PrivCheck: Privacy-Preserving Check-in Data Publishing for Personalized Location Based Services

Dingqi Yang[1], **Daqing Zhang**[2,3], **Bingqing Qu**[4], **Philippe Cudré-Mauroux**[1]
[1]eXascale Infolab, University of Fribourg, Fribourg, Switzerland
[2]Institut Mines-Télécom, Télécom SudParis, CNRS SAMOVAR, France; [3]Peking University, China
[4]University of Rennes 1, Renne, France
[1]{firstname.lastname}@unifr.ch, [2]daqing.zhang@it-sudparis.eu, [3]bqu@ina.fr

ABSTRACT
With the widespread adoption of smartphones, we have observed an increasing popularity of Location-Based Services (LBSs) in the past decade. To improve user experience, LBSs often provide personalized recommendations to users by mining their activity (i.e., check-in) data from location-based social networks. However, releasing user check-in data makes users vulnerable to inference attacks, as private data (e.g., gender) can often be inferred from the users' check-in data. In this paper, we propose PrivCheck, a customizable and continuous privacy-preserving check-in data publishing framework providing users with continuous privacy protection against inference attacks. The key idea of PrivCheck is to obfuscate user check-in data such that the privacy leakage of user-specified private data is minimized under a given data distortion budget, which ensures the utility of the obfuscated data to empower personalized LBSs. Since users often give LBS providers access to both their historical check-in data and future check-in streams, we develop two data obfuscation methods for historical and online check-in publishing, respectively. An empirical evaluation on two real-world datasets shows that our framework can efficiently provide effective and continuous protection of user-specified private data, while still preserving the utility of the obfuscated data for personalized LBSs.

ACM Classification Keywords
J.4 Computer Applications: Social and Behavioral Sciences

Author Keywords
Location based services; Privacy

INTRODUCTION
With the increasing popularity of smartphones and wireless networks, Location-Based Services (LBSs), such as location recommendation or search, have attracted millions of users. To develop efficient LBSs, personalization is a key ingredient to provide users with more pertinent recommendations, leading to better user experience. In the current literature, personalized LBSs are usually powered by mining users' activity data. In particular, Location Based Social Networks (LBSNs), such as Foursquare[1], have become primary data sources for developing personalized LBSs [3, 45, 42]. In LBSNs, users voluntarily share their real-time spatiotemporal activities by checking-in at certain Point of Interests (POIs). For example, a typical user check-in may indicate that the user is having dinner in a French restaurant (activity) with GPS coordinates [40.7586, -73.9791] (location) on a Friday evening (time). By analyzing large volumes of such digital footprints [48], user preferences can be extracted to enable personalized LBSs.

As large amounts of check-in data can be used to effectively mine user preferences, releasing such data often leads to serious privacy leakage. Previous work [15, 6, 17, 29, 34] has shown that the disclosure of such spatiotemporal user activity data may reveal sensitive information, such as a user's identity, health status, income level, friendship or home address, etc. Therefore, privacy protection is a critical issue in this context.

Existing privacy protection techniques for LBSs mainly focus on location privacy, i.e., obfuscating user locations before using them. A typical example is private location based query [13], where a user does not want to reveal her location information when sending a location-based query. To achieve this goal, the user's location data may be slightly altered or generalized using data obfuscation techniques such as cloaking region [15] and dummy location [30], in order to avoid revealing her real position. However, such a privacy mechanism should not be adopted by LBSNs as it hinders key benefits for their users. Specifically, users of LBSNs intendedly share their presence within their social circles by checking in at POIs. Hence, they might not appreciate the fact that the service hides their location information (even any part of their check-in data) from their friends. For example, when a user is watching a football match at a bar and wants to share her presence to potentially attract some nearby friends, she does not want her location to be obfuscated in any sense.

Since check-in data should not be obfuscated before being sent to LBSNs, an alternative solution is to protect user privacy *when publishing check-in data from LBSNs to any other third-party LBS providers*. Many LBSN third-party services require access to user check-ins in order to provide them with

[1]https://foursquare.com/

UbiComp '16, September 12-16, 2016, Heidelberg, Germany
 ISBN 978-1-4503-4461-6/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2971648.2971685

personalized LBSs. In addition to the check-in data, these services may optionally require access to other user data, such as users' profiles. While many users are willing to release their check-in data (public data) to obtain high-quality recommendation services, they may consider part of the data in their profile as private (private data), like gender or marital status. Although they may refuse to release such data, the correlations between check-in data and private data may cause privacy leakage. For example, based on user check-in data in New York City from Foursquare, we found that one's gender information can be inferred with high accuracy (see Figure 7 for details). Such an observation shows that private data often suffers from *inference attacks* [23], where an adversary analyzes a user's public data to illegitimately gain knowledge about her private data. Therefore, it is indispensable to provide privacy protection when releasing check-in data.

In the current literature, the privacy-preserving data publishing has been widely studied by database research community [12]. Its basic idea is to provide privacy guarantees by distorting the data before its publication, at the expense of a loss of *utility* of the data in the latter processing stages. In a personalized LBS scenario, the *utility of check-in data* refers to the personalization performance [18]. We consider a typical LBS, i.e., context-aware activity recommendation [45], as our target scenario. Specifically, by learning a user's preferences from her check-in data, it tries to recommend her a list of activities according to her current context (i.e., location and time). We define the utility here as the quality of the recommended ranking list, compared to the user' actual preferences.

In this paper, we study the privacy-preserving check-in data publishing problem by considering both the specific requirements of user privacy in LBSNs and the utility of the data for enabling personalized LBSs. Towards this goal, we face the following two challenges. First, since users often have different privacy concerns [19], a specific type of data (e.g., gender) may be considered as private by some users, while other users may prefer to consider it public in order to get better personalized services. Therefore, the first challenge is to provide users with *customizable privacy protection*, i.e., to protect user-specified private data only. Second, when subscribing to third-party services, users often allow the service providers to access not only their historical check-in data, but also their future check-in data streams. Although the obfuscated historical check-in data can efficiently reduce privacy leakage, the continuous release of user check-in streams will incrementally increase such leakage (see Figure 6 for details). Therefore, the second challenge is to provide *continuous privacy protection over check-in data streams*.

Aiming at overcoming the above challenges, we propose PrivCheck, a customizable and continuous privacy-preserving check-in data publishing framework against inference attacks for enabling personalized LBSs. Our framework provides continuous protection of user-specified private data against inference attacks by obfuscating both the historical and streaming check-in data before releasing them, while still preserving the utility of the published data for enabling personalized LBSs. Our main contributions are summarized as follows:

- First, unlike existing LBS privacy protection approaches that mainly focus on location privacy, we identify a privacy-preserving check-in data publishing problem by analyzing the specific privacy requirements and users' benefits of LBSNs. To the best of our knowledge, this is the first work addressing privacy-preserving data publishing for check-in data.
- Second, we propose a customizable and continuous data obfuscation framework for user check-in data from LBSNs. The key idea is to measure the privacy leakage of *user-specified private data* from public check-in data based on mutual information, and then obfuscate check-in data such that the privacy leakage is minimized under a given data distortion budget, which can ensure the utility of the released data. To handle real-world use case of third-party services built on top of LBSNs, our framework considers both historical and online check-in data publishing.
  - *Historical check-in data publishing*: When a user subscribes to a third-party service for the first time, the service provider has access to the user's entire check-in history. To obfuscate the user's historical check-in data, we minimize the privacy leakage from her historical check-in data by replacing her check-in history by that of another user whose check-in history is similar but with less privacy leakage.
  - *Online check-in data publishing*: After the user subscribed to third-party services, the service provider also has real-time access to her future check-in stream. Due to efficiency consideration, online data publishing should be performed only based on each incoming check-in itself, without accessing the user's historical check-in data. Therefore, we minimize the privacy leakage from individual check-ins by obfuscating each check-in from the check-in stream on-the-fly.
- Finally, based on real-world LBSN datasets, we conduct an extensive evaluation of our framework. The results show that our framework can continuously provide customized protection of user-specified private data, while the obfuscated data can still be exploited to enable high-quality LBSs.

RELATED WORK

Privacy issues of LBSs have been widely studied over the past decade. Most of the research efforts focus on location privacy [38, 21, 37], i.e., protecting user location information before using them. To this end, the most popular approach is location obfuscation. Its key idea is to obfuscate user location with imprecise locations, using techniques such as cloaking region [15] and dummy location [30]. However, in LBSNs, it might be inappropriate to obfuscate a user check-in data before being sent to the LBSNs, since the user intendedly shares her real time presence with her friends. Therefore, an alternative solution is to protect users' privacy when publishing their check-in data from LBSNs to any third-party service providers for enabling personalized LBSs. As we focus on this new problem in our study, our objective is to protect user private data by obfuscating publicly released data, which differs from the approaches tackling only location privacy.

In order to protect user privacy when publishing user data, the current practice mainly relies on policies or user agreements, e.g., on the use and storage of the published data [12]. However, this approach cannot guarantee that the users' sensitive information is actually protected from a malicious attacker. Therefore, to provide effective privacy protection when releasing user data, privacy-preserving data publishing has been widely studied. The key idea in this context is to obfuscate user data such that published data remains practically useful for some application scenarios while the individual's privacy is preserved. According to the attacks considered, existing work can be classified into two categories.

The first category is based on heuristic techniques to protect ad-hoc defined user privacy [12]. Specific solutions mainly tackle the privacy threat when attackers are able to link the data owner's identity to a record, or an attribute in the published data. For example, to protect user privacy from identity disclosure, K-anonymity [39] obfuscates the released data so that each record cannot be distinguished from at least k-1 other records. However, since these techniques usually have ad-hoc privacy definitions, they have been proven to be non-universal and can only be successful against limited adversaries [36].

The second category is theory-based and focuses on the uninformative principle [26], i.e., on the fact that the published data should provide attackers with as little additional information as possible beyond background knowledge. Differential privacy [9] is a well-known technique that is known to guarantee user privacy against attackers with arbitrary background knowledge. Information-theoretic privacy protection approaches have also been proposed in that context. They try to quantitatively measure privacy leakage based on various entropy-based metrics such as conditional entropy [36] and mutual information [8], and to design privacy-protection mechanisms based on those measures. Although the concept of differential privacy is stricter than that of information-theoretic approaches, the latter is intuitively more accessible and fits the practical requirements of many application domains[36]. For example, information theory can provide intuitive guidelines to quantitatively measure the amount of a user's private information (e.g., gender) that an adversary can learn by observing and analyzing the user's public data (e.g., check-in data).

In this study, we advocate the information-theoretic approach. In the current literature, information-theoretic metrics were adopted to design privacy mechanisms [11, 32, 8, 35]. Evfimievski et al. [11] leveraged randomization techniques to limit privacy leakage, without explicitly considering data utility loss constraints. Rebollo et al. [32] proposed a Kullback-Leibler divergence-based metric to evaluate the privacy disclosure and designed a privacy-preserving mechanism based on rate distortion theory. Calmon et al. [8] introduced a general framework to measure privacy leakage based on mutual information, and to protect user private data against statistic inference by considering data distortion constraints for user public data. Salamatian et al. [35] further applied this idea to study user privacy when releasing users' TV viewing records. By considering the fact that different users often have different privacy concerns [19], our approach differs from these work

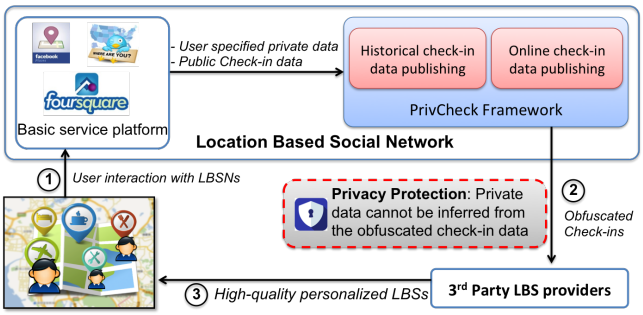


Figure 1. System workflow for privacy-preserving check-in data publishing in LBSNs: 1) Users report their check-ins to LBSNs; 2) PrivCheck publishes the obfuscated check-in data to third-party LBS providers; 3) The third-party LBS providers can still deliver high-quality personalized LBSs to users.

in offering users customizable privacy protection. Moreover, as an intrinsic streaming data source, users continuously report their check-ins online, which requires continuous privacy protection over time. Therefore, besides offering privacy protection when publishing static historical check-in data, our framework is also designed to offer users continuous privacy protection when publishing their check-in streams.

PRELIMINARIES

In this section, we first introduce our system workflow, and then a typical LBS as our targeted application. Finally, we present the data collection and preprocessing process.

System Workflow

Figure 1 illustrates the end-to-end workflow of our system. Our PrivCheck framework is implemented as a supplementary module to existing LBSN platforms. When users interact with each other in a LBSN, they voluntarily share their check-in data. When a user wants to subscribe to third-party services, she typically needs to give them access to her check-in data. Specifically, right after the user's subscription, third-party services can immediately access the user's historical check-in data. Before releasing her historical check-in data and according to the user's own criteria, the *historical check-in publishing* module obfuscates her historical check-in data to protect user-specified private data against inference attacks. Afterwards, when the user continuously share her check-ins on the LBSN, the *online check-in data publishing* module obfuscates each check-in from the check-in streams before sending to third-party services. Despite receiving obfuscated check-in data, the third-party services can still provide high-quality personalized LBSs to the users. The most important advantage of such an architecture lies on the easy integration of our framework into the existing LBSN platform, where the latter does not need to be changed significantly.

Targeted Location Based Service

In this study, we consider a typical LBS, i.e., context-aware activity recommendation [50, 45], as our target application. More precisely, knowing a user's current context (i.e., location and time), it tries to recommend her a list of activities that she may be interested in, where the activities are represented as POI categories (e.g., restaurant or nightlife spot) [31, 45]. To

deliver personalized recommendations, existing approaches mainly resort to mining users' check-in data. Specifically, check-in data can be represented as quadruples, user-time-location-activity. By aggregating and discretizing the context (i.e., time slots and location grid cells) of check-in data [50, 46] to build a user-context-activity tensor, many popular approaches leverages tensor factorization techniques for user preference prediction [33, 20, 43]. We advocate such an approach in this paper, and adopt the ranking tensor factorization algorithm from [44] to come up with a ranking list of activities for a user given her context.

Data Collection and Preprocessing

We use a check-in dataset collected from Foursquare by [40, 41]. It contains global-scale Foursquare check-ins collected via Twitter Public Streams[2] for about 18 months (from Apr. 2012 to Sep. 2013). A check-in is represented by a quadruple, user-time-location-activity. A user is uniquely identified by her ID. An activity is represented by the category corresponding to the POI [45], such as gym or restaurant. According to the aforementioned LBS scenario, we discretize the temporal and spatial dimensions of check-in data. Specifically, we map check-in time in a day onto three commonly-used time slots in LBSNs, namely morning (8:00-12:00), afternoon (12:00-20:00) and evening/night (20:00-8:00) [47]. Moreover, we empirically segment geographical areas (e.g., a city) into 1km$\times$1km grid cells. Formally, let $\mathcal{U}$, $\mathcal{A}$, $\mathcal{T}$, $\mathcal{L}$ denote the sets of users, activities, time slots, and location grid cells, respectively. We note that for a specific geographical area (e.g., a city), $\mathcal{A}$, $\mathcal{T}$, $\mathcal{L}$ are often of fixed size, while the size of $\mathcal{U}$ usually increases over time (e.g., incoming of new users).

We consider user profile data as private data in this paper. Due to Foursquare's privacy policy, only limited profile data (i.e., name and gender) is included in the check-ins[3]. Fortunately, as the dataset is collected via Twitter, we also have access to the corresponding Twitter profiles[4], which typically include additional information such as number of followers and "followings", etc. In this paper, due to the limited availability of user profile data in the collected dataset, we define two attributes as private, i.e, gender (male/female) and social status [5] (a yardstick to measure the popularity of a user in social network). For a user u, social status is computed as the ratio of the number of u's followers to the number of users u follows (i.e., "followings"): $social(u) = \frac{\#followers(u)}{\#followings(u)}$. We also discretize u' social status as popular ($social(u) > 1$) and non-popular ($social(u) \leq 1$). We note that our framework is not limited to these two types of private data, and it can incorporate any discretized/categorical attributes as private data (see Discussion section for more details).

THREAT MODEL

We consider the inference attack [23] in this study. Specifically, a user in a LBSN has two types of data, namely public check-in data (*public data*) that she is willing to publish for enabling personalized LBSs, as well as further private data that she wants to keep confidential (*private data*), such as gender. In the following, we denote public data as $X \in \mathcal{X}$, and user-specified private data as $Y \in \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are the sets of values that X and Y can take, respectively. Due to the fact that Y usually links to X by their joint probability $P(X,Y)$, an adversary who observes X is able to infer some information about Y. In order to reduce such privacy leakage, the basic idea is to release a distorted $\hat{X} \in \hat{\mathcal{X}}$ instead of X, so that it is hard to infer Y from $\hat{X}$.

We assume that an adversary has an inference attack method q that is used to get some information on Y. The adversary always tries to select q such that the inference cost of using q to infer Y is minimized. Therefore, before observing $\hat{X}$, q can be obtained by solving the following problem:

$$c = \min_{q} E_Y[C(Y,q)] \tag{1}$$

where $C(Y,q)$ is the expected cost function of inferring Y using q. After observing the distorted public data $\hat{X}$, q can be obtained by solving the following problem:

$$\hat{c} = \min_{q} E_{Y|\hat{X}}[C(Y,q)|\hat{X}] \tag{2}$$

The adversary's cost gain after observing $\hat{X}$ can be determined as follows:

$$\Delta C = c - \hat{c}, \tag{3}$$

which measures how much the adversary gains w.r.t. the inference of Y knowing $\hat{X}$. The idea of privacy protection is to find $\hat{X}$ such that the privacy leakage ΔC is minimized, while still enabling personalized LBSs based on $\hat{X}$.

Basic Idea of Our Solution

In order to reduce the privacy leakage ΔC, we obfuscate X to obtain $\hat{X}$ based on a probabilistic obfuscation function $p_{\hat{X}|X}$, which encodes the conditional probability of releasing $\hat{X}$ when observing X. Intuitively, $p_{\hat{X}|X}$ should be designed such that any inference attack on Y should be rendered weak. Meanwhile, it also keeps some utility of $\hat{X}$ by limiting the distortion budget in the obfuscation process, which can be modeled by a constraint ΔX as follows;

$$E_{\hat{X},X}(dist(\hat{X},X)) \leq \Delta X \tag{4}$$

where $dist(\hat{X},X)$ is a certain distance metric that measures the difference between $\hat{X}$ and X. ΔX limits the expected distortion w.r.t. the probabilistic obfuscation function $p_{\hat{X}|X}$. The data distortion budget can ensure the utility of the released data. Therefore, the key idea of our solution is to *learn* $p_{\hat{X}|X}$ *that minimizes* ΔC *under a given distortion budget* ΔX.

HISTORICAL CHECK-IN DATA PUBLISHING

To publish historical check-in data in a privacy-preserving way, the key idea is to probabilistically obfuscate a user's historical check-in data to that of another user, which are similar but have less privacy leakage. In this context, data obfuscation operates on one's historical check-in records, rather than obfuscating her check-in records one by one (for the incoming check-in stream). Compared to the streaming scheme, our study shows that such a historical check-in obfuscation

[2]https://dev.twitter.com/streaming/public

[3]https://developer.foursquare.com/docs/checkins/checkins

[4]https://dev.twitter.com/overview/api/users

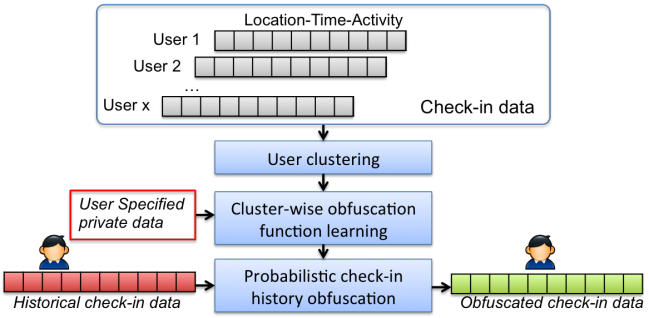


Figure 2. Historical check-in data publishing

scheme can achieve the same level of privacy protection with a lower data distortion budget (see Figure 6 for details). In the following, we first give an overview of our approach, and then describe its three key phases, i.e., user clustering, cluster-wise obfuscation function learning and probabilistic check-in history obfuscation.

Figure 2 shows the historical check-in data publishing process. First, aiming at reducing the problem complexity stemming from learning the optimal obfuscation function, we incorporate a clustering step in our framework to cluster a large number of users into a limited number of groups based on their check-in history. The underlying intuition is that similar activities from the social network, i.e., from the check-in history, usually imply similar user "lifestyles" [47], which further imply similar privacy leakage [7]. Second, based on the user clusters, we quantitatively measure the privacy leakage of user-specified private data (e.g. gender) from check-in data, and then learn the optimal obfuscation function by minimizing the privacy leakage under a given distortion budget. Finally, based on the learned obfuscation function, we perform probabilistic check-in history obfuscation. Customized privacy protection is achieved in the way that, for the specified private data (e.g., gender, social status, or both of them), a corresponding obfuscation function is generated.

User Clustering

We try to obfuscate a user's historical check-in data to that of another user. Directly learning the optimal obfuscation function $p_{\hat{X}|X}$ from individual user's check-in history incurs the complexity growing quadratically with the number of users $|\mathcal{U}|$. In order to reduce the problem complexity, the user clustering phase clusters users into a limited and fixed number of groups according to their check-in history. Then, the complexity related to learning the optimal obfuscation function between user clusters rather than between individual users is hence reduced and independent with $|\mathcal{U}|$.

Let H_u denote the check-in history of user $u(u \in \mathcal{U})$, which is a vector of size $|\mathcal{A}| * |\mathcal{T}| * |\mathcal{L}|$. Each element $H_u(a,t,l)$ indicates whether u has performed a certain spatiotemporal activity (a,t,l), where $a \in \mathcal{A}, t \in \mathcal{T}$ and $l \in \mathcal{L}$. Therefore, we can cluster the set of users $\mathcal{U}$ based on their check-in history $\{H_u | u \in \mathcal{U}\}$. As we adopt 0/1-based check-in history data, we apply average-linkage hierarchical clustering [10] using the Jaccard distance, and obtain a set of clusters $\mathcal{G}$. More complex clustering algorithms can also be applied, though this is beyond the scope of our paper.

Based on the clustering results, we then summarize the activity of a cluster to characterize the "lifestyle" of the users in that cluster. Specifically, for each cluster $g \in \mathcal{G}$, we first summarize the total number of users for each spatiotemporal activity: $D_g(a,t,l) = \sum_{u \in g} H_u(a,t,l)$, and then normalize $D_g(a,t,l)$ such that $\sum_{a \in \mathcal{A}, t \in \mathcal{T}, l \in \mathcal{L}} D_g(a,t,l) = 1$. The normalized $D_g(a,t,l)$ represents the empirical probability of the cluster g conducting activity a at time t and location l. Hence, D_g characterizes the "lifestyle" of the cluster.

Cluster-wise Obfuscation Function Learning

The optimal obfuscation function is learned based on user clusters, which implies that we try to obfuscate the users' "lifestyles" for privacy protection. *Therefore, values for the public data $\mathcal{X}$ and for the released public data $\hat{\mathcal{X}}$ refer to user clusters $\mathcal{G}$.* Without loss of generality, in the following derivation, we keep using X and Y for public and private data, respectively. The privacy leakage in this paper is measured by ΔC, which represents the information gain of an adversary after observing the released public data $\hat{X}$. When using a log-loss cost function, Calmon et al. [8] proved that ΔC becomes the mutual information between the release public data $\hat{X}$ and the specific private data Y:

$$\Delta C = I(\hat{X}, Y) = \sum_{\hat{x} \in \hat{X}, y \in Y} p(\hat{x}, y) \log \frac{p(\hat{x}, y)}{p(\hat{x}) p(y)} \tag{5}$$

As noted above, we use the probabilistic obfuscation function $p_{\hat{X}|X}$ to generate the released public data $\hat{X}$. Therefore, the joint probability of $\hat{X}$ and Y can be computed as:

$$p(\hat{x}, y) = \sum_{x \in X} p_{\hat{X}|X}(\hat{x}|x) p_{X,Y}(x, y) \tag{6}$$

The marginal probability $p_{\hat{X}}(\hat{x})$, $p_X(x)$ and $p_Y(y)$ can be calculated as follows:

$$p_{\hat{X}}(\hat{x}) = \sum_{x \in X, y \in Y} p_{\hat{X}|X}(\hat{x}|x) p_{X,Y}(x, y) \tag{7}$$

$$p_X(x) = \sum_{y \in Y} p_{X,Y}(x, y), \; p_Y(y) = \sum_{x \in X} p_{X,Y}(x, y) \tag{8}$$

Combined with the above Equations, the mutual information between the release public data $\hat{X}$ and the private data Y can be derived as:

$$I(\hat{X}, Y) = \sum_{\hat{x} \in \hat{X}, y \in Y} p(\hat{x}, y) \log \frac{p(\hat{x}, y)}{p(\hat{x})} - \sum_{y \in Y} p(y) \log p(y) \tag{9}$$

where the second term is the entropy of Y, i.e., $-\sum_{y \in Y} p(y) \log \; p(y)$, which is a constant for the specified private data (e.g., gender) in a given dataset. Hence, we ignore this term in the following derivations and obtain:

$$I(\hat{X}, Y) = \sum_{\hat{x} \in \hat{X}, y \in Y} p(\hat{x}, y) \log \frac{p(\hat{x}, y)}{p(\hat{x})} \tag{10}$$

Combined with Equations 6 and 7, the mutual information can then be derived as a function of only two factors, namely the joint probability $p_{X,Y}$ which can be empirically obtained from

a given dataset, and the obfuscation function $p_{\hat{X}|X}$:

$$I(\hat{X},Y) = \sum_{\substack{\hat{x}\in\hat{X}\\ x\in X\\ y\in Y}} p_{\hat{X}|X}(\hat{x}|x)p_{X,Y}(x,y)\log\frac{\sum_{x'\in X} p_{\hat{X}|X}(\hat{x}|x')p_{X,Y}(x',y)}{\sum_{\substack{x''\in X\\ y'\in Y}} p_{\hat{X}|X}(\hat{x}|x'')p_{X,Y}(x'',y')} \tag{11}$$

The optimal obfuscation function $p_{\hat{X}|X}$ is learned such that $I(\hat{X},Y)$ is minimized under a given distortion budget ΔX.

To quantitatively model the distortion budget ΔX for historical data publishing, we need to measure the difference between user clusters. Since we characterize user clusters using their activity distributions D_g, we resort to the Jensen-Shannon divergence [24], a symmetric and bounded metric, to quantitatively measure the difference between two activity probability distributions. Specifically, given two activity distributions D_g and $D_{\hat{g}}$, the Jensen-Shannon divergence is computed as follows:

$$JSD(D_g||D_{\hat{g}}) = \frac{1}{2}KLD(D_g||M) + \frac{1}{2}KLD(D_{\hat{g}}||M) \tag{12}$$

where $M = \frac{1}{2}(D_g + D_{\hat{g}})$, and $KLD(D_g||M)$ is the Kullback-Leibler divergence [22], which is calculated as:

$$KLD(D_g||M) = \sum_{a\in\mathcal{A},t\in\mathcal{T},l\in\mathcal{L}} \log(\frac{D_g(a,t,l)}{M(a,t,l)})\cdot D_{\hat{g}}(a,t,l) \tag{13}$$

In summary, for a given dataset, we can empirically determine $p_{G,Y}$ according to private data Y (e.g., gender, social status or their combination). Thus, the obfuscation function $p_{\hat{G}|G}$ can be learned by Algorithm 1, which contains a convex optimization problem with three constraints (can be solved by many solvers such as CVX [14]). The first constraint is for the distortion budget, and the last two constraints are probability constraints of $p_{G|\hat{G}}$. In order to stress the protected private data Y, we denote the corresponding optimal obfuscation function as $p_{\hat{G}|G,Y}$. We note that we do not assume any inference attack methods in our framework, and that any inference attacks on Y should be rendered weak by our solution.

As we try to find the optimal obfuscation probability between each pair of user clusters, the problem complexity of learning optimal $p_{\hat{G}|G}$ in Algorithm 1 is $O(n^2)$, where n is the number of user clusters $|\mathcal{G}|$ rather than the number of users $|\mathcal{U}|$. The later evaluation shows that a small number of $\mathcal{G}$ can indeed provide efficient privacy protection (see Figure 8 for details).

Probabilistic Check-in History Obfuscation

Since the learned obfuscation function is based on user clusters, we still need to bridge the gap between clusters and users to obfuscate individual user's check-in data. Algorithm 2 describes the probabilistic check-in data obfuscation process. Specifically, for a user u, we first obtain the corresponding obfuscation function $p_{\hat{G}|G,Y}$ to protect her private data Y (Line 2-3). We then obfuscate her cluster g to another $\hat{g}$ based on the obfuscation function $p_{\hat{G}|G,Y}(\hat{g}|g)$ (Line 4-5). Finally, since all users in $\hat{g}$ share the same lifestyle, we randomly select one user $\hat{u}$ in the cluster $\hat{g}$, and leverage her check-in history $H_{\hat{u}}$ to obfuscate (replace) H_u (Line 6-7).

Algorithm 1 Cluster-wise obfuscation function learning

Require: Joint probability $p_{G,Y}$, and distortion budget ΔX

1: Solve the optimization problem for $p_{\hat{G}|G}$

$$\begin{aligned} &\min_{p_{\hat{G}|G}} \ I(\hat{G},Y)\\ &s.t.,\ E_{\hat{G},G}(JSD(D_{\hat{g}}||D_g)) \le \Delta X\\ &\quad p_{\hat{G}|G}(\hat{g}|g)\in[0,1], \forall g,\hat{g}\in\mathcal{G}\\ &\quad \sum_{\hat{g}} p_{\hat{G}|G}(\hat{g}|g) = 1, \forall g\in\mathcal{G} \end{aligned}$$

2: **return** $p_{\hat{G}|G,Y}$

Algorithm 2 Probabilistic check-in history obfuscation

Require: Obfuscation functions $p_{\hat{G}|G}$ for all possible Y

1: **for** $u \in U$ **do**
2: Get u-specified private data Y
3: Get obfuscation function $p_{\hat{G}|G,Y}$ for Y
4: Get u's cluster g, where $u \in g$
5: Obfuscate the user's cluster g to $\hat{g}$ based on $p_{\hat{G}|G,Y}(\hat{g}|g)$
6: Randomly select a user $\hat{u}$ in cluster $\hat{g}$
7: Obfuscate H_u to $H_{\hat{u}}$
8: **end for**

ONLINE CHECK-IN DATA PUBLISHING

After a user's subscription to third-party services, the service providers have access to the user's future check-in streams. Therefore, we protect her private data by obfuscating her check-in streams. Different from historical check-in data publishing, the streaming nature of user check-ins imposes the following two constraints on online check-in data obfuscation. First, due to time and space efficiency requirements of real-time data publishing (i.e., single-pass processing with limited memory) [27], online check-in obfuscation can only be performed based on the incoming check-in data itself, without accessing the user's historical check-in data. Second, the temporal dimension of check-in data cannot be obfuscated, due to real-time publishing requirement. In other words, since third-party services can access user's real-time check-in streams, it is unreasonable to publish a check-in data with a time stamp other than the current time.

Considering these two constraints, the key idea of online check-in publishing is to obfuscate the spatial and activity dimensions for each incoming check-in in a real-time manner. Figure 3 shows the online check-in data publishing process. First, by measuring the privacy leakage of the user-specified private data from each activity-location pair, we learn the optimal obfuscation function between those pairs such that the privacy leakage is minimized under a given distortion budget. Second, for each incoming check-in data (i.e., user-time-location-activity) in the check-in streams, we obfuscate its activity and location according to the learned obfuscation function.

Check-in-wise Obfuscation Function Learning

The check-in-wise obfuscation function is learned based on activity-location pairs, denoted as (a,l). *Therefore, values for the public data $\mathcal{X}$ and for the released public data $\hat{\mathcal{X}}$*

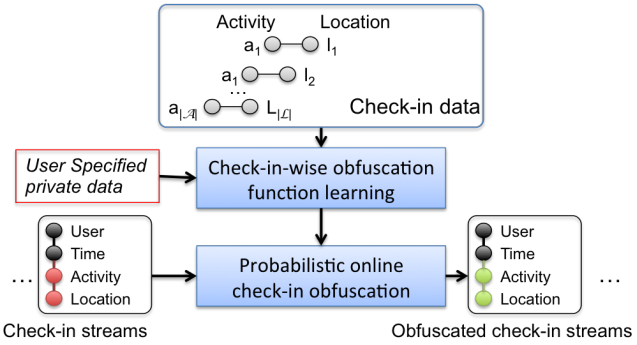


Figure 3. Online check-in data publishing

Algorithm 3 Check-in-wise obfuscation function learning

Require: Joint probability $p_{S,Y}$, distortion budget ΔX

1: Solve the optimization problem for $p_{\hat{S}|S}$

$$\min_{p_{\hat{S}|S}} \; I(\hat{S}|S,Y)$$
$$s.t., \; E_{\hat{S},S}(Dist((a,l),(\hat{a},\hat{l}))) \leq \Delta X$$
$$p_{\hat{S}|S}(\hat{a},\hat{l}|a,l) \in [0,1], \forall (a,l),(\hat{a},\hat{l}) \in \mathcal{S}$$
$$\sum_{(\hat{a},\hat{l})} p_{\hat{S}|S}(\hat{a},\hat{l}|a,l) = 1, \forall (a,l) \in \mathcal{S}$$

2: **return** $p_{\hat{S}|S,Y}$

refer to $\mathcal{S} = \{(a,l)|a \in \mathcal{A}, l \in \mathcal{L}\}$. In order to minimize the privacy leakage of private data Y from public data S ($S \in \mathcal{S}$), we follow a similar process as for cluster-wise obfuscation function learning, and try to minimize $I(\hat{S},Y)$.

To model the distortion budget for check-in streams, we only care about the percentage of the check-ins that have been modified. In other words, a check-in is modified if at least one of its dimensions has been changed (location or activity in our case). Therefore, to compute the distance between two activity-location pairs (a,l) and $(\hat{a},\hat{l})$, we use the following function:

$$Dist((a,l),(\hat{a},\hat{l})) = \begin{cases} 0, \; if \;\; a = \hat{a} \; and \; l = \hat{l} \\ 1, \; otherwise \end{cases} \quad (14)$$

In summary, we first empirically calculate $p_{S,Y}$, and learn the optimal obfuscation function $p_{\hat{S}|S,Y}$ using Algorithm 3 which is also a convex optimization problem. The complexity of solving the optimization problem in Algorithm 3 depends only on $|\mathcal{A}|$ and $|\mathcal{L}|$. Since $\mathcal{A}$ and $\mathcal{L}$ are of fixed size for a specific city, the complexity does not increase over time.

Probabilistic Online Check-in Obfuscation

Based on the learned obfuscation functions, we obfuscate each incoming check-in from the check-in stream using Algorithm 4. For each incoming check-in *u-a-t-l*, we first obtain the corresponding obfuscation function $p_{\hat{S}|S,Y}$ to protect u-specified private data Y (Line 1-2). We then obfuscate the activity and location of the check-in based on $p_{\hat{S}|S,Y}(\hat{a},\hat{l}|a,l)$, and return *u-â-t-l̂* as obfuscated check-in data (Line 3-4).

Algorithm 4 Probabilistic online check-in obfuscation

Require: Obfuscation functions $p_{\hat{S}|S}$ for all possible Y, a check-in *u-a-t-l*

1: Get u-specified private data Y
2: Get obfuscation function $p_{\hat{S}|S,Y}$ for Y
3: Obfuscate (a,l) to $(\hat{a},\hat{l})$ based on $p_{\hat{S}|S,Y}(\hat{a},\hat{l}|a,l)$
4: **return** obfuscated check-in *u-â-t-l̂*

Table 1. Characteristics of the experimental dataset

Dataset	New York City(NYC)	Tokyo(TKY)
User number	3,669	6,870
Check-in number	893,722	1,290,445
Activity number	141	113
Time slot number	3	3
Location grid cell number	179	106

EXPERIMENTAL EVALUATION

In this section, we empirically evaluate the effectiveness and efficiency of our framework. Specifically, based on two real-world LBSN datasets, we first evaluate the privacy protection performance of our framework by comparing it to state-of-the-art approaches, and analyze the trade-off between privacy protection and personalization performance in LBSs. Subsequently, we study the continuous privacy protection performance by evaluating the privacy leakage over time. Afterward, we evaluate the customization performance of privacy protection by comparing the privacy leakage of user-specified private data and that of other data. Finally, we evaluate the runtime performance of our framework. We start by introducing our experimental setup below, including descriptions of our datasets, baseline approaches and evaluation metrics, before reporting on the evaluation results.

Experimental Setup

Dataset

Following the aforementioned data collection and processing steps, we gather the check-in data and the corresponding private data (i.e., gender and social status) for users in New York city (NYC) and Tokyo (TKY). Table 1 shows the statistics of the resulting datasets.

Baseline Approaches

In order to demonstrate the effectiveness of our framework, we compare it with the following privacy-preserving obfuscation approaches:

- Random obfuscation (*Rand*). For historical check-in obfuscation, it randomly obfuscates each $H_u(a,t,l)$ with a given probability p_{rand}, i.e., if $H_u(a,t,l)$=1 (or 0), we obfuscate it to 0 (or 1) with probability p_{rand}. For online check-in obfuscation, it randomly obfuscates each check-in *u-a-t-l* to another *u-â-t-l̂* with probability p_{rand}. Here, p_{rand} controls the distortion budget in both cases.
- *Frapp* [1]. It is a generalized matrix-theoretic framework of data perturbation for privacy-preserving mining. Its key idea is to obfuscate one's check-in data to itself with higher probability than to others. For historical check-in obfuscation, it obfuscates a user u's check-in H_u to $H_{u'}$ with probability $p_{frapp} = \gamma e$ if $u = u'$, otherwise $p_{frapp} = e$. Here e is used

for probability normalization, i.e., $e = \frac{1}{\gamma+|\mathcal{U}|-1}$. For online check-in obfuscation, it obfuscates each check-in u-a-t-l to another u-$\hat{a}$-t-$\hat{l}$ with probability $p_{frapp} = \gamma e$ if $a = \hat{a}$ and $l = \hat{l}$, otherwise $p_{frapp} = e$ (here $e = \frac{1}{\gamma+|\mathcal{A}|*|\mathcal{L}|-1}$). The distortion budget is controlled by γ in both cases.

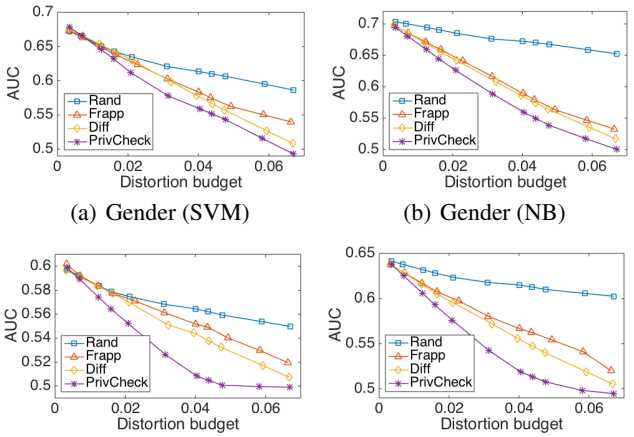
- Differential privacy (*Diff*). Differential privacy [9] is a state-of-the-art privacy protection method. Its key idea is to protect privacy regardless of the adversary's prior knowledge. We adopt exponential mechanism [16] for implementing differential privacy for discrete data. For historical check-in obfuscation, it obfuscates H_u to $H_{u'}$ with a probability that decreases exponentially with the distance $d(H_u, H_{u'})$, i.e., $p_{diff}(H_{u'}|H_u) \propto \exp(-\beta d(H_u, H_{u'}))$, where $\beta \geq 0$. β actually controls the distortion budget. The exponential mechanism satisfies $2\beta d_{max}$-differential privacy [16], where $d_{max} = \max_{u,u' \in U} d(H_{u'}, H_u)$. Since differential privacy directly obfuscates the individual's check-in data rather than the summarized activity distribution, we adopt the Jaccard distance to compute $d(H_u, H_{u'})$. For online check-in obfuscation, this method obfuscates each check-in with $p_{diff}(\hat{a}, \hat{l}|a, l) \propto \exp(-\beta Dist((a,l),(\hat{a},\hat{l})))$. Combining with the distance function in Equation 14, we derive that $p_{diff} \propto 1$ if $a = \hat{a}$ and $l = \hat{l}$, otherwise $p_{diff} \propto \exp(-\beta)$. Therefore, *Diff* is equivalent to *Frapp* in the case of online check-in obfuscation, only with a different way of controlling the distortion budget.

Evaluation Metrics

Privacy evaluation is traditionally based on simulations, and tries to show that the defined privacy is satisfied with a reasonable computation overhead [49]. In this paper, we take a step forward to quantitatively evaluate both our privacy protection and data utility. Specifically, we implement two inference attack methods to directly assess the performance of our privacy protection and use a real-world LBS to evaluate the resulting utility of the obfuscated data.

Privacy. Inference attacks [23] on private data try to infer a user's private information Y (e.g., gender) from her released public data $\hat{X}$, which can be regarded as a classification problem for discrete data. Therefore, we adopt here two common classification algorithms as inference attack methods, namely Support Vector Machine (SVM) and Naive Bayes (NB). We assume that adversaries have trained their classifiers based on the original public data X and private data Y from some non privacy-conscious users [8], who do not care about their privacy and publish all their data. We randomly sample 50% of all users as such non privacy-conscious users for training the classifiers, and then perform inference attacks on the private data Y of the rest of the users based on their obfuscated activity data $\hat{X}$. We use the *Area Under the Curve (AUC)*, which is a widely used metric for classification problem [25], to evaluate the performance of the inference attacks. *A better privacy protection implies lower AUC for both inference attack methods.* The ideal privacy protection is achieved when $AUC = 0.5$, which implies that any inference attack method performs no better than a random guess.

Utility. In this work, utility corresponds to the degree to which data can be used to power personalized LBSs. As aforementioned, we adopt the personalized recommendation algorithm from [44] to come up with a ranking of activities for a user given her context. To apply this algorithm, we first randomly split the original public data X into a training dataset X_{train} (80%) and a test set X_{test} (20%), and then use our framework to obfuscate X_{train} to $\hat{X}_{train}$. Subsequently, we apply the recommendation algorithm on the obfuscated data $\hat{X}_{train}$, which would be released to third-party services, and then make predictions for the test dataset X_{test}, which represents the users' true preference. The goal is to verify that the obfuscated data $\hat{X}_{train}$ can still be used to accurately predict the users' true preference in X_{test}. To evaluate the quality of the resulting recommendations, we use *Mean Average Precision (MAP)* [2], which is a widely used metric from Information Retrieval to assess the quality of rankings. *Higher value of MAP implies better personalization performance.*



(a) Gender (SVM) (b) Gender (NB)

(c) Social status (SVM) (d) Social status (NB)

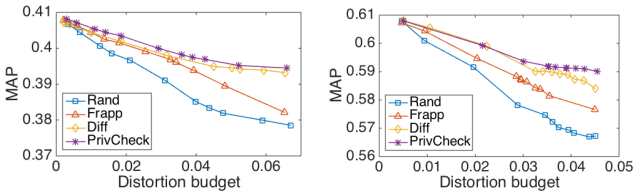
Figure 4. Privacy protection performance for different distortion budgets on NYC dataset

Privacy & Utility Trade-off for Historical Data Publishing

In this experiment, we vary the parameters that control the distortion budget for our method and baselines, and observe the resulting trade-off between privacy (AUC) and utility (MAP). As our framework and baselines use different parameters to control the distortion budget, we report all the experiment results with a unified $\Delta X = \frac{|X-\hat{X}|}{|X|}$ (the relative difference between X and $\hat{X}$). We consider both user gender and social status as private data. We set the number of user clusters to 200 (the selection of this parameter is discussed later). Each result we report is the mean value of ten repeated trials.

Figure 4 shows the results of the privacy protection performance with different distortion budgets for the NYC dataset. In general, higher distortion budgets imply a better privacy protection. Compared to the baselines and for a give distortion budget, our framework systematically achieves a better privacy protection (i.e., lower AUC) for both types of private data against any inference attack. We observe similar results on the TKY dataset, but do not show them due to space limitations.

Figure 5 gives the utility results for different distortion budgets. We observe that the higher data distortion budget leads



(a) New York dataset (b) Tokyo dataset

Figure 5. Utility performance for different distortion budgets

to a lower performance in LBS personalization. Compared to the baselines, our framework, which considers the users' "lifestyles" when obfuscating their activity data, always results in higher MAP values under the same distortion budget. Particularly, with the results shown in Figure 4, we see that PrivCheck achieves better privacy protection and data utility at the same time compared to differential privacy [28].

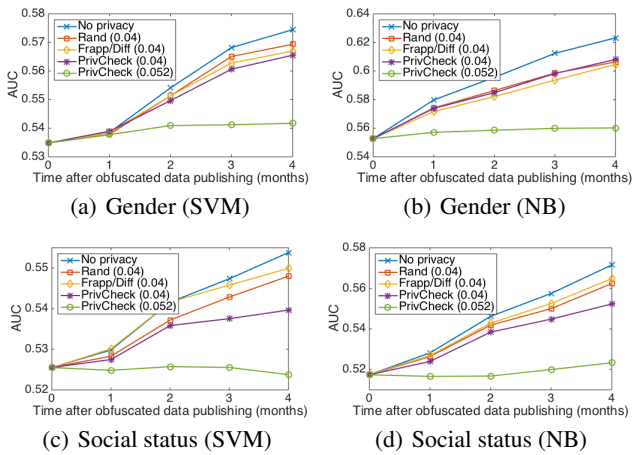
Privacy Protection over Check-in Streams

To study the privacy protection performance over time, we first obfuscate the historical check-in data using our method with a distortion budget of 0.04, and then compare different online obfuscation methods for the future check-in streams. As our dataset contains check-ins for 18 months, we select the first 14 month data as historical check-ins, and the last 4 month data as check-in streams.

Figure 6 shows the privacy performance over time on the NYC dataset. (We obtain similar results on the TKY dataset, which are not shown due to space limitations). As noted previously, *Frapp* and *Diff* are equivalent for online data publishing and have the same results under the same data distortion budget. We observe in Figure 6 that although our obfuscated historical check-ins effectively preserve user privacy, the privacy leakage incrementally increases over time with the release of further check-ins. When keeping the same distortion budget (0.04) for all online obfuscation methods, PrivCheck outperforms other baselines by achieving lower AUC. However, all the methods still show increasing privacy leakage over time. In order to keep the same level of privacy protection, we need to increase the distortion budget to 0.052 for our method. Compared to historical data obfuscation, online obfuscation needs more distortion budget (0.052 rather than 0.04) to keep same level of privacy protection. This is due to the fact that online obfuscation can only be done based on the incoming check-in itself without an overview of users' whole check-in history (real-time data publishing constraint).

Performance of Customized Privacy Protection

Since users often have different privacy requirements, our framework is designed to protect user-specific private data. According to the two types of private data, users have three options according to the types of private data to be protected: 1) protecting gender only (PrivCheck-Gender); 2) protecting social status only (PrivCheck-Social); 3) protecting both gender and social status (PrivCheck-Both). In this experiment, we configure our framework with those three settings, and report on the customized privacy protection performance. We fix the distortion budget to 0.04 for all historical data obfuscation methods, and to 0.052 for all online data obfuscation methods.



(a) Gender (SVM) (b) Gender (NB)

(c) Social status (SVM) (d) Social status (NB)

Figure 6. Privacy protection performance over time on NYC dataset

In this way, we keep the same level of privacy protection over time. The reported results are the average values over months.

Figure 7 shows the privacy protection results for both gender and social status, using the New York city and Tokyo datasets, respectively. We observe that PrivCheck-Gender (or PrivCheck-Social) outperforms all other methods when protecting the targeted gender data (or social status), by achieving the lowest AUC. Particularly, compared to PrivCheck-Both, which treats both data as private, PrivCheck-Gender (or PrivCheck-Social) can achieve a lower AUC for gender (or social status); Indeed, better privacy protection can be achieved under a given distortion budget when less private data has to be protected.

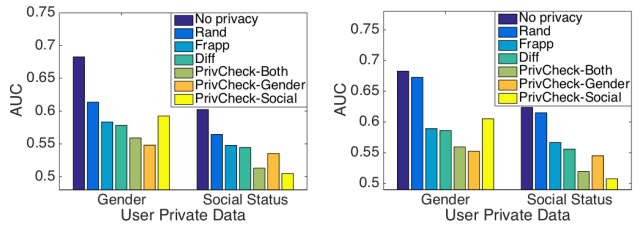
In addition, we observe that different types of private data suffer from different levels of privacy leakage. For example, Figure 7 shows that a user's gender can be inferred with a higher AUC than that of her social status. In practice, this observation can be used to help users decide which private data should be protected, by providing them with a quantitative metric based on AUC to indicate the privacy leakage of all potential private data.

Runtime Performance

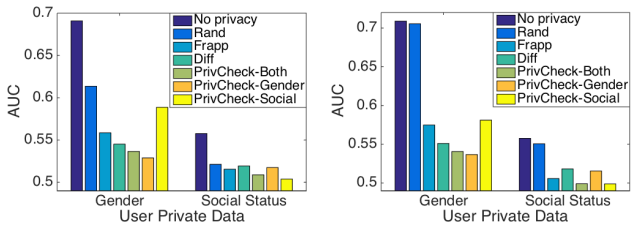
As our framework includes both a historical and an online check-in publishing modules, we separately discuss their runtime performance. The prototype of our framework is implemented on a commodity PC (Intel Core i7-4770HQ@2.20GHz, 16GB RAM, OS X), running MATLAB and CVX [14] library to solve the optimization problems in Algorithm 1 and 3.

Historical Check-in Data Publishing

For obfuscation function learning, we adopt a user clustering step to reduce the problem complexity. In this experiment, we study the learning time of the obfuscation function using Algorithm 1 w.r.t. the number of user clusters, and its influence on privacy protection performance. We fix the distortion budget to 0.04, and vary the number of user clusters. Figure 8 shows the AUC results and the time needed for learning the optimal obfuscation function for different numbers of user clusters on both datasets. On the one hand, we observe that AUC decreases with an increasing number of clusters. A larger number

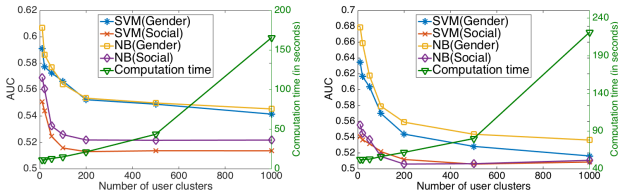


(a) SVM on NYC dataset (b) NB on NYC dataset



(c) SVM on TKY dataset (d) NB on TKY dataset

Figure 7. Customization performance of privacy protection



(a) New York city dataset (b) Tokyo dataset

Figure 8. Runtime and privacy performance for different number of user clusters

of user clusters usually implies finer-grained "lifestyles". By learning the obfuscation function over fine-grained clusters (more obfuscation lifestyle candidates), our method is more probable to find the optimal function that achieves a better privacy protection under the same distortion budget. Furthermore, we observe no significant reduction in AUC when the number of clusters gets higher than 200, which implies that 200 "lifestyles" are sufficient to characterize the privacy leakage between user activity data and their private data. On the other hand, the computation time continuously increases with the increasing number of clusters. Therefore, we selected 200 as the number of user clusters in the previous experiments. We note that learning the optimal obfuscation function is an offline step, and only depends on the joint probability $p_{G,Y}$. In practice, we can regularly update the obfuscation function in order to sustain its effectiveness.

Based on the learned obfuscation function, the probabilistic check-in history obfuscation in Algorithm 2 can be efficiently performed (0.49 seconds on both datasets).

Online Check-in Data Publishing

The complexity of obfuscation function learning for online check-in data publishing (Algorithm 3) depends only on $\mathcal{A}$ and $\mathcal{L}$, which are constant for a specific city. As shown in Table 2, our test PC is able to learn the optimal obfuscation function in 120 and 276 seconds for NYC and TKY datasets, respectively. We note again that the obfuscation function learning is an offline step, which can be regularly updated.

Table 2. Runtime performance for online check-in data publishing

Dataset	NYC	TKY
Obfuscation function learning time	120 sec	276 sec
Online check-in obfuscation speed	2,200 check-in/sec	

Due to the streaming nature of check-in data, the efficiency of probabilistic online check-in obfuscation is particularly important. As shown in Table 2, our method (Algorithm 4) is able to perform the obfuscation process with a speed of 2,200 check-in/second on both datasets, which can easily accommodate the current Foursquare check-in stream where the peak-day record shows 7 million check-ins/day [4] (about 81 check-ins/sec on average).

DISCUSSION

Private data type. As PrivCheck is designed to handle discrete data, it can incorporate any discrete/categorical attributes as private data. In this study, due to the data collection limitation, the selected two attributes are all with a small set of values (i.e., male/female for gender, and popular/non-popular for social status). Theoretically, attributes with a smaller set of values will be better protected under the same distortion budget. The evaluation also shows this effect. When considering both gender and social status as private data (Figure 7), we actually try to obfuscate for four cases (i.e., the combination of the two value sets). This is equivalent to one attribute with a set of four values. The results show that privacy is better protected when obfuscating for two values than for four values.

Data discretization. We note that the discretization of check-in data is a common and practical approach in developing LBSs [50, 46]. Although the discretization of the temporal and spatial dimensions introduces certain data obfuscation, our study shows that the discretized check-in data still causes privacy leakage (see Figure 7), and PrivCheck can provide effective protection against such privacy threats.

CONCLUSIONS AND FUTURE WORK

In this paper, we introduced PrivCheck, a customizable and continuous privacy-preserving check-in data publishing framework. It can protect user-specified data against inference attacks by releasing obfuscated check-in data, while still ensuring the utility of the released data to power personalized location-based services. Based on real-world scenarios, our framework considers not only historical check-in data publishing, but also online check-in publishing for check-in streams. We showed through extensive experiments on two real-world LBSN datasets that PrivCheck can provide an efficient and effective protection of private data against different inference attacks, while still preserving the utility of the published data for context-aware activity recommendation.

In the future, we plan to extend our framework by considering the data types with continuous values rather than discretized values, and explore other data utility other than personalized LBSs, such as providing aggregate demographic metrics.

ACKNOWLEDGMENTS

This work was supported by the Swiss National Science Foundation under grant number PP00P2_153023, and the National Natural Science Foundation of China Grant No. 61572048.

REFERENCES

1. Shipra Agrawal and Jayant R Haritsa. 2005. A framework for high-accuracy privacy-preserving mining. In *Proc. of the ICDE'05*. IEEE, 193–204.
2. Ricardo Baeza-Yates, Berthier Ribeiro-Neto, and others. 1999. *Modern information retrieval*. Vol. 463. ACM press New York.
3. Jie Bao, Yu Zheng, David Wilkie, and Mohamed Mokbel. 2015. Recommendations in location-based social networks: a survey. *GeoInformatica* 19, 3 (2015), 525–565.
4. Foursquare Blog. 2015. Foursquare Check-in Record. Available: http://blog.foursquare.com/post/130625318273/7-million-check-ins. (2015). Accessed: 2016-01-18.
5. Zhiyuan Cheng, James Caverlee, Kyumin Lee, and Daniel Z Sui. 2011. Exploring Millions of Footprints in Location Sharing Services. *Proc. of ICWSM'11* 2011 (2011), 81–88.
6. Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *Proc. of KDD'11*. ACM, 1082–1090.
7. Ratan Dey, Cong Tang, Keith Ross, and Nitesh Saxena. 2012. Estimating age privacy leakage in online social networks. In *Proc. of INFOCOM'12*. IEEE, 2836–2840.
8. Flávio du Pin Calmon and Nadia Fawaz. 2012. Privacy against statistical inference. In *Proc. of Allerton'12*. IEEE, 1401–1408.
9. Cynthia Dwork. 2006. Differential privacy. In *Automata, languages and programming*. Springer, 1–12.
10. Michael B Eisen, Paul T Spellman, Patrick O Brown, and David Botstein. 1998. Cluster analysis and display of genome-wide expression patterns. *PNAS* 95, 25 (1998), 14863–14868.
11. Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. 2003. Limiting privacy breaches in privacy preserving data mining. In *Proc. of PODS'03*. ACM, 211–222.
12. Benjamin Fung, Ke Wang, Rui Chen, and Philip S Yu. 2010. Privacy-preserving data publishing: A survey of recent developments. *ACM Computer Survey* 42, 4 (2010), 14.
13. Gabriel Ghinita, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, and Kian-Lee Tan. 2008. Private queries in location based services: anonymizers are not necessary. In *Proc. of SIGMOD'08*. ACM, 121–132.
14. Michael Grant and Stephen Boyd. 2008. Graph implementations for nonsmooth convex programs. In *Recent Advances in Learning and Control*. Springer, 95–110.
15. Marco Gruteser and Dirk Grunwald. 2003. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proc. of MobiSys'03*. ACM, 31–42.
16. Zhiyi Huang and Sampath Kannan. 2012. The exponential mechanism for social welfare: Private, truthful, and nearly optimal. In *Proc. of FOCS'12*.
17. Zheng Huo, Xiaofeng Meng, and Rui Zhang. 2013. Feel free to check-in: Privacy alert against hidden location inference attacks in GeoSNs. In *Database Systems for Advanced Applications*. Springer, 377–391.
18. Stratis Ioannidis, Andrea Montanari, Udi Weinsberg, Smriti Bhagat, Nadia Fawaz, and Nina Taft. 2014. Privacy tradeoffs in predictive analytics. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 42. ACM, 57–69.
19. Iris A Junglas, Norman A Johnson, and Christiane Spitzmüller. 2008. Personality traits and concern for privacy: an empirical study in the context of location-based services. *European Journal of Information Systems* 17, 4 (2008), 387–402.
20. Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. 2010. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proc. of RecSys'10*. ACM, 79–86.
21. Vassilis Kostakos, Jayant Venkatanathan, Bernardo Reynolds, Norman Sadeh, Eran Toch, Siraj A Shaikh, and Simon Jones. 2011. Who's your best friend?: targeted privacy attacks In location-sharing social networks. In *Proc. of UbiComp'11*. ACM, 177–186.
22. Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The Annals of Mathematical Statistics* (1951), 79–86.
23. Chen Li, Houtan Shirani-Mehr, and Xiaochun Yang. 2007. Protecting individual information against inference attacks in data publishing. In *Advances in Databases: Concepts, Systems and Applications*. Springer, 422–433.
24. Jianhua Lin. 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory* 37, 1 (1991), 145–151.
25. Charles X Ling, Jin Huang, and Harry Zhang. 2003. AUC: a better measure than accuracy in comparing learning algorithms. In *Advances in Artificial Intelligence*. Springer, 329–341.
26. Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. 2007. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data* 1, 1 (2007), 3.
27. Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G Lindsay. 1998. Approximate medians and other quantiles in one pass and with limited memory. In *ACM SIGMOD Record*, Vol. 27. ACM, 426–435.

28. Frank McSherry and Kunal Talwar. 2007. Mechanism design via differential privacy. In *Proc. of FOCS'07*. IEEE, 94–103.
29. Elham Naghizade, James Bailey, Lars Kulik, and Egemen Tanin. 2015. How Private Can I Be Among Public Users?. In *Proc. of UbiComp'15*. 1137–1141.
30. Ben Niu, Qinghua Li, Xiaoyan Zhu, Guohong Cao, and Hui Li. 2014. Achieving k-anonymity in privacy-aware location-based services. In *Proc. of INFOCOM'14*. IEEE, 754–762.
31. Anastasios Noulas and Cecilia Mascolo. 2013. Exploiting foursquare and cellular data to infer user activity in urban environments. In *Proc. of MDM'13*, Vol. 1. IEEE, 167–176.
32. David Rebollo-Monedero, Jordi Forne, and Josep Domingo-Ferrer. 2010. From t-closeness-like privacy to postrandomization via information theory. *IEEE Transactions on Knowledge and Data Engineering* 22, 11 (2010), 1623–1636.
33. Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *Proc. of SIGIR'11*. ACM, 635–644.
34. Luca Rossi, Matthew J. Williams, Christopher Stich, and Mirco Musolesi. 2015. Privacy and the City: User Identification and Location Semantics in Location-Based Social Networks. In *Proc. of ICWSM'15*. AAAI.
35. Salman Salamatian, Amy Zhang, Flavio du Pin Calmon, Sandilya Bhamidipati, Nadia Fawaz, Branislav Kveton, Pedro Oliveira, and Nina Taft. 2013. How to hide the elephant-or the donkey-in the room: Practical privacy against statistical inference for large data. In *Proc. of GlobalSIP*. IEEE.
36. Lalitha Sankar, S Raj Rajagopalan, and H Vincent Poor. 2013. Utility-privacy tradeoffs in databases: An information-theoretic approach. *IEEE Transactions on Information Forensics and Security* 8, 6 (2013), 838–852.
37. Marcello Paolo Scipioni and Marc Langheinrich. 2012. To Share or Not To Share? An Activity-centered Approach for Designing Usable Location Sharing Tools. In *Proc. of U-PriSM'12*.
38. Reza Shokri, George Theodorakopoulos, Jean-Yves Le Boudec, and Jean-Pierre Hubaux. 2011. Quantifying location privacy. In *Proc. of the 2011 IEEE Symposium on Security and Privacy*. IEEE, 247–262.
39. Latanya Sweeney. 2002. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 05 (2002), 557–570.
40. Dingqi Yang, Daqing Zhang, Longbiao Chen, and Bingqing Qu. 2015. NationTelescope: Monitoring and visualizing large-scale collective behavior in LBSNs. *Journal of Network and Computer Applications* 55 (2015), 170–180.
41. Dingqi Yang, Daqing Zhang, and Bingqing Qu. 2016. Participatory Cultural Mapping Based on Collective Behavior Data in Location-Based Social Networks. *ACM Transactions on Intelligent Systems and Technology (TIST)* 7, 3 (2016), 30.
42. Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhu Wang. 2013a. A sentiment-enhanced personalized location recommendation system. In *Proc. of HT'13*. ACM, 119–128.
43. Dingqi Yang, Daqing Zhang, Zhiyong Yu, and Zhiwen Yu. 2013b. Fine-grained preference-aware location search leveraging crowdsourced digital footprints from LBSNs. In *Proc. of UbiComp'13*. ACM, 479–488.
44. Dingqi Yang, Daqing Zhang, Zhiyong Yu, Zhiwen Yu, and Djamal Zeghlache. 2014. SESAME: Mining user digital footprints for fine-grained preference-aware social media search. *ACM Transactions on Internet Technology (TOIT)* 14, 4 (2014), 28.
45. Dingqi Yang, Daqing Zhang, Vincent. W. Zheng, and Zhiyong Yu. 2015. Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs. *IEEE Transactions on System, Man, Cybernetics: System* 45, 1 (2015), 129–142.
46. Jihang Ye, Zhe Zhu, and Hong Cheng. 2013. WhatâĂŹs your next move: User activity prediction in location-based social networks. In *Proceedings of the SIAM International Conference on Data Mining. SIAM*. SIAM.
47. Nicholas Jing Yuan, Fuzheng Zhang, Defu Lian, Kai Zheng, Siyu Yu, and Xing Xie. 2013. We know how you live: exploring the spectrum of urban lifestyles. In *Proc. of COSN'13*. ACM, 3–14.
48. Daqing Zhang, Bin Guo, and Zhiwen Yu. 2011. The emergence of social and community intelligence. *IEEE Computer* 44, 7 (2011), 21–28.
49. Xinxin Zhao, Lingjun Li, and Guoliang Xue. 2013. Checking in without worries: Location privacy in location based social networks. In *Proc. of INFOCOM'13*. IEEE, 3003–3011.
50. Vincent W Zheng, Yu Zheng, Xing Xie, and Qiang Yang. 2010. Collaborative location and activity recommendations with gps history data. In *Proc. of WWW'10*. ACM, 1029–1038.