with privacy-preservation. Inform. Sci. **515**, 91–102 (2019). https://doi.org/10.1016/j.ins.2019.11.021
27. X. Xu, Y. Xue, L. Qi, et al., An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles. Future Generation Comput. Syst. **96**, 89–100 (2019)
28. R. Bassily, A. Smith, Local, private, efficient protocols for succinct histograms. In, Proc. Forty-seventh Annu. ACM Symp. Theory Comput., 127–135. ACM (2015)
29. X. Ren, C.-M. Yu, W. Yu, et al., LoPub: High-Dimensional Crowdsourced Data Publication with Local Differential Privacy. IEEE Trans. Inform. Forensics Secur. **13**(9), 2151–2166 (2018)
30. T. Wang, M. Xu, B. Ding, et al., Practical and Robust Privacy Amplification with Multi-Party Differential Privacy. arXiv preprint arXiv **1908**, 11515 (2019)
31. X. Gu, M. Li, Y. Cheng, et al., PCKV: Locally Differentially Private Correlated Key-Value Data Collection with Optimized Utility. arXiv preprint arXiv **1911**, 12834 (2019)
32. K. Nissim, S. Raskhodnikova, A. Smith, Smooth sensitivity and sampling in private data analysis. In, Proc. Thirty-ninth Annu. ACM Symp Theory Comput., 75–84. ACM (2007)
33. Y. Wang, X. Wu, D. Hu, in EDBT/ICDT Workshops. *Using Randomized Response for Differential Privacy Preserving Data Collection*, 1558–2016 (2016).
34. A. Roth, C. Dwork, The algorithmic foundations of differential privacy. Foundations and Trends® in. Theor. Comput. Sci. **9**(3-4), 211–407 (2014). https://doi.org/10.1561/0400000042
35. S.L. Warner, Randomized response: a survey technique for eliminating evasive answer bias. J. Am. Stat. Assoc. **60**(309), 63–69 (1965). https://doi.org/10.1080/01621459.1965.10480775
36. S.F. Steven Ruggles, *Ronald Goeken, Josiah Grover, Erin Meyer, Jose Pacas and Matthew Sobek.: IPUMS USA: Version 9.0* (IPUMS, Minneapolis, MN, 2019)

Publisher's Note