Beyond Value Perturbation: Local Differential Privacy in the Temporal Setting

Qingqing Ye [1,2], Haibo Hu [1,3], Ninghui Li [4], Xiaofeng Meng [2], Huadi Zheng [1], Haotian Yan [1]
[1] Hong Kong Polytechnic University; [2] Renmin University of China;
[3] PolyU Shenzhen Research Institute; [4] Purdue University
{*qqing.ye, haibo.hu, huadi.zheng, hao-tian.yan*}*@polyu.edu.hk, ninghui@purdue.edu, xfmeng@ruc.edu.cn*

***Abstract*—Time series has numerous application scenarios. However, since many time series data are personal data, releasing them directly could cause privacy infringement. All existing techniques to publish privacy-preserving time series perturb the values while retaining the original temporal order. However, in many value-critical scenarios such as health and financial time series, the values must not be perturbed whereas the temporal order can be perturbed to protect privacy. As such, we propose "local differential privacy in the temporal setting" (TLDP) as the privacy notion for time series data. After quantifying the utility of a temporal perturbation mechanism in terms of the costs of a missing, repeated, empty, or delayed value, we propose three mechanisms for TLDP. Through both analytical and empirical studies, we show the last one, Threshold mechanism, is the most effective under most privacy budget settings, whereas the other two baseline mechanisms fill a niche by supporting very small or large privacy budgets.**

***Index Terms*—Local differential privacy; time series data; data sanitization**

I. Introduction

A time series is a sequence of values indexed in a discrete time order. Time series analysis has numerous applications in multimedia, internet of things, location services, financial trading, geophysics, and other domains that involve temporal measurements. However, many time series originate from personal data and releasing them for analysis could infringe privacy. For example, in recent COVID-19 pandemic, social networking and location-service providers such as Apple, Google and Facebook are called on by the Trump administration to provide people's location history for contact and social distance tracing. But the Congress has raised privacy concerns on tracking and collecting individual's location history [1]. As another example, medical researchers have successfully re-identified individuals from their anonymous fitness data (e.g., 20-minute aggregated physical activities by accelerometers) using machine learning techniques [2]. To address this issue, many privacy-preserving time series publishing techniques have been proposed [3], [4], most of which are based on differential privacy [5], [6]. They perturb the values of a released time series to guarantee no value at any timestamp can be restored with high confidence [4], [5], [7]. However, such perturbation can lead to significant distortion of the values as the noise can overshadow them or even be unbounded. In this paper, we propose **temporal** perturbation mechanisms to achieve differential privacy, and bound the perturbation of timestamp by a time window k. Since many time series manipulations, such as aggregation, window transformation (e.g., smoothing, rolling), and resampling in Apache Flink [8], are operated on time windows, temporal perturbation can have less or even no impact on the accuracy of these operators than value perturbation. The following are several real-life examples of publishing and analzying sensitive time series.

Example I: (*Biosensors in Health Monitoring*) Many personal health and fitness monitoring systems such as Apple Watch, Fitbit, and Omron HeartGuide continuously monitor vital metrics including heart rate, ECG, and blood pressure. The accuracy of these sensor values is essential for diagnostics. On the other hand, many diagnostics are based on the statistics of aggregated values, e.g., the maximum and average heart rate or blood pressure [9], of which temporal perturbation only causes a minimum change or even just a delay.

Example II: (*Mobility Tracking*) In recent COVID-19 pandemic, through mobile operators and location-service providers, governments collect users' (especially those in self-quarantine) mobility data to monitor their social distancing. For example, the moving speed can be used to infer whether they are stationary, walking, jogging, driving or taking a train [1]. The accuracy of the moving speed is essential for the monitoring. On the other hand, temporal perturbation only causes a delay of the monitoring.

Example III: (*Sensor Readings in Smart Home*) Environmental sensors are deployed in smart home to monitor the temperature, humidity, light, sound, air quality. The accuracy of sensor readings is essential to the operation of smart home. On the other hand, temporal perturbation only causes a delay of the reading and thus the operation.

In this paper, we first define local differential privacy in the temporal setting (TLDP) for time series data, denoted by ϵ-TLDP. TLDP ensures that the original value at any timestamp cannot be restored by a released time series with high confidence. This definition is in the context of a k-timestamp window, which denotes the maximum delay the system can tolerate a user to release a value. Then we quantify the utility of a released time series in terms of the **missing, repetition, empty, and delay** cost, and present three temporal perturbation mechanisms that dispatch values in the temporal axis. The first two, namely, Backward Perturbation mechanism and Forward Perturbation mechanism, dispatch a value to each of the k timestamps with probabilities that follow the Generalized Randomized Response [10] mechanism. However, they both suffer from value missing, repetition or empty. The third one, Threshold mechanism, is not only free of missing, repetition, and empty costs, but also has the lowest delay costs when ϵ is moderate. Through both analytical and empirical studies, we show the effectiveness of Threshold mechanism and provide a holistic strategy on when to use which mechanism for privacy-preserving time series release. To summarize, our technical contributions are three-folded.

- We formulate the problem of local differential privacy in the temporal setting (TLDP) for time series, and point out the issues arising from value-perturbation mechanisms.
- We present two baseline temporal perturbation mechanisms that satisfy ϵ-TLDP, with detailed analysis on their costs of value missing, repetition, empty, and delay.
- We design Threshold mechanism that can satisfy ϵ-TLDP without any value missing, repetition or empty. Through both analytical and empirical studies, we show when this mechanism should be used for time series release.

The rest of the paper is organized as follows. Section II formulates the problem of TLDP on time series, together with preliminaries on DP and LDP. Section III presents the two baseline mechanisms and analyzes their costs. Section IV introduces the Threshold mechanism with theoretical analysis on its TLDP privacy guarantee and total cost. Section V presents experimental results and demonstrations on both real and synthetic datasets. Finally, we review existing work in Section VI and conclude this paper in Section VII.

II. Preliminaries and Problem Definition

A. Preliminary: Differential Privacy

Differential Privacy in the Centralized Setting. Differential privacy (DP) [11] is initially defined on a randomized algorithm $\mathcal{A}$ of a sensitive database. $\mathcal{A}$ is said to satisfy ϵ-differential privacy, if for any two neighboring databases D and D' that differ only in one tuple, and any possible output v of $\mathcal{A}$:

$$\Pr(\mathcal{A}(D)=v) \le e^{\epsilon} \cdot \Pr(\mathcal{A}(D')=v), \tag{1}$$

where the parameter ϵ is called privacy budget and denotes the degree of privacy retained after releasing the output (the lower the better privacy). To satisfy ϵ-DP, two major perturbation mechanisms are developed, namely, Laplace mechanism [11] and Exponential Mechanism [12].

Differential Privacy in the Local Setting. Local Differential Privacy (LDP) [13] lets each user locally perturb her data before sending it to an untrusted data collector. Formally, $\mathcal{A}$ is said to satisfy ϵ-local differential privacy, if for any two tuples t and t', and any possible output t^*:

$$\Pr(\mathcal{A}(t)=t^*) \le e^{\epsilon} \cdot \Pr(\mathcal{A}(t')=t^*) \tag{2}$$

To satisfy ϵ-LDP, both Laplace mechanism and Generalized Randomized Response [10] are predominant perturbation mechanisms.

B. Problem Definition: Local Differential Privacy in the Temporal Setting

In this paper, we assume a time series is an infinite sequence of values $S = \{S_{t_1}, S_{t_2}, ..., S_{t_n}, ...\}$ in the temporal domain $T = \{t_1, t_2, ..., t_n, ...\}$. For ease of presentation, in the sequel we assume S is univariate (i.e., the values are one-dimensional) and T has equal time intervals, i.e., $t_2 - t_1 = ... = t_n - t_{n-1} = ...$. So a time series is simply denoted by $S = \{S_1, S_2, ..., S_n, ...\}$. As with other differential privacy notions, we first define neighboring time series. Since a time series has both values and timestamps, neighbors can be defined either on values or on timestamps, which leads to local differential privacy in the value setting (VLDP) and in the temporal setting (TLDP). VLDP is a straightforward extension of LDP in time series to protect the true value in each timestamp from being restored with high confidence. It can be satisfied by existing value perturbation mechanisms for LDP, such as Laplace mechanism or Randomized Response. On the other hand, the privacy goal of this paper is TLDP, which protects the true timestamp of any released value from being restored. As such, two time series are defined as neighbors if their *transposition distance* in the temporal domain is 1, that is, they turn into one another by swapping the values in both timestamps. To further restrict the swapping, we require the two timestamps within a **time window of length *k***, which is the maximum delay the system can tolerate a user to release a value. As will be elaborated in Section II-D, k is a system parameter based on the time series analysis task.

Definition 2.1: (**Neighboring Time Series**) Two time series S and S' are neighbors if there exist two timestamps $t_i \neq t_j$ such that

1) $|i-j| \le k$, and
2) $S_i = S'_j$ and $S_j = S'_i$, and
3) for any other timestamp $t_l (l \neq i, l \neq j)$, $S_l = S'_l$.[1]

Then we define local differential privacy in the temporal setting as follows.

Definition 2.2: (**Local Differential Privacy in the Temporal Setting, TLDP**) Given privacy budget ϵ, a randomized algorithm $\mathcal{A}$ satisfies ϵ-TLDP, if and only if for any two neighboring time series S and S', and any possible output R of $\mathcal{A}$, the following inequality holds:

$$\Pr(\mathcal{A}(S)=R) \le e^{\epsilon} \cdot \Pr(\mathcal{A}(S')=R) \tag{3}$$

As with other LDP notions, the degree of privacy is controlled by the privacy budget ϵ. Since the whole time series is released for analysis, the output R of $\mathcal{A}$ is simply a perturbed time series $\{R_1, R_2, ..., R_n, ...\}$ from the original one $S = \{S_1, S_2, ..., S_n, ...\}$.

C. Relationship between TLDP and VLDP

Although TLDP and VLDP are two independent privacy models derived from the original notion of LDP, they can convert to each other under certain conditions. The following

[1]We assume S and S' are infinite time series. For finite time series, we treat the beginning and ending k timestamps as warmup and cooldown periods and exclude them from this equality test.

Theorem 2.3 shows that any perturbation that satisfies VLDP also satisfies TLDP with a doubled privacy budget, and Theorem 2.4 shows that any temporal perturbation that satisfies ϵ-TLDP of window k can satisfy $\epsilon/2$-VLDP for any value in this window, as long as the skewness of value is bounded. In the interest of space, the proofs of lemmas and theorems in this paper are included in our technical report [14].

Theorem 2.3: If a randomized algorithm $\mathcal{A}$ satisfies $\epsilon/2$-VLDP for each value S_i, $\mathcal{A}$ will also satisfy ϵ-TLDP for the time series S.

Theorem 2.4: Let $\mathcal{A}$ be a randomized algorithm that satisfies ϵ-TLDP, then $\mathcal{A}$ also satisfies ϵ^*-VLDP, where $\epsilon^* = \max\{\epsilon/2, \hat{\epsilon}\}$ and $\hat{\epsilon}$ denotes the skewness of value. Specifically, in any time window with length k, the ratio of the maximum and minimum occurrences of two values is bounded by $\exp(\hat{\epsilon})$.

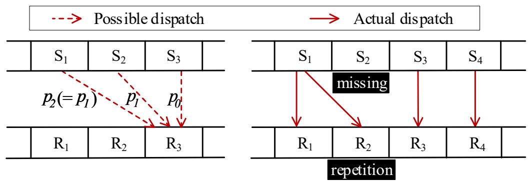
D. *Cost (Utility) of Released Time Series*

The cost of TLDP is the inaccuracy of time series analysis introduced by the released time series R. Since R deviates from S by missing, repeating, emptying, and delaying values, we quantify this cost in terms of these four costs as below.

1) **Missing Cost** C_M. This occurs when a value S_i is missed, i.e., it does not appear in the time window starting from R_i, i.e., $\{R_i, R_{i+1}, ..., R_{i+k-1}\}$. For simplicity, we assume each missing value bears a unit cost of M.
2) **Repetition Cost** C_N. This occurs when a value S_i is repeatedly released in the time window starting from R_i, i.e., $\{R_i, \ldots, R_{i+k-1}\}$. For simplicity, we assume each occurrence of a repeated value bears a unit cost of N.
3) **Empty Cost** C_E. This occurs when no value is set for R_i when it needs to be released at timestamp t_i. For simplicity, we assume each empty release of a timestamp bears a unit cost of E.
4) **Delay Cost** C_D. This occurs when a value S_i is released at a delayed timestamp $t_j (j > i)$. For simplicity, we assume one timestamp delay of a value bears a unit cost of D. As such, the delay cost of S_i being released to R_j is $D(j-i)$. To avoid double counting of costs, all above costs supersede delay cost. For example, a repeated value has no delay cost.

To illustrate how to derive these costs and their unit costs M, N, E, and D, for a time series analysis task in practice, we use two common time series manipulations [15], [16] as examples, namely, frequency counting (FC) and simple moving average (SMA). At each timestamp, $FC(v)$ returns the number of occurrences of value v in the past time series; whereas $SMA(d)$ takes the arithmetic mean over the past d timestamps.

Case 1: Frequency Counting $\boldsymbol{FC(v)}$. Let $f(v)$ be the probability of value v in the original time series. A missing value in the original time series leads to a repeated or an empty value in the released time series. For the first case, suppose the repeated value is v', this causes inaccuracy in all of the subsequent l counts and the expected mean square error (MSE) is $[(f(v) - f(v'))\frac{l}{2}]^2$. For the second case, it causes underestimation to all the subsequent l counts by $f(v)$, thus the expected MSE is $[f(v)\frac{l}{2}]^2$. As for a delayed value v, it causes



(a) Perturbation protocol (b) Dispatching example
Fig. 1. Backward Perturbation mechanism

inaccuracy in the subsequent $k-1$ counts at most, hence the expected MSE is $[f(v)\frac{k-1}{2}]^2$. Therefore, $M + N = [(f(v) - f(v'))\frac{l}{2}]^2$, or $M + E = [f(v)\frac{l}{2}]^2$, and $D = [f(v)\frac{k-1}{2}]^2$.

Case 2: Simple Moving Average $\boldsymbol{SMA(d)}$. A missing value v causes inaccuracy in the subsequent d timestamps, which amounts to an expected MSE of $d\frac{(v-\overline{v})^2}{d} = (S_i - \overline{v})^2$, where $\overline{v}$ is the arithmetic mean in the neighborhood timestamps. The expectation of $(S_i - \overline{v})^2$ can be derived as $\overline{v^2} - \overline{v}^2$, where $\overline{v^2}$ is the mean square in the neighborhood timestamps. Similarly, a repeated value v also causes an expected MSE of $\overline{v^2} - \overline{v}^2$. An empty value causes an expected MSE of $(\overline{v} - \frac{d\overline{v}-v}{d-1})^2$, which can be approximated to 0. A delayed value S_i for t timestamps, on the other hand, causes inaccuracy in $2t$ timestamps,[2] whose accumulated error is thus $\frac{2t}{d}(\overline{v^2} - \overline{v}^2)$. Therefore, $M = N = \overline{v^2} - \overline{v}^2$, $E = 0$, and $D = kM/d$.

III. Baseline Approach

In this section, we present two baseline temporal perturbation mechanisms for TLDP — the Backward and the Forward Perturbation mechanisms. As their names suggest, the Backward Perturbation mechanism probabilistically selects a value from previous k timestamps in the original time series to release at the current timestamp, whereas the Forward Perturbation mechanism probabilistically dispatches the value at the current timestamp to one of the subsequent k timestamps to release. In what follows, we first show how to set these probabilities to satisfy ϵ-TLDP, and then analyze their costs.

A. *Backward Perturbation Mechanism*

1) Perturbation Protocol: At each timestamp t_i, the protocol probabilistically releases the value R_i drawn from $\{S_{i-k+1}, S_{i-k+2}, ..., S_i\}$, which are the values at the k most recent timestamps. Fig. 1(a) shows an example where $k = 3$. At t_3, the released value R_3 is drawn from $\{S_1, S_2, S_3\}$ with probabilities $\{p_2, p_1, p_0\}$. To satisfy ϵ-TLDP, we set these probabilities as

$$\Pr(R_i = S_{i-j}) := p_j = \begin{cases} \frac{e^{\epsilon/2}}{k-1+e^{\epsilon/2}}, & j = 0 \\ \frac{1}{k-1+e^{\epsilon/2}}, & j \in \{1, 2, ..., k-1\} \end{cases} \tag{4}$$

As there are only two probabilities, we simply use p_0 and p_1 to denote them in the sequel. Note that since any two neighboring time series only differ two timestamps at most, this protocol satisfies ϵ-TLDP.

2) Cost Analysis: There are three costs in the Backward Perturbation mechanism — missing, repetition and delay.

• **Missing cost** C_M. A value S_i in the time series is missing if it is not selected as an output in all subsequent k timestamps

[2]We assume $d \geq k > t$.

(e.g., S_2 in Fig. 1(b)), whose probability is $(1-p_0)(1-p_1)^{k-1}$. Therefore, the expected missing cost $\mathbb{E}[C_M]$ of each value is

$$\mathbb{E}[C_M] = (1-p_0)(1-p_1)^{k-1} \cdot M \tag{5}$$

• **Repetition cost** C_N. The probability that a value S_i is repeated r times, i.e., has $r+1$ occurrences (e.g., S_1 in Fig. 1(b) has two occurrences), can be calculated in two cases: (1) S_i is dispatched to R_i (so there are r occurrences in subsequent $k-1$ timestamps), and (2) S_i is not dispatched to R_i (so there are $r+1$ occurrences in subsequent $k-1$ timestamps). Therefore,

$$\begin{aligned}\mathbb{E}[C_N] &= N \sum_{r=1}^{k-1} r p_0 \cdot \binom{k-1}{r} \cdot p_1^r \cdot (1-p_1)^{k-1-r} \\ &+ N \sum_{r=1}^{k-2} r(1-p_0) \cdot \binom{k-1}{r+1} \cdot p_1^{r+1} \cdot (1-p_1)^{k-2-r} \\ &= (1-p_0)(1-p_1)^{k-1} N\end{aligned} \tag{6}$$

From Eqs. 5 and 6, we observe that the expected counts of missing and repetition are the same, because a value repetition always causes a value missing, and vice versa. To suppress the repetition (and thus missing) counts, one might be tempted to revise the Backward Perturbation mechanism protocol and avoid choosing a value that has already been selected. Unfortunately, this collision-free variant undermines the ϵ-TLDP guarantee. We will elaborate on this in Section IV-A.

• **Delay cost** C_D. We first derive $p(S_i \to R_{i+j})$, the probability of R_{i+j} being the first occurrence of S_i in the released series. This is a joint probability of two events — S_i being chosen for R_{i+j}, and S_i not being chosen for any R_{i+l} ($l < j$). Summing this probability over all $j \in \{1, 2, \ldots, k-1\}$, we can derive the expected delayed timestamp counts and thus $\mathbb{E}[C_D]$:

$$\begin{aligned}\mathbb{E}[C_D] &= D \sum_{j=1}^{k-1} p(S_i \to R_{i+j}) \prod_{l=0}^{j-1} (1 - p(S_i \to R_{i+l}))(j-i) \\ &= p_1(1-p_0) \sum_{j=1}^{k-1} j \cdot (1-p_1)^{j-1} \cdot D\end{aligned} \tag{7}$$

To sum up the above, the expected total cost of Backward Perturbation mechanism on any value S_i is

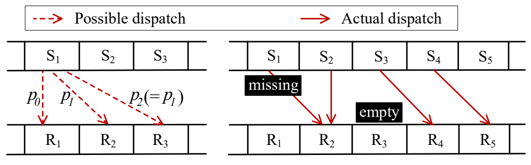
$$\mathbb{E}[C] = \mathbb{E}[C_M] + \mathbb{E}[C_N] + \mathbb{E}[C_D] \tag{8}$$

B. Forward Perturbation Mechanism

1) Perturbation Protocol: As opposed to Backward Perturbation mechanism which finds for each R_i a previous S_{i-j} to dispatch to, the Forward Perturbation mechanism dispatches each S_i to one of the R_{i+j}'s in the next k timestamps. The protocol is illustrated in Fig. 2(a), whose perturbation probabilities are the same as Eq. 4. That is, S_i is dispatched to R_i with probability $p_0 = \frac{e^{\epsilon/2}}{k-1+e^{\epsilon/2}}$ and dispatched to R_{i+j} ($j \in \{1, 2, ..., k-1\}$) with probability $p_1 = \frac{1}{k-1+e^{\epsilon/2}}$. Under this mechanism, multiple S_i's can be dispatched to the same R_{i+j}, causing value overwritten, and only the last dispatched S_i is released for R_{i+j} while all other S_i's dispatched to R_{i+j} become missing. Similar to Backward Perturbation mechanism, Forward Perturbation mechanism trivially satisfies ϵ-TLDP.

2) Cost Analysis: The costs in Forward Perturbation mechanism include missing, empty and delay.

• **Missing cost** C_M. Missing cost occurs when an earlier dispatched value S_i is overwritten by a subsequent one (e.g.,



(a) Perturbation protocol (b) Dispatching example

Fig. 2. Forward Perturbation mechanism

TABLE I
COST ANALYSIS OF DIFFERENT MECHANISMS

Cost	Backward Pert. mechanism	Forward Pert. mechanism	Threshold mechanism
Missing	$(1-p_0)(1-p_1)^{k-1}M$	$(1-p_0)(1-p_1)^{k-1}M$	0
Repetition	$(1-p_0)(1-p_1)^{k-1}N$	0	0
Empty	0	$(1-p_0)(1-p_1)^{k-1}E$	0
Delay	$p_1(1-p_0) \cdot \sum_{j=1}^{k-1} j(1-p_1)^{j-1}D$	$p_1(1-p_0) \cdot \sum_{j=1}^{k-1} j(1-p_1)^{j-1}D$	$(k-c_0)D$

S_1 in Fig. 2(b)). To derive this probability, we find that it only happens when S_i is dispatched to R_{i+j} ($j > 0$), and it can be overwritten by $S_{i+1}, S_{i+2}, ...,$ or S_{i+j} with probability $1-(1-p_0)(1-p_1)^{j-1}$. By summing over all j, we derive the expectation of the missing cost for S_i as:

$$\begin{aligned}\mathbb{E}[C_M] &= p_1 \sum_{j=1}^{k-1} \left(1-(1-p_0)(1-p_1)^{j-1}\right) \cdot M \\ &= (1-p_0)(1-p_1)^{k-1} \cdot M\end{aligned} \tag{9}$$

• **Empty cost** C_E. The probability that value R_i is empty, i.e., not dispatched from any S_{i-j} (e.g., R_3 in Fig. 2(b)), is $(1-p_0)(1-p_1)^{k-1}$. Therefore, the empty cost is

$$\mathbb{E}[C_E] = (1-p_0)(1-p_1)^{k-1} \cdot E \tag{10}$$

• **Delay cost** C_D. We derive the expected delayed timestamp counts and thus $\mathbb{E}[C_D]$ for value S_i by deriving the probability of R_{i+j} being the first occurrence of S_i in the released series, and summing over all R_{i+j}:

$$\mathbb{E}[C_D] = p_1 \sum_{j=1}^{k-1} j \cdot (1-p_0)(1-p_1)^{j-1} \cdot D \tag{11}$$

To sum up the above, the expected cost of Backward Perturbation mechanism on any value S_i is

$$\mathbb{E}[C] = \mathbb{E}[C_M] + \mathbb{E}[C_E] + \mathbb{E}[C_D] \tag{12}$$

C. Comparison of Backward and Forward Perturbation

Table I summarizes the cost analysis of both mechanisms, where they achieve the same missing and delay costs, and only differ in the type of cost they can avoid — the Backward Perturbation mechanism trades repetition cost for zero empty cost, whereas the Forward Perturbation mechanism does the opposite. As such, **the choice of both mechanisms simply depends on whose unit cost is larger, a repetition (N) or an empty value (E).**

Neither mechanism can avoid value missing, which is very costly or even unacceptable in many time series applications. In the next section, we propose the Threshold mechanism that completely eliminates the missing, repetition, and empty costs, and retains low delay cost under most ϵ and k settings.

IV. THRESHOLD MECHANISM

In this section, we first show the collision-free variant of Forward Perturbation mechanism violates ϵ-TLDP, which in

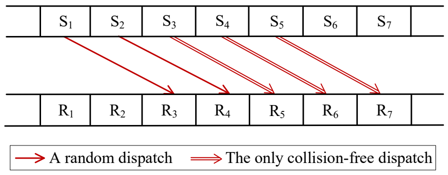


Fig. 3. Collision-free Forward Perturbation violates ϵ-TLDP.

turn motivates the Threshold mechanism. We then present the perturbation protocol, its privacy guarantee, and the optimal choice of the threshold parameter that minimizes the overall cost. Finally, we show an extended version that resolves the infeasible ϵ issue for this mechanism.

A. Motivation

The missing cost in two baseline mechanisms is due to repeatedly selecting from or dispatching to the same timestamp, which essentially causes a "collision". To eliminate the missing cost, one is tempted to avoid collision by restraining the perturbation protocol from selecting from a repeated timestamp (Backward Perturbation) or dispatching to an occupied timestamp (Forward Pertubation). However, the counterexample in Fig. 3 shows that **the collision-free variant of Forward Perturbation mechanism no longer satisfies ϵ-TLDP.** In this example, $k = 3$, so S_i can be dispatched to R_i, R_{i+1}, or R_{i+2}. However, after S_1 is dispatched to R_3 and S_2 to R_4, S_3 has only one collision-free choice R_5, because R_3 and R_4 have already been occupied but R_5 must be empty (since no prior value can be dispatched to it). Even worse, this effect propagates to all subsequent values — each of them has only one collision-free choice — $S_4 \to R_6$, $S_5 \to R_7$, and so on. Since the time series is infinite, it is just a matter of time for this effect to occur, which means an adversary can infer, with almost 100% confidence, that a released value must come from $k-1$ timestamps earlier. As such, ϵ-TLDP is violated.

As a key observation, this effect is caused by the **ever-decreasing empty timestamps for dispatching c** in the time window $\{t_i, t_{i+1}, ..., t_{i+k-1}\}$ until it stabilizes at 1 (because the last timestamp t_{i+k-1} is always empty). Inspired by this, we propose the Threshold mechanism that keeps c at a healthy level, i.e., above or equal to a threshold c_0, so that there are always enough choices to dispatch a value to satisfy ϵ-TLDP. Obviously, c_0 is in the range of $[2, k-1]$ and Section IV-D will show how to derive it from the input k and ϵ. For now we just regard it as an input parameter.

B. Perturbation Protocol

A naive perturbation protocol for the Threshold mechanism is to add the following rule to the collision-free Forward Perturbation — when dispatching S_i, if the number of empty timestamps $c = c_0$, we will ignore S_i. However, this rule re-introduces missing cost. In what follows, we design a more delicate perturbation protocol that can guarantee $c \geq c_0$ without any missing cost.

The idea is to adopt a less aggressive strategy than the naive one by only keeping c from dropping further to $c_0 - 1$. Algorithm 1 shows the pseudo-code. If $c > c_0$, S_i is dispatched

Algorithm 1 Perturbation protocol: $Perturb(\cdot)$

```
Input:      Original time series $S = \{S_1, S_2, ..., S_n, ...\}$
            Time window length $k$
            Threshold $c_0$
Output:     $R = Perturb(S, k, c_0)$ is the released time series
Procedure:
 1: Initialize $\boldsymbol{x} = \mathbf{0}$, and $R = \emptyset$
 2: for each value $S_i \in S$ do
 3:     Count the number of "0"s in $\{x_i, x_{i+1}, ..., x_{i+k-1}\}$, denoted by $c$
 4:     if $c > c_0$ then
 5:         Randomly select an index $l$ from $\mathbb{X} = \{j | x_j = 0, i \leq j \leq i+k-1\}$
 6:         Dispatch $S_i$ to $R_l$, and set $x_l = 1$
 7:     else
 8:         if $x_i = 0$ then
 9:             Dispatch $S_i$ to $R_i$, and set $x_i = 1$
10:         else
11:             Randomly select an index $l$ from $\mathbb{X} = \{j | x_j = 0, i < j \leq i+k-1\}$
12:             Dispatch $S_i$ to $R_l$, and set $x_l = 1$
13:     Release $R_i$
14: return $R = \{R_1, R_2, ..., R_n, ...\}$
```

to any empty timestamp with the same probability $1/c$ (Lines 4-6), and c will be decreased by 1 if R_i is still empty after dispatching S_i and thus expired. If $c = c_0$, to prevent c from dropping any further, if the current timestamp t_i is empty, it should not be expired without being dispatched. As such, the protocol always dispatches S_i to R_i if t_i is empty (Lines 8-9); otherwise, it dispatches S_i to any empty timestamp with the same probability $1/c_0$ (Lines 11-12). In either case, a non-empty R_i is released (Line 13). Note that once c drops to c_0, this protocol guarantees c remains c_0 forever.

C. Privacy Analysis

In this subsection, we prove Threshold mechanism satisfies ϵ-TLDP in Theorem 4.3. To facilitate the proof, we first derive the **dispatching probability** p_j, $j \in \{0, 1, ..., k-1\}$, which is the probability of S_i being dispatched to R_{i+j}. Let bit x_j denote whether t_{i+j} is empty ($x_j = 0$ means empty), and p_j^o denote this empty probability. Then

$$p_0 = p(S_i \to R_i) = p(x_i = 0) = p_0^o$$
$$p_j = p(S_i \to R_{i+j}) = \frac{(1 - p_0^o) \cdot p(x_{i+j} = 0 | x_i = 1)}{c_0} \tag{13}$$

To calculate p_0^o and $p(x_{i+j} = 0 | x_i = 1)$ where $j \in \{1, 2, ..., k-1\}$, we observe a recursion that $p(x_{i+j} = 0 | x_i = 1)$ with window length k equals to $p(x_{i+j-1} = 0)$ with length $k-1$. And to begin with, when $k = c_0 + 1$, we have $p(x_{i+j} = 0 | x_i = 1) = 1$. Based on this observation, we can derive the following lemma to recursively calculate p_0^o.

Lemma 4.1: Given time window length k, threshold c_0 and $m = k - c_0$, the probability that the first timestamp t_i is not empty, denoted by $g(k, m) = 1 - p_0^o$, can be recursively derived as

$$g(k,m) = \frac{m}{1 - \sum_{l=1}^{m} (-\frac{1}{c_0})^l \cdot \prod_{i=1}^{l+1} \frac{k-i}{i} \cdot \prod_{i=1}^{l-1} g(k-i, m-i)}$$

where $g(k, 1) = \frac{2}{k}$.

Now we present the following theorem on the dispatching probability of p_j.

Theorem 4.2: Given time window length k, threshold c_0 and $m = k - c_0$, for any $j \in \{0, 1, ..., k-1\}$, the dispatching probability of $S_i \to R_{i+j}$, denoted by p_j, is

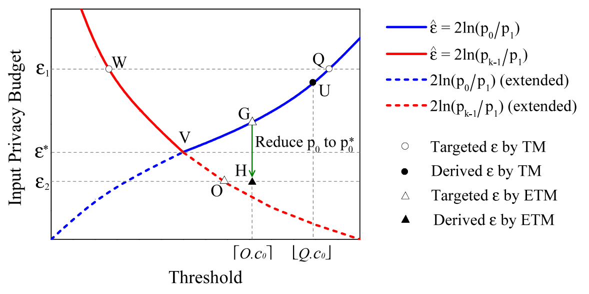


Fig. 4. Impact of threshold on privacy budget

$$p_0 = 1 - g(k,m)$$

$$p_1 = g(k,m) + \sum_{l=1}^{m} (\frac{-1}{c_0})^l \prod_{i=0}^{l-1} g(k-i, m-i) \prod_{i=1}^{l} \frac{k-1-i}{i}$$

$$p_{j>1} = -\sum_{l=1}^{m} (\frac{-1}{c_0})^l \prod_{i=0}^{l-1} g(k-i, m-i) \prod_{i=1}^{l-1} \frac{k-j-i}{i+1} \cdot l$$

To satisfy ϵ-TLDP, the ratio of any two dispatching probabilities should be bounded by $\exp(\frac{\epsilon}{2})$. The following theorem shows how to derive the $\hat{\epsilon}$ achieved by the Threshold mechanism, given window length k and threshold c_0 as inputs. In the sequel, $\hat{\epsilon}$ is called a derived privacy budget, as opposed to ϵ, which is the user input privacy budget.

Theorem 4.3: (**TLDP Guarantee of Threshold Mechanism**). Given time window length k and threshold c_0, the Threshold mechanism satisfies $\hat{\epsilon}$-TLDP, where $\hat{\epsilon} = 2\max\{\ln\frac{p_0}{p_1}, \ln\frac{p_{k-1}}{p_1}\}$, and p_i's are defined in Theorem 4.2.

D. Cost Minimization and Derivation of c_0^*

One remaining issue in Threshold mechanism is the choice of parameter c_0 that can minimize the total cost. Since Threshold mechanism does not have missing, repetition, and empty cost,[3] the expected total cost is:

$$\begin{aligned}\mathbb{E}[C] &= \sum_{j=1}^{k-1} j \cdot p_j \cdot D = \sum_{j=1}^{k-1} j \cdot (p_j^o - p_{j-1}^o) \cdot D \\ &= \left(k \cdot p_{k-1}^o - \sum_{j=0}^{k-1} p_j^o\right) \cdot D = (k - c_0) \cdot D\end{aligned} \tag{14}$$

The above equation shows that the total cost decreases with c_0. Therefore, **the optimal threshold c_0^* should be the largest c_0 that satisfies ϵ-TLDP** according to Theorem 4.3.

Unfortunately, Theorem 4.3 does not provide a closed-form solution to derive c_0^* from ϵ and k. Nonetheless, according to Theorem 4.3, the derived privacy budget $\hat{\epsilon}$ first decreases and then increases with c_0. Fig. 4 illustrates this. The red line denotes $\hat{\epsilon} = 2\ln\frac{p_{k-1}}{p_1}$, the blue line denotes $\hat{\epsilon} = 2\ln\frac{p_0}{p_1}$, and then the solid (red and blue) lines denote the derived privacy budget $\hat{\epsilon}$ by the Threshold mechanism, which is $2\max\{\ln\frac{p_0}{p_1}, \ln\frac{p_{k-1}}{p_1}\}$. The following theorem shows that both lines are monotonic. So instead of calculating $\hat{\epsilon}$ for each $c_0 \in [2, k-1]$ and finding c_0^* as the largest c_0 whose $\hat{\epsilon} \leq \epsilon$ (the user input privacy budget), we can use a binary search of c_0^* in $[2, k-1]$. The details of this binary search are in the complete description of Threshold mechanism in Section IV-F.

Theorem 4.4: The derived privacy budget $\hat{\epsilon}$ by Threshold mechanism first monotonically decreases with c_0 and then monotonically increases.

[3]The empty cost is amortized to 0 for an infinite time series, because $c = c_0$ always holds and no empty timestamp is wasted in the stable state.

E. Extended Threshold Mechanism

Fig. 4 also points out one limitation of Threshold mechanism — the intersection of the two solid lines, i.e., point V, dictates the lowest privacy budget ϵ^* this mechanism can achieve. For an input privacy budget $\epsilon_1 \geq \epsilon^*$, Threshold mechanism can use a binary search to find Q that intersects with ϵ_1,[4] and then Q is rounded down to $U = \lfloor Q.c_0 \rfloor$. However, for an input privacy budget $\epsilon_2 < \epsilon^*$, Threshold mechanism cannot find any feasible c_0 because to optimize the total cost this mechanism inherently favors the first (t_i) and last (t_{i+k-1}) timestamps to dispatch S_i. As such, the derived privacy budget, which is the maximum ratio (in logarithm) of any two dispatching probabilities, cannot be arbitrarily low no matter what c_0 is chosen.

To address this problem, we extend the Threshold mechanism. As shown in Fig. 4, for $\epsilon_2 < \epsilon^*$, we still find O, the intersection point with the red line, and then round it up (because ϵ in the red line is decreasing) to $\lceil O.c_0 \rceil$. To satisfy ϵ_2-TLDP at this c_0, we must move G, the intersection point with the blue line, to H, the intersection point with the horizontal line ϵ_2. This is achieved by reducing the dispatching probability p_0 to p_0^* such that $2\ln(\frac{p_0^*}{p_1}) = \epsilon_2$. The extended Threshold mechanism implements this reduction by probabilistically ignoring value S_i when dispatching it to the current timestamp. In other words, when the current timestamp t_i is empty, it dispatches S_i to R_i with probability $\frac{p_0^*}{p_0}$, which is $\frac{e^{\epsilon_2/2} \cdot p_1}{p_0}$, and ignores S_i with probability $1 - \frac{e^{\epsilon_2/2} \cdot p_1}{p_0}$. In essence, the extended Threshold mechanism re-introduces missing and empty costs to increase randomness and thus satisfy very small ϵ. Note that the drop of p_0 does not affect the other dispatching probabilities p_1, p_2, ..., and p_{k-1}, so the extended Threshold mechanism still satisfies ϵ-TLDP as shown in the following theorem.

Theorem 4.5: Given input privacy budget ϵ and time window length k, the extended Threshold mechanism satisfies ϵ-TLDP.

F. Overall Algorithm

Algorithm 2 summarizes the overall algorithm of the (extended) Threshold mechanism. The core of this algorithm is to find c_0^*, the optimal threshold of this mechanism, using a binary search. It first initializes the lower bound l and upper bound r for this search (Line 1). Then in each recursion, the current c_0 is the mid-point of $[l, r]$. Based on this c_0 and the time window length k, it derives the privacy budget $\hat{\epsilon}_1$ by Theorem 4.3 (Line 3). If $\hat{\epsilon}_1 < \epsilon$, this means the next recursion should search in the range of $[c_0, r]$ for even larger c_0 (Line 4). Otherwise, we should first determine whether the current c_0 is on the decreasing or increasing solid line as in Fig. 4. This is done by testing whether privacy budget $\hat{\epsilon}_2$ derived from k and $c_0 - 1$ is larger than $\hat{\epsilon}_1$. If this is true, c_0 is on the decreasing line and the next recursion should search $[c_0, r]$; otherwise, c_0 is on the increasing line and we should search $[l, c_0]$ (Line 7).

[4]According to Eq. 14, Threshold mechanism always chooses the largest c_0 whose $\hat{\epsilon} \leq \epsilon_1$, so Q is chosen instead of W.

Algorithm 2 (Extended) Threshold mechanism

```
Input:      Original time series $S = \{S_1, S_2, ..., S_n, ...\}$
            Time window length $k$
            Input privacy budget $\epsilon$
Output:     A perturbed time series $R$
Procedure:
1:  Initialize binary search range $l = 2$, and $r = k - 1$
2:  while $l < r$ do
3:      Calculate $c_0 = \lfloor \frac{l+r}{2} \rfloor$, and $\hat{\epsilon}_1 \leftarrow (k, c_0)$ according to Theorem 4.3
4:      if $\hat{\epsilon}_1 < \epsilon$ then $l = c_0$
5:      else
6:          Calculate $\hat{\epsilon}_2 \leftarrow (k, c_0 - 1)$
7:          if $\hat{\epsilon}_2 > \hat{\epsilon}_1$ then $l = c_0$ else $r = c_0$
8:      if $l + 1 = r$ then
9:          Calculate $\hat{\epsilon}_3 \leftarrow (k, l)$ and $\hat{\epsilon}_4 \leftarrow (k, r)$
10:         if $\hat{\epsilon}_3 \leq \epsilon$ then $c_0^* = l$, break the loop
11:         else if $\hat{\epsilon}_4 \leq \epsilon$ then $c_0^* = r$, break the loop
12:         else                    /* extended Threshold mechanism */
13:             $l = r, r = k - 1$
14:             while $l < r$ do
15:                 Calculate $c_0 = \lfloor \frac{l+r}{2} \rfloor$, $\hat{\epsilon}_5 \leftarrow (k, c_0)$
16:                 if $\hat{\epsilon}_5 \leq \epsilon$ then $r = c_0$ else $l = c_0$
17:                 if $l + 1 = r$ then
18:                     return $R = ExtendedPerturb(S, k, r)$
19: return $R = Perturb(S, k, c_0^*)$
```

The binary search terminates when $l+1 = r$. The algorithm derives both $\hat{\epsilon}_3$ and $\hat{\epsilon}_4$ from l and r, respectively (Line 9). If either of them is no larger than the input privacy budget ϵ, then the optimal threshold c_0^* is found and Algorithm 1 is invoked to perturb the original time series using c_0^* (Line 19). Otherwise, Threshold mechanism cannot achieve ϵ and we have to use the extended Threshold mechanism (Lines 13-18). Since this line must be decreasing as in Fig. 4, the optimal threshold c_0^* must reside in $[r, k-1]$ and can be found by a similar binary search (Lines 15-16). The search terminates when $l + 1 = r$ and the extended Threshold mechanism will be invoked to perturb the original time series (Lines 17-18). The perturbation is the same as Algorithm 1 except at Line 9, S_i is dispatched to R_i with probability $\frac{e^{\epsilon/2} \cdot p_1}{p_0}$ instead of probability 1.

V. Experimental Evaluation

A. Experimental Setting

1) Datasets: We conduct experiments on two real and one synthetic time series datasets: (1) **U.S. Stocks** [17] is about historical daily prices of all U.S. stocks. We select a stock that has the longest duration with 14,058 trading days and derive a time series from its daily close price, where each value indicates "up" or "down" of daily price; (2) **Taxi Trajectories** [18] consists of 6307 taxi trajectories, each of which has GPS coordinates in a 15-second interval and has at least 300 timestamps; and (3) **SyntheticTS** is a generated synthetic time series that consists of 10^6 timestamps, whose values S_i are integers and randomly drawn from $[0, 100]$.

2) Experiment Design: We design two sets of experiments. The first set evaluates the overall cost of the three TLDP perturbation mechanisms, namely, Backward Perturbation mechanism **BPM**, Forward Perturbation mechanism **FPM**, and Threshold mechanism **TM** (**ETM** denoting the extended version) under various datasets and system parameters, including the time window length k, privacy budget ϵ and unit cost of missing, repetition, empty and delay M, N, E and D. As costs are all relative, we fix the unit delay cost $D = 1$, so the delay cost of a value is less then k. Then we use $\frac{M}{k}$, $\frac{N}{k}$ and $\frac{E}{k}$ to denote the relative cost of a missing, repeated and empty value compared to a delayed value.

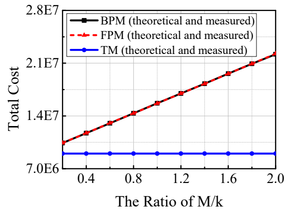
The second set compares the real utility of TLDP-based temporal perturbation (i.e., BPM, FPM and TM/ETM) against VLDP-based value perturbation (i.e., Randomized Response or Laplace mechanism) in three real-world applications, namely, frequency counting, simple moving average, and trajectory clustering. For frequency counting, since the values are binary, we use Randomized Response [19] for VLDP-based perturbation, where the flipping probability is set to $p = \frac{e^{\epsilon/2}}{1+e^{\epsilon/2}}$ to satisfy ϵ-TLDP according to Theorem 2.3. For simple moving average and trajectory clustering, since values are numerical data, we apply Laplace mechanism for VLDP-based perturbation. To be fair to it, the sensitivity $S(F)$ is calculated from those exist in the time series, instead of the whole value domain. A Laplace noise $Lap(\frac{2 \cdot S(F)}{\epsilon})$ is then added to each value to satisfy ϵ-TLDP according to Theorem 2.3.

We implement all mechanisms in Python and conduct experiments on a desktop computer with Intel Core i7-6700 3.40 GHz CPU, 32G RAM running Windows 10 operating system.

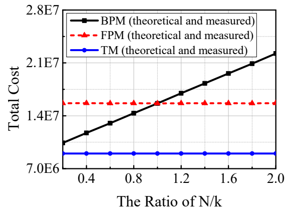
B. Results of Cost Analysis

1) Impact of relative cost: We first investigate the impact of relative cost of missing, repetition and empty on total cost. We set $k = 20$, and $\epsilon = 5$, which is equivalent to perturbing values with $\epsilon = 5/2$ according to Theorem 2.3. Both measured and theoretical results derived from Eqs. 8, 12 and 14 are plotted in Fig. 5(a)-(c). We observe that both results coincide with each other in all settings. In Fig. 5(a), we vary $\frac{M}{k}$ from 0.2 to 2.0, while keeping $\frac{N}{k} = \frac{E}{k} = 1$. BPM and FPM have exactly the same total cost, because the two mechanisms trade repetition with empty cost, or vice versa. In Fig. 5(b), we vary $\frac{N}{k}$ from 0.2 to 2.0 while keeping $\frac{M}{k} = \frac{E}{k} = 1$. The total cost of BPM grows with the ratio of $\frac{N}{k}$, while that of FPM stays unchanged. This is because repetition cost only exists in the former. Fig. 5(c) shows opposite results to Fig. 5(b) when $\frac{E}{k}$ is varied from 0.2 to 2.0. On the other hand, the total cost of **TM does not change over $\boldsymbol{\frac{M}{k}}$, $\boldsymbol{\frac{N}{k}}$ or $\boldsymbol{\frac{E}{k}}$** in Figs. 5(a)-(c), and **remains the lowest** among all three mechanisms.

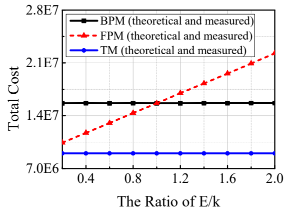
2) Impact of k and ϵ: We set all relative costs equally as $r = \frac{M}{k} = \frac{N}{k} = \frac{E}{k} = 1$, $\epsilon = 5$, and vary the time window length k from 10 to 160. Fig. 5(d) shows the impact of k on individual costs as bars and the total cost as lines. Overall, all costs increase with k. TM always has the lowest total cost, even though its delay cost is higher than the two baseline mechanisms. This is attributed to the fact that TM does not incur other costs, which are large in baseline mechanisms. Note that by setting all relative costs to 1 we already favor the two baselines, because the rationale behind any temporal perturbation mechanism assumes a missing/repeated/empty value is more severe than a delayed value.



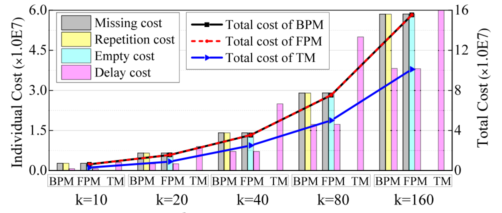
(a) Total cost varying M



(b) Total cost varying N



(c) Total cost varying E



(d) Impact of k on individual and total costs

Fig. 5. Impact of relative cost and sliding window length on total cost

In Figs. 6(a) and (b), we compare the total cost of three mechanisms by varying the input privacy budget ϵ from 1 to 16. We still set all relative costs equally as $r = \frac{M}{k} = \frac{N}{k} = \frac{E}{k} = 1$ and plot the results under $k = 10$ and 50, respectively. Overall, the total costs of three mechanisms all decrease with ϵ, and TM outperforms the other two except for very small and large ϵ. When ϵ becomes small, it cannot be satisfied by TM, so the extended version ETM (the dashed lines) is adopted. The gain of this mechanism gradually shrinks as ϵ gets smaller until at very small ϵ, e.g., $\epsilon = 1$ for $k = 10$ case, the baseline mechanisms outperform it. On the other hand, as ϵ becomes very large, all three mechanisms tend to saturate. For TM, this is because there is an upper bound of privacy budget it can expend, which is the maximum derived privacy budget of all $c_0 \in [2, k-1]$. As k increases, this bound is lifted — 10 for $k = 10$, and 16 for $k = 50$. As such, TM can still gain better performance as k increases whereas the two baseline mechanisms cannot.

3) Valid range of (extended) Threshold mechanism: To better understand the range of input privacy budget where TM/ETM can derive and where it outperforms the two baselines, we plot them in Fig. 6(c) for k from 5 to 200. The dark blue bar denotes the range of derived privacy budget, and the light blue bar denotes the range where TM/ETM outperform the baselines. As k increases, TM can derive a wider range of input privacy budget, for which it can be adopted. On the other hand, as k increases, the range of input privacy budget where the TM/ETM outperforms the baselines expands, which is attributed to the increasing upper bound and stable lower bound (as low as 2.5 even when $k = 200$). To further investigate the lower bound, we also plot the threshold c_0^* that corresponds to the lowest derived privacy budget in the red line. As k increases, c_0^* almost increases proportionally, so the delay cost $k - c_0^*$ increases proportionally, as opposed to the drastic increase of missing/repetition/empty costs of baselines as in Fig. 5(d).

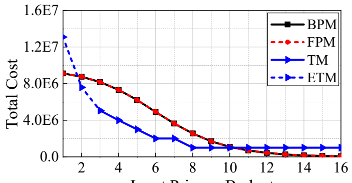
4) Conclusion: choosing suitable mechanisms: To summarize the results, we shall use the following strategy to choose a temporal perturbation mechanism for given ϵ and k. If this ϵ can be derived by the Threshold mechanism under k, then we should choose this mechanism. Otherwise, if ϵ is beyond the lower bound of derivable privacy budget, we should still choose the (extended) Threshold mechanism unless ϵ is extremely small (e.g. < 1) or k is very large (e.g., $k > 100$). Otherwise, if ϵ is beyond the upper bound of derivable privacy budget, we should still choose the Threshold mechanism unless ϵ is extremely large (e.g., > 20) or k is very small (e.g., $k < 10$). For other cases, we should choose either Forward or Backward perturbation mechanism, depending on which unit cost is higher, repetition or empty.

C. Real Applications: TLDP vs. VLDP

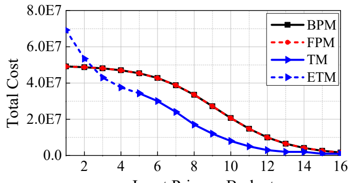
To compare the effectiveness of TLDP (e.g., BPM, FPM, TM/ETM) against VLDP (e.g., Laplace mechanism or Randomized Response), we measure their utilities in three real-world time series applications — frequency counting, simple moving average, and trajectory clustering.

1) Frequency Counting: We conduct frequency counting of value "up" in the US stock's time series of daily close. At each timestamp, we calculate the true count since the first timestamp and use it as the ground truth. Then we adopt VLDP-based Randomized Response and TLDP-based BPM, FPM and TM/ETM, and measure their deviation from the ground truth. Fig. 7(a) plot the MSE of these mechanisms, where the input privacy budget varies from 1 to 8, and the time window length $k = 10$. We observe that TM/ETM outperforms the other three mechanisms significantly in most cases. On the other hand, the MSE of FPM always stays high, which is consistent to our analysis in Section II-D that a timestamp being empty continuously causes underestimation to the subsequent counts. This phenomenon is also reflected on ETM when given a small privacy budget. As for BPM, it has both missing and repeated values, which can cancel each other to some extent and thus leads to a lower MSE than FPM. In addition, to verify the accuracy of our parameter setting of M, N, E and D for frequency counting in Section II-D, we plot the predicted and measured ratios of BPM to TM/ETM in terms of $Log(MSE)$ in Fig. 7(b). The red dash line indicates where BPM and TM/ETM are equivalently accurate. We observe that the predicted ratio shows similar trend as the measured ratio and can accurately predict at which point TM/ETM outperforms BPM.

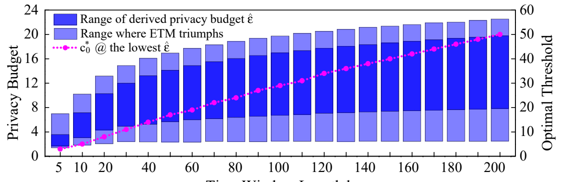
2) Simple Moving Average: We conduct simple moving average on the stock's daily close price and plot the MSE of estimated results in Fig. 7(c). The time window length k is set to 10 and the averaging range $d = k$. We observe that all three TLDP mechanisms, i.e., BPM, FPM and TM/ETM, significantly outperfosrm the VLDP-based Laplace mechanism, and TM/ETM achieves the lowest MSE in most cases. As opposed to frequency counting, FPM achieves higher accuracy than BPM, because an empty (i.e., skipped) timestamp in FPM does not cause as much deviation to the true average as a repeated value in BPM. Similar to frequency counting, we plot the predicted and measured ratios of FPM to TM/ETM



(a) $k = 10$

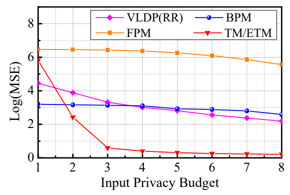


(b) $k = 50$

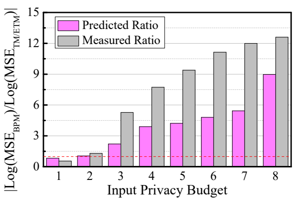


(c) Range of derived and triumphing privacy budgets of TM/ETM

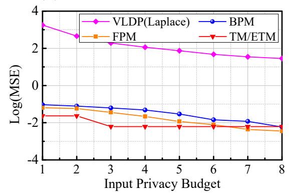
Fig. 6. Impact of sliding windows length k on total cost



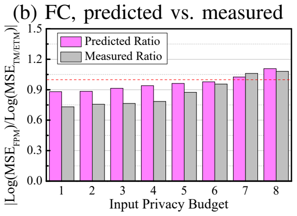
(a) FC, VLDP vs. TLDP



(b) FC, predicted vs. measured



(c) SMA, VLDP vs. TLDP



(d) SMA, predicted vs. measured

Fig. 7. Results of frequency counting and simple moving average

TABLE II
NMI on Trajectory Clustering

Input ϵ	1	2	3	4	5	6	7	8
Laplace	0.003	0.003	0.004	0.005	0.006	0.005	0.006	0.007
TM/ETM	0.572*	0.590*	0.610*	0.616	0.699	0.705	0.706	0.768

*: Extended Threshold Mechanism (ETM) is used.

under simple moving average in Fig. 7(d). We observe that the predicted ratio is closely aligned with the measure ratio, which verifies the correctness of our parameter setting of M, N, E and D for simple moving average in Section II-D.

3) Trajectory Clustering: We adopt the k-medoids algorithm [20] to cluster all 6307 trajectories into 6 groups. The clustering result over original data is regarded as the ground truth. Then we perturb these trajectories by TLDP-based Threshold mechanism (i.e., TM/ETM) and VLDP-based Laplace mechanism respectively, and apply the same k-medoids algorithm to cluster them again. For TM/ETM, we set $k = 10$. To measure the similarity between the ground-truth clusters and the clusters from perturbed trajectories, we adopt a classic metric Normalized Mutual Information (NMI) [21]. A larger NMI means more similarity. Table II shows the results with privacy budget from 1 to 8. TM/ETM always achieves a higher NMI than Laplace mechanism, which indicates a more similar clustering result to the ground truth.

VI. Related Work

In this section, we review existing works on differential privacy, and then with a focus on time series data release.

Differential Privacy. Differential privacy was first proposed in the centralized setting [22], [23]. To avoid relying on a trusted data collector, local differential privacy (LDP) was proposed to let each user perturb her data locally [13], [24]. In the literature, many LDP techniques have been proposed for various statistical collection tasks, such as frequency of categorical data [10], [25]–[27], and mean of numerical data [28]–[30]. Recently, the research focus in LDP has been shifted to more complex tasks, such as heavy hitter identification [31]–[33], itemset mining [34], [35], marginal release [36], [37], graph data mining [38]–[40], and key-value data collection [41], [42]. Some works also focus on learning problems [30], [43], [44].

Differential Privacy for Time Series Data. Existing work on time series focuses on differentially private aggregate statistics, e.g., frequency count. Depending on the privacy requirement, a perturbation mechanism can satisfy either *event-level* or *user-level* privacy [5], [45]. The former protects a user's presence at a single timestamp, while the later hides a user in the whole time series. Rastogi *et al.* [15] proposed to achieve differential privacy by injecting noise into the discrete Fourier coefficients. The work of [46], [47] proposed differentially private statistical release schemes that combine sampling or smoothing techniques. The works of [45], [48] studied continual counting queries on time series, and [49] further considers the correlation of continuously released time series data. Fan *et al.* [7] proposed a framework to release real-time aggregate statistics. Further, Kellaris *et al.* [6] merged the gap between event-level and user-level privacy by proposing a model call *w-event* privacy over infinite streams, which protects any event sequence occurring in w successive timestamps.

VII. Conclusion

This paper proposes local differential privacy in temporal setting (TLDP) for time series release. Since values must not be perturbed in many time-series applications, we propose three temporal-perturbation based mechanisms, i.e., Forward and Backward Perturbation mechanisms, and (extended) Threshold mechanism. We compare them through theoretical and empirical analysis under various privacy budgets and time window sizes. We also demonstrate the advantages of temporal perturbation over value perturbation through three real-world time series applications.

As for the future work, we plan to extend this work to multivariate time series data, where each timestamp comes with more than one time-dependent values that are correlated, for example, heart rate and blood pressure.

Acknowledgment

This work was supported by National Natural Science Foundation of China (Grant No: 61941121, 91646203, 62072390 and U1636205), the Research Grants Council, Hong Kong SAR, China (Grant No: 15238116, 15222118, 15218919, 15203120 and C1008-16G), the United States National Science Foundation (Grant No: 1931443).

References

[1] "COVID-19 community mobility reports," https://www.google.com/covid19/mobility/.

[2] L. Na, C. Yang, C.-C. Lo, F. Zhao, Y. Fukuoka, and A. Aswani, "Feasibility of reidentifying individuals in large national physical activity data sets from which protected health information has been removed with use of machine learning," *JAMA Network Open*, vol. 1, 2018.

[3] Y. Yang, M. Shao, S. Zhu, and G. Cao, "Towards statistically strong source anonymity for sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 3, p. 34, 2013.

[4] E. Shi, H. Chan, E. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *ISOC Network and Distributed System Security Symposium*. Citeseer, 2011.

[5] C. Dwork, "Differential privacy in new settings," in *SODA*. SIAM, 2010, pp. 174–183.

[6] G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias, "Differentially private event sequences over infinite streams," *Proceedings of the VLDB Endowment*, vol. 7, no. 12, pp. 1155–1166, 2014.

[7] L. Fan and L. Xiong, "An adaptive approach to real-time aggregate monitoring with differential privacy," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 26, no. 9, pp. 2094–2106, 2013.

[8] "Apache flink," https://flink.apache.org/.

[9] "High blood pressure (hypertension) - diagnosis - nhs," https://www.nhs.uk/conditions/high-blood-pressure-hypertension/diagnosis/.

[10] P. Kairouz, S. Oh, and P. Viswanath, "Extremal mechanisms for local differential privacy," in *NIPS*, 2014, pp. 2879–2887.

[11] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *TCC*. Springer, 2006, pp. 265–284.

[12] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *FOCS*. IEEE, 2007, pp. 94–103.

[13] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *FOCS*. IEEE, 2013, pp. 429–438.

[14] Tech. Rep., http://www.qingqingye.net/wp-content/uploads/2020/12/TechReport.pdf.

[15] V. Rastogi and S. Nath, "Differentially private aggregation of distributed time-series with transformation and encryption," in *SIGMOD*. ACM, 2010, pp. 735–746.

[16] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control, 5th Edition*. Holden-Day, 2015.

[17] "Historical daily prices of u.s. stocks," https://www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs.

[18] "Taxi trajectories," https://www.kaggle.com/crailtap/taxi-trajectory.

[19] S. L. Warner, "Randomized response: A survey technique for eliminating evasive answer bias," *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965.

[20] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3336–3341, 2009.

[21] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[22] C. Dwork, "Differential privacy," in *ICALP*. Springer, 2006, pp. 1–12.

[23] N. Li, M. Lyu, D. Su, and W. Yang, "Differential privacy: From theory to practice," *Synthesis Lectures on Information Security, Privacy, & Trust*, vol. 8, no. 4, pp. 1–138, 2016.

[24] Q. Ye and H. Hu, "Local differential privacy: Tools, challenges, and opportunities," in *WISE*. Springer, 2020, pp. 13–23.

[25] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *CCS*. ACM, 2014, pp. 1054–1067.

[26] R. Bassily and A. Smith, "Local, private, efficient protocols for succinct histograms," in *STOC*. ACM, 2015, pp. 127–135.

[27] T. Wang, J. Blocki, N. Li, and S. Jha, "Locally differentially private protocols for frequency estimation," in *USENIX Security*, 2017, pp. 729–745.

[28] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," in *NIPS*, 2017, pp. 3574–3583.

[29] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Minimax optimal procedures for locally private estimation," *Journal of the American Statistical Association*, vol. 113, no. 521, pp. 182–201, 2018.

[30] N. Wang, X. Xiao, Y. Yang, J. Zhao, S. C. Hui, H. Shin, J. Shin, and G. Yu, "Collecting and analyzing multidimensional data with local differential privacy," in *ICDE*. IEEE, 2019.

[31] Z. Qin, Y. Yang, T. Yu, I. Khalil, X. Xiao, and K. Ren, "Heavy hitter estimation over set-valued data with local differential privacy," in *CCS*. ACM, 2016, pp. 192–203.

[32] R. Bassily, U. Stemmer, A. G. Thakurta *et al.*, "Practical locally private heavy hitters," in *NIPS*, 2017, pp. 2285–2293.

[33] M. Bun, J. Nelson, and U. Stemmer, "Heavy hitters and the structure of local privacy," in *PODS*. ACM, 2018, pp. 435–447.

[34] T. Wang, N. Li, and S. Jha, "Locally differentially private frequent itemset mining," in *S&P*. IEEE, 2018, pp. 127–143.

[35] N. Wang, X. Xiao, Y. Yang, T. D. Hoang, H. Shin, J. Shin, and G. Yu, "Privtrie: Effective frequent term discovery under local differential privacy," in *ICDE*. IEEE, 2018, pp. 821–832.

[36] G. Cormode, T. Kulkarni, and D. Srivastava, "Marginal release under local differential privacy," in *SIGMOD*. ACM, 2018, pp. 131–146.

[37] Z. Zhang, T. Wang, N. Li, S. He, and J. Chen, "CALM: Consistent adaptive local marginal for marginal release under local differential privacy," in *CCS*. ACM, 2018, pp. 212–229.

[38] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, "Generating synthetic decentralized social graphs with local differential privacy," in *CCS*. ACM, 2017, pp. 425–438.

[39] Q. Ye, H. Hu, M. H. Au, X. Meng, and X. Xiao, "Towards locally differentially private generic graph metric estimation," in *ICDE*. IEEE, 2020, pp. 1922–1925.

[40] ——, "LF-GDPR:graph metric estimation with local differential privacy," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 10.1109/TKDE.2020.3047124.

[41] Q. Ye, H. Hu, X. Meng, and H. Zheng, "PrivKV: Key-value data collection with local differential privacy," in *S&P*. IEEE, 2019, pp. 294–308.

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

[43] H. Zheng, Q. Ye, H. Hu, C. Fang, and J. Shi, "BDPL: A boundary differentially private layer against machine learning model extraction attacks," in *ESORICS*. Springer, 2019, pp. 66–83.

[44] ——, "Protecting decision boundary of machine learning model with differentially private perturbation," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 10.1109/TDSC.2020.3043382.

[45] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum, "Differential privacy under continual observation," in *STOC*. ACM, 2010, pp. 715–724.

[46] G. Acs and C. Castelluccia, "A case study: Privacy preserving release of spatio-temporal density in paris," in *SIGKDD*. ACM, 2014, pp. 1679–1688.

[47] Y. Chen, A. Machanavajjhala, M. Hay, and G. Miklau, "Pegasus: Data-adaptive differentially private stream processing," in *CCS*. ACM, 2017, pp. 1375–1388.

[48] T.-H. H. Chan, E. Shi, and D. Song, "Private and continual release of statistics," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 3, p. 26, 2011.

[49] Y. Cao, M. Yoshikawa, Y. Xiao, and L. Xiong, "Quantifying differential privacy in continuous data release under temporal correlations," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 31, no. 7, pp. 1281–1295, 2019.