Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

An efficient online histogram publication method for data streams with local differential privacy*

Tao TAO[1,2,3], Funan ZHANG[1,2,3], Xiujun WANG[†‡1,2], Xiao ZHENG[1,2], Xin ZHAO[4]

[1]*School of Computer Science and Technology, Anhui University of Technology, Maanshan, China*

[2]*Anhui Engineering Research Center for Intelligent Applications and Security of Industrial Internet, Maanshan, China*

[3]*Engineering Research Institute, Anhui University of Technology, Maanshan, China*

[4]*Shengli No.1 Middle School of Dongying City, Shandong, China*

[†]E-mail: wxj@mail.ustc.edu.cn

Received May 23, 2023; Revision accepted Jan. 18, 2024; Crosschecked

unedited

Abstract: Many areas are now experiencing data streams that contain privacy-sensitive information. Although the sharing and release of these data are of great commercial value, if these data are released directly, the private user information in the data will be disclosed. Therefore, how to continuously generate publishable histograms (meeting privacy protection requirements) based on sliding data-stream windows has become a critical issue, especially when sending data to an untrusted third party. Existing histogram publication methods are unsatisfactory in terms of time and storage costs, because they must cache all elements in the current sliding window (SW). Our work addresses this drawback by designing the Efficient Online Histogram Publication (EOHP) method for LDP data streams. Specifically, in the EOHP method, the data collector first crafts a histogram of the current SW using an approximate counting method. Second, the data collector reduces the privacy budget by using the Optimized Budget Absorption (OBA) mechanism and adds appropriate noise to the approximate histogram, making it possible to publish the histogram while retaining satisfactory data utility. Extensive experimental results on two different real datasets show that the EOHP algorithm significantly reduces the time and storage cost and improves data utility compared to other existing algorithms.

Key words: Data stream; Differential privacy; Sliding windows; Approximate counting
https://doi.org/10.1631/FITEE.2300368 **CLC number:**

1 Introduction

With the rapid development of the internet and big data technologies (Dwork, 2008), real-time data streams frequently appear in many application domains. For example, many large hospitals must release a daily update on the current infection rate of a certain contiguous disease in a contiguous area, so the government and local residents can respond in an appropriate and timely manner. In another example, a search engine might have to share information related to recent user interest trends with manufacturing companies to help them design new products or develop precise sales plans (Liu et al., 2022). Of course, all these kinds of data grow quickly and continuously (Sultani and Ghani, 2015; Liu et al., 2020). It is very meaningful for us to understand such a data stream better with the aim of extracting critical demographic information and trend information(Wang

‡ Corresponding author

* Project supported by the Anhui Provincial Natural Science Foundation (No. 2108085MF218), the Anhui Province University Natural Science Research Project (No. 2022AH040052), the University Synergy Innovation Program of Anhui Province (No. GXXT-2023-021), the Key Program of the Natural Science Foundation of the Educational Commission of Anhui Province of China (No. 2022AH050319), and the National Natural Science Foundation of China (Nos. 62172003 and 61402008)

ORCID: Xiujun WANG, https://orcid.org/0000-0002-8758-5763

© Zhejiang University Press 2024

et al., 2023). At the same time, the data contains private, personal information. If this information is released directly without protection, private personal information leakage can occur. Thus, how to rapidly extract critical cluster information from real-time data streams without compromising individual privacy is a fundamental challenge in the privacy protection research.

1.1 Background for privacy protection in data streams

Currently, there are roughly three approaches to protecting data privacy: anonymization (Narayanan and Shmatikov, 2008), encryption (Dwork et al., 2014), and differential privacy(DP) (Dwork et al., 2014; Yang et al., 2018; Wang et al., 2018; Liu et al., 2021). Among them, DP defines a very strict attack model and has a rigorous mathematical theory proof that can support quantitative analysis of the background knowledge possessed by attackers. With DP, an appropriate amount of noise can be added to the data to prevent an attacker from obtaining private information, regardless of the amount of background knowledge the attacher possesses. Therefore, DP has become a hot research topic in data privacy protection. DP can be further divided into two categories: centralized differential privacy (CDP) and local differential privacy (LDP) (Dwork, 2008). CDP requires a trusted third party, which may not be available in practical applications. In reality, such a trusted third party may seem to be reliable, but will actually, with no hesitation, leak all such data to others for profit (Dwork, 2008; Wang et al., 2016; Labs and Terry, 2020; Dwork et al., 2010).

Compared to CDP (Narayanan and Shmatikov, 2008; Dwork et al., 2014; Duchi et al., 2013), LDP (Errounda and Liu, 2018; Erlingsson et al., 2019; Cormode et al., 2018; Ren et al., 2022) has the advantage of protecting massive end user privacy data locally without directly relying on a trusted third party. More specifically, within LDP, the user send perturbed data to the third part, rather than the original data, so the third part does not represent a functional part of the process. Hence the original data is retained by the user which is certainly better user privacy protection. Therefore, LDP has been successfully deployed by many well-known corporations; for example, Google (Cormode et al., 2018) uses the RAPPOR version of LDP in Chrome to collect information about users' preferred homepages, and Apple (Thakurta et al., 2017; Cormode et al., 2018) has implemented LDP in recent iOS and Mac OS versions to collect information about user search records.

1.2 Prior Arts and Limitations

So far, the LDP research has focused on data perturbation or data publication. For data perturbation, existing works usually rely on random response mechanisms (Erlingsson et al., 2014), which also serve as a foundation for other related perturbed methods. According to the types of data to be processed, the data publication methods are divided into two categories. The first category contains the methods that are used to handle discrete data, and aim to estimate data frequencies (i.e., the number of occurrences of each possible data value) (Erlingsson et al., 2014; Kairouz et al., 2014). The second category includes a method for continuous data, and attempts to estimate the mathematical means of intervals of interest (Nguyên et al., 2016; Ye et al., 2019).

To the best of our knowledge, References (Nguyên et al., 2016; Yang et al., 2018; Fan and Xiong, 2013; Ren et al., 2022) are the only research works that deal with publishing histograms based on sliding windows (SWs) in a data stream. For example, Geng Yang et al. (Yang et al., 2018) focused on discrete data and studied how to publish histograms by adding appropriate noise. Specifically, they adopted the Budget Absorption (BA) mechanism to dynamically allocate part of the privacy budget to each element in a data stream, in an effort to construct a publishable histogram at each time instance. Liyue Fan et al. (Fan and Xiong, 2013) proposed a framework called FAST for data publication in data streams. In the FAST framework, they developed a sampling component to reduce the original data stream to a subset of sampled elements, from which a publishable histogram could always be constructed. Then they designed a filtering component to determine when to update the sampled elements. F.Z. Errounda et al. (Errounda and Liu, 2018) studied how to combine w-event privacy and LDP, and proposed a continuous location statistics sharing algorithm with LDP. The proposed algorithm first computed the similarity between two consecutive windows, and perturbed this result using random response method (Erlingsson et al., 2014). Fi-

unedited

nally, it generated a histogram by adding additional noise. In (Ren et al., 2022), the authors proposed a privacy budget partitioning framework, LDP-IDS, for histogram publication in real-time data streams. More specifically, in the LDP-IDS framework, they try to divide the privacy budget based on how a data stream varies within a period. This method was able to boost data utility, while leading to a better allocation of the privacy budget.

However, these existing methods typically require buffering all elements contained in each SW (at each time instance), which inevitably leads to a heavy burden on time and storage. Furthermore, they usually need to exhaustively process every element within each window when constructing a publishable histogram, which causes a prohibitive computational burden. It should be noted that time and storage costs are fundamental measurements of algorithms, especially for online algorithms in real-time data streams where massive data arrive continuously at a very fast speed (Wang et al., 2016; Kim et al., 2019; Qin et al., 2016).

Sooeon Lee et al. (Lee et al., 2023) proposed a pioneering study that successfully implemented DP barriers for virtual emotion detection in the maritime transportation system. The authors ingeniously used mobile robots and drones to maximize the number of privacy barriers, thereby enhancing the systemąŕs security. H. Kim et al. (Kim et al., 2020) proposed a groundbreaking system named 5G-I-VEmoSYS, which integrates artificial intelligence and 5G technology to enable virtual emotion detection in maritime transportation systems. However, these methods were not suitable for application in our work and were not compared with it. Salna Joy et al. (Joy et al., 2023) introduced a smart security patrol robot based on Raspberry Pi that can use sound sensors and night vision cameras to monitor and protect a designated area. The robot can patrol a preset path, and if it detects any abnormal sound or face, it will send real-time video and location information to the user, so the user can take action in a timely manner. Because we use local differential privacy (LDP) to protect the data in our work, we can resist attacks based on background knowledge and prevent third-party leakage of user privacy. Therefore, the methods just discussed were not suitable for application in our work and were not compared with it.

1.3 Proposed solution and main contributions

To address the above challenges, we propose the Efficient Online Histogram Publication algorithm (EOHP) for LDP data streams, and combine the approximate counting methods with LDP. Our goal is to achieve online histogram publication, while avoiding disclosure of private user data by an untrusted third privacy due to device failure or profit seeking. Specifically, the proposed algorithm consists of three key steps as follows.

First, all users locally perturb their own data and then send the perturbed data to the collector. Consequently, we can prevent an untrusted third party from performing other related activities that compromise user privacy. Second, the data collector adaptively uses the approximate counting method (Datar et al., 2002) to quickly generate estimated histograms for each SW at each time instance. Therefore, we can avoid the heavy storage costs of caching each SWs, and also provide a well-estimated histogram with a guaranteed error bound, which can be further used in the next step to generate a publishable histogram. Third, we designed an Optimized Budget Absorption (OBA) mechanism that adds appropriate noise based on the error bound, to meet DP protection requirements. This step allows privacy allocation to proceed at a steady rate and also avoids excessive consumption of the privacy budget.

1.4 Summary of Experimental results

We extensively compare the real performance of our algorithm (EOHP) with three other state-of-the-art algorithms DDHP (Yang et al., 2018), k-means-dp (Zhang et al., 2014), and ClSA-LDP (Errounda and Liu, 2018) for histogram publication in data streams. In this section, we summarize the algorithms' performance over typical real-world datasets (UK Car Accident Dataset and New York Taxi Dataset) to show the superiority of our algorithm.

- *Time cost*

Using the same privacy budget, the average time cost of the EOHP algorithm for processing the UK Car Accident Dataset is about 0.54×10^5 ms, while the other algorithms require at least 2.18×10^5 ms. Thus, it can be concluded that the time cost of the EOHP algorithm is one quarter of the other algorithms. This experimental improvement verifies the

unedited

theoretical analysis we present in section 4.2.4.

• *Storage cost*

With the same privacy budget, the average storage cost of the EOHP algorithm over the New York Taxi Dataset is approximately 2.85×10^5 bits, while the other algorithms require at least 5.13×10^5 bits. Therefore, the EOHP algorithm outperforms the others by reducing the storage cost by a factor of about $\frac{11}{25}$, in comparison with other algorithms. This experimental result also verifies the theoretical proprieties of the EOHP algorithm as shown in Section 4.2.5.

• *Data utility*

We use mean squared error (MSE) to measure the data utility of the four algorithms. Using the same privacy budget, the MSE of the EOHP algorithm is only about 33.5% of the MSE of the other algorithms over the two datasets. More specifically, for the UK Car Accident Dataset, the MSE of the EOHP algorithm is about 39.0% of the maximum availability of the other three algorithms; for the New York Taxi Dataset, the MSE of the EOHP algorithm is about 28.0% of the ClSA-LDP algorithm. This improvement is further supported by the theoretical analysis in Section 3.5.

In summary, using identical privacy budgets, the proposed EOHP algorithm can not only reduce the time and storage cost significantly, but can also achieve much better data utility.

The rest of this paper is organized as follows. Section 2 presents the required definitions of DP, data streams, and SW. In Section 3, we present the EOHP algorithm and analyze its theoretical proprieties. In Section 4, we conduct extensive experiments on two typical real datasets and then show the EOHP algorithm superiority over other algorithms. Finally, we conclude the paper in Section 5.

2 Preliminaries

In this section, we first present the definitions of DP, data streams, and SWs, which are all foundations of our work. We list all the frequently used symbols in Table. 1.

2.1 Sliding Window Model

A SW (Errounda and Liu, 2018) in a data stream is a fixed size interval over a constant stream of data that generates a fixed size window at each passing time instance, and each window generates a histogram. New data arrives and old data expires, constantly updating and publishing histograms that satisfy DP in real time. Fig. 1 shows the process of publishing a satisfactory DP histogram of data streams when using the sliding-window model.

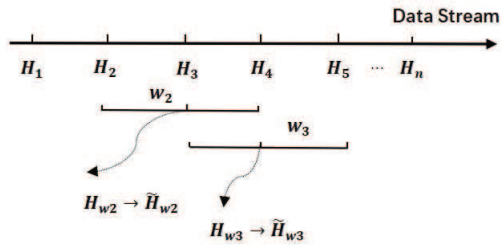

Fig. 1 Histogram publishing process using the sliding-window model

For the convenience of discussion, the parameters involved in this paper are defined as follows:

A real-time data stream is denoted by H:

$$H = \{H_1, H_2, \cdots, H_i | 1 \leq i \leq N\}. \tag{1}$$

There are d attributes of user i:

$$x(i) = \{x_1(i), x_2(i), \cdots, x_d(i)\} \ \ (i \in [1, N]). \tag{2}$$

Data in the SW at time instance i is represented by

$$H_{w_i} = \{H_i, H_{i+1}, \cdots, H_{i+w-1}\}. \tag{3}$$

Preliminary histogram obtained by the approximate counting algorithm at time instance i is denoted by

$$\bar{H} = \{\bar{H}_1, \bar{H}_2, \cdots, \bar{H}_i\}. \tag{4}$$

The histogram protected by the EOHP algorithm at time instance i is represented as

$$\widetilde{H} = \{\widetilde{H}_1, \widetilde{H}_2, \cdots, \widetilde{H}_i\}. \tag{5}$$

2.2 DP

DP is a strong privacy protection technique based on strict mathematical theory. Specifically, DP can quantitatively analyze the attacker's background knowledge; even if an attacker has all the data information except the target data, after DP protection, the attacker cannot obtain the precise target data. The research directions of DP techniques include CDP (Narayanan and Shmatikov, 2008; Dwork et al., 2014; Duchi et al., 2013) and

unedited

Table 1 Definition of relevant symbols

Symbol	Description
H	A data stream, i.e., an infinite sequence of data elements: $H = H_1, H_2, \cdots$
N	Number of data elements in the data set
w	Sliding window size
H_{w_i}	The sliding window at time instance i, i.e., $H_{w_i} = e_{i-w+1}, e_{i-w+2}, \cdots, e_i$.
$\bar{H}_i$	Approximate histogram at time instance i
$\widetilde{H}_i$	Histogram of additive noise at time instance i
L	The number of statistical intervals
r	The number of buckets (a data structure) used by the EOHP algorithm
ε	Privacy budget

LDP (Errounda and Liu, 2018; Erlingsson et al., 2019; Cormode et al., 2018; Ren et al., 2022); despite their different data processing methods, both can achieve very high data utility. The two mechanisms are described below.

2.2.1 CDP

In CDP, users send their real data to the data collector on demand; then, the collector publishes histograms that meet DP protection requirements. CDP not only processes data in real time to extract valuable information for distribution, but also protects user privacy. By adding a very small amount of noise to the data, a very high level of data utility can be achieved. However, CDP has a drawback: the third-party collector must be credible, otherwise, private data is still at risk of being leaked by untrusted third parties. The relevant definitions of CDP as applied to data streams are as follows.

Definition 1 *ε-DP.* A non-interactive privacy algorithm A gives ε-DP if, for any datasets H and H' that differ on at most one recode, and for any possible anonymized dataset $y \in range(A)$ that satisfies Eq.6

$$\frac{Pr(A(H) = y)}{Pr(A(H') = y)} \leq e^{\varepsilon}, \tag{6}$$

the smaller the ε value is, the closer the probability of the $A(H) = y$ and $A(H') = y$ output results, which indicates that algorithm A has stronger data privacy protection strength.

Definition 2 *(Global Sensitivity)* For any function f, $H \in R^d$, the global sensitivity of function f is

$$\Delta f = \max_{\{H,H'\}} \{||f(H) - f(H')||\}, \tag{7}$$

where H and H' are adjacent datasets and d is the query scope.

2.2.2 LDP

LDP differs from CDP because it can not only defend against maximum background knowledge attacks, but also avoid privacy attacks from untrusted third parties. In LDP, the user first perturbs their real data through the mechanism M, and then sends the $M(x)$ obtained from the perturbation process to the collector, who statistically analyzes and releases it.

Definition 3 *LDP.* Given a perturbation mechanism M, a definition domain $Dom(M)$, and a value domain $Range(M)$, the mechanism M satisfies ε-LDP if, given any input pair: x and x' both of which satisfy the definition domain $Dom(M)$, the output $y \in Range(M)$ after algorithmic perturbation satisfies Eq.8.

$$\frac{Pr(M(x) = y)}{Pr(M(x') = y)} \leq e^{\varepsilon}. \tag{8}$$

The ε (privacy budget) value is inversely proportional to the degree of privacy protection and positively proportional to data utility. The smaller the ε value, the better the privacy protection of the data and the worse the data utility. Therefore, because the attacker also cannot determine what the input result is based on the output, further strong protection of data privacy is obtained. There are two important combinatorial properties in LDP: Sequence Combinability and Parallel Combinability.

Sequence Combinability. Given a data stream H, $H = \{H_1, H_2, \cdots, H_i \mid 1 \leqslant i \leqslant N\}$ and N privacy algorithms $A = \{A_1, A_2, \cdots, A_i | 1 \leq i \leq N\}$, the privacy parameters corresponding to the algorithms in privacy algorithm A are $\varepsilon_1, \varepsilon_2, \ldots \varepsilon_i$, respectively. When these algorithms are applied to data set H, they form a combination of $A_1(H), A_2(H), \ldots A_i(H)$ and provide $\sum_{i=1}^{n} \varepsilon_i$-

unedited

difference privacy.

Parallel Combinability. Given a data stream H, $H = \{H_1, H_2, \cdots, H_i | 1 \leq i \leq N\}$, and assuming that each subset of the data set is independent of all others, there are n privacy algorithms $A = \{A_1, A_2, \ldots A_i \mid 1 \leqslant i \leqslant N\}$, and the corresponding privacy parameters of the algorithms in A are $\varepsilon_1, \varepsilon_2, \ldots \varepsilon_i$, respectively. When these algorithms operate simultaneously on the independent set H, they form a combination of $A_1(H), A_2(H), \ldots A_i(H)$ and provide $\max(\varepsilon_i)$-difference privacy.

2.3 Noise addition mechanism

This section focuses on the LDP and CDP mechanisms for adding noise to real-time data streams.

2.3.1 CDP noise addition mechanism

The Laplacian mechanism protects user privacy information by adding Laplacian-distributed random noise to real data; the perturbed data is then sent to the data collector.

Theorem 1 *Laplace Mechanism.* For any function f, $H \in R^d$, if the output of algorithm A satisfies Eq.9, then A satisfies ε-difference privacy.

$$A(H) = f(H) + \left\langle Lap_1\left(\frac{\Delta f}{\varepsilon}\right), \cdots, Lap_d\left(\frac{\Delta f}{\varepsilon}\right)\right\rangle, \tag{9}$$

where $Lap\left(\frac{\Delta f}{\varepsilon}\right)$ is a separate Laplace function, and scale parameter $b > 0$, can be used to measure the added amount of noise. Generally, let $b = \Delta f/\varepsilon$; the larger b is, the smaller ε is, indicating that the data privacy protection strength is better; when b is too large, the noise will also cause poor data utility. Therefore, the algorithm needs a suitable ε to achieve a balance between data protection and data utility.

2.3.2 LDP noise addition mechanism

Currently, the random response technique (Erlingsson et al., 2014) is the dominant perturbation mechanism in LDP. Random response is a foundation perturbation process on the user side of the data; the data received by the data collector is the perturbed data, so the risk of privacy leakage from an untrusted third party is avoided.

Theorem 2 *Random Response.* For any input $B \in X$, if the response output of $B' \in X$ satisfies Eq.10, then the random response method satisfies ε-difference privacy.

$$P(B' \mid B) = \frac{1}{k-1+\mathrm{e}^s}\begin{cases}\mathrm{e}^{\varepsilon}, & \text{if } B' = B\\ 1, & \text{if } B' \neq B\end{cases} \tag{10}$$

For the case where the variable contains k $(k > 2)$ candidate values, users provide their own true data with probability $\frac{\mathrm{e}^{\varepsilon}}{k-1+\mathrm{e}^{\varepsilon}}$ and randomly provide any one of the $k-1$ false data with probability $\frac{1}{k-1+\mathrm{e}^{\varepsilon}}$. Compared with the Laplace mechanism, random response not only ensures that the data is scrambled before it exits the mobile device, but also reduces the addition of noise at a later stage, making it more capable of meeting the strength requirement of data privacy protection.

3 EOHP Method

In this section, we propose an EOHP algorithm that can process LDP data streams online while satisfying different privacy strength requirements.

3.1 Problem description

Given a data stream $H_i = \{H_1, H_2, \cdots, H_i | 1 \leq i \leq N\}$, for each H_i at time moment i, release a noised version $\widetilde{H}_i$ in real time such that the continued release $\widetilde{H}_i = \{\widetilde{H}_1, \widetilde{H}_2, \cdots, \widetilde{H}_i | 1 \leq i \leq N\}$, satisfies ε-DP.

3.2 EOHP Algorithm

Data protection for a given data stream H consists of three main steps: perturbation processing, approximation processing, and noise addition processing. First, the user adopts the random response method to perturb their own data on the local side and then sends it to the data collector. Second, the collector uses Algorithm 2 to quickly obtain an approximate histogram online at the current SW (note that i is the current time instance). Finally, to improve publishing efficiency, the data collector can decide whether to publish the histogram at the current SW based on the similarity result. When the data in the current window differs significantly from that in the previous window, we choose to publish the histogram in the current window by adding a small amount of noise to the data; otherwise, the data collector publishes a histogram with noise at the previous SW.

unedited

The specific process is shown in Algorithm 1. Please note that, in the following description, a bucket is the combination of two counters that record the number of '1's in a binary data stream and its timestamp.

Algorithm 1 EOHP Algorithm

Input: H_i, T_0, ε

1: Get local side disturbance data $\ddot{H}_i$:
2: **for** each user **do**
3: send your own real data or fake data to the data collector with probability p as Eq.10
4: **end for**
5: Obtain preliminary histogram $\bar{H}_i$:
$\bar{H}_i \longrightarrow$ Approximate counting algorithm($\ddot{H}_i$);
6: **if** $T > T_0$ **then**
7: $\varepsilon_i \longrightarrow$ OBA(ε);
8: Add Laplace noise $\widetilde{H}_i = \bar{H}_i + Lap(1/\varepsilon_i)$;
9: $\widehat{H}_i \longrightarrow$ Dynamic programming grouping;
10: $\widetilde{H}_i = \widehat{H}_i$;
11: **else**
12: $\widetilde{H}_i = \widetilde{H}_{i-1}$;
13: **end if**
14: **return** $\widetilde{H}_i$

Let us explain Algorithm 1. Please note that we use $\ddot{H}_i$ to represent the data perturb on the local side, and $\bar{H}_i$ to represent the preliminary histogram of the current time instance i. Furthermore, we use $\widetilde{H}_{i-1}$ to denote the histogram with noise at the time instance $i-1$, and T_0 to denote the threshold (the initial value of the threshold is determined by pre-sampling). With these notations in mind, we explain Algorithm 1 as follows. Lines 1-4 obtain the local side perturbed data($\ddot{H}_i$), so we can perfectly avoid privacy attacks from any untrusted third party. Line 5 obtains the approximate histogram within the current SW (H_{wi} is the SW at time instance i). With this step, we can very rapidly obtain an approximate histogram ($\bar{H}_i$) for each SW. In lines 6-13, we first compute the similarity between the current SW and the previous one. If this computed value exceeds the predefined threshold, we publish a new histogram by adding suitable noise to the generated approximate histogram ($\widetilde{H}_i$) in lines 6-10. Conversely, if the similarity value does not surpass the threshold, we republish the previous histogram from the time instance $i-1$ in lines 11-13 (where i denotes the current time instance). In addition, if we decide to publish a new histogram, we use the dynamic grouping method to reduce MSE (thus boosting data utility).

3.3 Approximate counting algorithm

In this phase, to quickly obtain the statistical value of each interval in SW at the current time instance, the data collector uses the approximate counting algorithm to initially process the perturbed data, because this process only needs to cache the data in the current SW. This saves significant time and space costs, and can also improve the overall performance of the EOHP algorithm. The algorithm process is shown as Algorithm 2.

Algorithm 2 Approximate counting algorithm

1: **if** $len(B) > 0$ and $t - w == B[0]['time']$ **then**
2: del $B[0]$;
3: **end if**
4: $bit = f.readline()$;
5: **if** $bit == '1\backslash n'$ **then**
6: $B = \{'time' : i+1, 'sum' : 1\}$;
7: **for** $i = len(B) - 1$ **to** $max - 1$ **do**
8: **if** $B[i]['sum'] == B[i-max]['sum']$ **then**
9: $B[i-max]['sum'] += B[i-max+1]['sum']$
10: $B[i-max]['time'] = B[i-max]['sum']$
11: del $B[i-max+1]$
12: **end if**
13: **end for**
14: **end if**
15: **if** $len(B) > 0$ **then**
16: **for** $i = 0$ **to** $len(B)$ **do**
17: sum += $B[i]['sum']$;
18: sum -= $B[0]['sum']/2$;
19: **end for**
20: **end if**
21: **return** sum

Let us explain Algorithm 2. Please note that we use $B = []$ to represent the buckets, and use max to represent the maximum number of buckets. Furthermore, t denotes the current time instance i, sum denotes the number of '1's in a binary data stream, and $time$ denotes the timestamp for a bucket. With these notations in mind, we can explain Algorithm 2 as follows. Lines 1-3 determine whether there is an expired bucket in the current window; if so, the bucket is deleted. Lines 4-5 read the file and create a bucket, while setting the end timestamp equal to current time. In lines 6-10, we first update the bucket in the SW; if there are max buckets of the same size, the two oldest buckets are merged at the

unedited

current SW and the expired bucket is deleted. In lines 11-15, the approximate value of each interval in the current window at the current time instance i is obtained. In addition, the time is divided into 9 intervals, and a total of 9 runs are performed to obtain the approximate histogram $\bar{H}_i$ at the current time.

Because Algorithm 2 is an approximate counting method in real-time data streams, there is a certain error between the final statistics and the true frequency number, and the following conclusions can be drawn according to Algorithm 2.

Theorem 3 For $H_i \in H$, the approximate counting frequency satisfies: $\left|\bar{H}_i - H_i\right| \leq \frac{1}{r-1}\left|H_i\right|$.

Proof: When the leftmost bucket has only one '1's in the query range, the query error result is the maximum. Specifically, getting the real statistics of the data in the SW at the current time instance i: $\ddot{H}_i$=$1+(r-1)(1+2+\cdots+2^{j-2}+2^{j-1}) = 1+(r-1)(2^j-1)$, where 2^j is the size of the bucket with the earliest timestamp and r is the preset number of buckets of the same size to be determined. At this time, $2^{j-1}-1$ is overestimated, and we can obtain $|\bar{H}_i - \ddot{H}_i| = 2^{j-1}-1$. Therefore, it can be proved that $|\bar{H}_i - \ddot{H}_i| \leq \frac{1}{r-1}|H_i|$. In summary, to ensure that errors do not affect the accuracy of the final data release, the preset number of buckets of the same size should not exceed $\frac{\ddot{H}_i}{\left|\bar{H}_i - \ddot{H}_i\right|}+1$. Based on this, we can constantly adjust the preset number of buckets of the same size to optimize the final statistical results.

3.4 Privacy Allocation Policy

We propose an Optimized Budget Absorption (OBA) mechanism for real-time data streams. In OBA, a privacy budget, denoted as ε_i, is assigned in advance to the data within the SW at each time instance, where $\varepsilon_i = \frac{\varepsilon}{\text{len(data)}-w+1}$. If the correlation distance T between the sliding-window data at adjacent moments is less than a threshold T_0, the previous moment's noise histogram $\widetilde{H}_{i-1}$ is published, its privacy budget ε_i is retained, and the $\varepsilon_i = \frac{\varepsilon-\varepsilon_i}{\text{len(data)}-(i-1)-w+1}$ is used to publish the histogram with noise that meets the noise conditions at the next time instance. Otherwise, the ε_i value for publishing the current time instance of histogram $\widetilde{H}_i$ is obtained, constantly updating ε_i for data noise processing at each time instance. This approach allows maximization of privacy budgets saving upfront, thus addressing the problem of excessive consumption of privacy budgets. To visualize how OBA allocates a privacy budget, Fig. 2 shows the OBA privacy budget allocation mechanism process when the window size $w = 3$.

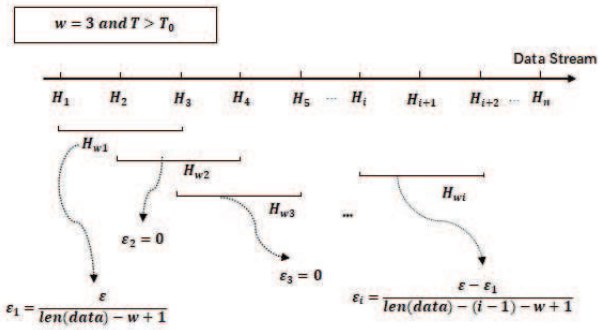

Fig. 2 The OBA privacy budget allocation process

As seen in Fig. 2, the data must be noised and published at the first time instance, and the privacy budget to be allocated is $\varepsilon_1 = \frac{\varepsilon}{\text{len(data)}-w+1}$. However, the histogram in the window from time instance 2 to time instance $i-1$ does not meet the conditions for adding noise, and the pre-allocated privacy budget will always be retained. At this time, the histograms in the window meet the noise condition at time instance i and are allocated $\varepsilon_i = \frac{\varepsilon-\varepsilon_1}{\text{len(data)}-(i-1)-w+1}$. If the histogram at a subsequent time continuously meets conditions for adding noise, the privacy budget value is allocated. Otherwise, $\varepsilon_i = \frac{\varepsilon-\varepsilon_i}{\text{len(data)}-i+w+1}$ is constantly updated to publish the histogram that satisfies the noise condition at the next time instance.

3.5 Algorithm performance analysis

To analyze the performance of the proposed EOHP algorithm, it is compared with DDHP (Yang et al., 2018), k-means-dp (Zhang et al., 2014), and ClSA-LDP (Errounda and Liu, 2018) algorithms. The DDHP algorithm directly adds noise to the accurate histogram, while adopting the BA privacy budget allocation strategy. The k-means-dp algorithm combines a MapReduce framework with k-means clustering to add noise to clustered data for publication. The ClSA-LDP algorithm adds noise to the accurate histogram with the random perturbation processing for publication. The results are shown in TABLE 2.

As can be seen from Table. 2, with the same

unedited

Table 2 Comparison of histogram publication algorithms

Method name	Design idea	Time cost	Storage cost	Mean squared error (MSE)
DDHP	Adding noise to the exact histogram and publishing.	$O(w)$	$O(w\log(\mathrm{L}))$	$1.5*10^5-3.5*10^5$
k-means-dp	Combining MapReduce framework with k-means clustering to add noise to clustered data for publishing	$O(w)$	$O(w\log(\mathrm{L}))$	$1.2*10^5-2.6*10^5$
CLSA-LDP	Adding noise to the exact histogram with random perturbation processing for release.	$O(w)$	$O(w\log(\mathrm{L}))$	$0.8*10^5-1.7*10^5$
EOHP	Adaptive noise enhancement and publishing of the approximate histogram online after random combustion.	$O(L\log(w))$	$O(\frac{L}{r-1}\log^2(w))$	$0.6*10^5-1.4*10^5$

privacy budget, the MSE, time, and storage cost of the EOHP algorithm are all smaller than those of the other three algorithms. Specifically, for the time and storage cost, the other three algorithms need to cache w data in the whole window and then process it. Due to the time needed process w data, the time cost is $O(w)$. The storage cost of w data is $O(w\log(\mathrm{L}))$, because all data in the window is divided into L intervals, and w data needs $w\log(\mathrm{L})$ bits. In contrast, the EOHP algorithm uses the approximate counting method to cache and process w data in the window online. The time and storage costs of this algorithm are $O(L\log(w))$ and $O(\frac{L}{r-1}\log^2(w))$, respectively. The reason for this is that the approximate counting method transforms each data point into a binary representation and then performs the computation, so the time cost is $L\log(w)$. In addition, the window needs to cache $\log(w)$ buckets, each bucket needs to use $\log(w)$ bits, and each binary bit is divided into L intervals, so the storage cost needs $L\log^2(w)$ bits. Furthermore, the statistical error value in the approximate counting method is $\frac{1}{r-1}$ times the real value. Therefore, the total storage costs of the EOHP algorithm are $\frac{L}{r-1}\log^2(w)$ bits. For the MSE, compared to other algorithms, the EOHP algorithm adopts the dynamic planning grouping method to reduce the overall publication error, which can improve data utility. Therefore, the MSE of the EOHP algorithm is smaller of that of the other algorithms over the two datasets.

4 EXPERIMENT

In this section, we evaluate the performance of the proposed EOHP algorithm and compare it with three recent algorithms for privacy protection in real-time data streams.

4.1 Experimental settings

All experiments in this paper were conducted on a 64 GB RAM Intel(R) Core(TM) i9 10980XE CPU @ 3.0 GHz 3.50 GHz, and the experimental codes are implemented in Python.

During the experiment, we utilized two real datasets: the UK Car Accident Dataset and the New York Taxi Dataset. Specifically, the UK Car Accident Dataset contains 1048576 traffic accident records, where the age range of the car accident is $[0, 90]$ and the monthly revenue range is $[2000, 20000]$; the New York Taxi Dataset contains 1640597 passenger count records, where the passenger counting interval of all these records is $[0, 6]$.

The monthly income range of the UK Car Accident Dataset is from 2,000 to 20,000, which is also a relatively wide range. In the experiment, this range was used to compare the spatial complexity of different algorithms. The passenger range of the New York Taxi Dataset is from 0 to 6, which is a relatively narrow range, and the data is more concentrated. In the experiment, this range was used to compare the temporal complexity of different algorithms in a practical application setting, and observe the change in data error as a result of changing the value of T_0 (maximum distances between SW data at adjacent moments).

We compare our EOHP algorithm with three algorithms for histogram publication: the DDHP, k-means-dp, and CLSA-LDP algorithms. These algorithms are either state-of-the-art solutions for dynamic data or recent and relevant algorithms for data streams (Yang et al., 2018)(Zhang et al., 2014)(Errounda and Liu, 2018). To make these algorithms more suitable for processing real-time data streams, three main improvements are made our work: First, users obfuscate their personal data using the LDP

unedited

algorithm before sending it to the data collector. Second, to obtain the approximate histogram of the current time instance, the collector adopts the approximate counting method to initially process the perturbed data. Finally, the OBA mechanism is used to allocate a reasonable privacy budget for noise processing of the histogram, which maximizes the privacy budget savings upfront and avoids excessive consumption of the privacy budget.

Our work uses Mean squared error (MSE) (Dwork et al., 2014; Nguyên et al., 2016; Ren et al., 2022) to test the utility of all algorithms and measure the availability of the EOHP algorithm, which is expressed as:

$$Err(H_{wi}) = E\left(\sum_{H_i \in w_i} (H_i - \widetilde{H}_i)\right)^2, \qquad (11)$$

where H_i is the true value of the data at the current time instance i , and $\widetilde{H}_i$ is the value obtained by the EOHP algorithm.

4.2 Experimental results

The data utility of EOHP algorithms in terms of data publishing was measured based on two real-world datasets, different threshold T_0 values, privacy budget ε values, sliding-window size w values, and comparisons with the DDHP (Yang et al., 2018), k-means-dp (Zhang et al., 2014), and CLSA-LDP (Errounda and Liu, 2018) algorithms. The DDHP algorithm adds noise to the accurate histogram and adopts the BA mechanism to allocate reasonable privacy budget. The k-means-dp (Zhang et al., 2014) algorithm adopts the MapReduce framework while using k-means clustering in line point noise addition. The CLSA-LDP (Errounda and Liu, 2018) algorithm first perturbs the true differences in the accurate histogram at an adjacent moment SW, and then adds appropriate noise by using an adaptive privacy distribution mechanism.

4.2.1 The impact of varying threshold T_0

In this group of experiments, w=200, ε=1, and 10,000 records were selected from the New York Taxi Dataset. The purpose is to observe the change in data error as a result of changing the value of T_0 (maximum distances between sliding-window data at adjacent moments). The results are shown in Fig. 3.

Fig. 3 shows the relationship between the T_0

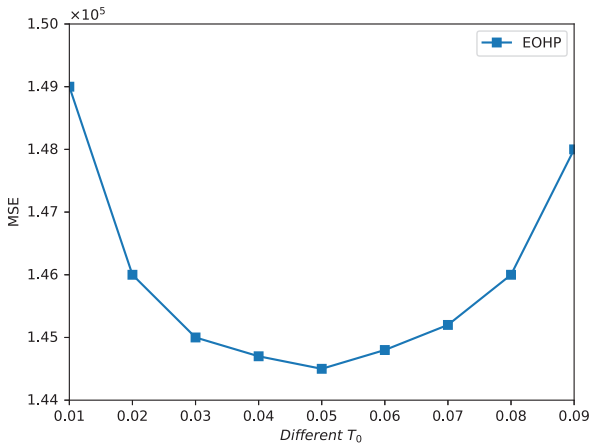

Fig. 3 The Impact of different threshold (T_0) on data utility: MSE analysis for the New York Taxi Dataset

value and the total error of the relevant distance. The T_0 value is a parameter that controls the amount of noise added to the data to achieve DP. The total error is the sum of the noise error and the reconstruction error. The noise error is the difference between the original data and the noisy data. The reconstruction error is the difference between the noisy data and the reconstructed data.[1]

As seen in Fig. 3, as the T_0 value changes, the overall error trend of the correlation distance decreases first and then increases. Therefore, the T_0 value has an impact on the overall error, and the overall error is composed of the noise error and reconstruction error. When the T_0 value is small, the overall error is mainly determined by the noise error, and when the noise error is larger, the overall error is greater. When the T_0 value begins to increase,

[1] The optimal T_0 value depends on the characteristics of the data and the reconstruction algorithm. In our work, we first used the approximate technique algorithm to process the data online and obtain the preliminary histogram of each time interval. In the approximate technique algorithm, we try to find a preset number of same-sized buckets that should not be exceeded. We can continuously adjust the preset number of same-sized buckets to optimize the final statistical results, so the optimal T_0 value will vary according to the preset number of same-sized buckets. In addition, we use the dynamic programming grouping method to reconstruct the data. In the dynamic programming grouping method, we try to find the minimum data grouping error, so we can find the optimal grouping when the data reaches the minimum error, so the optimal T_0 value will vary according to the optimal grouping. Therefore, we find an optimal T_0 value that minimizes the total error of the data by adjusting the preset number of same-sized buckets and finding the optimal grouping.

unedited

the overall error is mainly determined by the reconstruction error, and when the reconstruction error decreases, the overall error will also decrease. As the T_0 value continues to increase, the noise error and the reconstruction error increase, and the overall error increases. As a result of the above analysis, in this experimental validation, we need to find an optimal T_0 value so that the overall error is relatively small. Therefore, the New York Taxi Dataset uses the T_0 value of 0.05 and the UK Car Accident Dataset uses the T_0 value of 0.04.

The highlights of our work include the following: using the approximate counting Algorithm 2 to process data online saves a lot of time and space cost, and adopting the OBA mechanism allocates the privacy budget appropriately, while grouping methods to reduce the overall error and improve the overall data accuracy.

4.2.2 The impact of varying privacy budget ε

In this group of experiments, w=200, different privacy budget ε values were used for the New York Taxi Dataset and the UK Car Accident Dataset, and the impact on data utility was observed by selecting different ε values. The experimental results are shown in Figs. 4 and 5:

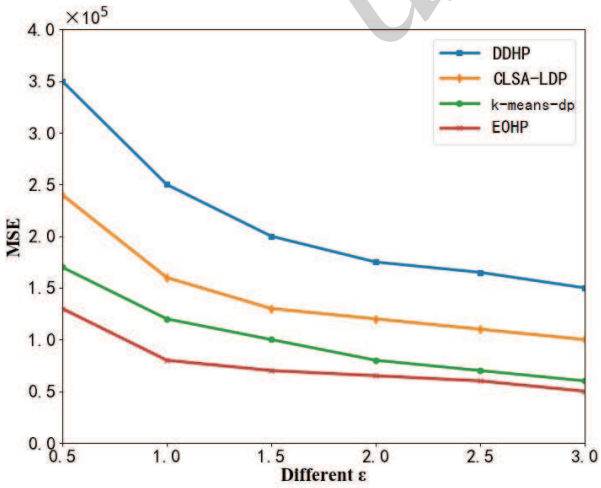

Fig. 4 The Impact of different privacy budget (ε) on data utility: MSE analysis for the New York Taxi Dataset

In Figs. 4 and 5, the total error of the EOHP, DDHP (Yang et al., 2018), k-means-dp (Zhang et al., 2014), and CLSA-LDP (Errounda and Liu, 2018) algorithms all decrease as the ε value increases, which is consistent with Eq.6. It can clearly be seen that the MSE of the DDHP (Yang et al., 2018), k-means-dp (Zhang et al., 2014), and CLSA-LDP (Errounda and Liu, 2018) algorithms are all greater than the EOHP algorithm. The reason is that although the DDHP (Yang et al., 2018) and CLSA-LDP (Errounda and Liu, 2018) algorithms use similarity measurement, they do not use the correlation method to reduce the error when the correlation distance is greater than the threshold, and directly add noise to the data at the current time instance. The k-means-dp (Zhang et al., 2014) algorithm causes a large data error due to the large distance between the noise-added new centroid and the original centroid. In contrast, in EOHP we use the dynamic groupings mechanism to optimize histogram grouping, which attempts to reduce the global data release error. Therefore, the overall error of the EOHP algorithm is the lowest, and the data utility on both the New York Taxi and the UK Car Accident datasets is highest.

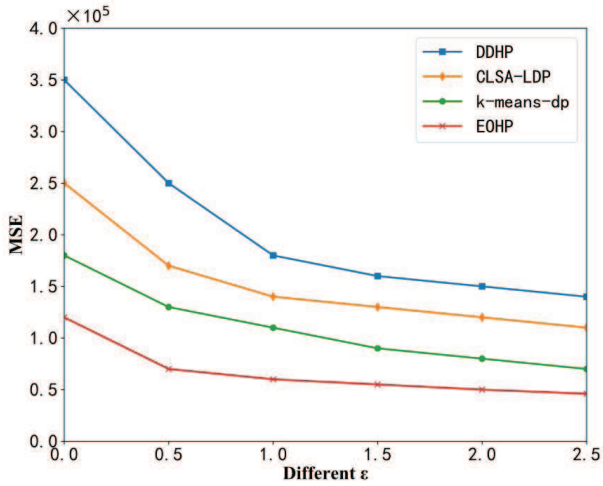

Fig. 5 The Impact of different privacy budget (ε) on data utility: MSE analysis for the UK Crash Dataset

4.2.3 The impact of varying window size w

In this group of experiments, ε=1, different w values were used for the New York Taxi and the UK Car Accident datasets, and a mean squared error trend is observed with the change in w. The experimental results are shown in Figs. 6 and 7.

In Figs. 6 and 7, when w increases, the total error of the EOHP, DDHP (Yang et al., 2018), k-means-dp (Zhang et al., 2014), and CLSA-LDP (Er-

unedited

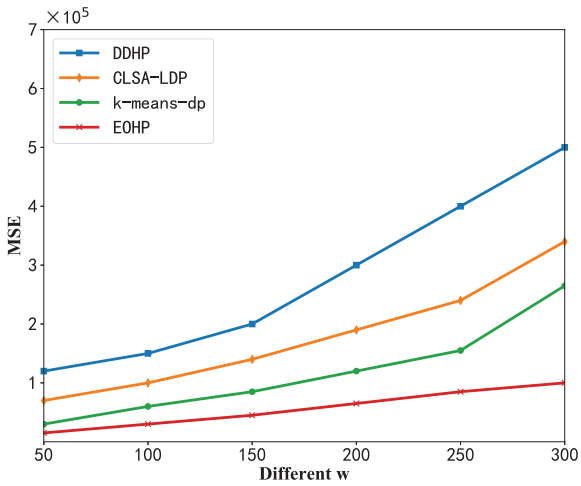

Fig. 6 The Impact of different w values (different window size) on data utility: MSE analysis for the New York Taxi Dataset

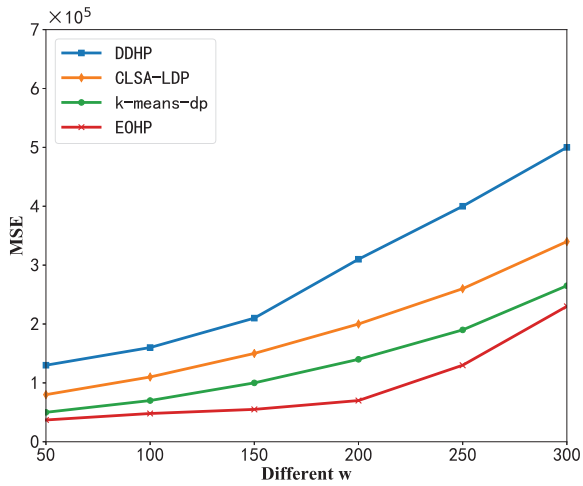

Fig. 7 The Impact of different w values (different window size) on data utility: MSE analysis for the UK Crash Dataset

rounda and Liu, 2018) algorithms increases. Larger w values cause increases in the approximate error and the noise error, which result in a greater overall error. The EOHP algorithm error growth rate is significantly lower than that of the other three algorithms, as shown in the figure. This is a more concise and clearer way to display the main point of this information. The EOHP algorithm first uses the approximate statistical value in the place of noise to avoid the addition of too much noise in the SW. Then the dynamic programming method is used to reduce the overall publication error , so the increasing data error trend will be relatively flat, and the EOHP algorithm average total error is reduced by 50%, compared to the other three algorithms.

4.2.4 Experimental evaluation of the time cost

In this group of experiments, ε=1, $L = 6$, and the EOHP algorithm is compared with other algorithms to observe the time cost using passenger counts from the UK Car Accident Dataset. The results are shown in Fig. 8.

From Fig. 8, it can be concluded that the running time of the EOHP algorithm is less than the other three algorithms, because the EOHP algorithm adopts the Algorithm 2 (the approximate counting algorithm) to process the data stream online, and the processing dimension is an interval of data rather than a single data value. Therefore, the EOHP algorithm requires less run time.

4.2.5 Experimental evaluation of the storage cost

In this group of experiments, ε=1, $L = 1000$, $r = 10$, and the EOHP algorithm is compared with other algorithms to observe the storage cost using the monthly revenue data from the New York Taxi Dataset. The results are shown in Fig. 9.

From Fig. 9, it can be concluded that the storage cost of the EOHP algorithm is less than the other algorithms, because the EOHP algorithm adopts the Algorithm 2 (the approximate counting algorithm) to process the data flow online, and the processing dimension is an interval of data rather than a single data value. Therefore, the EOHP algorithm has a lower storage cost.

In summary, the above experimental results demonstrate that our proposed EOHP algorithm has lower time and storage costs and a smaller data publication error when compared with the existing state-of-the-art histogram publishing methods.

5 Conclusions

In this paper, we propose an EOHP algorithm for LDP data streams. First, we used the random response method to obfuscate user data on the local side, which can avoid privacy attacks by an untrusted third party. Second, we used the approximate counting method to obtain the preliminary histograms at the current SW online, which saves significant time and storage costs. Finally, to avoid

unedited

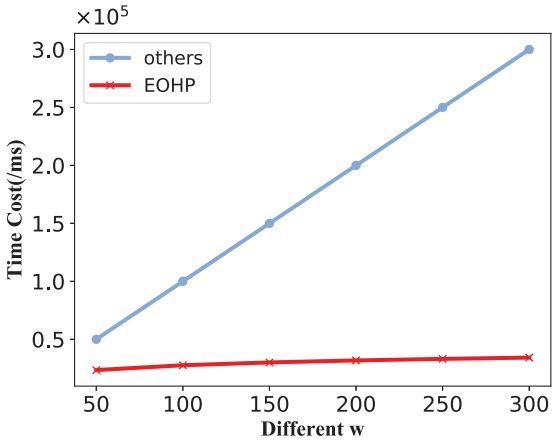

Fig. 8 Comparing time costs: EOHP algorithm versus three other algorithms (DDHP, k-means-dp, and CLSA-LDP)

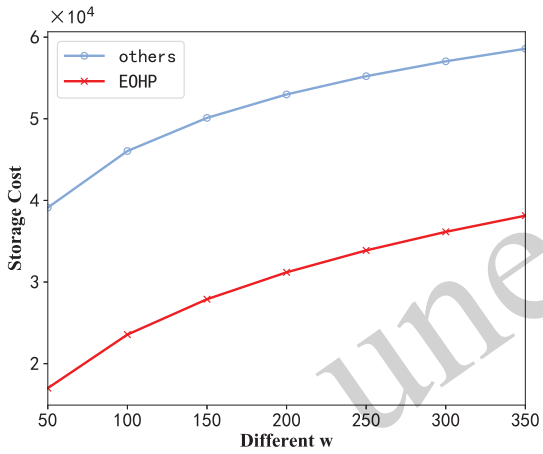

Fig. 9 Comparing storage costs: EOHP algorithm versus three other algorithms (DDHP, k-means-dp, and CLSA-LDP)

excessive consumption of the privacy budget, we propose the OBA mechanism to add a suitable amount of noise to the approximate histogram that will be published, which can achieve the privacy protection strength requirement. In conclusion, compared with the privacy protection algorithms that already exist in real-time data streams, the EOHP algorithm not only avoids privacy leakage by an untrusted third party, but also greatly improves data processing efficiency and achieves better data utility.

In future work, we will study personalized privacy budget allocation algorithms for privacy protection requirements for the data itself. In addition, we will study multi-dimensional data sets and consider more factors, such as data relevance, the dimension importance, and privacy sensitivity, to design more reasonable and effective multi-dimensional data release protection algorithms.

Contributors

Xiujun WANG and Tao TAO designed the research. Funan ZHANG processed the data. Tao TAO, Funan ZHANG, and Xiujun WANG drafted the paper. Xiao ZHENG and Xin ZHAO helped organize the paper, process the images, and verify the results in Sections 3 and 4. Funan ZHANG and Xiujun WANG revised and finalized the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

unedited

References

Cormode G, Jha S, Kulkarni T, et al., 2018. Privacy at scale: Local differential privacy in practice. Proc. Int Conf on Management of Data, p.1655-1658. https://doi.org/10.1145/3183713.3197390

Datar M, Gionis A, Indyk P, et al., 2002. Maintaining stream statistics over sliding windows. *SIAM J Comput*, 31(6):1794-1813. https://doi.org/10.1137/S0097539701398363

Duchi JC, Jordan MI, Wainwright MJ, 2013. Local privacy and statistical minimax rates. Proc IEEE 54th Annual Symp on Foundations of Computer Science p.429-438. https://doi.org/10.1109/FOCS.2013.53

Dwork C, Naor M, Pitassi T, et al., 2010. Differential privacy under continual observation. Proc 42nd ACM Symp on Theory of Computing p.715-724. https://doi.org/10.1145/1806689.1806787

Dwork C, Roth A, 2014. The algorithmic foundations of differential privacy. *Found Trends Theor Comput Sci*, 9(3-4):211-407. https://doi.org/10.1561/0400000042

Dwork C, 2008. Differential privacy: a survey of results. Proc 5th Int Conf on Theory and Applications of Models of Computation, p.1-19. https://doi.org/10.1007/978-3-540-79228-4_1

Erlingsson Ú, Pihur V, Korolova A, 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. Proc ACM SIGSAC Conf on Computer and Communications Security, p.1054-1067. https://doi.org/10.1145/2660267.2660348

Erlingsson Ú, Feldman V, Mironov I, et al., 2019. Amplification by shuffling: from local to central differential privacy via anonymity. Proc 30th Annual ACM-SIAM Symp on Discrete Algorithms, p.2468-2479. https://doi.org/10.1137/1.9781611975482.151

Errounda FZ, Liu Y, 2018. Continuous location statistics sharing algorithm with local differential privacy. Proc IEEE Int Conf on Big Data, p.5147-5152. https://doi.org/10.1109/BigData.2018.8621876

Fan LY, Xiong L, 2014. An adaptive approach to real-time aggregate monitoring with differential privacy. *IEEE Trans Knowl Data Eng*, 26(9):2094-2106. https://doi.org/10.1109/TKDE.2013.96

Joy S, Paulraj RL, Punith M, et al., 2023. A raspberry pi based smart security patrol robot. Proc 7th Int Conf on Computing Methodologies and Communication, p.1140-1145. https://doi.org/10.1109/ICCMC56507.2023.10083908

Kairouz P, Oh S, Viswanath P, 2014. Extremal mechanisms for local differential privacy. Proc 27th Int Conf on Neural Information Processing Systems, p.2879-2887. https://papers.nips.cc/paper/5392-extremal-mechanisms-for-local-differential-privacy

Kim H, Ben-Othman J, Mokdad L, 2019. Udipp: a framework for differential privacy preserving movements of unmanned aerial vehicles in smart cities. *IEEE Trans Veh Technol*, 68(4):3933-3943. https://doi.org/10.1109/TVT.2019.2897509

Kim H, Ben-Othman J, Mokdad L, et al., 2020. Research challenges and security threats to AI-driven 5G virtual emotion applications using autonomous vehicles, drones, and smart devices. *IEEE Netw*, 34(6):288-294. https://doi.org/10.1109/MNET.011.2000245 https://doi.org/10.1109/MNET.011.2000245

Labs J, Terry S, 2020. Privacy in the coronavirus era. *Genet Test Mol Biomarkers*, 24(9):535-536. https://doi.org/10.1089/gtmb.2020.29055.sjt

Lee S, Lee S, Kim H, 2023. Differential security barriers for virtual emotion detection in maritime transportation stations with cooperative mobile robots and UAVs. *IEEE Trans Intell Transport Syst*, 24(2):2461-2471. https://doi.org/10.1109/tits.2022.3172668

Liu H, Liu JY, Chen F, et al., 2022. Progressive residual learning with memory upgrade for ultrasound image blind super-resolution. *IEEE J Biomed Health Inform*, 26(9):4390-4401. https://doi.org/10.1109/jbhi.2022.3142076

Liu Z, Li J, Chen XF, et al., 2020. Fuzzy logic-based adaptive point cloud video streaming. *IEEE Open J Comput Soc*, 1:121-130. https://doi.org/10.1109/OJCS.2020.3006205

Liu Z, Zhan C, Cui Y, et al., 2021. Robust edge computing in UAV systems via scalable computing and cooperative computing *IEEE Wirel Communs*, 28(5):36-42. https://doi.org/10.1109/MWC.121.2100041

Narayanan A, Shmatikov V, 2008. Robust deanonymization of large sparse datasets. Proc IEEE Symp on Security and Privacy, p.111-125. https://doi.org/10.1109/SP.2008.33

Nguyên TT, Xiao XK, Yang Y, et al., 2016. Collecting and analyzing data from smart device users with local differential privacy. https://arxiv.org/abs/1606.05053

Qin Z, Yang Y, Yu T, et al., 2016. Heavy hitter estimation over set-valued data with local differential privacy. Proc ACM SIGSAC Conf on Computer and Communications Security, p.192-203. https://doi.org/10.1145/2976749.2978409

Ren XB, Shi L, Yu WR, et al., 2022. LDP-IDS: local differential privacy for infinite data streams. *Proc Int Conf on Management of Data*, p.1064-1077. https://doi.org/10.1145/3514221.3526190

Sultani ZN, Ghani RF, 2015. Kinect 3D point cloud live video streaming. *Proc Comput Sci*, 65:125-132. https://doi.org/10.1016/j.procs.2015.09.090

Thakurta AG, Vyrros AH, Vaishampayan US, et al., 2017. Emoji Frequency Detection and Deep Link Frequency. US Patent 9894089 B2.

Wang Q, Zhang Y, Lu X, et al., 2016. Real-time and spatio-temporal crowd-sourced social network data publishing with differential privacy. *IEEE Trans Dependable Secure Comput*, 15(4):591-606. https://doi.org/10.1109/TDSC.2016.2599873

Wang S, Sinnott R, Nepal S, 2018. Privacy-protected statistics publication over social media user trajectory streams. *Future Gener Comput Syst*, 87:792-802. https://doi.org/10.1016/j.future.2017.08.002

Wang XJ, Liu Z, Liu AX, et al., 2023. A near-optimal protocol for continuous tag recognition in mobile rfid systems. *IEEE/ACM Trans Networking*, in press. https://doi.org/10.1109/TNET.2023.3317875

Yang G, Xia CT, Bai YL, 2018. Differential privacy histogram publishing techniques for real-time data streams. *J Nanjing Univ Posts Telecommun (Nat Sci Ed)*, 38(2):69-77 (in Chinese). https://doi.org/10.14132/j.cnki.1673-5439.2018.02.012

Ye QQ, Hu HB, Meng XF, et al., 2019. Privkv: Key-value data collection with local differential privacy. Proc IEEE Symp on Security and Privacy, p.317-331. https://doi.org/10.1109/SP.2019.00018

Zhang XJ, Chen R, Xu JL, et al., 2014. Towards accurate histogram publication under differential privacy. Proc SIAM Int Conf on Data Mining, p.587-595. https://doi.org/10.1137/1.9781611973440.68

Unedited