Quantifying Differential Privacy in Continuous Data Release under Temporal Correlations

Yang Cao, Masatoshi Yoshikawa, Yonghui Xiao, and Li Xiong

Abstract—Differential Privacy (DP) has received increasing attention as a rigorous privacy framework. Many existing studies employ traditional DP mechanisms (e.g., the Laplace mechanism) as primitives to continuously release private data for protecting privacy at each time point (i.e., event-level privacy), which assume that the data at different time points are independent, or that adversaries do not have knowledge of correlation between data. However, continuously generated data tend to be temporally correlated, and such correlations can be acquired by adversaries. In this paper, we investigate the potential privacy loss of a traditional DP mechanism under temporal correlations. First, we analyze the privacy leakage of a DP mechanism under temporal correlation that can be modeled using Markov Chain. Our analysis reveals that, the event-level privacy loss of a DP mechanism may *increase over time*. We call the unexpected privacy loss *temporal privacy leakage* (TPL). Although TPL may increase over time, we find that its supremum may exist in some cases. Second, we design efficient algorithms for calculating TPL. Third, we propose data releasing mechanisms that convert any existing DP mechanism into one against TPL. Experiments confirm that our approach is efficient and effective.

Index Terms—Differential Privacy, Correlated data, Markov Model, Time series, Streaming data.

1 Introduction

With the development of IoT technology, vast amount of temporal data generated by individuals are being collected, such as trajectories and web page click streams. The continual publication of statistics from these temporal data has the potential for significant social benefits such as disease surveillance , real-time traffic monitoring and web mining. However, privacy concerns hinder the wider use of these data. To this end, *differential privacy under continual observation* [1] [2] [3] [4] [5] [6] [7] [8] has received increasing attention because differential priavcy provides a rigorous privacy guarantee. Intuitively, differential privacy (DP) [9] ensures that the change of any single user's data has a "slight" (bounded in ϵ) impact on the change in outputs. The parameter ϵ is defined to be a positive real number to control the level of privacy guarantee. Larger values of ϵ result in larger privacy leakage. However, most existing works on differentially private continuous data release assume data at different time points are independent, or attackers do not have knowledge of correlation between data. Example 1 shows that temporal correlations may degrade the expected privacy guarantee of a DP mechanism.

Example 1. *Consider the scenario of continuous data release with DP illustrated in Figure 1. A trusted server collects users' locations at each time point in Figure 1(a) and tries to continuously publish differentially private aggregates (i.e., the private counts of people at each location). Suppose that each user appears at only one location at each time point. According to the Laplace mechanism [10], adding $Lap(2/\epsilon)$ noise*[1] *to perturb each count in Figure1(c) can achieve ϵ-DP at each time point. It is because the modification of each cell (raw data) in Figure 1(a) affects at most two cells (true counts) in Figure 1(c), i.e., the global sensitivity is 2. However, it may not be true under the existence of temporal correlations. For example, due to the nature of road networks, user may have a particular mobility pattern such as "always arriving at loc_5 after visiting loc_4" (shown in Figure 1(b)), leading to the patterns illustrated in solid lines of Figure 1(a)(c). Such temporal correlation can be represented as $\Pr(l^t = loc_5 | l^{t-1} = loc_4) = 1$ where l^t is the location of a user at time t. In this case, adding $Lap(2/\epsilon)$ noise only achieves 2ϵ-DP because the change of one cell in Figure 1(a) may affect four cells in Figure 1(c) in the worst case, i.e., the global sensitivity is 4.*

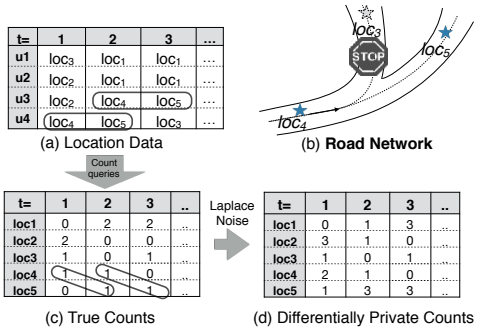


Fig. 1. Continuous Data Release with DP under Temporal Correlations.

Few studies in the literature investigated such potential privacy loss of event-level ϵ-DP under temporal correlations as shown in Example 1. A direct method (without finely utilizing the probability of correlation) involves amplifying the perturbation in terms of *group differential privacy* [11] [10], i.e., protecting the correlated data as a group. In Example 1, for temporal correlation $\Pr(l^t = loc_5 | l^{t-1} = loc_4) = 1$, we can

- *Yang Cao, Xiong Li are with the Department of Math and Computer Science, Emory University, Atlanta, GA, 30322. E-mail: {ycao31, lxiong}@emory.edu.*
- *Masatoshi Yoshikawa is with the Department of Social Informatics, Kyoto University, Kyoto, Japan, 606-8501. E-mail: yoshikawa@i.kyoto-u.ac.jp.*
- *Yonghui Xiao is with Google Inc., Mountain View, CA, 94043 E-mail: yohuxiao@gmail.com.*

Manuscript received XXX; revised XXX.

1. $Lap(b)$ denotes a Laplace distribution with variance $2b^2$.

protect the counts of loc_4 at time $t-1$ and loc_5 at time t in a group by increasing the scale of the perturbation (using $Lap(4/\epsilon)$) at each time point. However, this technique is not suitable for *probabilistic* correlations to finely prevent privacy leakage and may over-perturb the data as a result. For example, regardless of whether $\Pr(l^t = loc_i | l^{t-1} = loc_i)$ is 1 or 0.1, it always protects the correlated data in a bundle.

Although a few studies investigated the issue of differential privacy under probabilistic correlations, existing works are not appropriate for *continuous data release* because of two reasons. First, most of existing works focused on correlations between users (i.e., user-user correlation) rather than correlation between data at different time points (i.e., temporal correlations). Yang et al. [12] proposed Bayesian differential privacy, which measures the privacy leakage under probabilistic correlations between users using a Gaussian Markov Random Field without taking time factor into account. Liu et al. [13] proposed *dependent differential privacy* by introducing two parameters of *dependence size* and *probabilistic dependence relationship* between tuples. It is not clear whether we can specify these parameters for temporally correlated data since it is not commonly used probabilistic model. A concurrent work [14] proposed Markov Quilt Mechanism when data correlation is represented by a Bayesian Network (BN). Although BN is possible to model both user-user correlation (when the nodes in BN are individuals) and temporal correlation (when the nodes in BN are individual data at different time points), [17] assumes the private data are released only at one time. This is the second major difference between our work and all above works: they focus on the setting of "one-shot" data release, which means the private data are released only at one time. To the best of our knowledge, no study reported the dynamical change of the privacy guarantee in continuous data release.

In this work, we call the adversary with knowledge of probabilistic temporal correlations *adversary*$_\mathcal{T}$. Rigorously quantifying and bounding the privacy leakage against adversary$_\mathcal{T}$ in continuous data release remains a challenge. Therefore, our goal is to solve the following problems:

- How do we *analyze the privacy loss* of DP mechanisms against *adversary*$_\mathcal{T}$? (Section 3)
- How do we *calculate* such privacy loss efficiently? (Section 4)
- How do we *bound* such privacy loss in continuous data release? (Section 5)

1.1 Contributions

Our contributions are summarized as follows.

First, we show that the privacy guarantee of data release *at a single time* is not on its own or static; instead, due to the existence of temporal correlation, it may be increasing with previous release and even future release. We first adopt a commonly used model Markov Chain to describe temporal correlations between data at different time points, which includes *backward* and *forward* correlations, i.e., $\Pr(l_i^{t-1}|l_i^t)$ and $\Pr(l_i^t|l_i^{t-1})$ where l_i^t denotes the value (e.g., location) of user i at time t. We then define *Temporal Privacy Leakage (TPL)* as the privacy loss of a DP mechanism at time t against adversary$_\mathcal{T}$ who has knowledge of the above Markov model and observes the continuous data release. We show that TPL includes two parts: *Backward Privacy Leakage (BPL)* and *Forward Privacy leakage (FPL)*. Intuitively, BPL at time t is affected by previously releases that are from time 1 to $t-1$, and FPL at time t will be affected by future releases that are from time $t+1$ to the end of release. We define α-*differential privacy under temporal correlation*, denoted as α-$DP_\mathcal{T}$, to formalize the privacy guarantee of a DP mechanism against adversary$_\mathcal{T}$, which implies the temporal privacy leakage should be bounded in α. We prove a new form of sequential composition theorem for α-$DP_\mathcal{T}$, which reveals interesting connections among event-level DP, w-event DP [7] and user-level DP [4] [5].

Second, we design efficient algorithms to calculate TPL under given backward and forward temporal correlations. Our idea is to transform such calculation into finding an optimal solution of a *Linear-Fractional Programming* problem. This type of optimization can be solved by well-studied methods, e.g., simplex algorithm, in exponential time. By exploiting the constraints of this problem, we propose fast algorithms to obtain the optimal solution without directly solving the problem. Experiments show our algorithms outperform the off-the-shelf optimization software (CPLEX) by several orders of magnitude with the same optimal answer.

Third, we design novel data release mechanisms against TPL effectively. Our scheme is to carefully calibrate the privacy budget of the traditional DP mechanism at each time point to make them satisfy α-DP$_\mathcal{T}$. A challenge is that TPL is dynamically changing due to previous and even future allocated privacy budgets so that α-DP$_\mathcal{T}$ is hard to achieve. In our first solution, we prove that, even though TPL may increase over time, its supremum may exist when allocating a calibrated privacy budget at each time. That is to say, TPL will never be greater than such supremum α no matter when the release ends. However, when the releases are too short (TPL is far from reaching its supremum), we may over-perturb the data. In our second solution, we design another budget allocation scheme that exactly achieves α-DP$_\mathcal{T}$ at each time point.

Finally, experiments confirm the efficiency and effectiveness of our TPL quantification algorithms and data release mechanisms. We also demonstrate the impact of different degree of temporal correlations on privacy leakage.

2 PRELIMINARIES

2.1 Differential Privacy

Differential privacy [9] is a formal definition of data privacy. Let D be a database and D' be a copy of D that is different in any one tuple. D and D' are *neighboring databases*. A differentially private output from D or D' should exhibit little difference.

Definition 1 (ϵ-DP)**.** *$\mathcal{M}$ is a randomized mechanism that takes as input D and outputs $\boldsymbol{r}$, i.e., $\mathcal{M}(D) = \boldsymbol{r}$. $\mathcal{M}$ satisfies ϵ-differential privacy if the following inequality is true for any pair of neighboring databases D, D' and all possible outputs $\boldsymbol{r}$.*

$$\log \frac{\Pr(\boldsymbol{r} \in Range(M)|D)}{\Pr(\boldsymbol{r} \in Range(M)|D')} \leq \epsilon. \quad (1)$$

The parameter ϵ, called the *privacy budget*, represents the degree of privacy offered. A commonly used method to achieve ϵ-DP is the Laplace mechanism as shown below.

Theorem 1 (Laplace Mechanism). *Let $Q : D \to \mathbb{R}$ be a statistical query on database D. The sensitivity of Q is defined as the maximum L_1 norm between $Q(D)$ and $Q(D')$, i.e., $\Delta = \max_{D,D'} ||Q(D) - Q(D')||_1$. We can achieve ϵ-DP by adding Laplace noise with scale Δ/ϵ, i.e., $Lap(\Delta/\epsilon)$.*

2.2 Privacy Leakage

In this section, we define the privacy leakage of a DP mechanism against a type of adversaries A_i, who targets user i's value in the database, i.e., $l_i \in [loc_1, \ldots, loc_n]$, and has knowledge of $D_{\mathcal{K}} = D - \{l_i\}$. The adversary A_i observes the private output $\boldsymbol{r}$ and attempts to distinguish whether user i's value is loc_j or loc_k where $loc_j, loc_k \in [loc_1, \ldots, loc_n]$. We define the privacy leakage of a DP mechanism w.r.t. such adversaries as follows.

Definition 2. *The privacy leakage of a DP mechanism $\mathcal{M}$ against one A_i with a specific i and all $A_i, i \in \boldsymbol{U}$ are defined, respectively, as follows in which l_i and l_i' are two different possible values of i-th data in the database.*

$$PL_0(A_i, \mathcal{M}) \stackrel{\text{def}}{=\!=} \sup_{\boldsymbol{r}, l_i, l_i'} \log \frac{\Pr(\boldsymbol{r}|l_i, D_{\mathcal{K}})}{\Pr(\boldsymbol{r}|l_i', D_{\mathcal{K}})}$$

$$PL_0(\mathcal{M}) \stackrel{\text{def}}{=\!=} \max_{\forall A_i, i \in \boldsymbol{U}} PL_0(A_i, \mathcal{M}) = \sup_{\boldsymbol{r}, D, D'} \log \frac{\Pr(\boldsymbol{r}|D)}{\Pr(\boldsymbol{r}|D')}$$

In other words, the privacy budget of a DP mechanism can be considered as a metric of *privacy leakage*. The larger ϵ, the larger the privacy leakage. We note that a ϵ'-DP mechanism automatically satisfies ϵ-DP if $\epsilon' < \epsilon$. For convenience, in the rest of this paper, when we say that $\mathcal{M}$ satisfies ϵ-DP, we mean that the supremum of privacy leakage is ϵ.

2.3 Problem Setting

We attempt to quantify the potential privacy loss of a DP mechanism under temporal correlations in the context of *continuous data release* (e.g., releasing private counts at each time as shown in Figure 1). For the convenience of analysis, let us assume the length of release time is T. Note that we do not need to know the exact T in this paper. Users in the database, denoted by $\boldsymbol{U}$, are generating data continuously. Let $\boldsymbol{loc} = \{loc_1, \ldots, loc_n\}$ be all possible values of user's data. We denote the value of user i at time t by l_i^t. A trusted server collects the data of each user into the database $D^t = \{l_1^t, \ldots, l_{|\boldsymbol{U}|}^t\}$ at each time t (e.g., the columns in Figure 1(a)). Without loss of generality, we assume that each user contributes only one data point in D^t. DP mechanisms $\mathcal{M}^t$ release differentially private outputs $\boldsymbol{r}^t$ independently at different time t. For simplicity, we let $\mathcal{M}^t$ to be the same DP mechanism but may with different privacy budgets at each $t \in [1, T]$. Our goal is to quantify and bound the potential privacy loss of $\mathcal{M}^t$ against adversaries with knowledge of temporal correlations. We note that while we use location data in Example 1, the problem setting is general for temporally correlated data. We summarize notations used in this paper in Table 1.

Our problem setting is identical to *differential privacy under continual observation* in the literature [1] [2] [3] [4] [5] [6] [7] [8]. In contrast to "one-shot" data release over a static database, the attackers can observe multiple private outputs, i.e., $\boldsymbol{r}^1, \ldots, \boldsymbol{r}^t$. There are typically two different privacy goals in the context of continuous data release: *event-level* and *user-level* [4] [5]. The former protects each user's single data point at time t (i.e., the neighboring databases are D^t and $D^{t\prime}$), whereas the latter protects the presence of a user with all her data on the timeline (i.e., the neighboring databases are $\{D^1, \ldots, D^t\}$ and $\{D^{1\prime}, \ldots, D^{t\prime}\}$). In this work, we start from examining event-level DP under temporal correlations, and we also extend the discussion to user-level DP by studying the sequential composability of the privacy leakage.

TABLE 1
Summary of Notations.

$\boldsymbol{U}$	The set of users in the database
i	The i-th user where $i \in [1, \lvert\boldsymbol{U}\rvert]$
$\boldsymbol{loc}$	Value domain $\{loc_1, \ldots, loc_n\}$ of all user's data
$l_i^t, l_i^{t\prime}$	The data of user i at time t, $l_i^t \in \boldsymbol{loc}$, $l_i^t \neq l_i^{t\prime}$
D^t	The database at time t, $D^t = \{l_1^t, \ldots, l_n^t\}$
$\mathcal{M}^t$	Differentially private mechanism over D^t
$\boldsymbol{r}^t$	Differentially private output at time t
A_i	Adversary who targets user i without temporal correlations
$A_i^{\mathcal{T}}$	Adversary A_i with temporal correlations
P_i^B	Transition matrix that represents $\Pr(l_i^{t-1}\lvert l_i^t)$, i.e., backward temporal correlation, known to $A_i^{\mathcal{T}}$
P_i^F	Transition matrix that represents $\Pr(l_i^t\lvert l_i^{t-1})$, i.e., forward temporal correlation, known to $A_i^{\mathcal{T}}$
$D_{\mathcal{K}}^t$	The subset of database $D^t - \{l_i^t\}$, known to $A_i^{\mathcal{T}}$

3 Analyzing Temporal Privacy Leakage

3.1 Adversay with Knowledge of Temporal Correlations

Markov Chain for Temporal Correlations. The Markov chain (MC) is extensively used in modeling user mobility. For a time-homogeneous first-order MC, a user's current value only depends on the previous one. The parameter of the MC is the *transition matrix*, which describes the probabilities for transition between values. The sum of the probabilities in each row of the transition matrix is 1. A concrete example of transition matrix and time-reversed one for location data is shown in Figure 2. As shown in Figure 2(a), if user i is at loc_1 now (time t); then, the probability of coming from loc_3 (time $t-1$) is 0.7, namely, $\Pr(l_i^{t-1} = loc_3 | l_i^t = loc_1) = 0.7$. As shown in Figure 2(b), if user i was at loc_3 at the previous time $t-1$, then the probability of being at loc_1 now (time t) is 0.6; namely, $\Pr(l_i^t = loc_1 | l_i^{t-1} = loc_3) = 0.6$. We call the transition matrices in Figure 2(a) and (b) as backward temporal correlation and forward temporal correlation, respectively.

Definition 3 (Temporal Correlations). *The backward and forward temporal correlations between user i's data l_i^{t-1} and l_i^t are described by transition matrices $P_i^B, P_i^F \in \mathbb{R}^{n \times n}$, representing $\Pr(l_i^{t-1}|l_i^t)$ and $\Pr(l_i^t|l_i^{t-1})$, respectively.*

It is reasonable to consider that the backward and/or forward temporal correlations could be acquired by adversaries. For example, the adversaries can learn them from user's historical trajectories (or the reversed trajectories) by well studied methods such as Maximum Likelihood estimation (supervised) or Baum-Welch algorithm (unsupervised). Also, if the initial distribution of l_i^1 is known (i.e., $\Pr(l_i^1)$), the backward temporal correlation (i.e., $\Pr(l_i^{t-1}|l_i^t)$) can be derived from the forward temporal correlation (i.e., $\Pr(l_i^t|l_i^{t-1})$) by Bayesian inference. We assume the attackers' knowledge about temporal correlations is given in our framework.

We now define an enhanced version of A_i (in Definition 2) with knowledge of temporal correlations.

(a) Transition Matrix $\Pr(l_i^{t-1}|l_i^t)$

time t \ time t-1	$\mathbf{loc_1}$	$\mathbf{loc_2}$	$\mathbf{loc_3}$
$\mathbf{loc_1}$	0.1	0.2	**0.7**
$\mathbf{loc_2}$	0	0	1
$\mathbf{loc_3}$	0.3	0.3	0.4

Backward Temporal Correlation P_i^B

(b) Transition Matrix $\Pr(l_i^t|l_i^{t-1})$

time t-1 \ time t	$\mathbf{loc_1}$	$\mathbf{loc_2}$	$\mathbf{loc_3}$
$\mathbf{loc_1}$	0.2	0.3	0.5
$\mathbf{loc_2}$	0.1	0.1	0.8
$\mathbf{loc_3}$	**0.6**	0.2	0.2

Forward Temporal Correlation P_i^F

Fig. 2. Examples of Temporal Correlations.

Definition 4 (Adversary$_\mathcal{T}$)**.** *Adversary$_\mathcal{T}$ is a class of adversaries who have knowledge of (1) all other users' data $D_\mathcal{K}^t$ at each time t except the one of the targeted victim, i.e., $D_\mathcal{K}^t = D^t - \{l_i^t\}$, and (2) backward and/or forward temporal correlations represented as transition matrices P_i^B and P_i^F. We denote Adversary$_\mathcal{T}$ who targets user i by $A_i^\mathcal{T}(P_i^B, P_i^F)$.*

There are three types of adversary$_\mathcal{T}$: (i) $A_i^\mathcal{T}(P_i^B, \emptyset)$, (ii) $A_i^\mathcal{T}(\emptyset, P_i^F)$, (iii) $A_i^\mathcal{T}(P_i^B, P_i^F)$, where $\emptyset$ denotes that the corresponding correlations are not known to the adversaries. For simplicity, we denote types (i) and (ii) as $A_i^\mathcal{T}(P_i^B)$ and $A_i^\mathcal{T}(P_i^F)$, respectively. We note that $A_i^\mathcal{T}(\emptyset, \emptyset)$ is the same as the traditional DP adversary A_i without any knowledge of temporal correlations.

3.2 Temporal Privacy Leakage

We now define the privacy leakage w.r.t. adversary$_\mathcal{T}$. The adversary $A_i^\mathcal{T}$ observes the differentially private outputs $\boldsymbol{r}^t$, which is released by a traditional differentially private mechanisms $\mathcal{M}^t$ (e.g., Laplace mechanism) at each time point $t \in [1, T]$, and attempts to infer the possible value of user i's data at t, namely l_i^t. Similar to Definition 2, we define the privacy leakage in terms of event-level differential privacy in the context of continual data release as described in Section 2.3.

Definition 5 (Temporal Privacy Leakage, TPL)**.** *Let $D^{t\prime}$ be a neighboring database of D^t. Let $D_\mathcal{K}^t$ be the tuple knowledge of $A_i^\mathcal{T}$. We have $D^{t\prime} = D_\mathcal{K}^t \cup \{l_i^t\}$ and $D^{t\prime} = D_\mathcal{K}^t \cup \{l_i^{t\prime}\}$ where l_i^t and $l_i^{t\prime}$ are two different values of user i's data at time t. Temporal Privacy Leakage (TPL) of $\mathcal{M}^t$ w.r.t. a single $A_i^\mathcal{T}$ and all $A_i^\mathcal{T}, i \in \boldsymbol{U}$ are defined, respectively, as follows.*

$$TPL(A_i^\mathcal{T}, \mathcal{M}^t) \overset{\text{def}}{=\!=} \sup_{l_i^t, l_i^{t\prime}, \boldsymbol{r}^1, \ldots, \boldsymbol{r}^T} \log \frac{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^T | l_i^t, D_\mathcal{K}^t)}{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^T | l_i^{t\prime}, D_\mathcal{K}^t)}. \quad (2)$$

$$TPL(\mathcal{M}^t) \overset{\text{def}}{=\!=} \max_{\forall A_i^\mathcal{T}, i \in \boldsymbol{U}} TPL(A_i^\mathcal{T}, \mathcal{M}^t) \quad (3)$$

$$= \sup_{D^t, D^{t\prime}, \boldsymbol{r}^1, \ldots, \boldsymbol{r}^T} \log \frac{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^T | D^t)}{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^T | D^{t\prime})}. \quad (4)$$

We first analyze $TPL(A_i^\mathcal{T}, \mathcal{M}^t)$ (i.e., Equation (2)) because it is key to solve Equation (3) or (4).

Theorem 2. *We can rewrite $TPL(A_i^\mathcal{T}, \mathcal{M}^t)$ as follows.*

$$\underbrace{\sup_{\substack{\boldsymbol{r}^1, \ldots, \boldsymbol{r}^t, \\ l_i^t, l_i^{t\prime}}} \log \frac{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^t | l_i^t, D_\mathcal{K}^t)}{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^t | l_i^{t\prime}, D_\mathcal{K}^t)}}_{\textit{backward privacy leakage}} + \underbrace{\sup_{\substack{\boldsymbol{r}^t, \ldots, \boldsymbol{r}^T, \\ l_i^t, l_i^{t\prime}}} \log \frac{\Pr(\boldsymbol{r}^t, \ldots, \boldsymbol{r}^T | l_i^t, D_\mathcal{K}^t)}{\Pr(\boldsymbol{r}^t, \ldots, \boldsymbol{r}^T | l_i^{t\prime}, D_\mathcal{K}^t)}}_{\textit{forward privacy leakage}}$$

$$- \underbrace{\sup_{\boldsymbol{r}^t, l_i^t, l_i^{t\prime}} \log \frac{\Pr(\boldsymbol{r}^t | l_i^t, D_\mathcal{K}^t)}{\Pr(\boldsymbol{r}^t | l_i^{t\prime}, D_\mathcal{K}^t)}}_{PL_0(A_i^\mathcal{T}, \mathcal{M}^t)} \quad (5)$$

It is clear that $PL_0(A_i^\mathcal{T}, \mathcal{M}^t) = PL_0(A_i, \mathcal{M}^t)$ because PL_0 indicates the privacy leakage w.r.t. one output $\boldsymbol{r}$ (refer to Definition 2). As annotated in the above equation, we define backward and forward privacy leakage as follows.

Definition 6 (Backward Privacy Leakage, BPL)**.** *The privacy leakage of $\mathcal{M}^t$ caused by $\boldsymbol{r}^1, ..., \boldsymbol{r}^t$ w.r.t. $A_i^\mathcal{T}$ is called backward privacy leakage, defined as follows.*

$$BPL(A_i^\mathcal{T}, \mathcal{M}^t) \overset{\text{def}}{=\!=} \sup_{l_i^t, l_i^{t\prime}, \boldsymbol{r}^1, \ldots, \boldsymbol{r}^t} \log \frac{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^t | l_i^t, D_\mathcal{K}^t)}{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^t | l_i^{t\prime}, D_\mathcal{K}^t)}. \quad (6)$$

$$BPL(\mathcal{M}^t) \overset{\text{def}}{=\!=} \max_{\forall A_i^\mathcal{T}, i \in \boldsymbol{U}} BPL(A_i^\mathcal{T}, \mathcal{M}^t). \quad (7)$$

Definition 7 (Forward Privacy Leakage, FPL)**.** *The privacy leakage of $\mathcal{M}^t$ caused by $\boldsymbol{r}^t, ..., \boldsymbol{r}^T$ w.r.t. $A_i^\mathcal{T}$ is called forward privacy leakage, defined by follows.*

$$FPL(A_i^\mathcal{T}, \mathcal{M}^t) \overset{\text{def}}{=\!=} \sup_{l_i^t, l_i^{t\prime}, \boldsymbol{r}^t, \ldots, \boldsymbol{r}^T} \log \frac{\Pr(\boldsymbol{r}^t, \ldots, \boldsymbol{r}^T | l_i^t, D_\mathcal{K}^t)}{\Pr(\boldsymbol{r}^t, \ldots, \boldsymbol{r}^T | l_i^{t\prime}, D_\mathcal{K}^t)}. \quad (8)$$

$$FPL(\mathcal{M}^t) \overset{\text{def}}{=\!=} \max_{\forall A_i^\mathcal{T}, i \in \boldsymbol{U}} FPL(A_i^\mathcal{T}, \mathcal{M}^t). \quad (9)$$

By substituting Equation (6) and (8) into (5), we have

$$TPL(A_i^\mathcal{T}, \mathcal{M}^t) = BPL(A_i^\mathcal{T}, \mathcal{M}^t) + FPL(A_i^\mathcal{T}, \mathcal{M}^t) - PL_0(A_i^\mathcal{T}, \mathcal{M}^t). \quad (10)$$

Since the privacy leakage is considered as the worst case among all users in the database, by Equations (7) and (9), we have

$$TPL(\mathcal{M}^t) = BPL(\mathcal{M}^t) + FPL(\mathcal{M}^t) - PL_0(\mathcal{M}^t). \quad (11)$$

Intuitively, BPL, FPL and TPL are the privacy leakage w.r.t. the adversaries $A_i^\mathcal{T}(P_i^B)$, $A_i^\mathcal{T}(P_i^F)$ and $A_i^\mathcal{T}(P_i^B, P_i^F)$, respectively. In Equation (11), we need to minus $PL_0(\mathcal{M}^t)$ because it is counted in both BPL and FPL. In the following, we will dive into the analysis of BPL and FPL.

BPL over time. For BPL, we first expand and simplify Equation (6) by Bayesian theorem, $BPL(A_i^\mathcal{T}, \mathcal{M}^t)$ is equal to

$$\sup_{\substack{l_i^t, l_i^{t\prime}, \\ \boldsymbol{r}^1, \ldots, \boldsymbol{r}^{t-1}}} \log \frac{\sum_{l_i^{t-1}} \Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^{t-1} | l_i^{t-1}, D_\mathcal{K}^{t-1}) \Pr(l_i^{t-1} | l_i^t)}{\sum_{l_i^{t-1\prime}} \underbrace{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^{t-1} | l_i^{t-1\prime}, D_\mathcal{K}^{t-1})}_{\text{(i) } BPL(A_i^\mathcal{T}, \mathcal{M}^{t-1})} \underbrace{\Pr(l_i^{t-1\prime} | l_i^{t\prime})}_{\text{(ii) } P_i^B}}$$

$$+ \sup_{l_i^t, l_i^{t\prime}, \boldsymbol{r}^t} \log \underbrace{\frac{\Pr(\boldsymbol{r}^t | l_i^t, D_\mathcal{K}^t)}{\Pr(\boldsymbol{r}^t | l_i^{t\prime}, D_\mathcal{K}^t)}}_{\text{(iii) } PL_0(A_i^\mathcal{T}, \mathcal{M}^t)}. \quad (12)$$

We now discuss the three annotated terms in the above equation. The first term indicates BPL at the previous time $t-1$, the second term is the backward temporal correlation determined by P_i^B, and the third term is equal to the privacy leakage w.r.t. adversaries in traditional DP (see Definition 2). Hence, BPL at time t depends on (i) BPL at time $t-1$, (ii) the backward temporal correlations, and (iii) the (traditional) privacy leakage of $\mathcal{M}^t$ (which is related to the privacy budget allocated to $\mathcal{M}^t$). By Equation (12), we know that if $t = 1$, $BPL(A_i^\mathcal{T}, \mathcal{M}^1) = PL_0(A_i, \mathcal{M}^1)$; if $t > 1$, we have the following, where $\mathcal{L}^B(\cdot)$ is a *backward temporal privacy loss function* for calculating the accumulated privacy loss.

$$BPL(A_i^\mathcal{T}, \mathcal{M}^t) = \mathcal{L}^B\big(BPL(A_i^\mathcal{T}, \mathcal{M}^{t-1})\big) + PL_0(A_i, \mathcal{M}^t) \quad (13)$$

Equation (13) reveals that BPL is calculated recursively and may *accumulate over time*, as shown in Example 2 (Fig.3(a)).

Example 2 (**BPL due to previous releases**)**.** *Suppose that a DP mechanism $\mathcal{M}^t$ satisfies $PL_0(\mathcal{M}^t) = 0.1$ for each time $t \in [1, T]$, i.e., 0.1-DP at each time point. We now discuss*

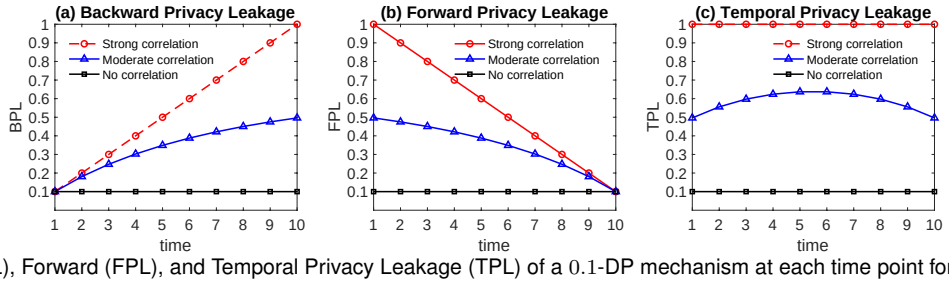


Fig. 3. Backward (BPL), Forward (FPL), and Temporal Privacy Leakage (TPL) of a 0.1-DP mechanism at each time point for Example 2 and 3.

BPL at each time point w.r.t. $A_i^{\mathcal{T}}$ with knowledge of backward temporal correlations P_i^B. In an extreme case, if P_i^B indicates the strongest correlation, say, $P_i^B = \left(\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right)$, then, at time t, $A_i^{\mathcal{T}}$ knows $l_i^t = l_i^{t-1} = \cdots = l_i^1$, i.e., $D^t = D^{t-1} = \cdots = D^1$ because of $D^t = \{l_i^t\} \cup D_{\mathcal{K}}^t$ for any $t \in [1, T]$. Hence, the continuous data release $\boldsymbol{r}^1, \ldots, \boldsymbol{r}^t$ is equivalent to releasing the same database multiple times; the privacy leakage at each time point will accumulate from previous time points and increase linearly (the red line with circle marker in Figure 3(a)). In another extreme case, if there is no backward temporal correlation that is known to $A_i^{\mathcal{T}}$ (e.g., for the A_i in Definition 2 or $A_i^{\mathcal{T}}(P_i^F)$), BPL at each time point is $PL_0(\mathcal{M}^t)$, as shown in Figure 3(a), the black line with rectangle marker. The blue line with triangle marker in Figure 3(a) depicts the backward privacy leakage caused by $P_i^B = \left(\begin{smallmatrix}0.8 & 0.2\\ 0 & 1\end{smallmatrix}\right)$, which can be finely quantified using our method (Algorithm 1) in Section 4.

FPL over time. For FPL, similar to the analysis of BPL, we expand and simplify Equation (6) by Bayesian theorem, $FPL(A_i^{\mathcal{T}}, \mathcal{M}^t)$ is equal to

$$\sup_{\substack{l_i^t, l_i^{t\prime},\\ \boldsymbol{r}^{t+1}, \ldots, \boldsymbol{r}^T}} \log \frac{\sum_{l_i^{t+1}} \Pr(\boldsymbol{r}^{t+1}, \ldots, \boldsymbol{r}^T | l_i^{t+1}, D_{\mathcal{K}}^{t+1}) \Pr(l_i^{t+1} | l_i^t)}{\sum_{l_i^{t+1\prime}} \underbrace{\Pr(\boldsymbol{r}^{t+1}, \ldots, \boldsymbol{r}^T | l_i^{t+1\prime}, D_{\mathcal{K}}^{t+1})}_{\text{(i) } FPL(A_i^{\mathcal{T}}, \mathcal{M}^{t+1})} \underbrace{\Pr(l_i^{t+1\prime} | l_i^{t\prime})}_{\text{(ii) } P_i^F}}$$
$$+ \sup_{l_i^t, l_i^{t\prime}, \boldsymbol{r}^t} \log \underbrace{\frac{\Pr(\boldsymbol{r}^t | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\boldsymbol{r}^t | l_i^{t\prime}, D_{\mathcal{K}}^t)}}_{\text{(iii) } PL_0(A_i^{\mathcal{T}}, \mathcal{M}^t)}. \tag{14}$$

By Equation (14), we know that if $t = T$, $FPL(A_i^{\mathcal{T}}, \mathcal{M}^T) = PL_0(A_i, \mathcal{M}^T)$; if $t < T$, we have the following, where $\mathcal{L}^F(\cdot)$ is a *forward temporal privacy loss function* for calculating the increased privacy loss due to FPL at the next time.

$$FPL(A_i^{\mathcal{T}}, \mathcal{M}^t) = \mathcal{L}^F\big(FPL(A_i^{\mathcal{T}}, \mathcal{M}^{t+1})\big) + PL_0(A_i, \mathcal{M}^t) \tag{15}$$

Equation (15) reveals that FPL is calculated recursively and may *increase over time*, as shown in Example 3 (Fig.3(b)).

Example 3 (FPL due to future releases). *Considering the same setting in Example 2, we now discuss FPL at each time point w.r.t. $A_i^{\mathcal{T}}$ with knowledge of forward temporal correlations P_i^F. In an extreme case, if P_i^F indicates the strongest correlation, say, $P_i^F = \left(\begin{smallmatrix}1 & 0\\ 0 & 1\end{smallmatrix}\right)$, then, at time t, $A_i^{\mathcal{T}}$ knows $l_i^t = l_i^{t+1} = \cdots = l_i^T$, i.e., $D^t = D^{t+1} = \cdots = D^T$ because of $D^t = \{l_i^t\} \cup D_{\mathcal{K}}^t$ for any $t \in [1, T]$. Hence, the continuous data release $\boldsymbol{r}^t, \ldots, \boldsymbol{r}^T$ is equivalent to releasing the same database multiple times; the privacy leakage at time t will increase when every time new release (i.e., $\boldsymbol{r}^{t+1}$,$\boldsymbol{r}^{t+2}$,...) happens, as shown in the red line with circle marker in Figure 3(b). We see that contrary to BPL, the FPL at time 1 is the highest (due to future releases at time 1 to 10) while FPL at time 10 is the lowest (since there is no future release with respect to time 10 yet). If $\boldsymbol{r}^{11}$ is released, all FPL at time $t \in [1, 10]$ will be updated. In another extreme case, if there is no forward temporal correlation that is known to $A_i^{\mathcal{T}}$ (e.g., for the A_i in Definition 2 or $A_i^{\mathcal{T}}(P_i^B)$), then the forward privacy leakage at each time point is $PL_0(\mathcal{M}^t)$, as shown in the black line with rectangle marker Figure 3(b). The blue line with triangle marker in Figure 3(b) depicts the forward privacy leakage caused by $P_i^F = \left(\begin{smallmatrix}0.8 & 0.2\\ 0 & 1\end{smallmatrix}\right)$, which can be finely quantified using our method (Algorithm 1) in Section 4.*

Remark 1. *The extreme cases shown in Examples 2 and 3 are the upper and lower bound of BPL and FPL. Hence, the temporal privacy loss functions $\mathcal{L}^B(\cdot)$ and $\mathcal{L}^F(\cdot)$ in Equations* (13) *and* (15) *satisfy $0 \leq \mathcal{L}^B(x) \leq x$, where x is BPL at the previous time, and $0 \leq \mathcal{L}^F(x) \leq x$, where x is FPL at the next time, respectively.*

From Examples 2 and 3, we know that: backward temporal correlation (i.e.,P_i^B) does not affect FPL, and forward temporal correlation (i.e.,P_i^F) does not affect BPL. In other words, adversary $A_i^{\mathcal{T}}(P_i^B)$ only causes BPL; $A_i^{\mathcal{T}}(P_i^F)$ only causes FPL; while $A_i^{\mathcal{T}}(P_i^B, P_i^F)$ poses a risk on both BPL and FPL. The composition of BPL and FPL is shown in Equations (10) and (11). Figure 3(c) shows TPL, which can be calculated using Equation (11).

3.3 α-DP$_{\mathcal{T}}$ and Its Composability

In this section, we define α-DP$_{\mathcal{T}}$ (differential privacy under temporal correlations) to provide a privacy guarantee against temporal privacy leakage. We prove its sequential composition theorem and discuss the connection between α-DP$_{\mathcal{T}}$ and ϵ-DP in terms of event-level/user-level privacy [4] [5] and w-event privacy [7].

Definition 8 (α-$DP_{\mathcal{T}}$, differential privacy under temporal correlations)**.** *If TPL of a differentially private mechanism is less than or equal to α, we say that such mechanism satisfies α-Differential Privacy under Temporal correlation, i.e., α-$DP_{\mathcal{T}}$.*

DP$_{\mathcal{T}}$ is an enhanced version of differential privacy on time-series data. If the data are temporally independent (i.e., for all user i, both P_i^B and P_i^F are $\emptyset$), an ϵ-DP mechanism satisfies ϵ-DP$_{\mathcal{T}}$. If the data are temporally correlated (i.e., existing user i whose P_i^B or P_i^F is not $\emptyset$), an ϵ-DP mechanism satisfies α-DP$_{\mathcal{T}}$ where α is the increased privacy leakage and can be quantified in our framework.

One may wonder, for a sequence of DP$_{\mathcal{T}}$ mechanisms on the timeline, what is the overall privacy guarantee. In the following, we suppose that $\mathcal{M}^t$ is a ϵ_t-DP mechanism at time $t \in [1, T]$ and poses risks of BPL and FPL as α_t^B and α_t^F, respectively. That is, $\mathcal{M}^t$ satisfies $(\alpha_t^B + \alpha_t^F - \epsilon_t)$-DP$_{\mathcal{T}}$ at time t according to Equation (11). Similar to definition of TPL w.r.t. a DP mechanism at a single time point, we define TPL of a sequence of DP mechanisms at consecutive time points as follows.

Definition 9 (TPL of a sequence of DP mechanisms)**.** *The temporal privacy leakage of DP mechanisms $\{\mathcal{M}^t, \ldots, \mathcal{M}^{t+j}\}$ where $j \geq 0$ is defined as follows.*

$$TPL(\{\mathcal{M}^t, \ldots, \mathcal{M}^{t+j}\}) \stackrel{\text{def}}{=} \sup_{\substack{D^t, \ldots, D^{t+j},\\ D^{t\prime}, \ldots, D^{t+j\prime},\\ \boldsymbol{r}^1, \ldots, \boldsymbol{r}^T}} \log \frac{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^T | D^t, \ldots, D^{t+j})}{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^T | D^{t\prime}, \ldots, D^{t+j\prime})}$$

It is easy to see that, if $j = 0$, it is event-level privacy; if $t = 1$ and $j = T - 1$, it is user-level privacy.

Theorem 3 (Sequential Composition under Temporal Correlations)**.** *A sequence of DP mechanisms* $\{\mathcal{M}^t, \ldots, \mathcal{M}^{t+j}\}$ *satisfies*

$$\begin{cases} (\alpha_t^B + \alpha_{t+1}^F)\text{-}DP_{\mathcal{T}} & j = 1 \\ (\alpha_t^B + \alpha_{t+j}^F + \sum_{k=t+1}^{k=t+j-1} \epsilon_k)\text{-}DP_{\mathcal{T}} & j \geq 2 \end{cases} \quad (16)$$

In Theorem 3, when $t = 1$ and $j = T - 1$, we have Corollary 1.

Corollary 1. *The temporal privacy leakage of a combined mechanism* $\{\mathcal{M}^1, \ldots, \mathcal{M}^T\}$ *is* $\sum_{k=1}^{k=T} \epsilon_k$ *where* ϵ_k *is the privacy budget of* $\mathcal{M}^k$*,* $k \in [1, T]$*.*

It shows that *temporal correlations do not affect the user-level privacy* (i.e., protecting all the data on the timeline of each user), which is in line with the idea of group differential privacy: protecting all the correlated data in a bundle.

Comparison between DP and DP$_\mathcal{T}$. As we mentioned in Section 2.3, there are typically two privacy notions in continuous data release: *event-level* and *user-level* [4] [5]. Recently, w-event privacy [7] is proposed to merge the gap between event-level and user-level privacy. It protects the data in any w-length sliding window by utilizing the sequential composition theorem of DP.

Theorem 4 (Sequential composition on independent data [15])**.** *Suppose that* $\mathcal{M}^t$ *satisfies* ϵ_t*-DP for each* $t \in [1, T]$*. A combined mechanism* $\{\mathcal{M}^t, \ldots, \mathcal{M}^{t+j}\}$ *satisfies* $\sum_{k=t}^{k=t+j} \epsilon_k$*-DP.*

For ease of exposition, suppose that $\mathcal{M}^t$ satisfies ϵ-DP for each $t \in [1, T]$. According to Theorem 4, it achieves $T\epsilon$-DP on user-level and $w\epsilon$-DP on w-event level. We compare the privacy guarantee of $\mathcal{M}^t$ on independent data and temporally correlated data in Table 2.

TABLE 2
The privacy guarantee of ϵ-DP mechanisms on different types of data.

Privacy Notion \ Data	independent	temporally correlated
event-level [4] [5]	ϵ-DP	α-DP$_\mathcal{T}$ ($\epsilon \leq \alpha \leq T\epsilon$)
w-event [7]	$w\epsilon$-DP	see Theorem 3
user-level [4] [5]	$T\epsilon$-DP	$T\epsilon$-DP$_\mathcal{T}$ (by Corollary 1)

It reveals that *temporal correlations may blur the boundary between event-level privacy and user-level privacy*. As shown in Table 2, the privacy leakage of a ϵ-DP mechanism at a single time point (event-level) on temporally correlated data may range from ϵ to $T\epsilon$ which depends on the strength of temporal correlation. For example, as shown in Figure 3(c), TPL of a 0.1-DP mechanism under strong temporal correlation at each time point (event-level) is $T * 0.1$, which is equal to the privacy leakage of a sequence of 0.1-DP mechanisms that satisfies user-level privacy. When the temporal correlations is moderate, TPL of a 0.1-DP mechanism at each time point (event-level) is less than $T * 0.1$ but still larger than 0.1, as shown in the blue line with triangle markers in Figure 3(c).

3.4 Connection with Pufferfish Framework

α-DP$_\mathcal{T}$ is highly related to Pufferfish framework [16] [17] and its instantiation [14]. Pufferfish provides rigorous and customizable privacy framework which consists of three components: a set of secrets, a set of discriminative pairs and data evolution scenarios (how the data were generated, or how the data are correlated). Note that the data evolution scenarios is essentially the adversary's background knowledge about data, such as data correlations. In Pufferfish framework [16] [17], they prove that when secrets to be all possible values of tuples, discriminative pairs to be the set of all pairs of potential secrets, and tuples to be independent, it is equivalent to DP; hence, Pufferfish becomes $DP_\mathcal{T}$ under the above setting of secrets and discriminative pairs, but with temporally correlated tuples.

A recent work [14] proposes Markov Quilt Mechanism for Pufferfish framework when data correlations can be modeled by Bayesian Network (BN). They further design efficient mechanisms MQMExact and MQMApprox when the structure of BN is Markov Chain. Although we also use Markov Chain as temporal correlation model, the settings of two studies are essentially different. In the motivating example of [14], i.e., Physical Activity Monitoring, the private histogram is "one-shot" (see Section 2.3) release: adapting to our setting in Figure 1, their example is equivalent to release location access statistics for a one-user database only at a given time T. Whereas, we focus on quantifying the privacy leakage in continuous data release. One important insight of our study is that, *the privacy guarantee of one-shot data release is not on its own or static; instead, it may be affected by previous release and even future release*, which are defined as Backward and Forward Privacy Leakage in our work.

3.5 Discussion

We make a few important observations regarding our privacy analysis. First, the temporal privacy leakage is defined in a personalized way. That is, the privacy leakage may be different for users with distinct temporal patterns (i.e., P_i^B and P_i^F). We define the overall temporal privacy leakage as the maximum one for all users, so that α-DP$_\mathcal{T}$ is compatible with the traditional ϵ-DP mechanisms (using one parameter to represent the overall privacy level) and we can convert them to protect against TPL. On the other hand, our definitions is also compatible with personalized differential privacy (PDP) mechanisms [18], in which the personalized privacy budgets, i.e., a vector $[\epsilon_1, \ldots, \epsilon_n]$, are allocated to each user. In other words, we can convert a PDP mechanism to bound the temporal privacy leakage for each user.

Second, in this paper, we focus on the temporally correlated data and assume that the adversary has knowledge of temporal correlations modeled by Markov chain. However, it is possible that the adversary has knowledge about more sophisticated temporal correlation model or other types of correlations. Especially, if the the assumption about Markov model is not the "ground truth". We may not protect against TPL appropriately. Song et al. [14] provided a formalization of the upper bound of privacy leakage if a set of possible data correlations Θ is given, which is *max-divergence* between any two conditional distributions of $\tilde{\theta} \in \Theta$ and $\theta \in \Theta$ given any secret that we want to protect. In their Markov Quilt Mechanism that models correlation using Bayesian Network (BN), the privacy leakage can be represented by *max-influence* between nodes (tuples). Hence, given a set of BNs as possible data correlations, the upper bound of privacy leakage is the largest max-influence under different

BNs. Similarly, in our case, given a set of Transition Matrices (TM), we can calculate the maximum TPL w.r.t. different TMs. However, it remains an open question how to find a set of possible data correlations Θ that includes the adversarial knowledge or the ground truth in a high probability, which may depend on the application scenarios. We believe that this question is also related to "how to identify appropriate concepts of neighboring databases in different scenarios" since properly defined neighboring databases can circumvent the affect of data correlations on privacy leakage in a certain extent (as we show in Section 3.3, the difference between event-level privacy, w-event privacy, and user-level privacy).

4 Calculating Temporal Privacy Leakage

In this section, we design algorithms for computing backward privacy leakage (BPL) and forward privacy leakage (FPL). We first show that both of them can be transformed to the optimal solution of a *linear-fractional programming* problem [19] in Section 4.1. Traditionally, this type of problem can be solved by simplex algorithm in exponential time [19]. By exploiting the constraints in this problem, we then design a polynomial algorithm to calculate it in Section 4.2. Further, based on our observation of some repeated computation when the polynomial algorithm runs on different time points, we precompute such common results and design quasi-quadratic and sub-linear algorithms for calculating temporal privacy leakage at each time point in Section 4.3 and Section 4.4, respectively.

4.1 Problem formulation

According to the privacy analysis of BPL and FPL in Section 3.2, we need to solve the backward and forward temporal privacy loss functions $\mathcal{L}^B(\cdot)$ and $\mathcal{L}^F(\cdot)$ in Equations (13) and (15), respectively. By observing the structure of the first term in Equations (12) and (14), we can see that the calculations for recursive functions $\mathcal{L}^B(\cdot)$ and $\mathcal{L}^F(\cdot)$ are virtually in the same way. They calculate the increment of the input values (the previous BPL or the next FPL) based on temporal correlations (backward or forward). Although different degree of correlations result in different privacy loss functions, the methods for analyzing them are the same.

We now quantitatively analyze the temporal privacy leakage. In the following, we demonstrate the calculation of $\mathcal{L}^B(\cdot)$, i.e., the first term of Equation (12).

$$\sup_{\substack{l_i^t, l_i^{t'}, \\ \boldsymbol{r}^1,\ldots,\boldsymbol{r}^{t-1}}} \log \frac{\sum_{l_i^{t-1}} \Pr(\boldsymbol{r}^1,\ldots,\boldsymbol{r}^{t-1}|l_i^{t-1}, D_{\mathcal{K}}^{t-1}) \Pr(l_i^{t-1}|l_i^t)}{\sum_{l_i^{t-1\prime}} \underbrace{\Pr(\boldsymbol{r}^1,\ldots,\boldsymbol{r}^{t-1}|l_i^{t-1\prime}, D_{\mathcal{K}}^{t-1})}_{BPL(A_i^{\mathcal{T}}, \mathcal{M}^{t-1})} \underbrace{\Pr(l_i^{t-1\prime}|l_i^{t\prime})}_{P_i^B}} \tag{17}$$

We now simplify the notations in the above formula. Let two arbitrary (distinct) rows in P_i^B be vectors $\boldsymbol{q} = (q_1, ..., q_n)$ and $\boldsymbol{d} = (d_1, ..., d_n)$. For example, suppose that $\boldsymbol{q}$ is the first row in the transition matrix of Figure 2(b); then, the elements in $\boldsymbol{q}$ are: $q_1 = \Pr(l_i^{t-1} = loc_1 | l_i^t = loc_1)$, $q_2 = \Pr(l_i^{t-1} = loc_2 | l_i^t = loc_1)$, $q_3 = \Pr(l_i^{t-1} = loc_3 | l_i^t = loc_1)$, etc. Let $\boldsymbol{x} = (x_1, ..., x_n)^{\mathrm{T}}$ be a vector whose elements indicate $\Pr(\boldsymbol{r}^1, ..., \boldsymbol{r}^{t-1} | l_i^{t-1}, D_{\mathcal{K}}^{t-1})$ with distinct values of $l_i^{t-1} \in \boldsymbol{loc}$, e.g., x_1 denotes $\Pr(\boldsymbol{r}^1, ..., \boldsymbol{r}^{t-1} | l_i^{t-1} = loc_1, D_{\mathcal{K}}^{t-1})$. We obtain the following by expanding $l_i^{t-1}, l_i^{t-1\prime} \in \boldsymbol{loc}$ in Equation (17).

$$\begin{aligned} \mathcal{L}^B\big(BPL(A_i^{\mathcal{T}}, \mathcal{M}^{t-1})\big) &= \sup_{\boldsymbol{q},\boldsymbol{d} \in P_i^B} \log \frac{q_1 x_1 + \cdots + q_n x_n}{d_1 x_1 + \cdots + d_n x_n} \\ &= \sup_{\boldsymbol{q},\boldsymbol{d} \in P_i^B} \log \frac{\boldsymbol{qx}}{\boldsymbol{dx}} \end{aligned} \tag{18}$$

Next, we formalize the objective function and constraints. Suppose that $BPL(A_i^{\mathcal{T}}, \mathcal{M}^{t-1}) = \alpha_{t-1}^B$. According to the definition of BPL (as the supremum), for any $x_j, x_k \in \boldsymbol{x}$, we have $e^{-\alpha_{t-1}^B} \leq \frac{x_j}{x_k} \leq e^{\alpha_{t-1}^B}$. Given $\boldsymbol{x}$ as the variable vector and $\boldsymbol{q}, \boldsymbol{d}$ as the coefficient vectors, $\mathcal{L}^B(\alpha_{t-1}^B)$ is equal to the logarithm of the objective function (19) in the following maximization problem.

$$\text{maximize} \quad \frac{\boldsymbol{qx}}{\boldsymbol{dx}} \tag{19}$$

$$\text{subject to} \quad e^{-\alpha_{t-1}^B} \leq \frac{x_j}{x_k} \leq e^{\alpha_{t-1}^B}, \tag{20}$$

$$0 < x_j < 1 \text{ and } 0 < x_k < 1, \quad \text{where } x_j, x_k \in \boldsymbol{x},\ j, k \in [1, n]. \tag{21}$$

The above is a form of *Linear-Fractional Programming* [19] (i.e., LFP), where the objective function is a ratio of two linear functions and the constraints are linear inequalities or equations. A linear-fractional programming problem can be converted into a sequence of linear programming problems and then solved using the simplex algorithm in time $O(2^n)$ [19]. According to Equation (18), given P_i^B as an $n \times n$ matrix, finding $\mathcal{L}^B(\cdot)$ involves solving $n(n-1)$ such LFP problems w.r.t. different permutation of choosing two rows $\boldsymbol{q}$ and $\boldsymbol{d}$ from the transition matrix P_i^B. Hence, the overal time complexity using simplex algorithm is $O(n^2 2^n)$.

As we mentioned previously, the calculations of $\mathcal{L}^B(\cdot)$ and $\mathcal{L}^F(\cdot)$ are identical. For simplicity, in the following part of this paper, we use $\mathcal{L}(\cdot)$ to represent the privacy loss function $\mathcal{L}^B(\cdot)$ or $\mathcal{L}^F(\cdot)$, use α to denote α_{t-1}^B or α_{t+1}^F, and use P_i to denote P_i^B or P_i^F.

4.2 Polynomial Algorithm

In this section, we design an efficient algorithm to calculate TPL, i.e., to solve the linear-fractional program in Equations (19) to (21). Intuitively, we prove that the optimal solutions always satisfy some conditions (Theorem 5), which enable us to design an efficient algorithm to obtain the optimal value without directly solving the optimization problem.

Properties of the optimal solutions. From Inequalities (20) and (21), we know that the feasible region of the constraints are not empty and bounded; hence, the optimal solution exists. We prove Theorem 5, which enables the optimal solution to be found in time $O(n^2)$.

We first define some notations that will be frequently used in our theorems. Suppose that the variable vector $\boldsymbol{x}$ consists of two parts (subsets): $\boldsymbol{x}^+$ and $\boldsymbol{x}^-$. Let the corresponding coefficients vectors be $\boldsymbol{q}^+, \boldsymbol{d}^+$ and $\boldsymbol{q}^-, \boldsymbol{d}^-$. Let $q = \sum \boldsymbol{q}^+$ and $d = \sum \boldsymbol{d}^+$. For example, suppose that $\boldsymbol{x}^+ = [x_1, x_3]$ and $\boldsymbol{x}^- = [x_2, x_4, x_5]$. Then, we have $\boldsymbol{q}^+ = [q_1, q_3]$, $\boldsymbol{d}^+ = [d_1, d_3]$, $\boldsymbol{q}^- = [q_2, q_4, q_5]$, and $\boldsymbol{d}^- = [d_2, d_4, q_5]$. In this case, $q = q_1 + q_3$ and $d = d_1 + d_3$.

Theorem 5. *If the following Inequalities* (22) *and* (23) *are satisfied, the maximum value of the objective function in the problem* (19)∼(21) *is* $\frac{q(e^{\alpha_{t-1}^B} - 1) + 1}{d(e^{\alpha_{t-1}^B} - 1) + 1}$.

$$\frac{q_j}{d_j} > \frac{q(e^{\alpha^B_{t-1}}-1)+1}{d(e^{\alpha^B_{t-1}}-1)+1}, \quad \forall j \in [1,n] \text{ where } q_j \in \boldsymbol{q}^+, d_j \in \boldsymbol{d}^+ \quad (22)$$

$$\frac{q_k}{d_k} \leq \frac{q(e^{\alpha^B_{t-1}}-1)+1}{d(e^{\alpha^B_{t-1}}-1)+1}, \quad \forall k \in [1,n] \text{ where } q_k \in \boldsymbol{q}^-, d_k \in \boldsymbol{d}^- \quad (23)$$

According to Equation (18), the increment of temporal privacy loss, i.e., $\mathcal{L}(\cdot)$, is the maximum value among the $n(n-1)$ LFP problems, which are defined by different 2-permutations $\boldsymbol{q}$ and $\boldsymbol{d}$ chosen from n rows of P_i.

$$\mathcal{L}(\alpha) = \max_{\boldsymbol{q},\boldsymbol{d} \in P_i} \log \frac{q(e^\alpha - 1)+1}{d(e^\alpha - 1)+1} \quad (24)$$

Further, we give Corollary 2 for finding $\boldsymbol{q}^+$ and $\boldsymbol{d}^+$.

Corollary 2. *If Inequalities* (22) *and* (23) *are satisfied, we have* $q_j > d_j$ *in which* $q_j \in \boldsymbol{q}^+$ *and* $d_j \in \boldsymbol{d}^+$.

Now, the question is how do we find q and d (or, $\boldsymbol{q}^+$ and $\boldsymbol{d}^+$) in Theorem 5 that give the maximum value of objective function. Inequalities (22) and (23) are sufficient conditions for obtaining such optimal value. Corollary 2 gives a necessary condition for satisfying Inequalities (22) and (23). Based on the above analysis, we design Algorithm 1 for computing BPL or FPL.

Algorithm 1: Calculate BPL or FPL by Theorem 5.

Input: P_i (i.e., P_i^B or P_i^F); α (i.e., α^B_{t-1} or α^F_{t+1}); ϵ_t (i.e., $PL_0(\mathcal{M}^t)$).
Output: BPL or FPL at time t

1 $\mathcal{L} = 0$; //the value of Equation (24).
2 **foreach** *two rows permutation* $\boldsymbol{q}, \boldsymbol{d}$ *from* n *rows in* P_i **do**
3 **foreach** $q_j \in \boldsymbol{q}, d_j \in \boldsymbol{d}$ **do** //Corollary 2.
4 **if** $q_j > d_j$ **then** add q_j to $\boldsymbol{q}^+$; add d_j to $\boldsymbol{d}^+$;
5 *update* $= false$;
6 **do** //find $\boldsymbol{q}^+$, $\boldsymbol{d}^+$ by Theorem 5.
7 $q = \sum \boldsymbol{q}^+$; $d = \sum \boldsymbol{d}^+$; //update q and d.
8 **foreach** $q_j \in \boldsymbol{q}^+, d_j \in \boldsymbol{d}^+$ **do**
 //if it does not satisfy Inequality (22).
9 **if** $q_j/d_j \leq (q*(e^\alpha-1)+1)/(d*(e^\alpha-1)+1)$
10 **then** $\boldsymbol{q}^+ \leftarrow \boldsymbol{q}^+ - q_j$; $\boldsymbol{d}^+ \leftarrow \boldsymbol{d}^+ - d_j$; *update* $= true$;
11 **while** *update*
12 **if** $\mathcal{L} < \log \frac{q*(e^\alpha-1)+1}{d*(e^\alpha-1)+1}$ **then** $\mathcal{L} = \log \frac{q*(e^\alpha-1)+1}{d*(e^\alpha-1)+1}$;
13 **return** $\mathcal{L} + \epsilon_t$ //by Equation (13) or (15).

Algorithm design. According to the definition of BPL and FPL, we need to find the maximum privacy leakage (Line 12) w.r.t. any 2-permutations selected from n rows of the given transition matrix P_i (Line 2). Lines 3∼11 are to solve a single linear-fractional programming problem w.r.t two specific rows chosen from P_i. In Lines 3 and 4, we divide the variable vector $\boldsymbol{x}$ into two parts according to Corollary 2, which gives the necessary condition for finding the maximum solution: if a pair of coefficients with the same subscript $q_j \leq d_j$, they are *not* in $\boldsymbol{q}^+$ and $\boldsymbol{d}^+$ that satisfy Inequalities (22) and (23). In other words, if $q_j > d_j$, they are "candidates" in $\boldsymbol{q}^+$ and $\boldsymbol{d}^+$ that gives the maximum objective function. In Lines 5∼11, we check the candidates $\boldsymbol{q}^+$ and $\boldsymbol{d}^+$ whether or not satisfying Inequalities (22) and (23). In Line 7, we update the values of q and d because they may be changed in the context. In Lines 8∼10, we check each bit in $\boldsymbol{q}^+$ and $\boldsymbol{d}^+$ whether or not satisfying Inequality (22). If any bit is removed, we set a flag updated to $true$ and do the loop again until every pairs of $\boldsymbol{q}^+$ and $\boldsymbol{d}^+$ satisfy Inequality (22).

A subtle question may arise regarding such "update". In Lines 8∼10, the algorithm may remove *several* pairs of q_j and d_j, say, $\{q_1, d_1\}$ and $\{q_2, d_2\}$, that do not satisfy Inequality (22) in one loop. However, one may wonder if it is possible that, after removing $\{q_1, d_1\}$ from $\boldsymbol{q}^+$ and $\boldsymbol{d}^+$, Inequality (22) can be satisfied for $\{q_2, d_2\}$ due to the update of q and d, i.e., $\frac{q_2}{d_2} > \frac{(q-q_1)*(e^\alpha-1)+1}{(d-d1)*(e^\alpha-1)+1}$. We show that this is impossible. If $\frac{q_1}{d_1} \leq \frac{q*(e^\alpha-1)+1}{d*(e^\alpha-1)+1}$, we have $\frac{q*(e^\alpha-1)+1}{d*(e^\alpha-1)+1} \leq \frac{(q-q_1)*(e^\alpha-1)+1}{(d-d1)*(e^\alpha-1)+1}$. Hence, $\frac{q_2}{d_2} \leq \frac{q*(e^\alpha-1)+1}{d*(e^\alpha-1)+1} \leq \frac{(q-q_1)*(e^\alpha-1)+1}{(d-d1)*(e^\alpha-1)+1}$. Therefore, we can remove multiple pairs of q_j and d_j that do not satisfy Inequality (22) at one time.

Theorems 6 and 7 provide insights on transition matrices that lead to the extreme cases of temporal privacy leakage, which are in accordance with Remark 1.

Theorem 6. *If for any two rows* $\boldsymbol{q}$ *and* $\boldsymbol{d}$ *chosen from* P_i *that satisfy* $q_i = d_i$ *for* $i \in [1,n]$, *we have* $\mathcal{L}(\cdot) = 0$.

Theorem 7. *If there exist two rows* $\boldsymbol{q}$ *and* $\boldsymbol{d}$ *in* P_i *that satisfy* $q_i = 1$ *and* $d_i = 0$ *for a certain index* i, $\mathcal{L}(\cdot)$ *in an identical function, i.e,* $\mathcal{L}(x) = x$.

Complexity. The time complexity for solving one linear-fractional programming problem (Lines 3∼11) w.r.t. two specific rows of the transition matrix is $O(n^2)$ because Line 9 may iterate $n(n-1)$ times in the worst case. The overall time complexity of Algorithm 1 is $O(n^4)$ since there are $n(n-1)$ permutations of different pairs of $\boldsymbol{q}$ and $\boldsymbol{d}$.

4.3 Quasi-quadratic Algorithm

When Algorithm 1 runs on different time points for continuous data release, we have a constant P_i and different α as inputs at each time point. Our observation is that, there may exist some common computations when Algorithm 1 takes different inputs of α. More specifically, we want to know that, for given $\boldsymbol{q}$ and $\boldsymbol{d}$ which are two rows chosen from P_i, whether or not the final q and d (when stopping update in Line 11 in Algorithm 1) keep the same for any input of α, so that we can precompute q and d and do not need Lines 5∼11 at next run with a different α. Since $\boldsymbol{q}, \boldsymbol{d}$ and α indicate a specific LFP problem in Equation (19)∼(21), we attempt to directly obtain the q and p in Theorem 5. Unfortunately, we find that such q and d do not keep the same for different α w.r.t. given $\boldsymbol{q}$ and $\boldsymbol{d}$. However, interestingly we find that, for any input α, there are only several possible pairs of q and d when the update in Lines 6∼11 of Algorithm 1 is terminated, as shown in Theorem 8.

Theorem 8. *Let* $\boldsymbol{q}$ *and* $\boldsymbol{d}$ *be two vectors drawn from rows of a transition matrix. Assume* $q_i \neq d_i, i \in [1,n]$, *for each* $q_i \in \boldsymbol{q}$ *and* $d_i \in \boldsymbol{d}$.[2] *Without loss of generality, we assume* $\frac{q_1}{d_1} > \cdots > \frac{q_k}{d_k} > 1 \geq \frac{q_{k+1}}{d_{k+1}} > \cdots > \frac{q_n}{d_n}$, *then there are only* k *pairs of* q *and* d *that satisfy Inequalities* (22) *and* (23) *for given* $\boldsymbol{q}$ *and* $\boldsymbol{d}$.

case 1: $\frac{q_k}{d_k} > \frac{q(e^\alpha-1)+1}{d(e^\alpha-1)+1} \geq 1$ *where* $q = \sum_{i=1}^k q_i, d = \sum_{i=1}^k d_i$;

case 2: $\frac{q_{k-1}}{d_{k-1}} > \frac{q(e^\alpha-1)+1}{d(e^\alpha-1)+1} \geq \frac{q_k}{d_k}$ *where* $q = \sum_{i=1}^{k-1} q_i, d = \sum_{i=1}^{k-1} d_i$;

$\vdots \qquad \vdots \qquad \vdots$

case k: $\frac{q_1}{d_1} > \frac{q(e^\alpha-1)+1}{d(e^\alpha-1)+1} \geq \frac{q_2}{d_2}$ *where* $q = q_1, d = d_1$.

2. The case of $q_i = d_i$ is proven in Theorem 6.

More interestingly, we find that the values of q and d are monotonically decreasing with the increase of α. That is, when α is increasing from 0 to ∞, the pairs of q and d is transiting from case 1 to case 2,..., until to case k. In other words, the final q and d is constant in a certain range of α. Hence, the mapping from a given α to the optimal solution of a LFP problem w.r.t. $\boldsymbol{q}$ and $\boldsymbol{d}$ can be represented by a piecewise function as shown in Theorem 9.

Theorem 9. *We can represent the function of the optimal solution of LPF problem w.r.t. given $\boldsymbol{q}$ and $\boldsymbol{d}$ by a piecewise function as follows, where q_i and d_i are the i-th elements of $\boldsymbol{q}$ and $\boldsymbol{d}$, respectively; and $\alpha_j = \log\left(\frac{q_{k-j+1}-d_{k-j+1}}{q^{k-j+1}*d_{k-j+1}-d^{k-j+1}*q_{k-j+1}}+1\right)$ for $j \in [1, k-1]$. We call $\alpha_1, ..., \alpha_{k-1}$ in the sub-domains as transition points; $q^1, \cdots, q^k$ and $d^1, \cdots, d^k$ as coefficients of the piecewise function.*

$$f_{\boldsymbol{q},\boldsymbol{d}}(\alpha) = \frac{q(e^\alpha - 1)+1}{d(e^\alpha - 1)+1} \text{ where}$$

$$q = \begin{cases} q^k = \sum_{i=1}^{k} q_i \\ q^{k-1} = \sum_{i=1}^{k-1} q_i \\ \vdots \\ q^1 = q_1. \end{cases} \quad d = \begin{cases} d^k = \sum_{i=1}^{k} d_i & 0 \le \alpha < \alpha_1; \\ d^{k-1} = \sum_{i=1}^{k-1} d_i & \alpha_1 \le \alpha < \alpha_2; \\ \vdots & \vdots \\ d^1 = d_1 & \alpha_{k-1} \le \alpha. \end{cases} \tag{25}$$

For convenience, we let $qArr = [q^1, ..., q^k]$, $dArr = [d^1, ..., d^k]$, and $aArr = [\alpha_{k-1}, ..., 0]$. Then, *qArr, dArr, aArr* determine a piecewise function in Equation (25). Also, we let qM, dM and aM be $n(n-1) \times n$ matrices in which rows are *qArr*, *dArr* and *aArr* w.r.t. distinct 2-permutations of $\boldsymbol{q}$ and $\boldsymbol{d}$ from n rows of P_i. In other words, the three matrices determine $n(n-1)$ piecewise functions.

In the following, we first design Algorithm 2 to obtain qM, dM and aM. We then design Algorithm 3 to utilize the precomputed qM, dM, aM for calculating backward or forward temporal privacy leakage at each time point.

Algorithm 2: Precompute the parameters.

Input: P_i (i.e., P_i^B or P_i^F)
Output: matrices *qM; dM; aM.*
1 **foreach** *two rows permutation* $\boldsymbol{q}$, $\boldsymbol{d}$ *from* n *rows in* P_i **do**
2 **foreach** $q_j \in \boldsymbol{q}, d_j \in \boldsymbol{d}$ **do**
3 **if** $q_j \le d_j$ **then** $q_j = 0; d_j = 0;$
4 **if** $\boldsymbol{q}$ *and* $\boldsymbol{d}$ *are* 0 **then** `//when Theorem 6 is true.`
5 qArr =$[0]^n$; dArr =$[0]^n$; aArr =$[0]^n$;
6 **else**
7 Permutate $\boldsymbol{q}$ and $\boldsymbol{d}$ in the same way that makes $\frac{q_1}{d_1} > \cdots > \frac{q_k}{d_k}$ where $q_1, ..., q_k$ are larger than 0;
8 Let $qArr = [q_1, ..., \sum_{i=1}^{n} q_i]$;
9 Let $dArr = [d_1, ..., \sum_{i=1}^{n} d_i]$;
10 **foreach** $i \in [1, n]$ **do**
11 aArr[i] = $\log \frac{q_i - d_i}{qArr[i]*d_i - dArr[i]*q_i}$;
12 Append *qArr, dArr, aArr* into new rows of *qM, dM ,aM*, respectively;
13 **return** *qM, dM, aM*

We now use an example of $\boldsymbol{q} = [0.2, 0.3, 0.5]$ and $\boldsymbol{d} = [0.1, 0, 0.9]$ to demonstrate two notable points in Algorithm 2. First, with such $\boldsymbol{q}$ and $\boldsymbol{d}$, Line 7 results in $\boldsymbol{q} = [0.3, 0.2, 0]$ and $\boldsymbol{q} = [0, 0.1, 0]$ since only $q_1 = 0.3 > 0$ and $q_2 = 0.2 > 0$. Second, after the operation of Lines 10 and 11, we have *aArr*= [Inf,1.47,NaN].

Algorithm 2 needs $O(n^3)$ time for precomputing the parameters *qM*, *dM* and *aM*, which only needs to be run one time and can be done offline. Algorithm 3 for calculating privacy leakage at each time point needs $O(n^2 \log n)$ time.

Algorithm 3: Calculate BPL or FPL by Precomputation.

Input: α (i.e., α_{t-1}^B or α_{t+1}^F); qM; dM; aM; ϵ_t (i.e., $PL_0(\mathcal{M}^t)$).
Output: BPL or FPL at time t
1 $\mathcal{L} = 0$;
2 **foreach** $j \in [1, n(n-1)]$ **do**
3 *qArr, dArr, aArr* as j-th rows in *qM, dM ,aM*, respectively;
4 Binary Search *aArr*[k] that *aArr*[k] >$\alpha \ge$ *aArr*[k+1];
5 PL = $\log \frac{qArr[k](e^\alpha - 1)+1}{qArr[k](e^\alpha - 1)+1}$;
6 **if** *PL*>$\mathcal{L}$ **then** $\mathcal{L}$ = PL;
7 **return** $\mathcal{L} + \epsilon_t$

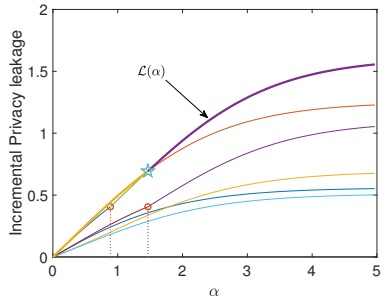


Fig. 4. Illustration of function $\mathcal{L}(\alpha)$ w.r.t. a 3×3 transition matrix in Example 4. Each lines is $f_{\boldsymbol{q},\boldsymbol{d}}(\alpha)$ w.r.t. distinct pairs of $\boldsymbol{q}$ and $\boldsymbol{d}$. The bold line is $\mathcal{L}(\alpha)$. The circle marks are transition points.

4.4 Sub-linear Algorithm

In this section, we further design a sub-linear privacy leakage quantification algorithm by investigating how to generate a function of $\mathcal{L}(\alpha)$, so that, given an arbitrary α, we can directly calculate the privacy loss.

Corollary 3. *Temporal privacy loss function $\mathcal{L}(\alpha)$ can be represented as a piecewise function:* $\max_{\boldsymbol{q},\boldsymbol{d} \in P_i} \log f_{\boldsymbol{q},\boldsymbol{d}}(\alpha)$.

Example 4. *Figure 4 shows the $\mathcal{L}(\alpha)$ function w.r.t. $\left(\begin{smallmatrix} 0.1 & 0.2 & 0.7 \\ 0.3 & 0.3 & 0.4 \\ 0.5 & 0.3 & 0.2 \end{smallmatrix}\right)$. Each line represents a piecewise function $f_{\boldsymbol{q},\boldsymbol{d}}(\alpha)$ w.r.t. distinct pairs of $\boldsymbol{q}$ and $\boldsymbol{d}$ chosen from P_i. Accoding to the definition of BPL and FPL, $\mathcal{L}(\alpha)$ function is a piecewise function whose value is not less than any other functions for any α, e.g., the bold line in Figure 4. The pentagram, which is not any transition point, indicates an intersection of two piecewise functions.*

Now, the challenge is how to find the "top" function which is larger than or equal to other piecewise functions. A simple idea is to compare the piecewise functions in every sub-domains. First, we list all transition points $\alpha_1, ..., \alpha_m$ of $n(n-1)$ functions $f_{\boldsymbol{q},\boldsymbol{d}}(\alpha)$ w.r.t. distinct pairs of $\boldsymbol{q}$ and $\boldsymbol{d}$ (i.e., all distinct values in *aM*); then, we find the "top" piecewise function on each range between two consecutive transition points, which requires computation time $O(n^3)$. Despite the complexity, this idea may not be correct because two piecewise functions may have an intersection between two consecutive transition points, such as the pentagram shown in Figure 4. Hence, we need a way to find such *additional* transition points. Our finding is that, if the top function is the same one at a_1 and a_2, then it is also the top function for any $\alpha \in [a_1, a_2]$, which is formalized as the following theorem.

Theorem 10. *Let $f(\alpha) = \frac{q(e^\alpha - 1)+1}{d(e^\alpha - 1)+1}$ and $f'(\alpha) = \frac{q'(e^\alpha - 1)+1}{d'(e^\alpha - 1)+1}$. If $f(a_1) \ge f'(a_1)$ and $f(a_2) \ge f'(a_2)$ in which $0 < a_1 < a_2$, we have $f(\alpha) \ge f'(\alpha)$ for any $a_1 \le \alpha \le a_2$.*

Based on Theorem 10, we design Algorithm 4 to generate a privacy loss function w.r.t. a given transition matrix. Al-

gorithm 4 outputs a piecewise function of $\mathcal{L}(\alpha)$: $qArr$, $dArr$ are coefficients of the sub-functions, and $aArr$ contains the corresponding sub-domains.

We now analyze Algorithm 4, which generate the privacy loss function $\mathcal{L}(\alpha)$ in given a domain $[a_1, a_m]$ by recursively find its sub-functions in sub-domains $[a_1, a_k]$ and $[a_k, a_m]$. In Line 1, we check whether the parameters *qM*, *dM* and *aM* are [0], which implied that P_i is uniform and $\mathcal{L}(\cdot)$ is 0 (see Lines 4 and 5 in Algorithm 2). For convenience of the following analysis, we denote the definition of $\mathcal{L}(\alpha)$ at point $\alpha = a_j$ by $\mathcal{L}^j(\alpha) = \log \frac{q^j(e^\alpha - 1)+1}{d^j(e^\alpha-1)+1}$ with coefficients q^j and d^j. In Lines 2, we obtain the definition of $\mathcal{L}^1(\alpha)$ with coefficients q^1 and d^1. In Line 3, we obtain the definition of $\mathcal{L}^m(\alpha)$ with coefficients q^m and d^m. From Line 4 to 20, we attempt to find the definition of $\mathcal{L}(\alpha)$ at every point in $[a_1, a_m]$. There are two modules in this part. The first one is from Line 5 to 12, in which we check that the definition of $\mathcal{L}(\alpha)$ in (a_1, a_m) is $\mathcal{L}^1(\alpha)$ or $\mathcal{L}^m(\alpha)$. The second module is from Line 13 to 19, in which we exam the definition of $\mathcal{L}(\alpha)$ at point a_k (which may be the intersection of two sub-functions $\mathcal{L}^1(\alpha)$ and $\mathcal{L}^m(\alpha)$) and in the sub-domains $[a_1, a_k]$ and $[a_k, a_m]$. Now, we dive into some details in these modules. In Line 5, the condition is true when $\mathcal{L}^1(\alpha)$ and $\mathcal{L}^m(\alpha)$ are the same, or they intersect at $\alpha = a_m$. Then, $\mathcal{L}^1$ is the top function in $[a_1, a_m]$ by Theorem 10 because of $\mathcal{L}^1(\alpha_1) \geq \mathcal{L}^m(\alpha_1)$ and $\mathcal{L}^1(\alpha_m) = \mathcal{L}^m(\alpha_m)$. In Line 8, the condition is true when $\mathcal{L}^1(\alpha)$ and $\mathcal{L}^m(\alpha)$ intersect at a_1 or they have no intersection in $[a_1, a_m]$ (this implies $q^1 = q^m$ and $d^1 = d^m$). Then, $\mathcal{L}^1$ is the top function in $[a_1, a_m]$ by Theorem 10. In Lines 6∼8 and 10∼12, we store the coefficients and sub-domains in arrays. From Line 14 to 19, we deal with the case of two functions intersecting at α_k that $a_1 < \alpha_k < a_m$ by recursively invoking Algorithm 4 to generate the sub-function of $\mathcal{L}(\alpha)$ in $[\alpha_1, \alpha_k]$ and $[\alpha_k, \alpha_m]$.

Algorithm 4: Generate Privacy Loss Function $\mathcal{L}(\alpha)$.

Input: P_i (i.e., P_i^B or P_i^F); a_1; a_m ($0 < a_1 \leq a_m$), *qM, dM, aM*.
Output: vectors $qArr$, $dArr$ (i.e., coefficients), $aArr$ (i.e., sub-domains).
1 **if** *qM, dM, aM are* [0] **then return** *qArr,dArr,aArr* as [0]
2 Find *maxPL1* , q^1, d^1 using **Algorithm 2** with a_1, *qM, dM, aM*, $\epsilon_t = 0$;
3 Find *maxPLm* , q^m, d^m using **Algorithm 2** with a_m, *qM, dM, aM*, $\epsilon_t = 0$;
4 $k = \frac{q^m + d^1 - q^1 - d^m}{q^1 * d^m - q^m * d^1}$;
5 **if** $a_1 = a_m$ **or** *maxPLm* = $\log \frac{q^1(e^{a_m}-1)+1}{d^1(e^{a_m}-1)+1}$ **then**
6 $aArr = [a_m]$; //initialize vectors.
7 $qArr = [q^1]$;
8 $dArr = [d^1]$;
9 **else if** *maxPL1* = $\log \frac{q^m(e^{a_1}-1)+1}{d^m(e^{a_1}-1)+1}$ **or** *k<=0* **then**
10 $aArr = [a_m]$; //initialize vectors.
11 $qArr = [q^m]$;
12 $dArr = [d^m]$;
13 **else**
14 $a_k = \log(k+1)$;
15 Find $qArr_k, dArr_k, aArr_k$ using **Algorithm 4** with P_i, a_1, a_k;
16 Find $qArr_m, dArr_m, aArr_m$ using **Algorithm 4** with P_i, a_k, a_m;
17 $aArr = [aArr_k, aArr_m]$; //concatenate two vectors.
18 $qArr = [qArr_k, qArr_m]$;
19 $dArr = [dArr_k, dArr_m]$;
20 **return** $qArr, dArr, aArr$.

When obtaining function $\mathcal{L}(\alpha)$, we can directly calculate BPL or FPL as shown in Algorithm 5. In Line 1 of Algorithm 5, we can perform a binary search because $aArr$ is sorted.

Complexity. The complexity of Algorithm 5 for calculating privacy leakage at one time point is $O(\log n)$, which makes it very efficient even for large n and T. Algorithm 4 itself requires $O(n^2 \log n + m \log m)$ time where m is the amount of transition points in $[a_1, a_m]$, and its parameters *qM*, *dM* and *aM* need to be calculated by Algorithm 2 which requires $O(n^3)$ time; hence, in total, generating $\mathcal{L}(\cdot)$ needs $O(n^3 + m \log m)$ times.

Algorithm 5: Calculate BPL or FPL by $\mathcal{L}(\alpha)$.

Input: α (i.e., α_{t-1}^B or α_{t+1}^F); $qArr$, $dArr$, $aArr$ (i.e., $\mathcal{L}(\alpha)$); ϵ_t.
Output: BPL or FPL at time t
1 **if** *qArr, dArr, aArr are* [0] **then return** ϵ_t
2 Binary Search α_k in $aArr$ that $\alpha_k \geq \alpha$ and $\alpha_{k+1} < \alpha$;
3 **return** $\log \frac{q_k(e^\alpha - 1)+1}{d_k(e^\alpha-1)+1} + \epsilon_t$

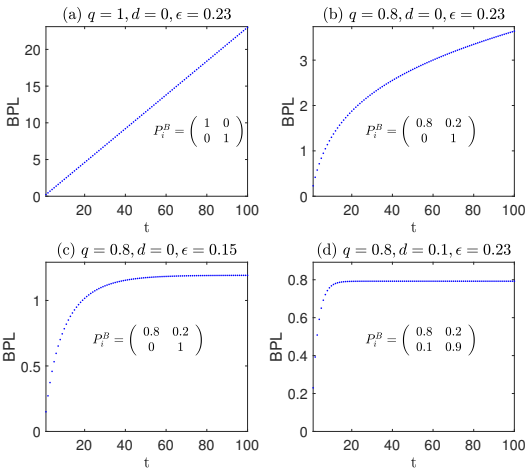


Fig. 5. Examples of the maximum BPL over time.

5 Bounding Temporal Privacy Leakage

In this section, we design two privacy budget allocation strategies that can be used to convert a traditional DP mechanism into one protecting against TPL.

We first investigate the upper bound of BPL and FPL. We have demonstrated that BPL and FPL may increase over time as shown in Figure 3. A natural question is that: is there a limit of BPL and FPL over time.

Theorem 11. *Given a transition matrix P_i^B (or P_i^F), let q and d be the ones that give the maximum value in Equation* (24) *and $q \neq d$. For $\mathcal{M}^t$ that satisfies ϵ-DP at each $t \in [1, T]$, there are four cases regarding the supremum of BPL (or FPL) over time.*

$$\begin{cases} \log \frac{\sqrt{4de^\epsilon(1-q)+(d+qe^\epsilon-1)^2}+d+qe^\epsilon-1}{2d} & d \neq 0 \\ \log \frac{(1-q)e^\epsilon}{1-qe^\epsilon} & d = 0 \text{ and } q \neq 1 \text{ and } \epsilon \leq \log(1/q) \\ \inf & d = 0 \text{ and } q \neq 1 \text{ and } \epsilon > \log(1/q) \\ \inf & d = 0 \text{ and } q = 1 \end{cases}$$

Example 5 (The supremum of BPL over time)**.** *Suppose that $\mathcal{M}^t$ satisfies ϵ-DP at each time point. In Figure 5, it shows the maximum BPL w.r.t. different ϵ and different transition matrices of P_i^B. In (a) and (b), the supremum does not exist. In (c) and (d), we can calculate the supremum using Theorem 11. The results are in line with the ones from computing BPL step by step at each time point using Algorithm 1.*

Algorithm 6 for finding supreme of BPL or FPL is correct because, according to Theorem 8, all possible q and d that give the maximum value in Equation (24) are in the matrices qM and dM. Algorithm 6 is useful not only for designing privacy budget allocation strategies in this section, but also

Algorithm 6: Find Supermum of BPL or FPL.

Input: ϵ_t ; qM; dM.
Output: the supremum of BPL or FPL over time, q, d
$sup = 0$; $q = 0$; $d = 0$
foreach $q_c \in qM$, $d_c \in dM$ **do**
  Calculate a candidate sup_c using Theorem 11 ;
  if $sup < sup_c$ **then** $sup = sup_c$; $q = q_c$; $d = d_c$
return sup, q, d

for setting an appropriate parameter a_m as the input of Algorithm 4 because we will show in experiments that a larger a_m makes Algorithm 4 time-consuming (while, too small a_m may result in failing to calculate privacy leakage from $\mathcal{L}(\alpha)$ if the input α may be larger than a_m).

Achieving α-DP$_\mathcal{T}$ by limiting upper bound. We now design a privacy budget allocation strategy utilizing Theorem 11 to bound TPL. Theorem 11 tells us that, if it is not the strongest temporal correlation (i.e., $d = 0$ and $q = 1$), we may bound BPL or FPL within a desired value by allocating an appropriate privacy budget to a traditional DP mechanism at each time point. In other words, we want a constant ϵ_t that guarantee the supremum of TPL, which is equal to the sum of the supremum of BPL and the supremum of BPL subtracting ϵ_t by Equation (11), will not larger than α. Based on this idea, we design Algorithm 7 for solving such ϵ_t.

Algorithm 7: Achieving α-DP$_\mathcal{T}$ by upper bound

Input: P_i^B and P_i^F; α (desired privacy level).
Output: Privacy budgets ϵ_t satisfying α-DP$_\mathcal{T}$ at each t
Find qM_B, dM_B using **Algorithm** 4 with P_i^B;
Find qM_F, dM_F using **Algorithm** 4 with P_i^F;
$bingo = false$; $range = \alpha$; $e = 0.5 * \alpha$;
do //binary search.
  $range = 0.5 * range$;
  Calculate sup_B by **Algorithm** 6 with e and qM_B, dM_B;
  Calculate sup_F by **Algorithm** 6 with e and qM_F, dM_F;
  if $sup_B + sup_F - e = \alpha$ **then** $bingo = true$
  else if $sup_B + sup_F - e > \alpha$ **then** $e = e - range$
  else $e = e + range$
while $bingo = false$
return $\epsilon_t = e$

Achieving α-DP$_\mathcal{T}$ by privacy leakage quantification. Algorithm 7 allocates privacy budgets in a conservative way: when T is short, the privacy leakage may not be increased to the supremum. We now design Algorithm 8 to overcome this drawback. Observing the supremum of backward privacy loss in Figure 5(c)(d), BPL at the first time point is much less than the supremum. Similarly, it is easy to see that FPL at the last time point is much less than its supremum. Hence, we attempt to allocate more privacy budgets to $\mathcal{M}^1$ and $\mathcal{M}^T$ so that the temporal privacy leakage at every time points are *exactly* equal to the desired level. Specifically, if we want that BPL at two consecutive time points are exactly the same value α^B, i.e., $BPL(\mathcal{M}^t) = BPL(\mathcal{M}^{t+1}) = \alpha^B$, we can derive that $\epsilon_t = \epsilon_{t+1}$ for $t \geq 2$ (it is true for $t = 1$ only when $\mathcal{L}^B(\cdot) = 0$). Applying the same logic to FPL, we have a new strategy for allocating privacy budgets: assigning larger privacy budgets at time points 1 and T, and constant values at time $[2, T]$ to make sure that BPL at time points $[1, T-1]$ are the same, denoted by α^B, and FPL at time $[2, T]$ are the same, denoted by α^F. Hence, we have $\epsilon_1 = \alpha^B$ and $\epsilon_T = \alpha^F$. Let the value of privacy budget at $[2, T]$ be ϵ_m. We have (i) $\mathcal{L}^B(\alpha^B) + \epsilon_m = \alpha^B$, (ii) $\mathcal{L}^F(\alpha^F) + \epsilon_m = \alpha^F$ and (iii) $\alpha^B + \alpha^F - \epsilon_m = \alpha$. Combing (i) and (iii), we have Equation (26); Combing (ii) and (iii), we have Equation (27).

$$\mathcal{L}^B(\alpha^B) + \alpha^F = \alpha \quad (26)$$

$$\mathcal{L}^F(\alpha^F) + \alpha^B = \alpha \quad (27)$$

Based on the above idea, we design Algorithm 8 to solve α^B and α^F. Since α^B should be in $[0, \alpha]$, we heuristically initialize α^B with $0.5 * \alpha$ in Line 3 and then use binary search to find appropriate α^B and α^F that satisfy Equations (26) and (27).

Algorithm 8: Achieving α-DP$_\mathcal{T}$ by quantification

Input: P_i^B and P_i^F ; α (desired privacy level for user i).
Output: Privacy budgets $\epsilon_t, t \in [1, T]$ satisfying α-DP$_\mathcal{T}$ at each t
Find qM^B, dM^B, aM^B using **Algorithm** 4 with P_i^B;
Find qM^F, dM^F, aM^F using **Algorithm** 4 with P_i^F;
$bingo = false$; $range = \alpha$; $a^B = 0.5 * \alpha$;
do //binary search.
  $range = 0.5 * range$;
  Find L^B by **Algorithm** 5 with a^B, qM^B, dM^B, aM^B, $\epsilon = 0$;
  $a^F = \alpha - L^B$; //Equation (26).
  Find L^F by **Algorithm** 5 with a^F with qM^F, dM^F, aM^F, $\epsilon = 0$;
  if $L^F + a^B = \alpha$ **then** $bingo = true$
  else if $L^F + a^B < \alpha$ **then** $a^B = a^B + range$ //Equation (27).
  else $a^B = a^B - range$
while $bingo = false$
return $\epsilon_1 = a^B$; $\epsilon_t = a^B + a^F - \alpha, t \in [2, T-1]$; $\epsilon_T = a^F$

6 Experimental Evaluation

In this section, we design experiments for the following: (1) verifying the runtime and correctness of our privacy leakage quantification algorithms, (2) investigating the impact of the temporal correlations on privacy leakage and (3) evaluating the data release Algorithms 7 and 8. We implemented all the algorithms[3] in Matlab2017b and conducted the experiments on a machine with an Intel Core i7 2.6 GHz CPU and 16G RAM running macOS High Sierra.

6.1 Runtime of Privacy Quantification Algorithms

In this section, we compare the runtime of our algorithms with IBM ILOG CPLEX[4], which is a well-known software for solving optimization problems, e.g., the linear-fractional programming problem (19)∼(21) in our setting.

For verifying the correctness of three privacy quantifying algorithms, Algorithm 1, Algorithm 3 and Algorithm 5, we generate 100 random transition matrices with dimension size $n = 30$ and comparing the calculation results with the one solving LFP problem using CPLEX. We verified that all results obtained from our algorithms are identical to the one using CPLEX w.r.t. the same transition matrix.

For testing the runtime of our algorithms, we run them 30 times with randomly generated transition matrices, and run CPLEX one time (because it is very time-consuming), and then calculate the average runtime for each of them. Since Algorithm 3 needs parameters that are precomputed by Algorithm 2, and Algorithm 5 needs $\mathcal{L}(\cdot)$ that can be obtained using Algorithm 4, we also test the runtime of these precomputations. The results are shown in Figure 6.

Runtime vs. n. In Figures 6(a) and (b), we show the runtime of privacy quantification algorithms and precomputation algorithms, respectively, In Figure 6(a), each algorithm takes inputs of $\alpha = 0.1$ and $n \times n$ random probability matrices. The runtime of all algorithms increase along with

3. Souce code: https://github.com/brahms2013/TPL
4. http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/. We use version 12.7.1.

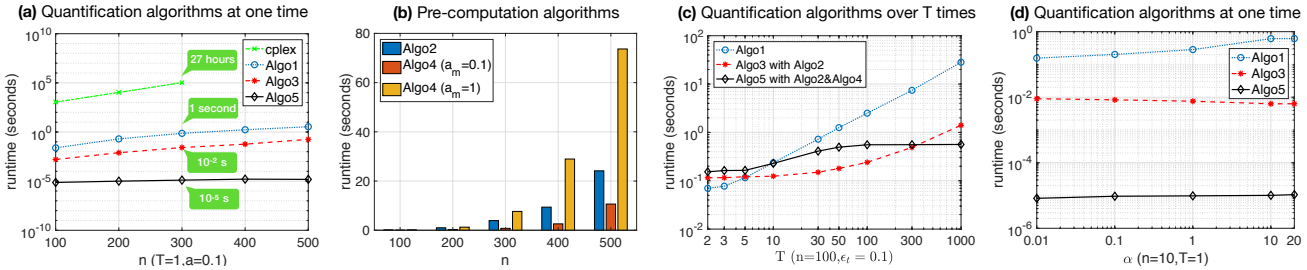


Fig. 6. Runtime of Temporal Privacy Leakage Quantification Algorithms.

n because the number of variables in our LFP problem is n. The proposed Algorithms 1, 3 and 5 significantly outperform CPLEX. In Figure 6(b), we test precomputation procedures. We observed that all algorithms are increasing with n, but Algorithm 4 is more susceptible to a_m. Algorithm 4 with a larger a_m results in higher runtime because it performs binary search in $[0, a_m]$.

These results are in line with our complexity analysis, which shows the complexity of Algorithms 2 and 4 are $O(n^3)$ and $O(n^2 \log n + m \log m)$ (m is the amount of transition points and increasing with a_m), respectively. However, when n or a_m is large, pre-computation Algorithms 2 and 4 are time-consuming because we need to find optimal solutions for $n*(n-1)$ LFP problems given a n-dimension transition matrix. This can be improved by advanced computation tools such as parallel computing because here each LFP problem is independently solved, and such computation only needs to be run one time before starting to release private data. Another interesting way to improve the runtime is to find the relationship between the optimal solutions of different LFP problems given a transition matrix (so that we can prune some computations). We defer this to future study.

Runtime vs. T. In Figure 6(c), we test the runtime of each privacy quantification algorithm integrated with their precomputations over different length of time points. We want to know how can we benefit from these precomputations over time. All algorithms take inputs of 100×100 matrices and $\epsilon_t = 0.1$ for each time point t. The parameters of Algorithm 4 need to be initialized by Algorithm 2, so we take them as an integrated module along with Algorithm 5. It shows that, Algorithm 1 runs fast if T is small. Algorithm 3 becomes the most preferable if T is in $[5, 300]$. However, when T is larger than 300, Algorithm 5 with its precomputation (Algorithm 4 with $a_m = (T+1) * \epsilon_t$ which is the worst case of supremum) is the fast one and its runtime is almost constant with the increase of T. Therefore, there is no best algorithm in efficiency without a known T, but we can choose appropriate algorithm adaptively.

Runtime vs. α. In Figure 6(d), we show that, a larger previous BPL (or the next FPL), i.e., α, may lead to higher runtime of Algorithm 1, whereas other algorithms are relatively stable for varying α. The reason is that, when α is large, Algorithm 1 may take more time in Lines 9 and 10 for updating each pair of $q_j \in \boldsymbol{q}^+$ and $d_j \in \boldsymbol{d}^+$ to satisfy Inequality (22). An update in Line 10 is more likely to occur due to a large α because $\frac{q(e^\alpha-1)+1}{d(e^\alpha-1)+1}$ is increasing with α. However, such growth of runtime along with α will not last so long because the update happens n times in the worse case. As shown in Figure 6(b), when $\alpha > 10$, the runtime of Algorithm 1 becomes stable.

6.2 Impact of Temporal Correlations on TPL

In this section, for the ease of exposition, we only present the impact of temporal correlations on BPL because the growth of BPL and FPL are in the same way.

The setting of temporal correlations. To evaluate if our privacy loss quantification algorithms can perform well under different degrees of temporal correlations, we find a way to generate the transition matrices to eliminate the effect of different correlation estimation algorithms or datasets. First, we generate a transition matrix indicating the "strongest" correlation that contains probability 1.0 in its diagonal cells (this type of transition matrix will lead to an upper bound of TPL). Then, we perform *Laplacian smoothing* [20] to *uniformize* the probabilities of P_i (the uniform transition matrix will lead to an low bound of TPL). Let p_{jk} be an element at the jth row and kth column of the matrix P_i. The uniformized probabilities $\hat{p}_{jk}$ are generated using Equation (28), where s is a positive parameter that controls the degrees of uniformity of probabilities in each row. Hence, a smaller s means stronger temporal correlation. We note that, different s are only comparable under the same n.

$$\hat{p}_{jk} = \frac{p_{jk} + s}{\sum_{u=1}^{n} (p_{ju} + s)} \tag{28}$$

We examined s values ranging from 0.005 to 1 and set n to 50 and 200. Let ε be the privacy budget of $\mathcal{M}^t$ at each time point. We test $\varepsilon = 1$ and 0.1. The results are shown in Figure 7 and are summarized as follows.

Privacy Leakage vs. s**.** Figure 7 shows that the privacy leakage caused by a non-trivial temporal correlation will increase over time, and such growth first increases sharply and then remains stable because it is gradually close to its supremum. The increase caused by a stronger temporal correlations (i.e., smaller s) is steeper, and the time for the increase is longer. Consequently, stronger correlations result in higher privacy leakage.

Privacy Leakage vs. ε**.** Comparing Figures 7(a) and (b), we found that 0.1-DP significantly delayed the growth of privacy leakage. Taking $s = 0.005$, for example, the noticeable increase continues for almost 8 timestamps when $\varepsilon = 1$ (Figures 7(a)), whereas it continues for approximately 80 timestamps when $\varepsilon = 0.1$ (Figures 7(b)). However, after a sufficient long time, the privacy leakage in the case of $\varepsilon = 0.1$ is not substantially lower than that of $\varepsilon = 1$ under stronger temporal correlations. This is because, although the privacy leakage is eliminated at each time point by setting a small privacy budget, the adversaries can eventually learn sufficient information from the continuous releases.

Privacy Leakage vs. n**.** Under the same s, TPL is smaller when n (dimension of the transition matrix) is larger, as

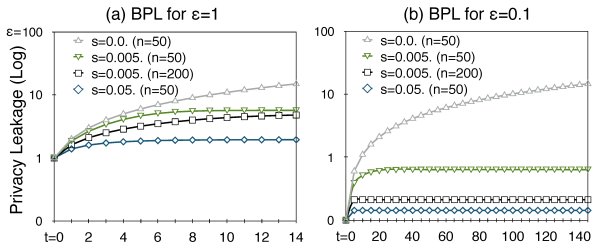


Fig. 7. Evaluation of BPL under different degrees of correlations.

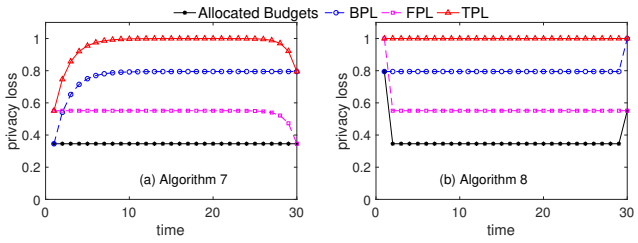


Fig. 8. Privacy Budget Allocation Schemes for 1-DP$_\mathcal{T}$.

shown in the lines $s = 0.005$ with $n = 50$ and $n = 200$ of Figure 7. This is because the transition matrices tend to be uniform (weaker correlations) when the dimension is larger.

6.3 Evaluation of Data Releasing Algorithms

In this section, we first show a visualization of privacy allocation of Algorithms 7 and 8, then we compare the data utility in terms of Laplace noise.

Figure 8 shows an example of budget allocation, w.r.t. $P_i^B = \left(\begin{smallmatrix}0.8 & 0.2\\ 0.2 & 0.8\end{smallmatrix}\right)$ and $P_i^F = \left(\begin{smallmatrix}0.8 & 0.2\\ 0.1 & 0.9\end{smallmatrix}\right)$. The goal is 1-DP$_\mathcal{T}$. It is easy to see that Algorithm 8 has better data utility because it exactly achieves the desired privacy level.

Figure 9 shows the data utility of Algorithms 7 and 8 with 2-DP$_\mathcal{T}$. We calculate the absolute value of the Laplace noise with the allocated budgets (as shown in Figure 8). Higher value of noise indicates lower data utility. In Figure 9(a), we test the data utility under backward and forward temporal correlation both with parameter $s = 0.001$, which means relatively strong correlation. It shows that, when T is short, Algorithm 8 outperforms Algorithm 7. In Figure 9(b), we investigate the data utility under different degree of correlations. The dash line indicates the absolute value of Laplace noise if no temporal correlation exists (privacy budget is 2). It is easy to see that the data utility significantly decays under strong correlation $s = 0.01$.

7 Related Work

Dwork et al. first studied *differential privacy under continual observation* and proposed event-level/user-level privacy [4] [5]. A plethora of studies have been conducted to investigate different problems in this setting, such as high dimensional data [1] [8], infinite sequence [7] [21] [22], and real-time publishing [6] [23]. To the best of our knowledge, no study has reported the risk of differential privacy under temporal correlations for the continuous aggregate release setting. Although [24] have considered a similar adversarial model in which the adversaries have prior knowledge of temporal correlations represented by Markov chains, they proposed a mechanism extending differential privacy for releasing a private location, whereas we focus on the scenario of continuous aggregate release with DP.

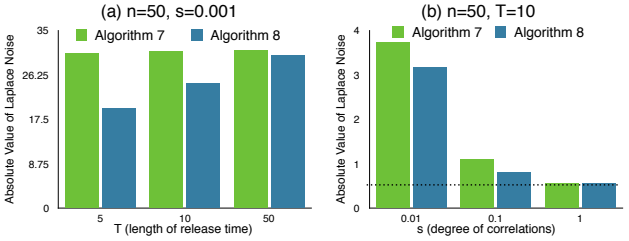


Fig. 9. Data utility of 2-DP$_\mathcal{T}$ mechanisms.

Several studies have questioned whether differential privacy is valid for correlated data. Kifer and Machanavajjhala [25] [16] [17] first raised the important issue that differential privacy may not guarantee privacy if adversaries know the data correlations between tuples. They [25] argued that it is not possible to ensure any utility in addition to privacy without making assumptions about the data-generating distribution and the background knowledge available to an adversary. To this end, they proposed a general and customizable privacy framework called *PufferFish* [16] [17], in which the potential secrets, discriminative pairs, and data generation need to be explicitly defined. Song et al. [14] proposed Markov Quilt Mechanism when the correlations can be modeled by Bayesian Network. Yang et al. [12] investigated differential privacy on correlated tuples described using a proposed Gaussian correlation model. The privacy leakage w.r.t. adversaries with specified prior knowledge can be efficiently computed. Zhu et al. [26] proposed correlated differential privacy by defining the sensitivity of queries on correlated data. Liu et al. [13] proposed dependent differential privacy by introducing dependence coefficients for analyzing the sensitivity of different queries under probabilistic dependence between tuples. Most of the above works dead with correlations between users in the database, i.e., user-user correlations, in the setting of one-shot data release, whereas we deal with the correlation among single user's data at different time points, i.e., temporal correlations, and focusing on the dynamic change of privacy guarantee in continuous data release.

On the other hand, it is still controversial [27] what should be the guarantee of DP on correlated data. Li et al. [27] proposed *Personal Data Principle* for clarifying the privacy guarantee of DP on correlated data. It states that an individual's privacy is not violated if no data about the individual is used. By doing this, one can ignore any correlation between this individual's data and other users' data, i.e.,user-user correlations. On the other hand, the question of "what is individual's data", or "what should be an appropriate notion of negiboring databases" is tricky in many application scenarios such as genomic data. If we apply Personal Data Principle to the setting of continuous data release, event-level privacy is not a good fit for protecting individual's privacy because a user's data at each time point is only a part of his/her whole data in the streaming database. Our work shares the same insight with Personal Data Principle on this point: we show that the privacy loss of event-level privacy may increase over time under temporal correlation, while the guarantee of user-privacy is as expected. We note that, when the data stream is infinite or the end of data release is unknown, we can only resort to event-level privacy or w-event privacy [7]. Our work provides useful tools against TPL in such setting.

8 Conclusions

In this paper, we quantified the risk of differential privacy under temporal correlations by formalizing, analyzing and calculating the privacy loss against adversaries who have knowledge of temporal correlations. Our analysis shows the privacy loss of event-level privacy may increase over time, while the privacy guarantee of user-level privacy is as expected. We design fast algorithms for quantifying temporal privacy leakage which enables private data release in real-time. This work opens up interesting future research directions, such as investigating the privacy leakage under temporal correlations combining with other type of correlation models, and use our methods to enhance previous studies that neglected the effect of temporal correlations in continuous data release.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 16K12437, 17H06099, JSPS Core-to-Core Program, A. Advanced Research Network, the National Institute of Health (NIH) under award number R01GM114612, the Patient-Centered Outcomes Research Institute (PCORI) under contract ME-1310-07058, and the National Science Foundation under award CNS-1618932.

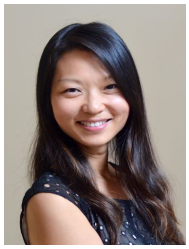
References

[1] G. Acs and C. Castelluccia, "A case study: Privacy preserving release of spatio-temporal density in paris," in *KDD*, 2014, pp. 1679–1688.
[2] J. Bolot, N. Fawaz, S. Muthukrishnan, A. Nikolov, and N. Taft, "Private decayed predicate sums on streams," in *ICDT*, 2013, pp. 284–295.
[3] T.-H. H. Chan, E. Shi, and D. Song, "Private and continual release of statistics," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 3, pp. 26:1–26:24, 2011.
[4] C. Dwork, "Differential privacy in new settings." in *SODA*, 2010, pp. 174–183.
[5] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum, "Differential privacy under continual observation," in *STOC*, 2010, pp. 715–724.
[6] L. Fan, L. Xiong, and V. Sunderam, "FAST: differentially private real-time aggregate monitor with filtering and adaptive sampling," in *SIGMOD*, 2013, pp. 1065–1068.
[7] G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias, "Differentially private event sequences over infinite streams," *PVLDB*, vol. 7, no. 12, pp. 1155–1166, 2014.
[8] Y. Xiao, L. Xiong, L. Fan, S. Goryczka, and H. Li, "DPCube: differentially private histogram release through multidimensional partitioning," *Trans. Data Privacy*, vol. 7, no. 3, pp. 195–222, 2014.
[9] C. Dwork, "Differential privacy," in *ICALP*, 2006, pp. 1–12.
[10] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *TCC*, 2006, pp. 265–284.
[11] R. Chen, B. C. Fung, P. S. Yu, and B. C. Desai, "Correlated network data publication via differential privacy," *VLDBJ*, vol. 23, no. 4, pp. 653–676, 2014.
[12] B. Yang, I. Sato, and H. Nakagawa, "Bayesian differential privacy on correlated data," in *SIGMOD*, 2015, pp. 747–762.
[13] Changchang Liu, Supriyo Chakraborty, and Prateek Mittal, "Dependence makes you vulnerable: Differential privacy under dependent tuples," in *NDSS*, 2016.
[14] S. Song, Y. Wang, and K. Chaudhuri, "Pufferfish privacy mechanisms for correlated data," in *SIGMOD*, 2017, pp. 1291–1306.
[15] F. D. McSherry, "Privacy integrated queries: an extensible platform for privacy-preserving data analysis," in *SIGMOD*, 2009, pp. 19–30.
[16] D. Kifer and A. Machanavajjhala, "A rigorous and customizable framework for privacy," in *PODS*, 2012, pp. 77–88.
[17] ——, "Pufferfish: A framework for mathematical privacy definitions," *ACM Trans. Database Syst.*, vol. 39, no. 1, pp. 3:1–3:36, 2014.
[18] Z. Jorgensen, T. Yu, and G. Cormode, "Conservative or liberal? personalized differential privacy," in *ICDE*, 2015, pp. 1023–1034.
[19] E. B. Bajalinov, *Linear-Fractional Programming Theory, Methods, Applications and Software*, 2003, vol. 84.
[20] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian surface editing," in *SGP*, 2004, pp. 175–184.
[21] Y. Cao and M. Yoshikawa, "Differentially private real-time data release over infinite trajectory streams," in *MDM*, vol. 2, 2015, pp. 68–73.
[22] ——, "Differentially private real-time data publishing over infinite trajectory streams," *IEICE Trans. Inf.& Syst.*, vol. E99-D, no. 1, pp. 163–175, 2016.
[23] H. Li, L. Xiong, X. Jiang, and J. Liu, "Differentially private histogram publication for dynamic datasets: An adaptive sampling approach," in *CIKM*, 2015, pp. 1001–1010.
[24] Y. Xiao and L. Xiong, "Protecting locations with differential privacy under temporal correlations," in *CCS*, 2015, pp. 1298–1309.
[25] D. Kifer and A. Machanavajjhala, "No free lunch in data privacy," in *SIGMOD*, 2011, pp. 193–204.
[26] T. Zhu, P. Xiong, G. Li, and W. Zhou, "Correlated differential privacy: Hiding information in non-IID data set," *IEEE Trans.Inf. Forensics Security.*, vol. 10, no. 2, pp. 229–242, 2015.
[27] N. Li, M. Lyu, D. Su, and W. Yang, "Differential privacy: From theory to practice," in *Synthesis Lectures on Information Security, Privacy, Trust*, 2016, pp. 1–138.
[28] W. Dinkelbach, "On nonlinear fractional programming," *Management Science*, vol. 13, no. 7, pp. 492–498, 1967.

Yang Cao received B.S. degree from the School of Software Engineering, Northwestern Polytechnical University, China, in 2008, the M.S. degree and Ph.D. degree from the Graduate School of Informatics, Kyoto University, Japan, in 2014 and 2017, respectively. He is currently a Postdoctoral Fellow at the department of Math and Computer Science, Emory University, USA. His research interests are privacy preserving data publishing and mining.

Masatoshi Yoshikawa received the B.E., M.E. and Ph.D. degrees from Department of Information Science, Kyoto University in 1980, 1982 and 1985, respectively. Before joining Graduate School of Informatics, Kyoto University as a professor in 2006, he has been a faculty member of Kyoto Sangyo University, Nara Institute of Science and Technology, and Nagoya University. His general research interests are in the area of databases. His current research interests include multi-user routing algorithms and services, theory and practice of privacy protection, and medical data mining. He is a member of ACM and IPSJ.

Yonghui Xiao received 3 B.S. degrees from Xi'an Jiaotong University, China, in 2005. He obtained the M.S. degree from Tsinghua University in 2011 after spending two years working in industry. He obtained the Ph.D. degree in the department of Math and Computer Science at Emory University in 2017. He is currently a software engineer at Google.



Li Xiong is a Winship Distinguished Research Professor of Computer Science (and Biomedical Informatics) at Emory University. She holds a PhD from Georgia Institute of Technology, an MS from Johns Hopkins University, and a BS from University of Science and Technology of China, all in Computer Science. She and her research group, Assured Information Management and Sharing (AIMS), conduct research that addresses both fundamental and applied questions at the interface of data privacy and security, spatiotemporal data management, and health informatics.

APPENDIX A
PROOF OF THEOREM 2

We need to prove

$$TPL(A_i^{\mathcal{T}}, \mathcal{M}^t) = \tag{29}$$

$$\sup_{l^t, l_i^{t'}} \left(\sup_{\boldsymbol{r}^1} \log \frac{\Pr(\boldsymbol{r}^1 | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\boldsymbol{r}^1 | l_i^{t'}, D_{\mathcal{K}}^t)} + \cdots + \sup_{\boldsymbol{r}^T} \log \frac{\Pr(\boldsymbol{r}^T | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\boldsymbol{r}^T | l_i^{t'}, D_{\mathcal{K}}^t)} \right) \tag{30}$$

$$= \sup_{l^t, l_i^{t'}, \boldsymbol{r}^1} \log \frac{\Pr(\boldsymbol{r}^1 | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\boldsymbol{r}^1 | l_i^{t'}, D_{\mathcal{K}}^t)} + \cdots + \sup_{l^t, l_i^{t'}, \boldsymbol{r}^T} \log \frac{\Pr(\boldsymbol{r}^T | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\boldsymbol{r}^T | l_i^{t'}, D_{\mathcal{K}}^t)} \tag{31}$$

$$= \sup_{\substack{\boldsymbol{r}^1, ..., \boldsymbol{r}^t, \\ l_i^t, l_i^{t'}}} \log \frac{\Pr(\boldsymbol{r}^1, ..., \boldsymbol{r}^t | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\boldsymbol{r}^1, ..., \boldsymbol{r}^t | l_i^{t'}, D_{\mathcal{K}}^t)} + \sup_{\substack{\boldsymbol{r}^t, ..., \boldsymbol{r}^T, \\ l_i^t, l_i^{t'}}} \log \frac{\Pr(\boldsymbol{r}^t, ..., \boldsymbol{r}^T | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\boldsymbol{r}^t, ..., \boldsymbol{r}^T | l_i^{t'}, D_{\mathcal{K}}^t)}$$
$$- \sup_{\boldsymbol{r}^t, l_i^t, l_i^{t'}} \log \frac{\Pr(\boldsymbol{r}^t | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\boldsymbol{r}^t | l_i^{t'}, D_{\mathcal{K}}^t)} \tag{32}$$

Because $\boldsymbol{r}^t$ at different $t \in [1, T]$ are independent given $D^t = \{l_i^t, D_{\mathcal{K}}^t\}$ or $D^{t'} = \{l_i^{t'}, D_{\mathcal{K}}^t\}$, we can derive Equation (30) from Equation (29), and derive Equation (32) from Equation (31). The remaining question is how to prove the derivation of Equation (31) from Equation (30). It is true because $\mathcal{M}^t$ is the same DP mechanism at different t, and the output domains $Range(\mathcal{M}^k) = Range(\mathcal{M}^j)$ for any $k, j \in [1, T]$. We explain in details in below.

Assume we have $\{\bar{l_i^t}, \bar{l_i^{t'}}\}$ maximizing the first item in Equation (31) with a certain value of $\boldsymbol{r}^1$, i.e.,

$$\{\bar{l_i^t}, \bar{l_i^{t'}}\} = \arg\max_{l_i^t, l_i^{t'}} \log \frac{\Pr(\boldsymbol{r}^1 | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\boldsymbol{r}^1 | l_i^{t'}, D_{\mathcal{K}}^t)}.$$

Then, $\{\bar{l_i^t}, \bar{l_i^{t'}}\}$ also maximizes other items in Equation (31) such as $\sup_{\boldsymbol{r}^k, l_i^t, l_i^{t'}} \log \frac{\Pr(\boldsymbol{r}^k | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\boldsymbol{r}^k | l_i^{t'}, D_{\mathcal{K}}^t)}$ with a certain value of $\boldsymbol{r}^k$ where $k \in [1, T]$.

Formally, that is to say, there exists $\bar{\boldsymbol{r}^k} \in Range(\mathcal{M}^k)$, for any $k \in [1, T]$, that we have

$$\log \frac{\Pr(\bar{\boldsymbol{r}^k} | \bar{l_i^t}, D_{\mathcal{K}}^t)}{\Pr(\bar{\boldsymbol{r}^k} | \bar{l_i^{t'}}, D_{\mathcal{K}}^t)} = \sup_{l^t, l_i^{t'}, \boldsymbol{r}^k} \log \frac{\Pr(\boldsymbol{r}^k | l_i^t, D_{\mathcal{K}}^t)}{\Pr(\boldsymbol{r}^k | l_i^{t'}, D_{\mathcal{K}}^t)}.$$

Hence, we can derive Equation (31) from Equation (30). The theorem follows.[5]

APPENDIX B
PROOF OF THEOREM 3

Proof. According to Definition 5 of TPL, the overall privacy leakage of $\mathcal{M}^t$ and $\mathcal{M}^{t+1}$ in terms of $\text{DP}_{\mathcal{T}}$ are

$$\sup_{D^t, D_{\mathcal{M}}^t, ..., D_{\mathcal{M}}^{t+j\prime}, \boldsymbol{r}^1, ..., \boldsymbol{r}^T} \log \frac{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^T | D^t, D^{t+1})}{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^T | D^{t\prime}, D^{t+1\prime})}$$
$$= \sup_{\substack{D^t, D^{t\prime}, \\ \boldsymbol{r}^1, \ldots, \boldsymbol{r}^t}} \log \frac{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^t | D^t)}{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^t | D^{t\prime})} + \sup_{\substack{D^t, D^{t\prime}, \\ \boldsymbol{r}^{t+1}, \ldots, \boldsymbol{r}^T}} \log \frac{\Pr(\boldsymbol{r}^{t+1}, \ldots, \boldsymbol{r}^T | D^{t+1})}{\Pr(\boldsymbol{r}^{t+1}, \ldots, \boldsymbol{r}^T | D^{t+1\prime})}$$
$$= BPL(\mathcal{M}^t) + FPL(\mathcal{M}^{t+1})$$
$$= \alpha_t^B + \alpha_{t+1}^F.$$

Therefore, the theorem follows when $j = 1$.

According to Definition 9, the overall privacy leakage of $\mathcal{M}^t, \ldots, \mathcal{M}^{t+j}$ in terms of $\text{DP}_{\mathcal{T}}$ is as follows.

$$\sup_{\substack{D^t, \ldots, D^{t+j}, \\ D^{t\prime}, \ldots, D^{t+j\prime}, \\ \boldsymbol{r}^1, \ldots, \boldsymbol{r}^T}} \log \frac{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^T | D^t, \ldots, D^{t+j})}{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^T | D^{t\prime}, \ldots, D^{t+j\prime})}$$
$$= \sup_{\substack{D^t, D^{t\prime}, \\ \boldsymbol{r}^1, \ldots, \boldsymbol{r}^t}} \log \frac{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^t | D^t)}{\Pr(\boldsymbol{r}^1, \ldots, \boldsymbol{r}^t | D^{t\prime})} + \sup_{\substack{D^{t+j}, D^{t+j\prime}, \\ \boldsymbol{r}^{t+j}, \ldots, \boldsymbol{r}^T}} \log \frac{\Pr(\boldsymbol{r}^{t+j}, \ldots, \boldsymbol{r}^T | D^{t+j})}{\Pr(\boldsymbol{r}^{t+1}, \ldots, \boldsymbol{r}^T | D^{t+j\prime})}$$
$$+ \sup_{\substack{D^{t+1}, \\ D^{t+1\prime}, \\ \boldsymbol{r}^{t+1}}} \log \frac{\Pr(\boldsymbol{r}^{t+1} | D^{t+1})}{\Pr(\boldsymbol{r}^{t+1} | D^{t+1\prime})} + \cdots + \sup_{\substack{D^{t+j-1}, \\ D^{t+j-1\prime}, \\ \boldsymbol{r}^{t+j-1}}} \log \frac{\Pr(\boldsymbol{r}^{t+j-1} | D^{t+j-1})}{\Pr(\boldsymbol{r}^{t+j-1} | D^{t+j-1\prime})}$$
$$= BPL(\mathcal{M}^t) + FPL(\mathcal{M}^{t+j}) + PL_0(\mathcal{M}^{t+1}) + \cdots PL_0(\mathcal{M}^{t+j-1})$$
$$= \alpha_t^B + \alpha_{t+j}^F + \sum_{k=1}^{k=j-1} \epsilon_{t+k}.$$

Therefore, the theorem follows when $j \geq 2$. □

APPENDIX C
PROOF OF THEOREM 5

We need Dinkelbach's Theorem and Lemma 1 in our proof.

Theorem 12 (Dinkelbach's Theorem [28])**.** *In a linear-fractional programming problem, suppose that the variable vector is $\boldsymbol{x}$ and the objective function is represented as $\frac{Q(\boldsymbol{x})}{D(\boldsymbol{x})}$. Vector $\boldsymbol{x}^*$ is an optimal solution if and only if*

$$\max\{Q(\boldsymbol{x}) - \lambda * D(\boldsymbol{x})\} = 0 \text{ where } \lambda = \frac{Q(\boldsymbol{x}^*)}{D(\boldsymbol{x}^*)}. \tag{33}$$

Lemma 1. *For the following maximization problem ($k_1, ..., k_n \in \mathbb{R}$) with the same constraints as the ones in the linear-fractional programming* (19)∼(21)*,*

$$\begin{aligned} \text{maximize } & k_1 x_1 + \cdots + k_n x_n \\ \text{subject to } & e^{-\alpha_{t-1}^B} * x_k \leq x_j \leq e^{\alpha_{t-1}^B} * x_k, \\ & 0 < x_j < 1 \text{ and } 0 < x_k < 1, \\ & \text{where } x_j, x_k \in \boldsymbol{x},\ j, k \in [1, n]. \end{aligned}$$

an optimal solution is as follows: if $k_i > 0$, let $x_i = e^{\alpha_{t-1}^B} m$ where m is a positive real number; if $k_i \leq 0$, let $x_i = m$.

Proof. Without loss of generality, we suppose that the smallest value in the optimal solution is x_n. Let y_j be $\frac{x_j}{x_n}$ for $j \in [1, n-1]$; then, $1 \leq y_j \leq e^{\alpha_{t-1}^B}$. Replacing x_j with y_j and setting $x_n = m$, we have a new objective function $\frac{1}{m} * (k_1 y_1 + \cdots + k_{n-1} y_{n-1} + k_n)$ whose solution is equivalent to the original one. Because the only constraint is $1 \leq y_j \leq e^{\alpha_{t-1}^B}$, the following is an optimal solution for the maximum objective function: if $k_j > 0$, let $y_j = e^{\alpha_{t-1}^B}$; if $k_j \leq 0$, let $y_j = 1$. □

Proof of Theorem 5. We first prove that, under the conditions shown in Theorem 5, i.e., Inequalities (22) and (23), an optimal solution of the problem (19)∼(21) is:

$$\boldsymbol{x}^* = \begin{cases} x_j = e^{\alpha_{t-1}^B} * m & x_j \in \boldsymbol{x}^+ \\ x_k = m & x_k \in \boldsymbol{x}^- \end{cases}, \tag{34}$$

where m is a positive real number.

In combination with Dinkelbach's Theorem, we rewrite our objective function as $\frac{Q(\boldsymbol{x})}{D(\boldsymbol{x})}$ in which $Q(\boldsymbol{x}) = \boldsymbol{q}\boldsymbol{x}$ and $D(\boldsymbol{x}) = \boldsymbol{d}\boldsymbol{x}$. Substituting $\boldsymbol{x}^*$ of Equation (34) into $Q(\boldsymbol{x})$ and $D(\boldsymbol{x})$, we have $Q(\boldsymbol{x}^*) = q(e^{\alpha_{t-1}^B} - 1) + 1$ and $D(\boldsymbol{x}^*) = d(e^{\alpha_{t-1}^B} - 1) + 1$ (recall that $q = \sum \boldsymbol{q}^+$ and $d = \sum \boldsymbol{d}^+$. Then,

5. We note that, if the DP mechanisms at each time are significantly different, or the sensitivity of queries at each time are different, Equation (31) may not be the supremum of TPL, but an upper bound of TPL.

we can rewrite Inequalities (22) and (23) in Theorem 5 as follows.

$$\frac{q_j}{d_j} > \frac{Q(\boldsymbol{x}^*)}{D(\boldsymbol{x}^*)}, \quad \forall j \in [1,n] \text{ where } q_j \in \boldsymbol{q}^+, d_j \in \boldsymbol{d}^+ \tag{35}$$

$$\frac{q_k}{d_k} \leq \frac{Q(\boldsymbol{x}^*)}{D(\boldsymbol{x}^*)}, \quad \forall k \in [1,n] \text{ where } q_k \in \boldsymbol{q}^-, d_k \in \boldsymbol{d}^- \tag{36}$$

Because $D(\boldsymbol{x}^*) > 0$, to prove $\boldsymbol{x}^*$ in (34) is an optimal solution for $\frac{Q(\boldsymbol{x})}{D(\boldsymbol{x})}$, we only need to prove the maximum value of the following equation is equal to 0.

$$\text{maximize } \{D(\boldsymbol{x}^*)Q(\boldsymbol{x}) - Q(\boldsymbol{x}^*)D(\boldsymbol{x})\} = 0. \tag{37}$$

We expand the above equation as follows.

$$\begin{aligned}\text{Eqn.(37)} &= D(\boldsymbol{x}^*)(\boldsymbol{q}^+\boldsymbol{x}^+ + \boldsymbol{q}^-\boldsymbol{x}^-) - Q(\boldsymbol{x}^*)(\boldsymbol{d}^+\boldsymbol{x}^+ + \boldsymbol{d}^-\boldsymbol{x}^-) \\ &= \big(D(\boldsymbol{x}^*)\boldsymbol{q}^+ - Q(\boldsymbol{x}^*)\boldsymbol{d}^+\big)\boldsymbol{x}^+ + \big(D(\boldsymbol{x}^*)\boldsymbol{q}^- - Q(\boldsymbol{x}^*)\boldsymbol{d}^-\big)\boldsymbol{x}^-\end{aligned} \tag{38}$$

By Equations (35) and (36)), we have $D(\boldsymbol{x}^*)\boldsymbol{q}^+ - Q(\boldsymbol{x}^*)\boldsymbol{d}^+ > 0$ and $D(\boldsymbol{x}^*)\boldsymbol{q}^- - Q(\boldsymbol{x}^*)\boldsymbol{d}^- \leq 0$. Hence, according to Lemma 1, we can obtain the maximum value in Equation (37) by setting $\boldsymbol{x}^+ = [e^{\alpha^B_{t-1}} * m]$ and $\boldsymbol{x}^- = [m]$ where m is a positive real number. We obtain the maximum value in Equation (37).

$$\begin{aligned}&\big((D(\boldsymbol{x}^*)q - Q(\boldsymbol{x}^*)d)e^{\varepsilon}m + \big(D(\boldsymbol{x}^*)(1-q) - Q(\boldsymbol{x}^*)(1-d)\big)\big)m \\ =&\big(D(\boldsymbol{x}^*)(qe^{\varepsilon} + (1-q)) - Q(\boldsymbol{x}^*)(de^{\varepsilon} + (1-d))\big)m \\ =&\big(D(\boldsymbol{x}^*)Q(\boldsymbol{x}^*) - Q(\boldsymbol{x}^*)D(\boldsymbol{x}^*)\big)m = 0\end{aligned}$$

Therefore, by Dinkelbach's Theorem, $\boldsymbol{x}^*$ is an optimal solution for the problem (19)∼(21). Substituting them into the objective function (19), we obtain the maximum value $\frac{q(e^{\alpha^B_{t-1}}-1)+1}{d(e^{\alpha^B_{t-1}}-1)+1}$. □

APPENDIX D
PROOF OF COROLLARY 2

Proof. The proof is by contradiction.

First, assume that $q = d$, i.e., $\sum \boldsymbol{q}^+ = \sum \boldsymbol{d}^+$. That is, there exist $q_j \in \boldsymbol{q}^+$ and $d_j \in \boldsymbol{d}^+$ satisfying $\frac{q_j}{d_j} \leq 1$. This leads to a contradiction to Inequality (22) because of $\frac{q_i}{d_i} > \frac{q(e^{\alpha^B_{t-1}}-1)+1}{d(e^{\alpha^B_{t-1}}-1)+1} = 1$. Hence, $q = d$ is false.

Second, assume that $q < d$, i.e., $\sum \boldsymbol{q}^+ < \sum \boldsymbol{d}^+$. Since $\sum(\boldsymbol{q}^+ + \boldsymbol{q}^-) = \sum(\boldsymbol{d}^+ + \boldsymbol{d}^-) = 1$, we have $\sum \boldsymbol{q}^- > \sum \boldsymbol{d}^-$. That is, there exist $q_k \in \boldsymbol{q}^-$ and $d_k \in \boldsymbol{d}^-$ satisfying $\frac{q_k}{d_k} > 1$. This leads to a contradiction to Inequality (23) because of $\frac{q_k}{d_k} < \frac{q(e^{\alpha^B_{t-1}}-1)+1}{d(e^{\alpha^B_{t-1}}-1)+1} < 1$. Hence, $q < d$ is false.

Therefore, $q > d$ is true. Then, we have $\frac{q(e^{\alpha^B_{t-1}}-1)+1}{d(e^{\alpha^B_{t-1}}-1)+1} > 1$. By Inequality (22), it follows that $\frac{q_j}{d_j} > \frac{q(e^{\alpha^B_{t-1}}-1)+1}{d(e^{\alpha^B_{t-1}}-1)+1} > 1$ in which $q_j \in \boldsymbol{q}^+$ and $d_j \in \boldsymbol{d}^+$. □

APPENDIX E
PROOF OF COROLLARY 3

Proof. The corollary is true because of Equation (24). □

APPENDIX F
PROOF OF THEOREM 6

Proof. It is clear that, if $q_i = d_i$ for $i \in [1,n]$ in $\boldsymbol{q}$ and $\boldsymbol{d}$, Line 9 in Algorithm 1 is always true so that the flag update is alway true until all bits are removed from $\boldsymbol{q}^+$ and $\boldsymbol{d}^+$. That is to say, $\boldsymbol{q}^+$ and $\boldsymbol{d}^+$ are empty. Hence, in this case, Algorithm 1 will be terminated with empty $\boldsymbol{q}^+$ and $\boldsymbol{d}^+$, which result in $q = d = 0$. Since for any two rows $\boldsymbol{q}$ and $\boldsymbol{d}$ in the transition matrix P_i, we have $q = d = 0$, it follows $\mathcal{L}(\cdot) = 0$. □

APPENDIX G
PROOF OF THEOREM 7

Proof. For such two rows $\boldsymbol{q}$ and $\boldsymbol{d}$ in P_i that satisfy $q_i = 1$ and $d_i = 0$, it is easy to see that Algorithm 1 results in $q = 1$, $d = 0$ and the flag update is false. Hence, we have a maximum solution of α (i.e., α^B_{t-1} or α^F_{t+1}) for such LFP problem. Since the maximum solution of LFP problem w.r.t. any two other rows in P_i cannot be larger than α, $\mathcal{L}(\cdot)$ is an identical function, i.e., $\mathcal{L}(x) = x$. □

APPENDIX H
PROOF OF THEOREM 8

Proof. According to Corollary 2, we have $\frac{q(e^{\alpha})+1}{d(e^{\alpha})+1} > 1$ because of $q > d$. By the pigeonhole principle, there only exists k cases as shown in the theorem. We only need to pove that the statements about q and d for each case are true. In case k, it is clear that $\frac{q_1}{d_1} > \frac{q_1(e^{\alpha})+1}{d_1(e^{\alpha})+1}$. In case $k-1$, the final $\boldsymbol{q}^+$ and $\boldsymbol{d}^+$ are impossible to include $\{q_3, \cdots, q_n\}$ and $\{d_3, \cdots, d_n\}$, respectively; otherwise, it violates Inequality (22). Additionally, the final $\boldsymbol{q}^+$ and $\boldsymbol{d}^+$ in case $k-1$ are impossible to exclude any elements in $\{q_1, q_2\}$ and $\{d_1, d_2\}$, respectively; otherwise, it violates Inequality (23). Similarly, we can verfiy all other cases are true. □

APPENDIX I
PROOF OF THEOREM 9

Proof. We now show that the pairs of q and d will transit from the case 1 to case k in Theorem 8 when α is increasing. When $\alpha = 0$, it is easy to see that $\frac{q(e^0-1)+1}{d(e^0-1)+1} = 1$; hence, q and d fall into case 1. We can see that $\frac{q(e^{\alpha}-1)+1}{d(e^{\alpha}-1)+1}$ continues to increase along with α untill $\alpha = \alpha_1$ which makes $\frac{q(e^{\alpha_1}-1)+1}{d(e^{\alpha_1}-1)+1} = \frac{q_k}{d_k}$ (i.e., $\alpha_1 = \log(\frac{q_k - d_k}{q*d_k - d*q_k} + 1)$), where $q = q^k = \sum_{i=1}^{k} q_i$ and $d = d^k = \sum_{i=1}^{k} d_i$. If $\alpha = \alpha_1$, q and d fall into case 2. Hence, when $0 \leq \alpha < \alpha_1$, q and d are the ones in case 1. We can also find the *transition point* α_2 by solving $\frac{q(e^{\alpha_2}-1)+1}{d(e^{\alpha_2}-1)+1} = \frac{q_{k-1}}{d_{k-1}}$ (i.e., $\alpha_2 = \log(\frac{q_{k-1} - d_{k-1}}{q*d_{k-1} - d*q_{k-1}} + 1)$), where $q = \sum_{i=1}^{k-1} q_i$ and $d = \sum_{i=1}^{k-1} d_i$. If $\alpha = \alpha_2$, q and d fall into case 3. Hence, when $\alpha_1 \leq \alpha < \alpha_2$, q and d are the ones in case 2. Similarly, we can use the same logic to find $\alpha_3, \cdots, \alpha_{k-1}$, which are called *transition points*. If $\alpha \geq \alpha_{k-1}$, q and d fall into the case k because of $\frac{q_1}{d_1} > \frac{q_1(e^{\alpha}-1)+1}{d_1(e^{\alpha}-1)+1}$ regardless of how large α is. Therefore, given $\boldsymbol{q}$ and $\boldsymbol{d}$, $f_{\boldsymbol{q},\boldsymbol{d}}(\alpha) = \frac{q(e^{\alpha}-1)+1}{d(e^{\alpha}-1)+1}$ is a piecewise function whose sub-domains and coefficients of q and d are shown in Equation (25). □

APPENDIX J
PROOF OF THEOREM 10

Proof. It is easy to see that functions $f(\alpha) = \frac{q(e^{\alpha}-1)+1}{d(e^{\alpha}-1)+1}$ and $f'(\alpha) = \frac{q'(e^{\alpha}-1)+1}{d'(e^{\alpha}-1)+1}$ are at most one intersection when $\alpha > 0$. Assume there is an intersection in $[a_1, a_2]$. Since $f(a_1) \geq f(a_1)$ and both of f and f' are increasing functions, we have $f(a_2) \leq f'(a_2)$, which is in contradiction with the given condition $f(a_2) \geq f'(a_2)$. Hence, the assumption fails and they have no intersection in $[a_1, a_2]$. In other words, $f(\alpha)$ is always larger than $f'(\alpha)$ when $a_1 < \alpha < a_2$. □

APPENDIX K
PROOF OF THEOREM 11

Proof. We denote the supremum of BPL or FPL over time by α. If $d = 0$ and $q = 1$, it is easy to see that $\mathcal{L}^B(\cdot) = \mathcal{L}^F(\cdot) = 1 * \cdot$. In other words, BPL and FPL will increase linearly without limit; hence, $\mathcal{F}$ does not exist. If BPL (or FPL) has a limit, since the current privacy leakage should be calculated based on the previous one (or the next one for FPL), we have $\alpha = \mathcal{L}^B(\alpha) + \epsilon$ (see Equations (13) or (15)), namely $\alpha = \log \frac{q(e^{\alpha}-1)+1}{d(e^{\alpha}-1)+1} + \epsilon$. By expanding this equation and letting $\mathcal{F}$ be e^{α}. we have an equation with a variable $\mathcal{F}$, which $\mathcal{F} > 1$.

$$d\mathcal{F}^2 + (qe^{\epsilon} + d - 1)\mathcal{F} + (qe^{\epsilon} - e^{\epsilon}) = 0 \tag{39}$$

Hence, the logarithm of the solution (if it exists) in the above equation is the supremum of BPL or FPL. If $d = 0$ and $q \neq 1$, we have $\mathcal{F} = \frac{(1-q)e^{\epsilon}}{1-qe^{\epsilon}}$. To ensure a positive $\mathcal{F}$, because of $(1-q)e^{\epsilon} > 0$, we need $1 - qe^{\epsilon} > 0$, i.e., $\epsilon < \log(1/q)$. Therefore, when $d = 0$, $q \neq 1$ and $\epsilon \geq \log(1/q)$, the supremum does not exist; when $d = 0$, $q \neq 1$ and $\epsilon < \log(1/q)$, we have the supremum $\log \frac{(1-q)e^{\epsilon}}{1-qe^{\epsilon}}$. If $d \neq 0$, by solving the quadratic equation (39), we can obtain a positive solution: $\frac{\sqrt{4de^{\epsilon}(1-q)+(d+qe^{\epsilon}-1)^2}+d+qe^{\epsilon}-1}{2d}$. It is easy to verify that this solution always exist and is greater than 1. Therefore, the theorem follows. □