CGM: An Enhanced Mechanism for Streaming Data Collection with Local Differential Privacy

Ergute Bao
National University of Singapore
ergute@comp.nus.edu.sg

Yin Yang
Hamad Bin Khalifa University
yyang@hbku.edu.qa

Xiaokui Xiao
National University of Singapore
xkxiao@nus.edu.sg

Bolin Ding
Alibaba Group
bolin.ding@alibaba-inc.com

ABSTRACT

Local differential privacy (LDP) is a well-established privacy protection scheme for collecting sensitive data, which has been integrated into major platforms such as iOS, Chrome, and Windows. The main idea is that each individual randomly perturbs her data on her local device, and only uploads the noisy version to an untrusted data aggregator. This paper focuses on the collection of streaming data consisting of regular updates, *e.g.*, daily app usage. Such streams, when aggregated over a large population, often exhibit strong *autocorrelations*, *e.g.*, the average usage of an app usually does not change dramatically from one day to the next. To our knowledge, this property has been largely neglected in existing LDP mechanisms. Consequently, data collected with current LDP methods often exhibit unrealistically violent fluctuations due to the added noise, drowning the overall trend, as shown in our experiments.

This paper proposes a novel *correlated Gaussian mechanism* (*CGM*) for enforcing (ϵ, δ)-LDP on streaming data collection, which reduces noise by exploiting public-known autocorrelation patterns of the aggregated data. This is done through non-trivial modifications to the core of the underlying Gaussian Mechanism; in particular, CGM injects temporally correlated noise, computed through an optimization program that takes into account the given autocorrelation pattern, data value range, and utility metric. CGM comes with formal proof of correctness, and consumes negligible computational resources. Extensive experiments using real datasets from different application domains demonstrate that CGM achieves consistent and significant utility gains compared to the baseline method of repeatedly running the underlying one-shot LDP mechanism.

PVLDB Reference Format:
Ergute Bao, Yin Yang, Xiaokui Xiao, and Bolin Ding. CGM: An Enhanced Mechanism for Streaming Data Collection with Local Differential Privacy. PVLDB, 14(11): XXX-XXX, 2021.
doi:10.14778/3476249.3476277

1 INTRODUCTION

Local differential privacy (LDP) [16, 30], which first appeared in Ref. [21], is a widely accepted framework for the collection and analysis of sensitive data. LDP provides a strong, information-theoretic guarantee on the user's privacy, and has been deployed in common software systems, including Apple's iOS, macOS and Safari [2], Microsoft Windows 10 [15], and Google Chrome [20]. Specifically, in the LDP setting, an untrusted data aggregator aims to collect sensitive information from individual users, and obtain meaningful statistics from the data, while ensuring the user's privacy. To do so, each individual randomly perturbs her data on her local device, and only reports the noisy version to the data aggregator. In other words, the data aggregator never has access to the exact values of the sensitive data. The scale of the random noise injected to each data record is calibrated to a pre-defined *privacy budget* of each individual, as well as the the range of the data values [17]. Intuitively, the random noise provides the user plausible deniability of what her true sensitive values are, given her reported randomized version.

So far, most existing mechanisms for enforcing LDP aim at collecting static data, such as the user's age and gender. In practice, the data aggregator often wants to collect information that changes over time through continuous updates, *e.g.*, "How many times have you visited the CDC website yesterday?" and "How long have you stayed in that area with a surge of COVID-19 cases in the past few hours?" In fact, Apple already collects daily energy usage and number of crashes of the Safari browser with LDP [2]. In these scenarios, a simple and commonly used approach is to repeatedly execute a one-shot LDP mechanism once at every timestamp. *The problem with this approach is that it fails to capture the inherent temporal autocorrelations of the data.* For instance, during this hard time of COVID-19, the number of visits to the government's public health website, when averaged over a large population, is unlikely to drop drastically from one day to the next. On the other hand, web traffic data collected with the above approach often exhibit unrealistically violent fluctuations, as shown in our experiments.

Specifically, as explained in Section 2, to satisfy LDP, the individual's data (*e.g.*, number of visits to a website) needs to be perturbed according to the entire range containing all possible data values (*e.g.*, from zero to the maximum daily visits of any user to any website), which leads to a high noise scale. Meanwhile, in many applications, the difference between two consecutive data updates is usually significantly smaller than the full range of the data item, *e.g.*, the number of visits to a website in two consecutive days tend to be similar when averaged over a large population, as mentioned above. Accordingly, we assume that a publicly known bound C that clips the maximum change (*i.e.*, the *differential*) between the

Proceedings of the VLDB Endowment, Vol. 14, No. 11 ISSN 2150-8097.
doi:10.14778/3476249.3476277

data item in two adjacent timestamps, and we aim to exploit this information to reduce the noise scale required to satisfy LDP. Note that this assumption concerns the maximum value differential of an *aggregate* stream over a large population, as elaborated in Section 3.1; in particular, individual time series may still exhibit high fluctuations between consecutive timestamps.

Utilizing the differential clipping bound C is non-trivial. Continuing the daily website visits reporting example, if on each day (say, day t), we simply let the user report the differential with respect to the previous day (*i.e.*, the difference between the number of visits on day t and that on day $t-1$), then the reported value is indeed bounded by C, leading to a smaller noise scale. However, this does not help the data aggregator, who needs the user's data item, not the differential. To reconstruct the user's number of website visits on day t, the data aggregator needs to sum up (i) her estimated number of visits on day $t-1$ and (ii) the reported differential on day t. The error of this sum is strictly higher than that of (i) alone due to the accumulated noise of the two components. In other words, the error of the reconstructed user's data item increases with time. As we elaborate later in Section 3.2, this scheme of reporting value differentials often leads to noisier data value estimates compared to the naive method of ignoring C and directly collecting the user's data with a one-shot LDP mechanism at each timestamp.

In this paper, we propose a novel solution, namely *correlated Gaussian mechanism* (*CGM*), that properly utilizes the knowledge of the differential bound C in continual data collection with LDP. Specifically, similar to the naive method, CGM lets the user directly report a perturbed version of her current sensitive value at each timestamp, and nothing else. Meanwhile, CGM makes non-trivial modifications to the core math of the underlying LDP mechanism to exploit the knowledge of C and reduce the noise scale. In particular, the random noise injected according to CGM is *correlated* at adjacent timestamps, and the noise vector is solved from an optimization program with the objective of maximizing a given data utility function, *e.g.*, mean square error of the perturbed values with respect to the exact ones.

We formally prove that CGM satisfies (ϵ, δ)-LDP, and analyze its expected utility gains compared to the baseline method of repeatedly applying a one-shot mechanism at every timestamp. In particular, the utility enhancement brought by CGM is more pronounced for longer sequences with stronger data correlations, *i.e.*, a smaller value differential bound C compared to the range of the corresponding data values. Further, the computational overhead of CGM is negligible, even for long sequences. Extensive experiments with real data from different application domains confirm the significant advantage of CGM in terms of result utility.

In the following, Section 2 provides necessary background on LDP. Section 3 defines the problem of streaming data collection with LDP, and discusses naive solutions. Section 4 presents the proposed solution CGM. Section 5 formally proves the correctness of CGM and analyzes its performance. Section 6 contains a thorough set of experimental evaluations. Section 7 reviews related work. Finally, Section 8 concludes the paper with future directions.

2 PRELIMINARIES

2.1 Local Differential Privacy

We follow the (ϵ, δ)-local differential privacy (LDP) framework for privacy preserving data collection and analysis, defined as follows.

Definition 2.1 ((ϵ, δ)-Local Differential Privacy [17]). A randomized mechanism $\mathcal{M}$ satisfies *(ϵ, δ)-local differential privacy (LDP)* if and only if

$$\Pr[\mathcal{M}(x) \in O] \leq \exp(\epsilon) \cdot \Pr[\mathcal{M}(x') \in O] + \delta, \tag{1}$$

for any set of output $O \subseteq Range(\mathcal{M})$ and any inputs x and x'.

In the above definition, the input x is the sensitive data held by an individual, and $\mathcal{M}(x)$ is the perturbed version reported to the data aggregator. Intuitively, an (ϵ, δ)-LDP algorithm ensures that the output distribution for an individual with data x is similar to that for an individual with data x', for all possible x and x' in the data domain. The similarity of the output distributions is quantified by parameters ϵ and δ, which determine the strength of privacy guaranteed by the mechanism $\mathcal{M}$. Smaller ϵ and δ indicate a stronger privacy guarantee, and vice versa.

The privacy parameter ϵ is commonly referred to as the *privacy budget* of the individual, and δ can be roughly viewed as a *probability of failure*, *i.e.*, when the individual's *privacy loss*, measured by $\log\left(\frac{\Pr[\mathcal{M}(x)\in O]}{\Pr[\mathcal{M}(x')\in O]}\right)$, exceeds ϵ. Parameter ϵ is considered as a budget in part due to a classic composition rule [19], which states that given any two mechanisms $\mathcal{M}_1$ and $\mathcal{M}_2$ satisfying (ϵ_1, δ_1)- and (ϵ_2, δ_2)-differential privacy respectively, their combination satisfies $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-differential privacy. In other words, when multiple mechanisms are applied (*e.g.*, for reporting multiple updates at different timestamps), their consumption of the privacy budget ϵ accumulates in an additive manner. This composition result, however, is in fact rather pessimistic, as we show in the next subsection.

Post-processing (either randomized or deterministic), performed after a differentially private mechanism, does not affect the privacy guarantee, according to the following lemma.

Lemma 2.2 (Post-processing [17]). *Let $\mathcal{M}$ be an (ϵ, δ)-LDP mechanism, and G be a function whose input is the output of $\mathcal{M}$. Then, $G(\mathcal{M})$ also satisfies (ϵ, δ)-LDP.*

2.2 Analytic Gaussian Mechanism

Let F be a function that maps the input from the domain $\mathbb{R}^d$ into $\mathbb{R}^d$, where d is the dimensionality of the underlying data. To convert F into a differentially private mechanism, a canonical approach is to inject random noise into the output of F. The scale of the noise is calibrated according to the *sensitivity* of F, defined as follows.

Definition 2.3 (Sensitivity [17]). The sensitivity of a function $F : \mathbb{R}^d \to \mathbb{R}^d$, denoted as $S(F)$, is defined as

$$S(F) = \max_{x,x'} \|F(x') - F(x)\|,$$

where x and x' are any pair of valid inputs to function F and $\|\cdot\|$ is a norm.

In this work, we focus on the L_2 norm, denoted as $\|\cdot\|$ henceforth. A classic mechanism for enforcing (ϵ, δ)-differential privacy is the Gaussian mechanism [19], which injects random Gaussian noise

Table 1: List of Notations

Symbol	Description	Section
$\mathcal{M}$	A differentially private mechanism	2
O	A set of outputs of mechanism $\mathcal{M}$	2
ϵ, δ	Privacy parameters	2
$S(F)$	L_2 sensitivity of function F	2
n	Number of individuals in the application	3
l	Length of each data stream	3
$x_{k,i}$	i-th element of the k-th data stream	3
d	Dimensionality of each element $x_{k,i}$	3
C	Differential bound of adjacent data items	3
$\Delta_{k,i}$	i-th differential of the k-th data stream	3
f_i	i-th function of interest	4
f_i^*	Perturbed version of f_i	4
ξ_i	Surrogate function for f_i	4
ξ_i^*	Perturbed version of ξ_i	4
γ_i	Noise injected into f_i	4
α_{ij}	Parameters in CGM	4
η_i	i-th fresh noise	4
σ_i	Scale of the i-th fresh noise	4
r_i	Reuse ratio in the i-th estimate	4

into the output of F based on its L_2 sensitivity. Subsequent work found that the noise scale of the original Gaussian mechanism can be reduced through a more careful analysis. The modern, state-of-the-art version is the *analytic Gaussian mechanism*, as follows.

Lemma 2.4 (Analytic Gaussian Mechanism [3, 42]). *Let $F : \mathbb{R}^d \to \mathbb{R}^d$ be a function. The analytic Gaussian mechanism that injects Gaussian noise $\mathcal{N}\left(\mathbf{0}, \sigma^2 \cdot \mathbf{I}\right)$ into the output of F satisfies (ϵ, δ)-differential privacy, if and only if*

$$\frac{S(F)}{\sigma} \leq \sqrt{2}\left(\sqrt{\chi^2 + \epsilon} - \chi\right), \tag{2}$$

where $\mathbf{0}$ and $\mathbf{I}$ are a zero vector and a $d \times d$ identity matrix, respectively, and χ is the solution to

$$\operatorname{erfc}(\chi) - \exp(\epsilon) \cdot \operatorname{erfc}\left(\sqrt{\chi^2 + \epsilon}\right) = 2\delta, \tag{3}$$

and erfc() *denotes the complementary error function, i.e.,*

$$\operatorname{erfc}(x) \triangleq 1 - \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \, \mathrm{d}t.$$

According to Eq. (2), the scale of the Gaussian noise, controlled by the standard deviation σ, is calibrated to the sensitivity $S(F)$ of the function of interest F. In our context, F is simply an identity function when the individual reports her data item x. Hence, when the user reports her data for l timestamps, the overall sensitivity is proportional to $\sqrt{l}$, meaning that the noise scale $\sigma = O(\sqrt{l})$. This is a much tighter bound than the classic composition rule mentioned at the end of Section 2.1, since if we were to split the privacy budget ϵ into l equal shares and apply the Gaussian mechanism on each timestamp using a share of the budget, then the overall noise scale would grow linearly with l. Table 1 summarizes frequent notations used throughout the paper.

3 PROBLEM SETTING

Section 3.1 clarifies the problem studied in this paper. Section 3.2 describes naive solutions and their limitations.

3.1 Problem Definition

Recall from Section 1 that in our target application setting, an untrusted data aggregator aims to collect sensitive data from a set of individual users at regular time intervals, while satisfying LDP. Note that according to Section 2, with finite overall privacy parameters ϵ and δ, the number of updates that a user can report to the aggregator is necessarily finite. In practice, major aggregators circumvent this problem by requiring the user to renew her privacy budget periodically, *e.g.*, $\epsilon = 2$ each day for some functionalities of iOS [2], which enables data collection indefinitely. The proposed solution is compatible with such settings, which is an important consideration in our mechanism design. Without loss of generality, in the following we focus on finite streams of length l, and privacy parameters ϵ, δ for the whole stream; we describe the adaptation to infinite streams and periodically renewing privacy parameters whenever necessary.

Formally, let n be the total number of users. Each user $k \in \{1, 2, \ldots, n\}$ possesses a private data item $x_{k,i}$ at each time instance $i \in \{1, 2, \ldots, l\}$. Each such data item $x_{k,i}$ is a vector of fixed dimensionality, denoted as d, *i.e.*, $x_{k,i} \in \mathbb{R}^d$. Without loss of generality, we assume that each data item $x_{k,i}$ lies within a d-dimensional hypersphere with unit diameter, *i.e.*, $\|x_{k,i}\| \leq \frac{1}{2}$, where $\|\cdot\|$ denotes the L_2 norm as mentioned in Section 2. This property can be ensured by scaling all data vectors using their maximum possible L_2 norm, based on public knowledge in the application domain. Note that for each individual k, her data items gradually arrive over time: at each timestamp i, she has access to all previous data items, namely, $x_{k,1}, \ldots, x_{k,i-1}$, and does not have access to future data items after time i.

As explained in Section 1, we focus on scenarios in which the data items in an aggregate stream exhibit publicly-known autocorrelations. In particular, we assume that the difference between any two adjacent data items in the aggregate stream is bounded by a public constant $C \in (0, 1)$, referred to as the *differential bound*.[1] In the LDP setting, the data aggregator collects perturbed values before computing the aggregate over the entire populations, meaning that the differential bound C on the aggregate stream cannot be directly utilized in the LDP mechanism. Hence, in our proposed solution, we enforce bound C on individual time series: formally, for any user k and any timestamp $i > 1$, we enforce that $\|x_{k,i} - x_{k,i-1}\| \leq C$, which can be done, *e.g.*, by clipping out-of-bound outlier items. Note that since the differential bound assumption is over the aggregate stream rather than individual series, such clipping (say, for data item $x_{k,i}$) may introduce a temporarily high bias in the estimate at the data aggregator for one particular stream (*i.e.*, stream k) at a particular timestamp (time i), when user k's data stream experiences a sudden fluctuation at time i.

[1] In some applications, the streaming data can also be autocorrelated with a lag higher than 1. For instance, for data exhibiting strong weekly periodicity patterns, a daily update $x_{k,i}$ can be more strongly correlated with the update one week ago (*i.e.*, $x_{k,i-7}$) than the one yesterday ($x_{k,i-1}$). We discuss such cases at the end of this subsection.

On the other hand, intuitively, as long as the differential bound C holds for the average user over a large population, the bias introduced in such clipping can be more than compensated by the reduction of noise scale required to satisfy LDP, as shown in our experiments in Section 6.

Our goal is to find a randomized mechanism $\mathcal{M}$ which allows each user to release her private data to the untrusted data aggregator, while satisfying (ϵ, δ)-LDP. Formally, the problem of streaming data collection with LDP is defined as follows.

Problem Definition. *Given n users, each possessing a data stream with l timestamps $x_{k,1}, \ldots, x_{k,l}$, satisfying, each data item $\|x_{k,i}\| \leq \frac{1}{2}$ $(i = 1, \ldots, l,$ and $k = 1, \ldots, n)$, design a mechanism $\mathcal{M}$ such that at each time instance i, $\mathcal{M}$ takes as input a user's data items: $(x_{k,1}, \ldots, x_{k,i})$, and outputs $x^*_{k,i}$, with the following objective:*

$$\min \frac{1}{l} \sum_{i=1}^{l} \mathbb{E}\left[\|x_{k,i} - x^*_{k,i}\|^2\right], \tag{4}$$

subject to the constraint that $\mathcal{M}$ satisfies (ϵ, δ)-LDP, i.e.,

$$\Pr\left[\mathcal{M}\left(x_{k,1}, \ldots, x_{k,l}\right) \in O\right] \leq \exp(\epsilon) \cdot \Pr\left[\mathcal{M}\left(x'_{k,1}, \ldots, x'_{k,l}\right) \in O\right] + \delta, \tag{5}$$

for any input data stream $(x_{k,1}, \ldots, x_{k,l})$, and any valid data stream $(x'_{k,1}, \ldots, x'_{k,l})$ from the same data domain, and any set of outputs $O \subseteq Range(\mathcal{M})$.

Discussions. So far, we have focused on the case where the differential bound C applies to two adjacent timestamps, *i.e.*, $\|x_{k,t} - x_{k,t-1}\| \leq C$. In applications where the data exhibit a periodic pattern, it is possible that a new data item is correlated with a previous one that is $j > 1$ timestamps behind, *i.e.*, $\|x_{k,t} - x_{k,t-j}\| \leq C$. For instance, with daily updates, we can set $j = 7$ to capture the situation where the data streams demonstrate strong weekly periodicity. This case can be transformed to our problem, by conceptually splitting each data stream into j sub-streams. In the above example where the data exhibit a weekly autocorrelation, we split the stream into 7 sub-streams corresponding to Monday, Tuesday, ..., Sunday. Then, each sub-stream would satisfy the value differential bound on adjacent timestamps, and our proposed solution directly applies.

Our setting assumes an additive bound C of the maximum difference between data values on two adjacent timestamps. When all data values are positive, the proposed solution can also be applied to the case with a multiplicative bound on the ratio between two consecutive timestamps, by taking logarithm of all data items in a pre-processing step. In the following, we focus on the case with an additive bound.

3.2 Naive Solutions

As mentioned in Section 1, a naive solution for our problem is to execute a one-shot LDP mechanism at every timestamp. Specifically, at each timestamp i, each user k applies the analytic Gaussian mechanism (defined in Lemma 2.4) to perturb her value $x_{k,i}$, as shown in Algorithm 1. To do so, the algorithm independently samples a noise vector from the Gaussian distribution $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, adds the noise to $x_{k,i}$, and outputs the perturbed value $x^*_{k,i}$.

Algorithm 1: Perturbing data items directly

Input: data items $\{x_{k,i}\}_{i=1}^{l}$, where $\|x_{k,i}\| \leq \frac{1}{2}$ and $\|x_{k,i} - x_{k,i-1}\| \leq C$, privacy parameters ϵ, δ.
Output: perturbed estimates $\{x^*_{k,i}\}_{i=1}^{l}$.

1 **for** $i = 1$ ***to*** l **do**
2 $\quad x^*_{k,i} \longleftarrow x_{k,i} + \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$; where σ is computed as in Eq. (7) and $\mathbf{I}$ is the $d \times d$ dimension identity matrix.
3 $\quad$ Output $x^*_{k,i}$;

Algorithm 2: Perturbing value differentials

Input: data items $\{x_{k,i}\}_{i=1}^{l}$, where $\|x_{k,i}\| \leq \frac{1}{2}$ and $\|x_{k,i} - x_{k,i-1}\| \leq C$, privacy parameters ϵ, δ.
Output: perturbed estimates $\{x^*_{k,i}\}_{i=1}^{l}$.

1 **for** $i = 1$ ***to*** l **do**
2 $\quad$ **if** $i = 1$ **then**
3 $\quad\quad$ Output $x^*_{k,i}$ computed as in Algorithm 1;
4 $\quad$ **else**
5 $\quad\quad \Delta^*_{k,i} \longleftarrow x_{k,i} - x_{k,i-1} + \mathcal{N}(\mathbf{0}, 4C^2\sigma^2\mathbf{I})$; where σ is computed as in Eq. (7) and $\mathbf{I}$ is the $d \times d$ dimension identity matrix.
6 $\quad\quad$ Output $x^*_{k,i} = x^*_{k,i-1} + \Delta^*_{k,i}$;

The following Lemma formally states the utility and privacy guarantee of Algorithm 1. The proof follows by applying Lemma 2.4 on the identity function with the sensitivity $\sqrt{l}$, since each data item $x_{k,i}$ satisfies $\|x_{k,i}\| \leq \frac{1}{2}$ as described in Section 3.1. In terms of utility, $x^*_{k,i}$ is an unbiased estimator of $x_{k,i}$, since the Gaussian noise injected is of mean $\mathbf{0}$. The expected mean squared error for the k-th individual with respect to the i-th timestamp is simply the sum of variance across all dimensions, *i.e.*, $\sigma^2 \cdot d$.

Lemma 3.1. *Algorithm 1 satisfies (ϵ, δ)-LDP with expected error:*

$$\frac{1}{l} \sum_{i=1}^{l} \mathbb{E}\left[\|x^*_{k,i} - x_{k,i}\|^2\right] = \sigma^2 d, \tag{6}$$

where d is the dimensionality for each element $x_{k,i}$, and l is the length of the data stream. In addition, σ satisfies

$$\frac{\sqrt{l}}{\sigma} \leq \sqrt{2}\left(\sqrt{\chi^2 + \epsilon} - \chi\right), \tag{7}$$

and χ is defined in Eq. (3)*. The expectation is taken over the randomness in the algorithm.*

Clearly, Algorithm 1 does not use the differential bound C at all. To utilize C, another naive solution is to let each user release value differentials, as mentioned in Section 1. We outline this approach in Algorithm 2. Specifically, at the first timestamp, each user k perturbs her first data item $x_{k,1}$ similarly as Algorithm 1. After that, at each timestamp $i > 1$, the user perturbs the differential $\Delta_{k,i} \triangleq x_{k,i} - x_{k,i-1}$ using the analytic Gaussian mechanism, where the scale of the noise is calibrated to the range of the differential $\Delta_{k,i}$, *i.e.*, $2C$, since for any input data stream $(x_{k,1}, \ldots, x_{k,l})$, any

valid data stream $(x'_{k,1}, \ldots, x'_{k,l})$ from the same data domain, we have $\|\Delta_{k,i} - \Delta'_{k,i}\| \leq \|\Delta_{k,i}\| + \|\Delta'_{k,i}\| \leq 2C$.

Algorithm 2 satisfies (ϵ, δ)-LDP, since (i) it performs the same operation as Algorithm 1 at timestamp 1, and (ii) at timestamp $i = 2, \ldots, l$, the algorithm perturbs $\Delta_{k,i}$, which has sensitivity $2C$; therefore, the required noise scale for perturbing $\Delta_{k,i}$ is $2C \cdot \sigma$, where σ is the noise scale needed for perturbing $x_{k,i}$, which leads to the noise variance $4C^2\sigma^2$ shown in Algorithm 2. We omit a formal proof for brevity.

It is easy to see that Algorithm 2 always yields worse utility than Algorithm 1, as mentioned in Section 1. This is due to the fact that the amount of noise in $x^*_{k,i}$ (where $i \geq 2$) is the sum of all noise in $x^*_{k,1}$ and $\Delta^*_{k,j}$, $2 \leq j \leq i$, which grows linearly with respect to i. Hence, Algorithm 2 incurs error:

$$\frac{1}{l}\sum_{i=1}^{l} \mathbb{E}\left[\|x^*_{k,i} - x_{k,i}\|^2\right] = \sigma^2 d\left(1 + 2C^2(l-1)\right), \tag{8}$$

which is larger than the error of Algorithm 1, *i.e.*, $\sigma^2 d$ (Lemma 3.1). In the next section, we show how to properly utilize the differential bound C to reduce error.

Discussion. In Algorithm 2, for each user k, the noise contained in an earlier data item, say, $x^*_{k,j}$, is accumulated to all later data items, $x^*_{k,i}$, $i > j$. Hence, it is theoretically possible to reduce overall error by letting the user report earlier data items with a lower amount of noise, and later ones with larger noise, so that the overall mechanism still satisfies (ϵ, δ)-LDP for all l timestamps. This modification, however, is incompatible with the setting where the user allows a fixed privacy budget *per timestamp*, a common practice in current LDP implementations as described in the beginning of Section 3.1. To see this, observe that in both Algorithms 1 and 2, the data release at each timestamp consumes the same amount of privacy budget, which satisfies (ϵ', δ')-LDP where $\epsilon' \approx \frac{\epsilon}{\sqrt{l}}$, $\delta' \approx \frac{\delta}{l}$ for a large l. Breaking the symmetry of noise at different timestamps would invalidate this property, and, thus, is incompatible with the periodically refreshing privacy budget setting.

4 CORRELATED GAUSSIAN MECHANISM

In this section, we present the proposed Correlated Gaussian Mechanism (CGM). In the following, we first use a simple example to explain the rationale of using correlated Gaussian noise in Section 4.1. Then, we formalize CGM in the general form in Section 4.2, and apply it to our problem setting in Section 4.3.

4.1 Rationale

Let F be a function that maps any input data x from a data universe to a vector $(f_1(x), f_2(x))$, where $f_1(x)$ and $f_2(x)$ can be any two values in $[-\frac{1}{2}, +\frac{1}{2}]$, as long as their differential satisfies that $|f_1(x) - f_2(x)| \leq C < \frac{1}{2}$ for some constant C. Then, the sensitivity of F is:

$$S(F) = \max_{x,x'} \|F(x) - F(x')\| = \sqrt{2}.$$

The maximum is taken overall all possible data pairs x and x' that are from the same data universe. For instance, we have $\|F(x) - F(x')\| = \sqrt{2}$ when $F(x) = (-\frac{1}{2}, -\frac{1}{2})$ and $F(x') = (+\frac{1}{2}, +\frac{1}{2})$.

Suppose that we are to release $F(x)$ under (ϵ, δ)-LDP using the analytic Gaussian mechanism in Lemma 2.4, with privacy parameters, say, $\epsilon = 1$ and $\delta = 10^{-5}$. Then, by Lemma 2.4, the analytic Gaussian mechanism would inject independent Gaussian noise $\mathcal{N}(0, \sigma^2)$ into $f_1(x)$ and $f_2(x)$, with parameter σ set to

$$\sigma = \frac{\sqrt{2}}{\sqrt{2}(\sqrt{\chi^2 + \epsilon} - \chi)} \approx 5.28, \tag{9}$$

where $\chi \approx 2.54$ is the solution to Eq. (3). Observe that σ is decided solely by the sensitivity of F, meaning that the analytic Gaussian mechanism fails to utilize the correlation between f_1 and f_2, *i.e.*, $|f_1(x) - f_2(x)| \leq C < \frac{1}{2}$ for any input x.

Next, consider an alternative approach that injects correlated Gaussian noise into f_1 and f_2 as follows. First, we sample a Gaussian noise η_1 from $\mathcal{N}(0, \sigma^2)$ with $\sigma \approx 5.28$ (as in Eq. (9)), and add it to f_1 to obtain a noisy value $f_1^* = f_1 + \eta_1$. After that, instead of directly injecting noise into f_2, we consider an alternative function

$$\xi_2 = \alpha \cdot f_2 - \beta \cdot f_1,$$

where $\alpha = 1 + (1 - 2C)^2$ and $\beta = (1 - 2C)$ are derived from public information C, and, thus, are not sensitive information. As will be clarified shortly, ξ_2 is used to facilitate the generation of noise for f_2, while α and β are constants that decide how the noise in f_2 should correlate with that in f_1.

Given ξ_2, we sample another Gaussian noise η_2 from $\mathcal{N}(0, \sigma^2)$ with the same $\sigma \approx 5.28$ as in Eq. (9), and compute a noisy value $\xi_2^* = \xi_2 + \eta_2$. After that, we use ξ_2^* and f_1^* to generate a noisy version f_2^* of f_2:

$$f_2^* = \frac{1}{\alpha}\xi_2^* + \frac{\beta}{\alpha} f_1^*.$$

Then, we have

$$\begin{aligned} f_2^* &= \frac{1}{\alpha}\xi_2 + \frac{1}{\alpha}\eta_2 + \frac{\beta}{\alpha} f_1 + \frac{\beta}{\alpha}\eta_1 \\ &= f_2 - \frac{\beta}{\alpha} f_1 + \frac{1}{\alpha}\eta_2 + \frac{\beta}{\alpha} f_1 + \frac{\beta}{\alpha}\eta_1 \\ &= f_2 + \frac{1}{\alpha}\eta_2 + \frac{\beta}{\alpha}\eta_1. \end{aligned}$$

In other words, the noise in f_2^* is a linear combination of the noise in ξ_2^* and f_1^*.

The benefit of this correlated noise approach is that it reduces the noise amount in f_2^*. In particular, the variance of f_2^* is:

$$\frac{1}{\alpha^2}\sigma^2 + \frac{\beta^2}{\alpha^2}\sigma^2 = \frac{\sigma^2}{1 + (1-2C)^2} \approx \frac{(5.28)^2}{1 + (1-2C)^2}.$$

In contrast, if we inject independent Gaussian noise into f_1 and f_2, the variance of the noise in f_2^* would be $\sigma^2 \approx (5.28)^2$, according to Eq. (9), which is strictly larger than $\frac{\sigma^2}{1+(1-2C)^2}$ since $C < \frac{1}{2}$.

In addition, the correlated noise approach still ensures (ϵ, δ)-DP with $\epsilon = 1$ and $\delta = 10^{-5}$, as explained in the following. Observe that for any two inputs x and x',

$$\begin{aligned} &\left|\xi_2(x) - \xi_2(x')\right| \\ =&\left|\alpha \cdot (f_2(x) - f_2(x')) - \beta \cdot (f_1(x) - f_1(x'))\right| \\ \leq&(\alpha - \beta) \cdot \left|f_2(x) - f_2(x')\right| + \beta \cdot |f_2(x) - f_1(x)| \\ &+ \beta \cdot \left|f_1(x') - f_2(x')\right| \\ \leq&(\alpha - \beta) + \beta \cdot C + \beta \cdot C \quad \leq 1. \end{aligned}$$

Let F' be a function takes any input data x, and maps it to a vector $[f_1(x), \xi_2(x)]$. The L_2 sensitivity of F' is

$$\begin{aligned} S(F') &= \max_{x,x'} \|F'(x) - F'(x')\| \\ &= \max_{x,x'} \sqrt{(f_1(x) - f_1(x'))^2 + (\xi_2(x) - \xi_2(x'))^2} \\ &\leq \max_{x,x'} \sqrt{1+1} \ = \sqrt{2}. \end{aligned}$$

Then, by Lemma 2.4, injecting Gaussian noise η_1 and η_2 into $f_1(x)$ and $\xi_2(x)$, respectively, achieves (ϵ, δ)-DP with $\epsilon = 1$ and $\delta = 10^{-5}$. Since $f_2^*(x)$ is obtained by post-processing $\xi_2^*(x)$ and $f_1^*(x)$, by Lemma 2.2, the generation of $f_1^*(x)$ and $f_2^*(x)$ also satisfies (ϵ, δ)-DP with $\epsilon = 1$ and $\delta = 10^{-5}$.

The above example shows that when two functions f_1 and f_2's outputs have a small difference $|f_1(x) - f_2(x)|$ for any input data x, we can reduce the amount of noise required to make f_1 and f_2 differentially private, by correlating the noise in f_2 with that in f_1. In what follows, we formalize this correlated noise approach for the general case when we inject correlated noise into an arbitrary number of functions.

4.2 Formalization of CGM in the General Form

Let $f_1, f_2, \ldots, f_l$ be a sequence of functions, each of which maps the input data x to a vector in $\mathbb{R}^d$. CGM takes as input $f_1, f_2, \ldots, f_l$ and constants σ_i, α_{ij} $(1 \leq j < i \leq l)$, and injects into each f_i a Gaussian noise γ_i as follows:

$$\gamma_i = \eta_i + \sum_{j=1}^{i-1} \alpha_{ij} \cdot \gamma_j, \tag{10}$$

where η_i is sampled from a Gaussian distribution $\mathcal{N}\left(\mathbf{0}, \sigma_i^2 \cdot \mathbf{I}\right)$. In other words, the noise γ_i in f_i is a linear combination of a "fresh" Gaussian variable η_i and the "old" noise γ_j injected into each f_j $(j \leq i-1)$. Note that in the special case when $\alpha_{ij} = 0$ for all $1 \leq j < i \leq l$, CGM degenerates to the vanilla Gaussian mechanism that injects independent Gaussian noise into each f_i.

To formulate the privacy guarantee of CGM, we first define a set of surrogate functions $\Xi = \{\xi_1, \xi_2, \ldots, \xi_l\}$, such that for any data x and any $1 \leq i \leq l$,

$$\xi_i(x) = \frac{1}{\sigma_i} \left(f_i(x) - \sum_{j=1}^{i-1} \left(\alpha_{ij} \cdot f_j(x)\right) \right). \tag{11}$$

Then, by Lemmas 2.4 and 2.2, we have the following result.

Lemma 4.1. *The correlated Gaussian mechanism that injects γ_i into f_i $(1 \leq i \leq l)$ ensures (ϵ, δ)-DP, if and only if*

$$S(\Xi) \leq \sqrt{2} \left(\sqrt{\chi^2 + \epsilon} - \chi \right), \tag{12}$$

where χ is the solution to

$$\operatorname{erfc}(\chi) - \exp(\epsilon) \cdot \operatorname{erfc}\left(\sqrt{\chi^2 + \epsilon} \right) = 2\delta,$$

and $\operatorname{erfc}()$ *denotes the complementary error function.*

Proof. Suppose that we inject i.i.d. Gaussian noise $\mathcal{N}(\mathbf{0}, \mathbf{I})$ into $\Xi = \{\xi_i\}$ (by Eq. (11)) for differential privacy. Then, by Lemma 2.4, we can ensure (ϵ, δ)-DP in Ξ if Eq. (12) holds. Let ξ_i^* denote the noisy version of ξ_i thus obtained. We show that we can obtain all f_i^* $(1 \leq i \leq k)$ from Ξ^* without incurring any more privacy loss. Hence we can prove that CGM is (ϵ, δ)-DP.

We obtain f_1^* from ξ_1^* directly. For every $i \geq 2$, we obtain f_i^* from ξ_i^* and $\sum_{j=1}^{i-1} \left(\alpha_{ij} \cdot f_j^*\right)$, namely:

$$f_i^* = \sigma_i \xi_i^* + \sum_{j=1}^{i-1} \left(\alpha_{ij} \cdot f_j^*\right). \tag{13}$$

Since DP is preserved under post-processing (Lemma 2.2), reusing the noisy $\sum_{j=1}^{i-1} \left(\alpha_{ij} \cdot f_j^*\right)$ does not incur any privacy loss. By induction, all f_i^* can be obtained without incurring any more privacy loss than releasing Ξ^*. Hence, CGM is (ϵ, δ)-DP and the reconstruction procedure is described as in (13). □

Intuitively, by Lemma 4.1, when the L_2 sensitivity of each $f_i - \sum_{j=1}^{i-1} \left(\alpha_{ij} \cdot f_j\right)$ is small, CGM may achieve a significantly reduced noise scale for perturbing each f_i, while satisfying (ϵ, δ)-LDP. This is done by setting σ_i to a small value for each Gaussian variable η_i in Eq. (10), so that the total noise γ_i injected into f_i is much smaller than that required when we apply the analytic Gaussian mechanism directly. Based on this idea, next we present a concrete algorithm that applies CGM to our problem setting, *i.e.*, streaming data collection under LDP.

4.3 Applying CGM to Streaming Data Collection

To instantiate CGM for the problem of releasing a data stream under LDP, a simple way would be to apply Lemma 4.1 and generate the noise to be injected to a particular data item (say, $x_{k,i}$) such that this noise is correlated with all the noise vectors injected into previous data items $x_{k,j}$ $(j < i)$ in the same stream. However, doing so incurs a high computational cost that grows with increasing number of timestamps i. Instead, we design an effective and low-cost instantiation of CGM with constant (and negligible) computational overhead, as follows.

Without loss of generality, we focus on a particular user k's data stream $x_k = (x_{k,1} \ldots, x_{k,l})$. Recall from the setting in Section 3 that for every item $x_{k,i}$ and $x_{k,i-1}$ we have that $\|x_{k,i} - x_{k,i-1}\| \leq C$. Meanwhile, in our problem setting, the function F to be released is the $d \times d$-dimensional identity function $\mathbf{I}$. Hence, the exact $x_{k,i}$ is directly correlated with $x_{k,i-1}$. To compute the noisy version $x_{k,i}^*$ of $x_{k,i}$, CGM generates a noise vector that is directly correlated with the noise contained in the previous noisy update $x_{k,i-1}^*$. To do so, the high level idea is to regard $x_{k,i}$ as a linear combination of $x_{k,i}$ (*i.e.*, itself) and $x_{k,i-1}$, and inject noise γ_i into $x_{k,i}$ as in Eq. (9), with parameters α_{ij} $(1 \leq j < i-1)$ fixed to 0. Hence, the noise injected to $x_{k,i}$ equals the sum of a fresh Gaussian noise η_i and a portion of the noise injected to $x_{k,i-1}$. This drastically simplifies CGM, since the generation of γ_i only relies on γ_{i-1}, not $\gamma_1, \gamma_2, \ldots, \gamma_{i-2}$. Consequently, we only need to keep track of the noise generated for the previous estimate $x_{k,i-1}^*$ when $x_{k,i}$ comes, rather than the whole history of noisy data updates. This design minimizes the computational cost of CGM. Further, this design also comes with no loss of utility compared to the general form in Eq. (10) that reuses the noise at all previous timestamps. A detailed proof can be found in Appendix A.1 in the technical report version [4].

Algorithm 3: CGM for streaming data

Input: data items $\{x_{k,i}\}_{i=1}^{l}$, where $\|x_{k,i}\| \leq \frac{1}{2}$ and $\|x_{k,i} - x_{k,i-1}\| \leq C$, privacy parameters ϵ, δ.

Output: private estimates $\{x^*_{k,i}\}_{i=1}^{l}$.

1 **for** $i = 1$ ***to*** l **do**
2 **if** $i = 1$ **then**
3 $x^*_{k,i} \longleftarrow x_{k,i} + \mathcal{N}(\mathbf{0}, \sigma_1^2 \cdot \mathbf{I})$, where σ_1 is computed as in Eq. (14).
4 $v_1 \longleftarrow 1$.
5 **else**
6 $r_i \longleftarrow \frac{1-2C}{(1-2C)^2+v_{i-1}}$.
7 $\sigma_i \longleftarrow ((1 - r_i) + r_i \cdot 2C) \cdot \sigma_1$.
8 $x^*_{k,i} \longleftarrow x_{k,i} + \mathcal{N}(\mathbf{0}, \sigma_i^2 \cdot \mathbf{I}) + r_i \cdot \gamma_{k,i-1}$, where $\gamma_{k,i-1}$ is the noise injected in $x^*_{k,i-1}$.
9 $v_i \longleftarrow \frac{v_{i-1}}{(1-2C)^2+v_{i-1}}$.
10 Output $x^*_{k,i}$

Algorithm 3 outlines our proposed method for collecting a data stream with (ϵ, δ)-LDP. For simplicity, Algorithm 3 focuses on the data stream possessed by one individual k. At the i-th time step, the algorithm the exact data item $x_{k,i}$, and outputs a noisy version $x^*_{k,i}$ by injecting correlated noise (Lines 3 and 8).

When $i = 1$, the algorithm injects a freshly generated noise into $x_{k,i}$ (Line 3), similarly as in Algorithm 1. Specifically, the noise is sampled from $\mathcal{N}(\mathbf{0}, \sigma_1^2 \cdot \mathbf{I})$, where σ_1 satisfies:

$$\frac{\sqrt{l}}{\sigma_1} \leq \sqrt{2}\left(\sqrt{\chi^2 + \epsilon} - \chi\right), \tag{14}$$

and χ is defined in Eq. (3).

When $i \geq 2$, our algorithm samples random noise for $x_{k,i}$ that is correlated with the noise contained in $x^*_{k,i-1}$ (Line 8). In particular, the noise injected to $x_{k,i}$ is the sum of a fresh noise from $\mathcal{N}(\mathbf{0}, \sigma_i^2 \cdot \mathbf{I})$ and r_i times the noise injected to $x_{k,i-1}$, denoted as $r_i \cdot \gamma_{i-1}$. To compute σ_i and r_i, we introduce another parameter v_i (Lines 4 and 9): v_1 is initialized to be 1 at timestamp 1 (Line 4) and v_i is updated at every timestamp (Line 9). v_i denotes the ratio between the variance of the overall noise injected to $x_{k,i}$ and that injected to $x_{k,1}$. Finally, the algorithm releases $x^*_{k,i}$ (Line 10).

It remains to clarify the computation of r_i (Line 6) and σ_i (Line 7), whose values are carefully chosen to minimize the overall error of CGM. The formulae for r_i and σ_i are the results of a careful theoretical analysis on the privacy-utility tradeoff of Algorithm 3, presented in the next section. In particular, the value of σ_i is given in Lemma 5.3, and the choice of r_i is clarified towards the end of Section 5.2.

Space and time complexity. Algorithm 3 incurs $O(1)$ extra space and $O(l)$ time for the entire stream, which is clearly optimal, as reporting the data updates itself requires the same costs. In particular, at each timestamp i, the algorithm only needs to store and update three additional variables r_i, v_i and $\gamma_{k,i}$ (Lines 4 and 9, 6 and 8). The updating of these variables takes constant time. Overall, the computational overhead of Algorithm 3 is negligible; hence, it is suitable for a mobile platform such as iOS and Android.

CGM for data streams with unknown length. In the case when l, the length of the input data stream, is not known in advance or is unbounded, one can choose to fix the value of σ_1 first and account for the incurred privacy cost at every timestamp using Lemma 5.3. The algorithm is terminated when the privacy cost exceeds the predefined privacy budget, or when the input data stream terminates.

5 THEORETICAL ANALYSIS

In this section, we analyze the privacy-utility trade-off of Algorithm 3.

5.1 Main Result

We first present our main theorem, which states the privacy-utility trade-off of Algorithm 3 in terms of the expected squared error for each x_i.

Theorem 5.1. *Algorithm 3 produces estimates $\{x^*_{k,i}\}_{i=1}^{l}$ under (ϵ, δ)-LDP, with error guarantee for each data item:*

$$\mathbb{E}\left[\|x^*_{k,i} - x_{k,i}\|^2\right] = \frac{4C - 4C^2}{1 - (1-2C)^{2i}}\sigma^2 d, \tag{15}$$

where d is the dimensionality for each element $x_{k,i}$, l is the length of the data stream, σ is defined in Eq. (7). The expectation is taken over the randomness of the Algorithm.

The proof of the above theorem is rather complicated, and we defer it in the next subsection. Note that the σ in Theorem 5.1 is the same as the σ in Algorithm 1. Meanwhile, the term $\frac{4C-4C^2}{1-(1-2C)^{2i}}$ quickly converges to $(4C - 4C^2)$ as the number of timestamps i increases. When $C < \frac{1}{2}$, this factor is strictly smaller than 1, giving CGM an advantage over the baseline approach in terms of utility. In addition, when $C \ll 1$, C^2 becomes negligible, and the error of CGM is approximately $4C$ times that of Algorithm 1.

Based on Theorem 5.1, the following corollary formally states the overall error of CGM with respect to the whole data stream.

Corollary 5.2. *Algorithm 3 produces estimates $\{x^*_{k,i}\}_{i=1}^{l}$ under (ϵ, δ)-LDP, with expected error:*

$$\frac{1}{l}\left(\sum_{i=1}^{l}\mathbb{E}\left[\|x^*_{k,i} - x_{k,i}\|^2\right]\right) = (4C - 4C^2)\sigma^2 d(1 + o(1)), \tag{16}$$

where σ is defined in Eq. (7) as in Theorem 5.1, and the expectation is taken over the randomness of the Algorithm.

Proof. It suffices to compute $\sum_{i=1}^{l}\frac{1}{1-(1-2C)^{2i}}$. We denote $(1 - 2C)^2$ as q. Then, for some i_0, we have:

$$\begin{aligned}\sum_{i=1}^{l}\frac{1}{1-(1-2C)^{2i}} &= \sum_{i=1}^{i_0}\frac{1}{1-q^i} + \sum_{i=i_0}^{l}\frac{1}{1-q^i} \\ &\leq \sum_{i=1}^{i_0}\frac{1}{1-q^i} + \sum_{i=i_0}^{l}(1 + \frac{1}{i}) \\ &= l + O(\log l)\end{aligned}$$

□

5.2 Proof of Theorem 5.1

In what follows, we prove Theorem 5.1. Without loss of generality, we let $d = 1$ and one may substitute any value of d into Theorem 5.1.

Proof of the privacy guarantee. To set up the stage for privacy analysis, observe that $x_{k,i}$ ($i \geq 2$) can be expressed as the sum of two terms, as follows.

$$\begin{aligned} x_{k,i} &= x_{k,i} - r_i \cdot x_{k,i-1} + r_i \cdot x_{k,i-1} \\ &= (1 - r_i) \cdot x_{k,i} + (r_i \cdot x_{k,i} - r_i \cdot x_{k,i-1}) + r_i \cdot x_{k,i-1} \\ &= \underbrace{(1 - r_i) \cdot x_{k,i} + r_i \cdot \left(x_{k,i} - x_{k,i-1}\right)}_{\text{first term}} \\ &+ \underbrace{r_i \cdot x_{k,i-1}}_{\text{second term}} . \end{aligned} \tag{17}$$

The way that Algorithm 3 injects noise into $x_{k,i}$ (Line 6 in Algorithm 3) is equivalent to applying a perturbation on each term in Eq. (17) separately. In particular, the first term is perturbed with a fresh Gaussian noise $\mathcal{N}(0, \sigma_i^2)$, while the second term is replaced by $r_i \cdot x^*_{k,i-1}$, where $x^*_{k,i-1} = x_{k,i-1} + \gamma_{i-1}$ is the noisy estimate for $x_{k,i-1}$. By the post-processing property of DP (Lemma 2.2), using $r_i \cdot x^*_{k,i-1}$ in the i-th time-step does not incur any privacy cost. Therefore, to analyze the privacy cost of releasing $x^*_{k,i}$, we only need to quantify the privacy cost of injecting $\mathcal{N}(0, \sigma_i^2)$ into the first term:

$$r_i \cdot \left(x_{k,i} - x_{k,i-1}\right) + (1 - r_i) \cdot x_{k,i} \tag{18}$$

Accordingly, we let V_i be the sensitivity of the first term, *i.e.*,

$$V_i = \begin{cases} 1, & \text{if } i = 1 \quad (19a) \\ 1 - r_i + r_i \cdot 2C, & \text{otherwise} \quad (19b) \end{cases}$$

where C is the differential bound, *i.e.*, $\|x_{k,i} - x_{k,i-1}\| \leq C$. When $i \geq 2$, the sensitivity of the first term follows from the triangle inequality: $x_{k,i}$ is of sensitivity 1 and the differential $(x_{k,i} - x_{k,i-1})$ is of sensitivity $2C$. Clearly, the whole vector of concatenating all

$$\left\{ \frac{1}{V_i} \cdot \left(r_i \cdot \left(x_{k,i} - x_{k,i-1}\right) + (1 - r_i) \cdot x_{k,i}\right) \right\}_{i=1}^{l}$$

is of sensitivity $\sqrt{l}$. Combining Lemma 2.4 and Lemma 2.2, we reach the following lemma.

Lemma 5.3. *The generation of $\{x^*_{k,i}\}_{i=1}^{l}$ in Algorithm 3 satisfies (ϵ, δ)-LDP, for σ_i such that,*

$$\frac{V_i \sqrt{l}}{\sigma_i} \leq \sqrt{2} \left(\sqrt{\chi^2 + \epsilon} - \chi \right), \tag{20}$$

and χ is defined in Eq. (3).

At the i-th timestamp, the noisy estimate for $x_{k,i}$ can be obtained by adding the noisy estimate for the first term in Eq. (18), i.e.,

$$r_i \cdot \left(x_{k,i} - x_{k,i-1}\right) + (1 - r_i) \cdot x_{k,i},$$

which is perturbed in this timestamp, and the noisy estimate for the second term, i.e., $r_i \cdot x_{k,i-1}$, which is retrieved from the previous timestamp with no additional privacy cost.

Choice of parameters r_i and V_i. Next, we clarify the values of r_i and V_i in Eq. (19b). Recall that the variance in the fresh noise injected to $x^*_{k,i}$ is σ_i^2, where σ_i is computed according to Eq. (20). We denote $\text{var}[x^*_{k,i}]$ as the variance in $x^*_{k,i}$. Note that $\text{var}[x^*_{k,i}]$ is the sum of the variance from the fresh noise γ_i and that of the reused noise in $x^*_{k,i-1}$. By Eq. (20), we have the following expression for the variance of $x^*_{k,i}$:

$$\begin{aligned} \text{var}\left[x^*_{k,i}\right] &= \sigma_i^2 + r_i^2 \cdot \text{var}\left[x^*_{k,i-1}\right] \\ &= (r_i \cdot 2C + 1 - r_i)^2 \cdot \sigma_1^2 + r_i^2 \cdot \text{var}\left[x^*_{k,i-1}\right] \end{aligned}$$

Hence, to minimize $\text{var}[x^*_{k,i}]$, we just need to find the optimal r_i that minimizes the above quadratic function of r_i when $0 < C < \frac{1}{2}$. In particular, we know that $\text{var}[x^*_{k,1}] = \sigma_1^2$, where σ_1^2 is computed from Eq. (20) with $V_1 = 1$. From this we can compute r_2 and V_2, which is then plugged into Eq. (20) for $i = 2$, which, in turn, is used to determine r_3, and so on. A little algebra gives us the general expressions for the optimal r_i and the corresponding $\text{var}\left[x^*_{k,i}\right]$:

$$r_i = \frac{1 - 2C}{(1 - 2C)^2 + \frac{4C - 4C^2}{1 - (1 - 2C)^{2i-2}}}, \tag{21}$$

and

$$\text{var}\left[x^*_{k,i}\right] = \frac{4C - 4C^2}{1 - (1 - 2C)^{2i}} \cdot \sigma_1^2. \tag{22}$$

Note that the closed form expression of r_i is in line with the (recursive) computation expression r_i in Line 6 of Algorithm 3 and we omit the details here. Next, based on r_i, we can compute V_i, which is used to determine σ_i according to Eq. (20). In addition, the distribution of $x^*_{k,i}$ is a Gaussian distribution, expressed as

$$\mathcal{N}(x_{k,i}, \frac{4C - 4C^2}{1 - (1 - 2C)^{2i}} \cdot \sigma_1^2),$$

since the summation of two independent Gaussian noises is still a Gaussian noise. The expected squared error for $x_{k,i}$ is $\frac{4C - 4C^2}{1 - (1 - 2C)^{2i}} \cdot \sigma_1^2$, which is in line with Eq. (22). Finally, we plug the values of V_i and σ_i into Lemma 5.3, and Theorem 5.1 follows.

Remark. CGM can also be applied to other differential privacy definitions such as Rényi differential privacy (RDP) [33] and zCDP [10], without changes in the algorithm. Here we provide the high level idea; a formal proof sketch can be found in Appendix A.5 of the technical report version [4]. According to [33], under Rényi differential privacy, injecting Gaussian noise to the original data, with a noise scale linear to the sensitivity of the function of interest, satisfies RDP. Hence, the above privacy analysis of CGM (based on the analytic Gaussian mechanism) can be adapted to RDP as well. Further, under RDP, with the reuse ratio of r_i, the scale of the fresh noise injected at each timestamp is still calibrated to $(x_{k,i} - r_i \cdot x_{k,i-1})$. Thus, the optimal reuse ratio for CGM under RDP is exactly calculated as in Eq. (21). Finally, the improvement of CGM over the baseline solution is independent of the privacy framework, as long as the linear relationship between the scale of the noise and the sensitivity of the function of interest holds.

6 EXPERIMENTS

We compare the utility performance of CGM (Algorithm 3) against the baseline solution Algorithm 1 using the following two real datasets.

- Kaggle Web Traffic Time Series [28]: The training dataset of this Kaggle competition consists of approximately 145k time series. Each of these time series contains the daily views of a Wikipedia article, during the time span from July 1st, 2015 to December 31st, 2016 (550 days).
- Beijing taxi GPS trajectories [31]: This classic time series dataset is collected from approximately 5k taxi drivers in Beijing. Each of these time series contains the GPS pings of a taxi driver in Beijing in May 2009. Each GPS ping includes the taxi ID, timestamp, latitude and longitude, as well as other information such as occupancy indicator, vehicle orientation, and speed.

In the following, Sections 6.1 and 6.2 present the experimental designs as well as evaluation results on the Kaggle Web Traffic and Beijing Taxi datasets, respectively. Section 6.3 investigates the effect of various values of the differential bound C on the performance of CGM.

6.1 Evaluations on Kaggle Web Traffic Dataset

Experiment design. In this set of experiments, we consider each Wikipedia page k as an individual, and its daily page view counts $x_{k,1}, \ldots, x_{k,l}$ as the private streaming data, where $l = 550$ is the number of timestamps. This setting corresponds to several real-world applications of LDP-compliant data collection. For instance, a data aggregator, such as a web traffic analytics service, may ask a large group of web site owners for their daily visit statistics, which might be considered as private information to the web site owner. Meanwhile, another related application (mentioned in Section 1) concerns the collection of web browser users' number of visits to a particular website (or a category of such websites) each day. In a broader sense, any app that collects private daily statistics (*e.g.*, app usage, energy consumption, number of crashes, *etc.*) is related to this setting.

The original data from Kaggle are highly skewed, in the sense that a small number of Wikipedia pages receive huge numbers of visits per day, where as the number of daily visits of other pages are far lower. Consequently, those outlier pages would cause extremely high sensitivity, leading to very noisy daily visit count estimates for other pages under LDP. To avoid this issue, in a preprocessing step, we filter out all pages with over 20000 views in any of the 550 days (85k pages remain after the filtering). Accordingly, the sensitivity of each item of each data stream is 20000.

Recall from Section 1 that the proposed method CGM is based on the observation that a publicly known constant C exists that bounds the value differentials between consecutive timestamps in a data stream. In the context of the Kaggle Web Traffic dataset, C bounds the maximum difference between number of views in two adjacent days. To ensure this property, when using CGM, each user k performs the following preprocessing step to clip her streaming data starting from the second timestamp, *i.e.*, $x_{k,2}, \ldots, x_{k,550}$, as follows.

$$x'_{k,i} = \begin{cases} x_{k,i-1} + C, & \text{if } x_{k,i} - x_{k,i-1} > C, \quad (23a) \\ x_{k,i-1} - C, & \text{if } x_{k,i} - x_{k,i-1} < -C, \quad (23b) \\ x_{k,i} & \text{otherwise.} \quad (23c) \end{cases}$$

After that, the user procceds to inject random noise to the clipped values using CGM. Note that the original, unclipped data value $x_{k,i}$ is considered the ground truth, and the clipping based on C is considered a part of the algorithm for enforcing LDP. In other words, the bias introduced in the clipping step is included in the overall error of CGM. We emphasize that this clipping step only applies to CGM, *not* to the naive solution (Algorithm 1) that directly releases each $x_{k,i}$ using the analytic Gaussian mechanism.

Regarding privacy parameters, in this set of experiments we follow the periodically refreshing scheme described in Section 3, which is commonly used in practice. In particular, the user allocates a privacy budget ϵ for each day, alongside a failure parameter δ. In our experiments, δ is fixed to a small value 10^{-5}; ϵ varies and can be either 0.25, 0.5, 1.0, or 2.0 per day. Note that when accumulated over a long period, *e.g.*, 550 days in this dataset, the equivalent overall privacy budget under the Gaussian mechanism increases proportionally the square root of number of timestamps, *i.e.*, $\sqrt{550} \approx 23.45$. Whether such a high accumulated privacy budget leads to large privacy leakage is out of the scope of this work, and we refer the reader to the discussion in a recent paper [7].

We compare the utility performance of CGM (Algorithm 3) and the navie solution (Algorithm 1) on the processed dataset in terms of the average mean squared error under the same level of privacy guarantee at each time stamp. The result is averaged over 1000 independent experiments. The mean squared error (MSE) for the i-th timestamp (day) is defined as:

$$\frac{1}{n} \sum_{k=1}^{n} |x^*_{k,i} - x_{k,i}|^2, \quad (24)$$

where $1 = 1, 2, \ldots, 550$.

Evaluation results. Figure 1 plots the utility of CGM (Algorithm 3) and that of the baseline solution (Algorithm 1) as a function of time, with varying values of the daily privacy budget ϵ. For CGM, we fix the differential bound to $C = 500$; considering that each page can be viewed up to 20,000 times in our setting, this means that the number of page visit can move in either direction for up to 2.5% per day. The choice of C depends on public knowledge in the application domain, and we evaluate the impact of various values of C on the utility of CGM in Section 6.3.

As demonstrated in Figure 1, CGM consistently outperforms the baseline solution by a wide margin, for all settings of ϵ. The performance gap between CGM and baseline expands as the amount of daily privacy budget ϵ decreases, *i.e.*, the stronger the privacy requirement, the more pronounced the advantage of CGM becomes. This is because CGM introduces two types of error: bias due to clipping with the differential bound C, and variance due to the injected noise, whereas the baseline solution outputs unbiased value updates, albeit with a far larger noise variance. With a small ϵ, the noise variance dominates the overall error, leading to a wider performance gap between CGM and baseline. Note that even for a

large privacy budget $\epsilon = 2$, which is the value used in some functionalities of iOS [2], CGM still significantly outperforms baseline. This demonstrates that CGM has clear benefit in terms of result utility in practical settings.

In particular, on day 1, the utility of CGM is identical to that of the baseline method, since the noise injected to the first data update is the same in both methods. Starting from day 2, the error of CGM drops rapidly, until reaching a relatively stable level after a few days. This is because in CGM, the variance of the reported noisy value at timestamp i drops with increasing i, according to our theoretical analysis result in Eq. (22).

6.2 Evaluations on Beijing Taxi Dataset

Experiment design. In the second set of experiments, we consider the situation where the data aggregator collects results of a continuous range aggregate query with LDP, which asks each taxi driver to report the amount of time she has stayed within a given spatial range (*e.g.*, a COVID-19 danger zone) since the last update; each taxi driver submits an update every 30 minutes. Each reported value is normalized to the range $[0, 1]$, leading to a sensitivity of 1 for each data update. In our experiments, we used the taxi trajectories in one day (we could not simulate a longer timespan since the data contains trajectories for different taxis on different days). Accordingly, there are $l = 48$ timestamps in total, corresponding to 24 hours with one update every 30 minutes. We report the average result for 30 different days in the dataset; for each day, every experiment is repeated 100 times, and the average results are reported.

Specifically, we normalize the latitude and longitude of all taxis at all times to a unit square, *i.e.*, $[0, 1] \times [0, 1]$. The query range is a rectangle of size $[0.45, 0.55] \times [0.45, 0.55]$, located at the center of the spatial range. We have also tested other query ranges, which led to similar results and the same conclusions on the relative performance of CGM and baseline; hence, we only report the results for one query range for brevity.

In the dataset, the GPS sampling rate varies for different taxis, and the taxi's location is unknown between two adjacent pings. In order to estimate the duration of time each taxi stays with the query range during every reporting period, we linearly interpolate the GPS coordinates between every two consecutive pings. Finally, for CGM, we enforce a differential bound C similarly as in Section 6.1, and compare the performance of CGM and the baseline method in terms of the mean squared error, defined in Eq. (24).

Regarding the privacy parameters, in this set of experiments, since all updates happen within one day, we assume that each user assigns an overall privacy budget covering all updates (unlike the previous set of experiments where the privacy budget is allocated to each timestamp). Note that this privacy setting also agrees with the current common practice of allocating a daily privacy budget. Specifically, we vary ϵ in $\{0.25, 0.5, 1, 2\}$ and fix δ to 10^{-5}. Finally, for CGM, we fix the differential bound to $C = 0.05$.

Evaluation results. Figure 2 compares the utility performance of CGM and the baseline approach under various privacy parameter settings. Once again, CGM consistently and signficiantly outperforms baseline, in all settings except for the first update of a stream, since CGM and baseline perform the same operations for the first update. Similar to the case of Kaggle Web Traffic, the performance gap between CGM and baseline is larger for smaller values of ϵ, which correspond to stronger privacy protection. The error of CGM drops from the same level as baseline to its stable level within 20 updates.

Finally, we mention that for the setting with $\epsilon = 2$ (the privacy budget value used in iOS [2]), the stable MSE of CGM is around 0.006, corresponding to an RMSE of $\sqrt{0.006} = 0.077$. Considering that the original data value has a range of $[0, 1]$, this level of error might be acceptable in practice. In contrast, the MSE of the baseline approach reaches as high as 0.028, corresponding to an RMSE of 0.167, which might be too high to be acceptable.

6.3 Effect of Differential Bound C

Next, we investigate the impact of the value of the differential bound C on the utility performance of CGM. Note that unlike in the non-private setting, tuning hyperparameter C with data is a challenging problem under LDP, since (i) the tuning process itself needs to be done with an LDP-compliant mechanism with a portion of the privacy budget, and (ii) the sensitivity of C, the impact of C on result utility, and the appropriate privacy parameters for the tuning process are all difficult math problems. This is an orthogonal topic, and we refer the reader to several recent works [26, 32] for generic hyperparameter tuning solutions. The purpose of this set of experiments is to provide a guideline to LDP practitioners for how to choose an appropriate C given the public domain knowledge of the application, without tuning C with data.

Figures 3 shows the utility performance of CGM with varying $C \in \{200, 500, 1000\}$ on the Kaggle Web Traffic dataset. Observe that with a small privacy budget, *e.g.*, $\epsilon = 0.25$, a smaller C leads to lower error. On the other hand, with a relatively large privacy budget $\epsilon = 2$, overly small values of C lead to large error fluctuations, whereas CGM with a larger C demonstrate more stable performance over time. To understand this behavior, note that the error of CGM includes two components: the bias introduced by clipping, and the variance due to the injected random noise necessary to satisfy LDP. The former occurs when the value of C is too low, and, consequently, the differential bound assumption over the aggregate stream, described in Section 3.1, no longer holds. The latter is given by Eq. (15). When ϵ is low, the noise variance dominates overall error; hence, a smaller C leads to reduced noise scale, and, thus, lower overall error. However, a smaller C, while reducing noise scale, also brings in higher clipping bias. As ϵ grows, the noise scale drops, and the effect of the clipping bias becomes more pronounced, which starts to favor larger values of C.

Figure 4 repeats the same experiments on the Beijing Taxi dataset, with varying $C \in \{0.01, 0.05, 0.1\}$. Unlike the case of Kaggle Web Traffic, the lowest value $C = 0.01$ is consistently the best choice according to the evaluation results. This is probably due to the fact that the level of value fluctuation in the Beijing Taxi dataset is lower than that in Kaggle Web Traffic, leading to generally lower clipping bias in the former case. In addition, in Appendix A.2 of the technical report version [4], we provide experimental results comparing the empirical and theoretical performance (given by Eq. (15)) of CGM.

Summing up the evaluation results with varying C on both datasets, an appropriate choice of C depends on both the privacy parameters and properties of the underlying data. In particular, for

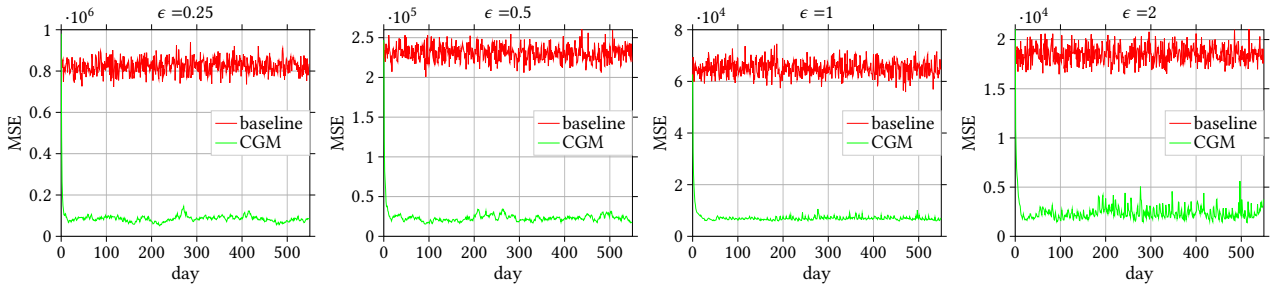


Figure 1: Utility performances of CGM (Algorithm 3) and the baseline (Algorithm 1) on the Kaggle Web Traffic dataset, with varying daily privacy budget $\epsilon \in \{0.25, 0.5, 1, 2\}$ and $\delta = 10^{-5}$. For CGM, the differential bound is fixed to $C = 500$.

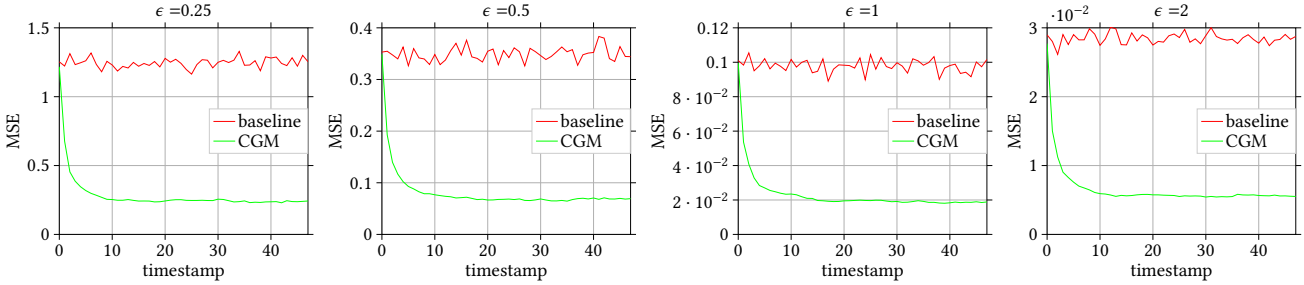


Figure 2: Utility performances of CGM (Algorithm 3) and the baseline (Algorithm 1) on the Beijing Taxi dataset, with varying total privacy budget for all updates $\epsilon \in \{0.25, 0.5, 1, 2\}$ and $\delta = 10^{-5}$. For CGM, the differential bound is fixed to $C = 0.05$. The whole space is normalized to $[0, 1] \times [0, 1]$, and the query region is $[0.45, 0.55] \times [0.45, 0.55]$.

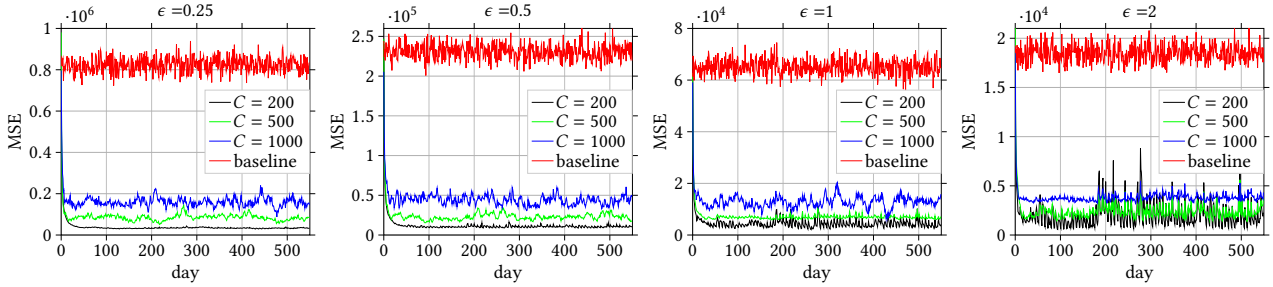


Figure 3: Impact of varying differential bound $C \in \{200, 500, 1000\}$ on the utility performance of CGM on the Kaggle Web Traffic dataset, where $\epsilon \in \{0.25, 0.5, 1, 2\}$ and $\delta = 10^{-5}$.

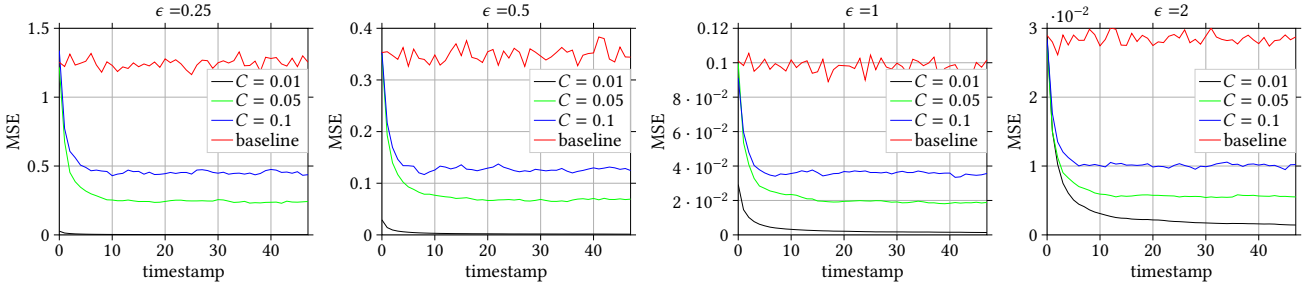


Figure 4: Impact of varying differential bound $C \in \{0.01, 0.05, 0.1\}$ on the utility performance of CGM on the Beijing Taxi dataset, where $\epsilon \in \{0.25, 0.5, 1, 2\}$ and $\delta = 10^{-5}$. The query region is $[0.45, 0.55] \times [0.45, 0.55]$.

relatively smooth streaming time series and/or a relatively low privacy budget, we can aggressively set a small value for C; conversely, a larger C might be preferred when the data exhibit a high level of fluctuations, and/or when ϵ is larger. Finally, we mention that for all choices of C, CGM consistently outperforms the baseline method, whose error is far higher.

7 RELATED WORK

The observation that data items in neighboring timestamps are correlated is also exploited in private temporal data analysis [11, 39], although their formalization of correlations is different from ours. Specifically, [11, 39] consider a temporal data streams from a discrete space, whereas we consider temporal data streams from a continuous space. Following the discrete setting, they model the correlation between an the locations of an item at neighboring time stamps with a transition matrix. For example, given the publicly known transition matrix M, an item currently at location x has probability $M_{x,y}$ of being at location y in the next timestamp. Their model of correlation can be applied to analyzing road networks and users' behavioral patterns, which are radically different from our target applications where each user continuously updates a numeric value. Hence, Ref. [11, 39] are orthogonal to this work.

Private streaming data analysis was first proposed by Dwork et al. [18]. Our setting differs from [18] in the following ways, although we both consider a data stream as the sensitive information. First, [18] considers element-level differential privacy. To be more specific, their mechanism outputs similar distributions for any two neighboring data streams, where two data streams are called neighbors if they differ by one element (up to many times of occurrences). Instead, we consider the local differential privacy setting, and our mechanism outputs similar distributions for any two data streams, as long as they are from the same data domain. Second and more importantly, [18] does not exploit the correlation between two adjacent data items in a stream, while this observation is the key for the proposed method CGM to achieve high utility in this work.

PeGaSus [13, 25], which considers the event-level privacy, is not directly comparable to this work, which considers the user-level privacy. Despite this major difference, both CGM and Pegasus aim to reduce the overall variance in the released statistics. While Pegasus groups similar data items and applies a smoothing post-processing step, CGM enforces such a condition so that it can inject correlated noises to adjacent data items. A combination of these techniques is a promising direction for future work.

FAST [22, 23] addresses a similar problem as ours, and reduces the amount of injected noise to satisfy differential privacy by sampling the data stream along the time dimension. In particular, FAST only perturbs and returns data items of the sampled timestamps; data at other timestamps are discarded and instead derived from the sample ones. This technique is orthogonal to CGM, and can be used in combination of CGM, *i.e.*, by perturbing the data items at sampled timestamps with CGM. In Appendix A.3 of the technical report version [4], we experimentally demonstrate that FAST+CGM obtains higher result utility than using FAST alone.

Estimating static sample statistics under the local differential privacy framework is a hot topic in the literature (*e.g.*, [5, 9, 12, 14, 29, 35, 37, 40, 41]). In this paper, we study the problem of estimating changing statistics over time; to be more specific, the sensitive data of every individual is a continuous stream of data items. This setting is very different from the existing works mentioned above; as a result, our proposed method CGM is not directly comprable with these existing solutions.

Analyzing evolving data with privacy guarantee is a relatively new topic in private data analysis, *e.g.*, [15, 27]. In particular, the solution in [15] does not provide a formal differential privacy guarantee. [27] assumes that the input data of participants are sampled from multiple changing Bernoulli distributions, and the goal is to estimate the average of mean of the changing Bernoulli distributions once every few time steps. Our work is not comparable with [27], since instead of considering an underlying distribution for the input sensitive data, we assume that the data stream is from a bounded data domain, which is the canonical assumption in private data analysis. In short, the input data changes over time in our setting, whereas the underlying distribution changes over time in [27].

In our problem setting, we regard the analytic Gaussian mechanism [3], an improvement over the original Gaussian mechanism [19], as a standard solution for achieving (ϵ, δ)-local differential privacy. Since the Gaussian noise is used as a building block in many (ϵ, δ)-differentially private applications (*e.g.*, [1, 6, 8, 24, 34, 36, 38]), we consider its most recent version [3] as the baseline solution to our problem.

8 CONCLUSION

In this work, we study the problem of streaming data collection under local differential privacy, which each individual possesses a stream of data items, and aim to release her the data stream under LDP. The naive approach requires each user to perturb the data item independently at each timestamp, which leads to an excessively large amount of noise. Addressing this issue, we exploit data autocorrelations common in many real data streams, and propose a novel mechanism CGM. CGM achieves rigorous accuracy and privacy guarantees with negligible computational overhead. Through both theoretical analysis and extensive experimental evaluations using real data from multiple application domains, we demonstrate that CGM consistently and significantly outperforms the baseline solution in terms of result utility.

Regarding future work, we plan to adapt CGM to more complex analysis types, *e.g.*, collecting heavy hitters and histograms. Another promising direction is to extend the idea of CGM to the ϵ-LDP setting.

ACKNOWLEDGMENTS

This work was supported by the the Ministry of Education Singapore (Number MOE2018-T2-2-091), Qatar National Research Fund Qatar Foundation (Number NPRP10-0208-170408), and by the National Research Foundation, Singapore under its Strategic Capability Research Centres Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the funding agencies.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *CCS*. 308–318.

[2] Apple. 2016. *Differential Privacy Overview.* Retrieved December 21, 2020 from https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf

[3] Borja Balle and Yu-Xiang Wang. 2018. Improving the Gaussian Mechanism for Differential Privacy: Analytical Calibration and Optimal Denoising. In *ICML*. 403–412.

[4] Ergute Bao, Yin Yang, Xiaokui Xiao, and Bolin Ding. 2021. *CGM: An Enhanced Mechanism for Streaming Data Collectionwith Local Differential Privacy (Technical report).* Retrieved May 15, 2020 from https://drive.google.com/file/d/164GVGSr2xz4x_xlbOhP1ciQW7axGAQKN/view?usp=sharing

[5] Raef Bassily and Adam Smith. 2015. Local, Private, Efficient Protocols for Succinct Histograms. In *STOC*. 127–135.

[6] Raef Bassily, Adam Smith, and Abhradeep Thakurta. 2014. Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds. In *FOCS*. 464–473.

[7] Abhishek Bhowmick, John C. Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. 2018. Protection Against Reconstruction and Its Applications in Private Federated Learning. *CoRR* abs/1812.00984.

[8] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnés, and Bernhard Seefeld. 2017. Prochlo: Strong Privacy for Analytics in the Crowd. In *SOSP*. 441–459.

[9] Mark Bun, Jelani Nelson, and Uri Stemmer. 2018. Heavy Hitters and the Structure of Local Privacy. In *PODS*. 435–447.

[10] Mark Bun and Thomas Steinke. 2016. Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds. In *TCC*, Vol. 9985. 635–658.

[11] Y. Cao, M. Yoshikawa, Y. Xiao, and L. Xiong. 2017. Quantifying Differential Privacy under Temporal Correlations. In *ICDE*. 821–832.

[12] Rui Chen, Haoran Li, A. Kai Qin, Shiva Prasad Kasiviswanathan, and Hongxia Jin. 2016. Private spatial data aggregation in the local setting. In *ICDE*. 289–300.

[13] Yan Chen, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2017. PeGaSus: Data-Adaptive Differentially Private Stream Processing. In *CCS*. 1375–1388.

[14] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. 2018. Marginal Release Under Local Differential Privacy. In *SIGMOD*. 131–146.

[15] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting Telemetry Data Privately. In *NeurIPS*. 3574–3583.

[16] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. 2013. Local Privacy and Minimax Bounds: Sharp Rates for Probability Estimation. In *NeurIPS*. 1529–1537.

[17] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC*. 265–284.

[18] Cynthia Dwork, Moni Naor, Toniann Pitassi, Guy N. Rothblum, and Sergey Yekhanin. 2010. Pan-Private Streaming Algorithms. In *ICS*. 66–80.

[19] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. In *TCS*, Vol. 9. 211–407. Issue 3-4.

[20] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *CCS*. 1054–1067.

[21] Alexandre V. Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. 2002. Privacy preserving mining of association rules. In *KDD*. 217–228.

[22] Liyue Fan and Li Xiong. 2012. Real-Time Aggregate Monitoring with Differential Privacy. In *CIKM*. 2169–2173.

[23] L. Fan and L. Xiong. 2014. An Adaptive Approach to Real-Time Aggregate Monitoring With Differential Privacy. In *TKDE*, Vol. 26. 2094–2106.

[24] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. 2017. Privacy-Preserving Distributed Linear Regression on High-Dimensional Data. In *PETS*, Vol. 2017. 345–364.

[25] Sameera Ghayyur, Yan Chen, Roberto Yus, Ashwin Machanavajjhala, Michael Hay, Gerome Miklau, and Sharad Mehrotra. 2018. IoT-Detective: Analyzing IoT Data Under Differential Privacy. In *SIGMOD*. 1725–1728.

[26] Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. 2010. Differentially Private Combinatorial Optimization. In *SODA*. 1106–1125.

[27] Matthew Joseph, Aaron Roth, Jonathan Ullman, and Bo Waggoner. 2018. Local Differential Privacy for Evolving Data. In *NeurIPS*. 2375–2384.

[28] Kaggle. 2018. *Web Traffic Time Series Forecasting.* Retrieved December 21, 2020 from https://www.kaggle.com/c/web-traffic-time-series-forecasting/data

[29] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. 2014. Extremal Mechanisms for Local Differential Privacy. In *NeurIPS*. 2879–2887.

[30] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. 2008. What Can We Learn Privately?. In *FOCS*. 531–540.

[31] Jing Lian and Lin Zhang. 2018. One-Month Beijing Taxi GPS Trajectory Dataset with Taxi IDs and Vehicle Status. In *DATA*. 3–4.

[32] Jingcheng Liu and Kunal Talwar. 2019. Private Selection from Private Candidates. In *STOC*. New York, NY, USA, 298–309.

[33] Ilya Mironov. 2017. Rényi Differential Privacy. In *CSF*. 263–275.

[34] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. 2018. Scalable Private Learning with PATE. In *ICLR*.

[35] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2016. Heavy Hitter Estimation over Set-Valued Data with Local Differential Privacy. In *CCS*. 192–203.

[36] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. 2017. Locally Differentially Private Protocols for Frequency Estimation. In *USENIX Security*. 729–745.

[37] Tianhao Wang, Bolin Ding, Jingren Zhou, Cheng Hong, Zhicong Huang, Ninghui Li, and Somesh Jha. 2019. Answering Multi-Dimensional Analytical Queries under Local Differential Privacy. In *SIGMOD*. 159–176.

[38] Xi Wu, Fengan Li, Arun Kumar, Kamalika Chaudhuri, Somesh Jha, and Jeffrey F. Naughton. 2017. Bolt-on Differential Privacy for Scalable Stochastic Gradient Descent-based Analytics. In *SIGMOD*. 1307–1322.

[39] Yonghui Xiao and Li Xiong. 2015. Protecting Locations with Differential Privacy under Temporal Correlations. In *CCS*. 1298–1309.

[40] Min Xu, Bolin Ding, Tianhao Wang, and Jingren Zhou. 2020. Collecting and Analyzing Data Jointly from Multiple Services under Local Differential Privacy. *PVLDB* 13, 11 (2020), 2760–2772.

[41] Min Xu, Tianhao Wang, Bolin Ding, Jingren Zhou, Cheng Hong, and Zhicong Huang. 2019. DPSAaS: Multi-Dimensional Data Sharing and Analytics as Services under Local Differential Privacy. *PVLDB* 12, 12 (2019), 1862–1865.

[42] Jun Zhao, Teng Wang, Tao Bai, Kwok-Yan Lam, Xuebin Ren, Xinyu Yang, Shuyu Shi, Yang Liu, and Han Yu. 2019. Reviewing and Improving the Gaussian Mechanism for Differential Privacy. In *CoRR*, Vol. abs/1911.12060.

A APPENDIX

A.1 Utility Analysis of CGM

The general form of CGM, described in Section 4.2, reuses noises from all previous timestamps, whereas in a simplified version presented in Section 4.3 (used in our implementation and the experiments), CGM only reuses the noise at last timestamp, which is clearly more efficient than the general form. A natural question is: does this simplification leads to a degradation in result utility? In this appendix, we prove that the simplified version in fact achieves the same utility as the general form of CGM.

For ease of presentation, we focus on the time series of a particular user, denoted as $(x_1, \ldots, x_l)$, where l is the length of the sequence. In what follows,we first describe the general correlated noise injection mechanism that utilizes all previous data items $x_{i-1}, x_{i-2}, \ldots, x_1$ when releasing the noisy version of x_i. Next, we briefly show that the optimal choice of reuse ratios for this general algorithm is actually the same as CGM in Section 4.3, *i.e.*, the optimal choice is only to utilize x_{i-1} when releasing the noisy x_i.

We first briefly introduce the general correlated noise injection mechanism that utilizes all previous data items $x_{i-1}, x_{i-2}, \ldots, x_1$ when releasing noisy x_i. Without loss of generality, we consider a data stream of scalars, where each item x_i satisfies $|x_i| \leq \frac{1}{2}$ and $|x_i - x_{i-1}| \leq C$ for $i \geq 2$ with some constant $C < \frac{1}{2}$. Recall that x_i can be written as:

$$\begin{aligned} x_i &= x_i - \sum_{j=1}^{i-1} r_i^{(j)} \cdot x_j + \sum_{j=1}^{i-1} r_i^{(j)} \cdot x_j \\ &= \underbrace{(1 - \sum_{j=1}^{i-1} r_i^{(j)}) \cdot x_i + \sum_{j=1}^{i-1} r_i^{(j)} \cdot (x_i - x_j)}_{\text{first term}} + \underbrace{\sum_{j=1}^{i-1} r_i^{(j)} \cdot x_j}_{\text{second term}}, \end{aligned}$$

where $r_i^{(j)}$ is the reuse ratio of the overall noise in noisy x_j when releasing noisy x_i. Without loss of generality, we only consider the case when all $0 \leq r_i^{(j)} \leq 1$ (for $1 \leq j < i$) and $\sum_{j=1}^{i-1} r_i^{(j)} \leq 1$. Note that for the sensitivity the first term, each $(x_i - x_j)$ contributes $r_i^{(j)} \cdot (2(j-i) \cdot C - 1)$. Since we want this to be as small as possible, we shall only consider utilizing items x_j, where $j > i - \frac{1}{2C}$. Without loss of generality we let $\frac{1}{2C}$ be an integer. Hence, the above expression for x_i can be re-written as:

$$x_i = \underbrace{(1 - \sum_{j=\max(i-\frac{1}{2C}+1,1)}^{i-1} r_i^{(j)}) \cdot x_i + \sum_{j=\max(i-\frac{1}{2C}+1,1)}^{i-1} r_i^{(j)} \cdot (x_i - x_j)}_{\text{first term}} + \underbrace{\sum_{j=i-\frac{1}{2C}}^{i-1} r_i^{(j)} x_j}_{\text{second term}}. \tag{25}$$

The general correlated noise injection mechanism reuses $r_i^{(j)}$ of the old noise injected to x_j, and injects a fresh noise of scale calibrated to the sensitivity of the first term in Eq. (25) when releasing noisy x_i. We denote the fresh noise first injected to noisy x_i as Z_i, where all Z_i's are independent. Then each noisy x_i has $i-1$ independent sources of noises, namely, $Z_1, \ldots, Z_{i-1}$. Both the scale of the noise Z_i and the coefficient of all the independent noises $Z_1, \ldots, Z_{i-1}$ in noisy x_i are determined at the i-th timestamp, as we will explain shortly.

At the first timestamp, we release noisy x_1. We denote a clean data item as x_i and its released version as x_i^*. For the first timestamp, we have that

$$x_1^* = x_1 + Z_1. \tag{26}$$

where the scale of Z_1 is calibrated to the sensitivity of x_1 and $var[Z_1]$ is determined accordingly. Without loss of generality, we can set $var[Z_1]$ to 1.

At the second timestamp, we see how correlated noise comes into play. Following Eq. (25), we have that

$$x_2^* = x_2 + r_2^{(1)} \cdot Z_1 + Z_2. \tag{27}$$

The overall variance in x_2^* can be expressed as:

$$var[x_2^*] = var[Z_2] + var[Z_1] \cdot \left(r_2^{(1)}\right)^2,$$

where we put $var[Z_1]$ in front of $\left(r_2^{(1)}\right)^2$ to emphasize that $var[Z_1]$ is a constant while $r_2^{(1)}$ is to be determined. Since the scale of Z_2 is calibrated to the sensitivity of $\left((1 - r_2^{(1)}) \cdot x_2 + r_2^{(1)} \cdot (x_2 - x_1)\right)$, i.e., $(1 - r_2^{(1)}) + 2C \cdot r_2^{(1)}$ and $var[Z_1] = 1$, we can write the explicit expression for $var[x_2^*]$ as:

$$var[x_2^*] = \left((1 - r_2^{(1)}) + 2C \cdot r_2^{(1)}\right)^2 + 1 \cdot \left(r_2^{(1)}\right)^2. \tag{28}$$

Following Eq. (28), the mechanism computes the optimal reuse ratio $r_2^{(1)}$ that minimizes Eq. (28). This problem can be seen as a quadratic programming instance of one dimension:

$$\begin{aligned} \min_{r_2^{(1)}} \quad & \left((1-2C)^2+1\right)\cdot (r_2^{(1)})^2 - 2(1-2C)\cdot r_2^{(1)} \\ \text{s.t.} \quad & 0 \le r_2^{(1)} \le 1 \end{aligned} \tag{29}$$

Solving this problem gives us the optimal value for $r_2^{(1)} = \frac{1-2C}{(1-2C)^2+1}$ and $var[x_2^*] = \frac{4C-4C^2}{1-(1-2C)^4}$, which are in line with Eq.(21) and Eq.(22) in our original draft, respectively. In addition, the variance of the fresh noise Z_2 is $var[Z_2] = \frac{1}{((1-2C)^2+1)^2}$. Bear in mind that Z_2 is independent with Z_1.

At the third timestamp, the release of noisy x_3 becomes tricky, since the noise in noisy x_1 is correlated with *those* (recall that Z_2 and Z_1 are two independent noises) noises in noisy x_2. We have that:

$$x_3^* = x_3 + r_3^{(2)}\cdot(r_2^{(1)}\cdot Z_1 + Z_2) + r_3^{(1)}\cdot Z_1 + Z_3, \tag{30}$$

where we have abused the notation and use $r_2^{(1)}$ to denote the optimal reuse ratio computed from the second timestamp. In Eq. (30), $r_2^{(1)}\cdot Z_1$ and Z_2 are the noises injected to x_2^*, Z_1 is the noise injected to x_1^*, and finally, Z_3 is the fresh noise injected to x_3^*. The overall variance in x_3^* can be expressed as:

$$var[x_3^*] = var[Z_3] + var[Z_2]\cdot\left(r_3^{(2)}\right)^2 + var[Z_1]\cdot\left(r_2^{(1)}\cdot r_3^{(2)} + r_3^{(1)}\right)^2,$$

where we put $var[Z_2]$ in front of $\left(r_3^{(2)}\right)^2$, $var[Z_1]$ in front of $\left(r_2^{(1)}\cdot r_3^{(2)} + r_3^{(1)}\right)^2$ and $r_2^{(1)}$ in front of $r_3^{(2)}$ to emphasize that $var[Z_2]$, $var[Z_1]$ and $r_2^{(1)}$ are constant while $r_3^{(2)}$ and $r_3^{(1)}$ are to be determined. Note that we group the noises contributed by Z_1 in both x_2^* and x_1^* together, since they are sampled from the same source of randomness. Overall, the noise in x_3^* comes from three sources, the reused noise of Z_1, Z_2, and the fresh noise Z_3. Similar to the previous timestamp, we can rewrite the explicit expression for $var[x_3^*]$ as:

$$var[x_3^*] = \left((1-r_3^{(1)}-r_3^{(2)}) + 4C\cdot r_3^{(1)} + 2C\cdot r_3^{(2)}\right)^2 + \frac{1}{\left((1-2C)^2+1\right)^2}\cdot\left(r_3^{(2)}\right)^2 + 1\cdot\left(\frac{1-2C}{(1-2C)^2+1}\cdot r_3^{(2)} + r_3^{(1)}\right)^2. \tag{31}$$

Following Eq. (31), the mechanism computes the optimal $(r_3^{(1)}, r_3^{(2)})$ that minimizes Eq. (31), and the value of $var[Z_3]$ is also determined in this way. Similar to the previous timestamp, this problem can be seen as a quadratic programming instance of two dimensions, denote the column vector $(r_3^{(1)}, r_3^{(2)})$ as r:

$$\begin{aligned} \min_{r} \quad & \frac{1}{2} r^T H r + f^T r \\ \text{s.t.} \quad & \mathbf{0} \le r \le \mathbf{1} \\ & r_3^{(1)} + r_3^{(2)} \le 1 \end{aligned} \tag{32}$$

where $f = (-2(1-4C), -2(1-2C))$, and the 2−by−2 matrix H is $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$, with $a = 2(1-4C)^2 + 2 > 0$, $b = c = 2(1-4C)(1-2C) + \frac{2(1-2C)}{(1-2C)^2+1}$ and $d = 2(1-2C)^2 + \frac{2}{(1-2C)^2+1} > 0$. The gradient of the objective function is $Hr + f$. In particular, we consider the gradient towards the direction of $(1-2C, -1+4C)$ at any point $r = (r_3^{(1)}, r_3^{(2)})$ in the feasible region. Through algebraic calculation, it is easy to see that this gradient is non-negative. In other words, the value of the objective function is non-decreasing along the direction of $(1-2C, -1+4C)$ (since that $1-2C > 1-4C$). Hence, the optimal (minimum) solution must be from the boundary $\{(r_3^{(1)}, r_3^{(2)}) : r_3^{(1)} = 0, 0 \le r_3^{(2)} \le 1\}$ (see Figure 5), and the optimal $r_3^{(2)}$ is determined as in our original draft. Accordingly, the optimal strategy is to utilize x_2 only when releasing noisy x_3. A similar argument of utilizing x_{i-1} only when releasing noisy x_i applies to all subsequent timestamps $i > 3$ and we omit the details here.

In conclusion, CGM, which utilizes only the noise from the previous timestamp, is optimal among all correlated noise injection mechanisms for privately releasing streaming data with differential bound constraints.

A.2 Empirical vs. Theoretical Performance of CGM

This appendix compares the empirical performance of CGM with its theoretical analysis through additional experiments of CGM with varying clipping threshold C. In particular, the empirical error of CGM comes from two sources: (i) the error due to clipping the differential between neighboring data items, and (ii) the error due to additive Gaussian noise to satisfy LDP. When (i) is negligible compared with (ii), the empirical performance of CGM is close to its theoretical guarantee as stated in Theorem 5.1. Conversely, when (i) is large, *i.e.*, when the differential bound assumption (explained in Section 3.1) is violated, the empirical error of CGM can be significantly higher than that given by Theorem 5.1.

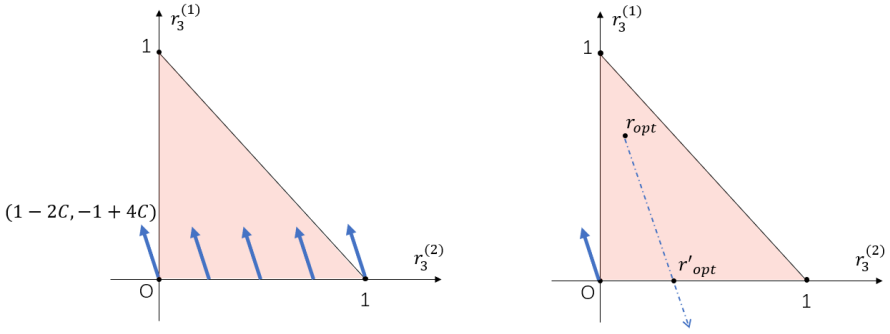


Figure 5: The illustration for the quadratic programming problem defined as in Eq. (32). The feasible region is colored in red. The value of the objective function is non-decreasing along the direction of $(1-2C,-1+4C)$ (colored in blue). Hence, the minimum solution to this problem must be from the boundary $\{(r_3^{(1)}, r_3^{(2)}) : r_3^{(1)} = 0, 0 \le r_3^{(2)} \le 1\}$. If, to the contrary, the minimum solution r_{opt} is not from the boundary $\{(r_3^{(1)}, r_3^{(2)}) : r_3^{(1)} = 0, 0 \le r_3^{(2)} \le 1\}$, then one can always find another solution r'_{opt} by starting from r_{opt} and go along the direction of $(-1+2C, 1-4C)$ (along the blue dashed line) until crossing the boundary $\{(r_3^{(1)}, r_3^{(2)}) : r_3^{(1)} = 0, 0 \le r_3^{(2)} \le 1\}$. It is easy to see that r'_{opt} is at least as good as r_{opt} since the value of the objective function does not increase along the direction of $(-1+2C, 1-4C)$ and we've reached a contradiction. Hence, the minimum solution must be from the boundary $\{(r_3^{(1)}, r_3^{(2)}) : r_3^{(1)} = 0, 0 \le r_3^{(2)} \le 1\}$.

In Figure 6 (resp. Figure 7), we plot the error due to bias by clipping in CGM, the empirical error of CGM (which includes the error due to clipping bias), the theoretical error of CGM, and the empirical error of the baseline solution on the Kaggle Web Traffic dataset (resp. Beijing taxi dataset), with varying clipping threshold $C \in \{100, 200, 500, 1000\}$ (resp. $C \in \{0.01, 0.02, 0.05, 0.1\}$), privacy parameter $\epsilon \in \{0.25, 0.5, 1, 2\}$ and δ fixed to 10^{-5}. From the results on Beijing taxi dataset (Figure 7), we observe that the difference between the theoretical and empirical errors of CGM is negligible. This is because the error due to bias is small compared with the noise due to DP, as is evident from the figure.

Regarding the Kaggle web dataset (Figure 6), the difference between the theoretical and empirical errors of CGM is negligible compared with the error of the baseline under a strong privacy guarantee (see Figures 6a and 6b); while the error due to bias in CGM is comparable to the error due to privacy in CGM under a relatively weak privacy guarantee (see Figures 6c and 6d). As a result, there is a notable difference between the empirical error of CGM and the theoretical analysis in Figures 6a and 6b. Overall, the difference between the empirical performance and the theoretical guarantee of CGM decreases when C increases. In addition, the difference is more notable for a larger value of ϵ. Despite this, by exploiting the enforced correlation, CGM significantly reduces the error due to privacy, which more than compensates the error due to bias. As a result, CGM consistently outperforms the baseline approach under various settings of the clipping threshold C for both datasets.

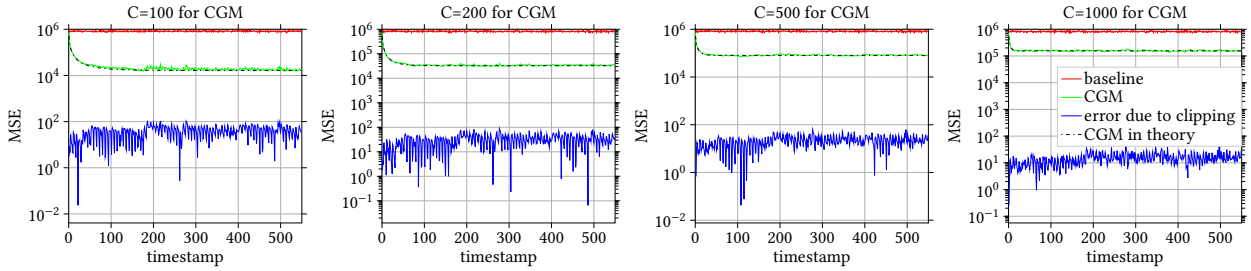
A.3 CGM with Data Stream Subsampling

FAST [22, 23] samples the data stream and only perturbs data items of the sampling points. This sampling technique is orthogonal to CGM. In this subsection, we provide experimental results of combining CGM with this data stream subsampling technique.

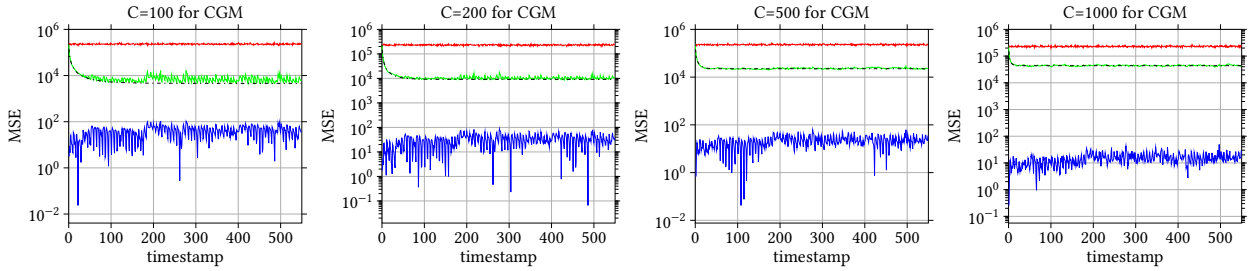
For the Beijing taxi dataset (see Figure 8), we only run CGM and the baseline approach for the odd timestamps with the same overall privacy budget of $\epsilon \in \{0.25, 0.5, 1, 2\}$ and $\delta = 10^{-5}$. For the Kaggle web traffic dataset (see Figures 9 and 10), we have two experiment settings. In the first setting, we double (without loss of generality) the daily privacy budget for the odd days. Namely, we allocate a daily privacy budget of $\epsilon \in \{0.5, 1, 2, 4\}$ and $\delta = 10^{-5}$ to only the odd days and run CGM and the baseline approach only for those days. In the second setting, we allocate an overall privacy budget of $\epsilon \in \{5, 10, 20, 50\}$ and $\delta = 10^{-5}$ to only the odd days and run CGM and the baseline approach only for those days. Note that the differential bound for adjacent odd timestamps in CGM are set twice as much as in the original draft, since we only need to process the odd timestamps. Finally, the error is calculated on all odd and even timestamps. From Figures 8, 9 and 10, it is clear that CGM still improves the utility when combined with the data stream subsampling technique.

A.4 CGM with Lagged Temporal Autocorrelations

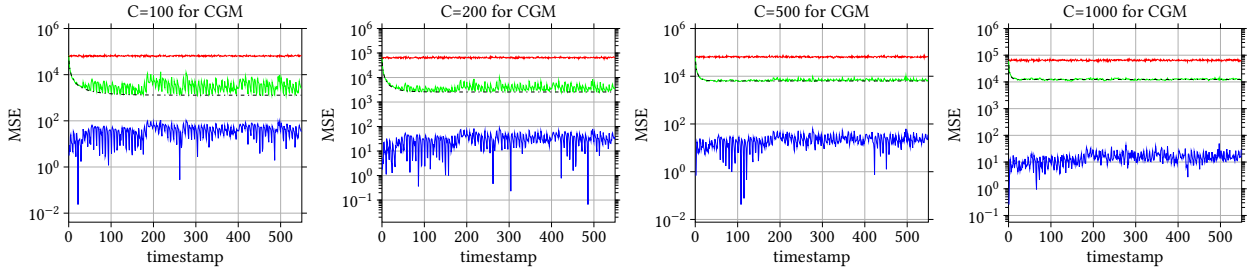
In this appendix, we outlines the application of CGM on data streams with temporal autocorrelations, where the lag is more than 1 timestamp, which is briefly mentioned in Section 3.1. Let us consider a lag of h. Without loss of generality, we focus on the case that h divides l (the total length of the data stream). Namely, for the k-th user, $\|x_{k,h\cdot i+j} - x_{k,h\cdot(i+1)+j}\| \le C$ for all $1 \le i \le \frac{l}{h}$ and $1 \le j \le h$. For example, if $h = 7$, we can think of the data items as they arrive daily; to be more specific, the $(h \cdot i + j)$-th data item arrives on the j-th day of the i-th week.



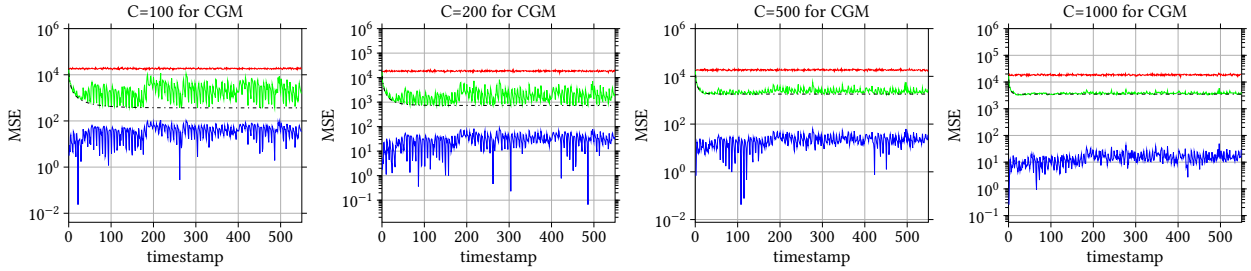
(a) Daily privacy budget $\epsilon = 0.25$ and $\delta = 10^{-5}$



(b) Daily privacy budget $\epsilon = 0.5$ and $\delta = 10^{-5}$



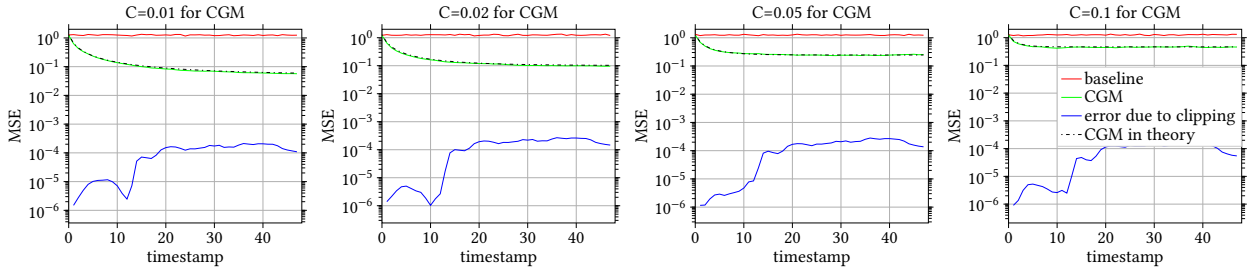
(c) Daily privacy budget $\epsilon = 1$ and $\delta = 10^{-5}$



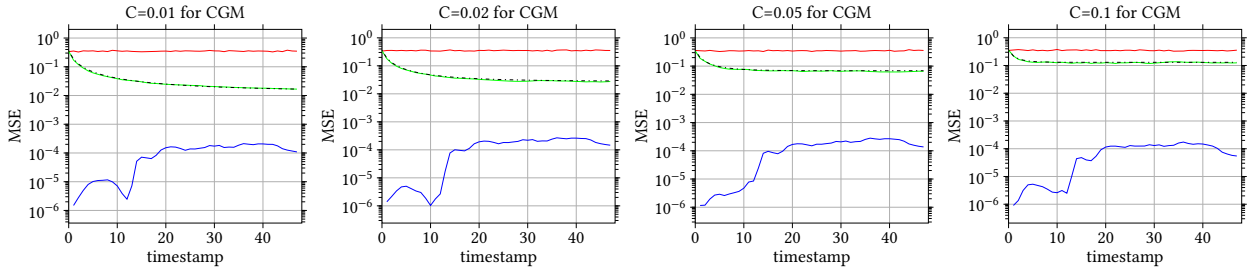
(d) Daily privacy budget $\epsilon = 2$ and $\delta = 10^{-5}$

Figure 6: Empirical performance of CGM, which does take into the account of the error due to bias by clipping, and the baseline approach on the Kaggle Web Traffic dataset, with varying daily privacy budget $\epsilon \in \{0.25, 0.5, 1, 2\}$ and $\delta = 10^{-5}$. For CGM, the differential bound varies from $\{100, 200, 500, 1000\}$. We also include the error due to bias by clipping in CGM and the theoretical error of CGM, which does not take into the account of the error due to bias, in the figure for comparison.

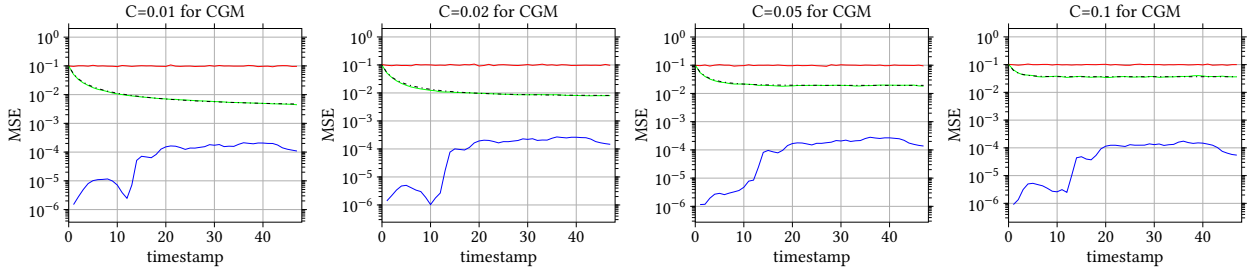
Next, we explain how to apply CGM to this data stream. In terms of privacy, we can view the entire data stream of length l as the output and apply Lemma 5.3, which is exactly the same as what we do for the case of $h = 1$. Next, we can just plug i into Eq. (21) and Eq. (22) to get the values of the reuse ratio and the scale of the fresh noise, respectively. Following the same example of $h = 7$, we have that the reuse ratio remains the same throughout a week, and the reused noise is from the same day in the previous week. The corresponding algorithm is outlined as Algorithm 4.



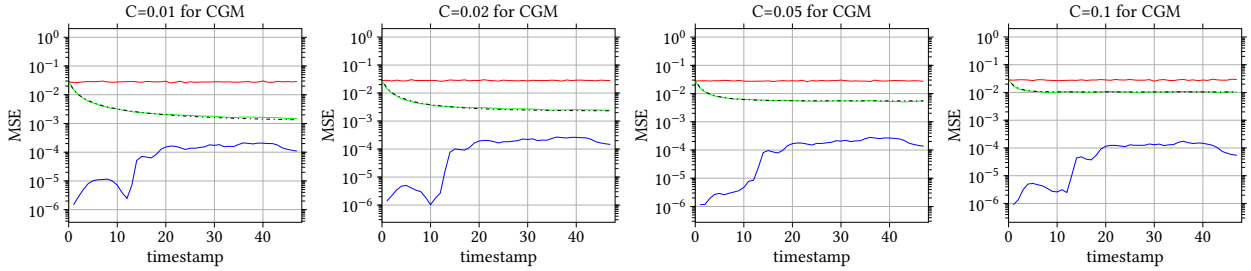
(a) Total privacy budget $\epsilon = 0.25$ and $\delta = 10^{-5}$



(b) Total privacy budget $\epsilon = 0.5$ and $\delta = 10^{-5}$



(c) Total privacy budget $\epsilon = 1$ and $\delta = 10^{-5}$



(d) Total privacy budget $\epsilon = 2$ and $\delta = 10^{-5}$

Figure 7: Empirical performances of CGM, which does take into the account of the error due to bias by clipping, and the baseline approach on the Beijing Taxi dataset, with varying total privacy budget $\epsilon \in \{0.25, 0.5, 1, 2\}$ and $\delta = 10^{-5}$. For CGM, the differential bound varies from $\{0.01, 0.02, 0.05, 0.1\}$. We also include the error due to bias by clipping in CGM and the theoretical error of CGM, which does not take into the account of the error due to bias, in the figure for comparison.

A.5 CGM under Rényi Differential Privacy

In this subsection, we show how to apply CGM under Rényi Differential Privacy (RDP) [33]. A similar reasoning applies for zCDP [10] and we omit the details here for brevity.

We first briefly explain why CGM works under RDP with no modifications to the reuse ratio. This is because the α-th order Rényi divergence between two Gaussian distributions $\mathcal{N}(0, \sigma^2)$ and $\mathcal{N}(0, \sigma^2)$ is equal to $\frac{\alpha\mu^2}{\sigma^2}$ (Proposition 7 in [33]). Essentially, this means that the

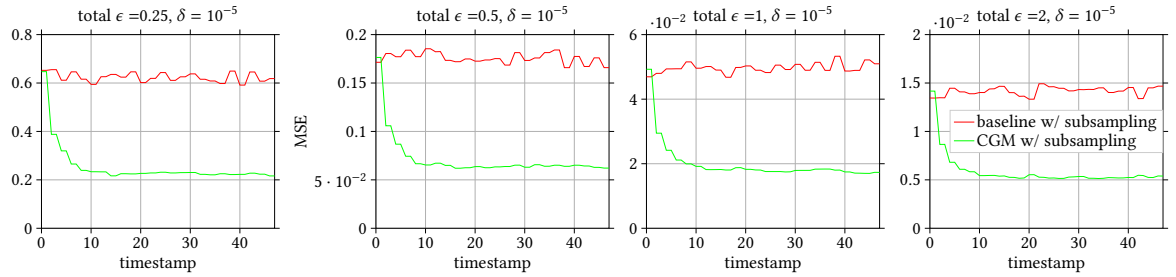


Figure 8: Utility performances of CGM at the odd timestamps and the baseline approach at the odd timestamps on the Beijing Taxi dataset, with varying total privacy budget for all updates $\epsilon \in \{0.25, 0.5, 1, 2\}$ and $\delta = 10^{-5}$. For CGM, the differential bound is fixed to $2 \times C = 0.1$ (where $C = 0.05$, as in the original draft) for adjacent odd timestamps. The whole space is normalized to $[0, 1] \times [0, 1]$, and the query region is $[0.45, 0.55] \times [0.45, 0.55]$.

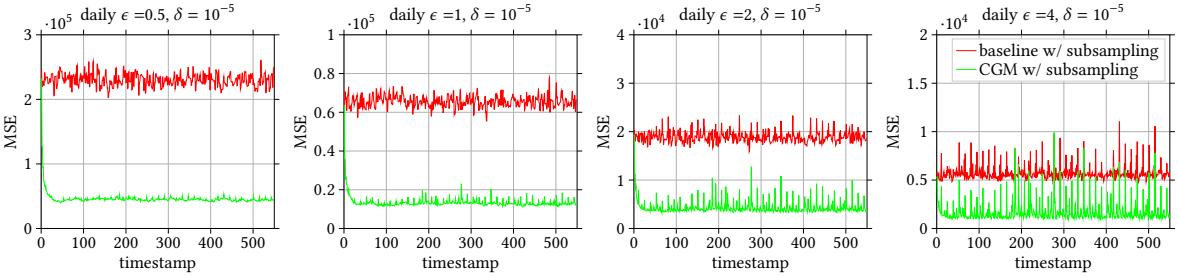


Figure 9: Utility performances of CGM at the odd timestamps and the baseline approach at the odd timestamps on the Kaggle Web Traffic dataset, with daily privacy budget $\epsilon \in \{0.5, 1, 2, 4\}$ and $\delta = 10^{-5}$ for each odd day. For CGM, the differential bound is fixed to $2 \times C = 1000$ (where $C = 500$, as in the original draft).

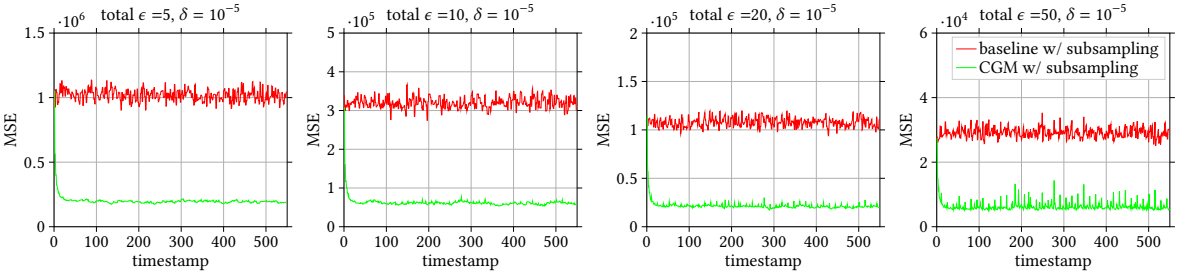


Figure 10: Utility performances of CGM at the odd timestamps and the baseline approach at the odd timestamps on the Kaggle Web Traffic dataset, with overall privacy budget $\epsilon \in \{5, 10, 20, 50\}$ and $\delta = 10^{-5}$ for all odd days. For CGM, the differential bound is fixed to $2 \times C = 1000$ (where $C = 500$, as in the original draft).

scale of the Gaussian noise is linear with respect to the sensitivity of the function. Hence, CGM, which is based on this linear relationship between scale of the noise and the sensitivity, adheres to the same set of reuse ratio under RDP and (ϵ, δ)-DP.

Next, we walk through the baseline approach for streaming data release under RDP. Recall from Corollary 3 in [33], adding a Gaussian noise $\mathcal{N}(0, \sigma^2\mathbf{I})$ to a function of sensitivity 1 satisfies $(\alpha, \frac{\alpha}{2\sigma^2})$-RDP. Hence, to achieve (α, ϵ)-RDP, it suffices to add a Gaussian noise $\mathcal{N}(0, \frac{\alpha\epsilon}{2}\mathbf{I})$ to the function. Similarly,

Corollary A.1. *Given a data stream $\{x_{k,1}, \ldots, x_{k,l}\}$ of length l, where each data item satisfies $\|x_{k,i}\| \leq \frac{1}{2}$ for $1 \leq i \leq l$, adding a Gaussian noise $\mathcal{N}(0, \frac{\alpha\epsilon l}{2}\mathbf{I})$ repeatedly at each timestamp to $x_{k,i}$ satisfies (α, ϵ)-RDP for the entire data stream.*

Algorithm 4: CGM for streaming data with periodic correlations

Input: data items $\{x_{k,h\cdot i+j}\}$ $(1 \le i \le \frac{l}{h}$ and $1 \le j \le h)$, where $\|x_{k,h\cdot i+j}\| \le \frac{1}{2}$ and $\|x_{k,h\cdot i+j} - x_{k,h\cdot(i+1)+j}\| \le C$, privacy parameters ϵ, δ.

Output: private estimates $\{x^*_{k,h\cdot i+j}\}$

for $i = 1$ ***to*** $\frac{l}{h}$ **do**
  for $j = 1$ ***to*** h **do**
    if $i = 1$ **then**
      $x^*_{k,h\cdot i+j} \longleftarrow x_{k,h\cdot i+j} + \mathcal{N}(\mathbf{0}, \sigma_1^2 \cdot \mathbf{I})$, where σ_1 is computed as in Eq. (14).
      $v_1 \longleftarrow 1$.
    else
      $r_i \longleftarrow \frac{1-2C}{(1-2C)^2+v_{i-1}}$.
      $\sigma_i \longleftarrow ((1-r_i) + r_i \cdot 2C) \cdot \sigma_1$.
      $x^*_{k,h\cdot i+j} \longleftarrow x_{k,h\cdot i+j} + \mathcal{N}(\mathbf{0}, \sigma_i^2 \cdot \mathbf{I}) + r_i \cdot \gamma_{k,h\cdot(i-1)+j}$, where $\gamma_{k,h\cdot(i-1)+j}$ is the noise injected in $x^*_{k,h\cdot(i-1)+j}$.
      $v_i \longleftarrow \frac{v_{i-1}}{(1-2C)^2+v_{i-1}}$.
    Output $x^*_{k,h\cdot i+j}$

Algorithm 5: Gaussian mechanism for streaming data under RDP

Input: data items $\{x_{k,i}\}_{i=1}^{l}$, where $\|x_{k,i}\| \le \frac{1}{2}$ and $\|x_{k,i} - x_{k,i-1}\| \le C$, privacy parameters ϵ, α.

Output: perturbed estimates $\{x^*_{k,i}\}_{i=1}^{l}$.

for $i = 1$ ***to*** l **do**
  $x^*_{k,i} \longleftarrow x_{k,i} + \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$; where σ is computed as in Corollary A.1 and $\mathbf{I}$ is the $d \times d$ dimension identity matrix.
  Output $x^*_{k,i}$;

Algorithm 6: CGM for streaming data under RDP

Input: data items $\{x_{k,i}\}_{i=1}^{l}$, where $\|x_{k,i}\| \le \frac{1}{2}$ and $\|x_{k,i} - x_{k,i-1}\| \le C$, privacy parameters ϵ, α.

Output: private estimates $\{x^*_{k,i}\}_{i=1}^{l}$.

for $i = 1$ ***to*** l **do**
  if $i = 1$ **then**
    $x^*_{k,i} \longleftarrow x_{k,i} + \mathcal{N}(\mathbf{0}, \sigma^2 \cdot \mathbf{I})$, where σ is computed as in Corollary A.1.
    $v_1 \longleftarrow 1$.
  else
    $r_i \longleftarrow \frac{1-2C}{(1-2C)^2+v_{i-1}}$.
    $\sigma_i \longleftarrow ((1-r_i) + r_i \cdot 2C) \cdot \sigma$.
    $x^*_{k,i} \longleftarrow x_{k,i} + \mathcal{N}(\mathbf{0}, \sigma_i^2 \cdot \mathbf{I}) + r_i \cdot \gamma_{k,i-1}$, where $\gamma_{k,i-1}$ is the noise injected in $x^*_{k,i-1}$.
    $v_i \longleftarrow \frac{v_{i-1}}{(1-2C)^2+v_{i-1}}$.
  Output $x^*_{k,i}$

Finally, to apply CGM under RDP, it suffices to compute the σ for the baseline approach according to Corollary A.1. And at each timestamp, the reuse ratio and the variance for the fresh noise are determined by the following two equations, respectively:

$$r_i = \frac{1-2C}{(1-2C)^2 + \frac{4C-4C^2}{1-(1-2C)^{2i-2}}}, \tag{33}$$

$$\sigma^2_{i,\text{fresh}} = (1 - r_i + r_i \cdot 2C)^2\sigma^2. \tag{34}$$

We outline the baseline approach and CGM in Algorithms 5 and 6, respectively.